Minimum Memory Implementations of the Lifting Scheme

Christos Chrysafis
Hewlett-Packard Laboratories
1501 Page Mill Road, Bldg.3U-3
Palo Alto, CA 94304-1126
chrysafi@hpl.hp.com

Antonio Ortega
Integrated Media Systems Center
University of Southern California
Los Angeles, CA 90089-2564
ortega@sipi.usc.edu

ABSTRACT

All publications on the lifting scheme up to now [1, 2] consider non-causal systems, where the assumption is that the whole input signal is buffered. This is problematic if we want to use lifting in a low memory scenario. In this paper we present an analysis for making a *lifting implementation of a filter bank causal*, while at the same time *reducing the amount of delay* (or memory) needed for the whole system. The amount of memory needed for the lifting implementation of any filter bank can be shown to be always smaller than the corresponding convolution implementation. The amount of memory savings is filter bank dependent, it ranges from no savings for the Haar transform to 40% for a $2-10$ filter bank. The amount of savings depends on the number of lifting steps as well as the length of the lifting steps used. We will also elaborate on the use of *boundary extensions* on each lifting step instead of the whole signal. This leads to lower memory requirements as well as simpler implementations.

Keywords: Lifting Scheme, Wavelet Transforms

1 INTRODUCTION

There have been a number of publications that propose constructing wavelet transforms as a series of *lifting* and *dual lifting* steps [1, 2], or *prediction* and *update* steps. A number of authors worked on different aspects of the lifting implementations of wavelets [4, 5]. The lifting approach to WT allows for arbitrary decompositions including nonlinear filters. Complexity trade-offs for this approach have not yet been studied in depth. There have been no general theoretical analyses of the memory reduction potential of using lifting implementation of wavelets. Consider the lifting structure in Figure 1, where filters $P_k(z)$ and $U_k(z)$ may or may not be causal. The structure provides an easy way to understand the lifting implementation for one dimensional signals. Prediction and update operators can have any form. For example we can apply linear or non linear operator and we can even quantize the different prediction values. The filter bank implemented using a lifting structure will always be perfect reconstruction.

A "naive" implementation would consist of applying each filter, for example $P_0(z)$, to the whole signal, storing the result, then applying $U_0(z)$ to the output that was stored. If we follow the update/predict steps in the sequence they appear for the whole signal length, one step at a time, memory of the order of the signal length is needed. Whole signal buffering is undesirable if not impossible in many applications such as real time audio processing or wavelet image compression of large images.

In this paper we will study how to implement such systems with minimal memory, i.e., such that outputs can be produced as soon as sufficient inputs are available. We will discuss causality in lifting structures. In section 2 we will rearrange the delay elements in a lifting structure in order to make the whole system causal. After causality has been achieved we will look into "sharing" certain delay elements in between adjacent lifting steps so that we can minimize the amount of delay (memory) needed. In section 3 we will discuss about the advantages of symmetric lifting steps for fast implementations and we will give a procedure to recursively construct symmetric filters with theorem 3.1 . In section 5 we will describe the process of applying symmetric extension in lifting structures, and in Appendix A we will give a number of examples of memory savings for different filter banks with the proposed approach.

2 LIFTING IMPLEMENTATIONS IN LOW MEMORY

The system in Figure 1 may have to be used as a causal system, for example for audio processing applications or for a low memory wavelet transform. That is, we may not have access to the whole signal at once and, instead, we may receive input samples one sample at a time. We would like to process data as we receive them and therefore we need to have a causal system. Our goal is to generate outputs while buffering the minimum number of data. In this case

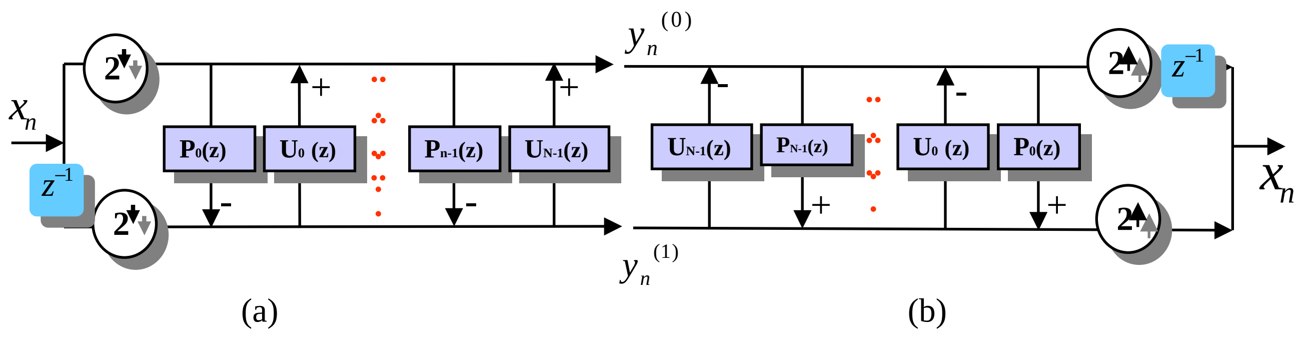

Figure 1. Lifting structure for a two channel analysis and synthesis filter banks.

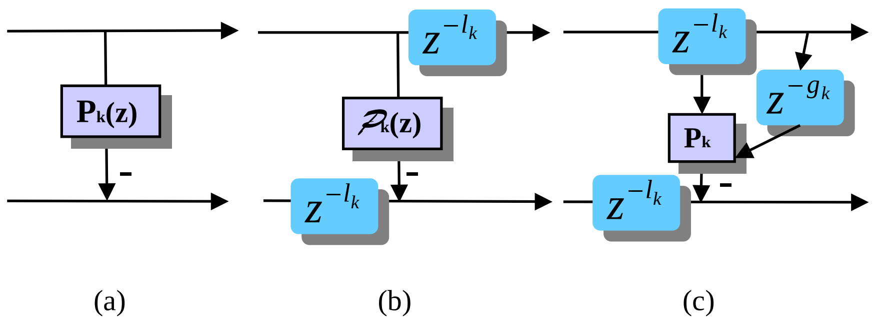

Figure 2. (a) Non causal (b) causal (c) modified causal lifting steps.

the systems in Figure 1 need to be modified with a few extra delay elements or memories having to be used in the system.

Assume that we have a series of N_p prediction steps and N_u update steps, where each prediction step is followed by an update step. If we have an odd total number of lifting steps the last one will be a prediction step and will not be followed by an update step. We assume that we start the lifting decomposition by first applying a prediction step and then an update step. If we happen to have two or more prediction steps following each other then we can combine them in a single prediction step. So, based on the above observations, without loss of generality we can assume that $N_p = N_u$ or $N_p = N_u + 1$, which are the only two possible situations for any two channel filter bank. That is, the number of prediction steps is either equal to the number of update steps or one more than the number of update steps.

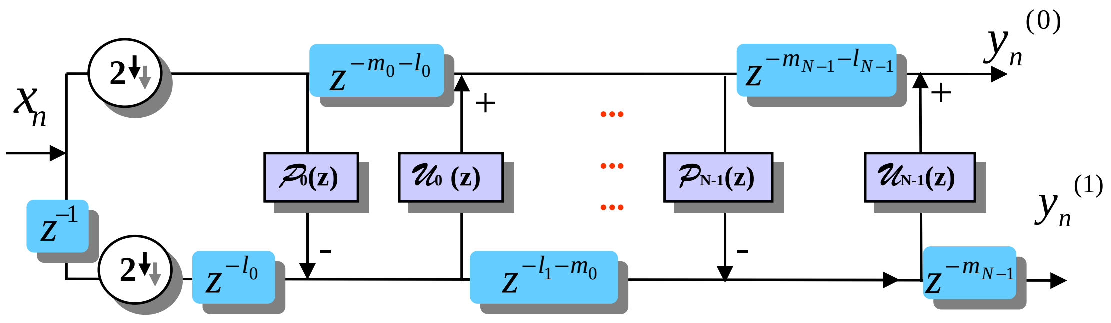

Figure 3. Analysis lifting structure with causal lifting steps $\mathcal{P}(z)$,$\mathcal{U}(z)$, the synthesis structure can be derived in a similar manner.

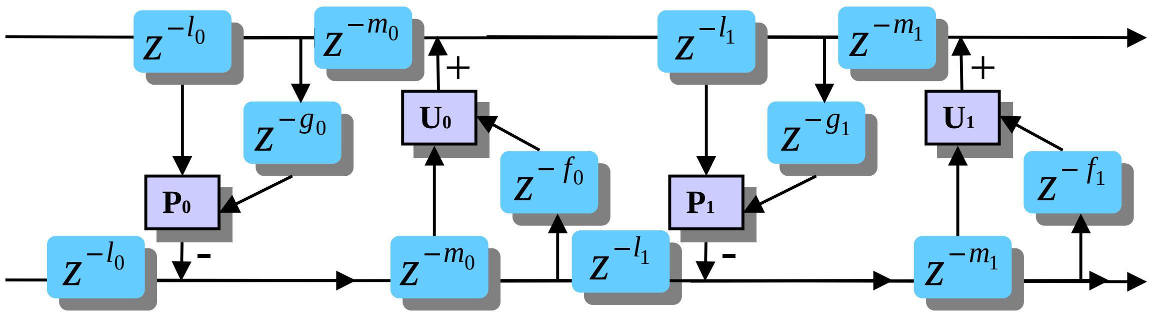

Figure 4. Modified lifting representation analysis filter bank, with filtering dis-jointed from delay units. Blocks $[P_i]$ do not include any delay, instead they reads multiple input data from the delay lines z^{-l_0} and z^{-g_0}.

Let the k^{th} prediction $P_k(z)$ and update steps $U_k(z)$ steps be:

$$P_k(z) = p_{-l_k} z^{l_k} + p_{-l_k+1} z^{l_k-1} + \cdots + p_0 z^0 + \cdots + p_{g_k-1} z^{-g_k+1} + p_{g_k} z^{-g_k} \tag{1}$$

and

$$U_k(z) = u_{-m_k} z^{m_k} + u_{-m_k+1} z^{m_k-1} + \cdots + u_0 z^0 + \cdots + u_{f_k-1} z^{-f_k+1} + u_{f_k} z^{-f_k} \tag{2}$$

Let $\mathcal{P}(z)$ be a causal version of $P(z)$ obtained by shifting by z^{-l_k}, and $\mathcal{U}(z)$ be a causal version of $U(z)$ obtained by shifting by z^{-m_k}:

$$\mathcal{P}_k(z) = z^{-l_k} P_k(z) = p_{-l_k} + p_{-l_k+1} z^1 \cdots + p_{g_k-1-l_k} z^{-g_k+1-l_k} + p_{g_k-l_k} z^{-g_k-l_k} \tag{3}$$

$$\mathcal{U}_k(z) = z^{-m_k} U_k(z) = u_{-m_k} + u_{-m_k+1} z^1 \cdots + u_{-m_k+f_k-1} z^{-f_k-m_k+1} + u_{f_k-m_k} z^{-f_k-m_k} \tag{4}$$

Let us introduce the operators $\mathcal{C}$ and $\mathcal{A}$ denoting the causal and anti-causal parts of a signal, that is:

$$\mathcal{C}\{\sum_{i=-\infty}^{\infty} a_i z^{-i}\} = \sum_{i=0}^{\infty} a_i z^{-i}, \qquad \mathcal{A}\{\sum_{i=-\infty}^{\infty} a_i z^{-i}\} = \sum_{i=1}^{\infty} a_i z^{i} \tag{5}$$

The orders of the causal and anti-causal parts of $P_k(z)$ and $U_k(z)$ are:

$$\text{order}\{\mathcal{C}\{P_k(z)\}\} = g_0, \qquad \text{order}\{\mathcal{A}\{P_k(z)\}\} = l_0 \tag{6}$$

$$\text{order}\{\mathcal{C}\{U_k(z)\}\} = f_0, \qquad \text{order}\{\mathcal{A}\{U_k(z)\}\} = m_0 \tag{7}$$

The lengths of the predict and update steps $P_k(z)$ and $U_k(z)$ are:

$$\text{Length}\{P_k(z)\} = 1 + \text{order}\{\mathcal{C}\{P_k(z)\}\} + \text{order}\{\mathcal{A}\{P_k(z)\}\} = 1 + g_0 + l_0 \tag{8}$$

$$\text{Length}\{U_k(z)\} = 1 + \text{order}\{\mathcal{C}\{U_k(z)\}\} + \text{order}\{\mathcal{A}\{U_k(z)\}\} = 1 + f_0 + m_0 \tag{9}$$

We can force all filtering steps in Figure 1 to be causal by introducing appropriate delays. In Figures 2(a) and 2(b) we see a single lifting step implementation with a non causal and a causal structure respectively. The complete modified analysis system with the use of causal building blocks is depicted in Figure 3. To understand the transition from a non causal to a causal system let us introduce a delay of z^{-l_k} on both inputs of the system in

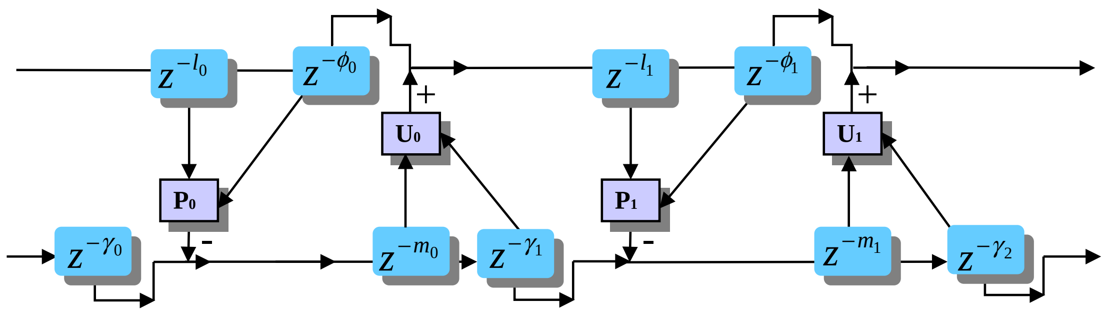

Figure 5. Analysis lifting structure with causal lifting steps and minimum memory requirements.

Figure 2(a), in the upper part of the input in Figure 2(a) we can distribute the delay into both branches. We know that $z^{-l_k}P_k(z) = \mathcal{P}_k(z)$, this leads to Figure 2(b) which is a causal system.

The main problem in forcing the system to be causal is the increased amount of memory or delay needed. The system in Figure 3 requires more delay than the one in Figure 1(a), due to the z^{-l_k} and z^{-m_k} delays that were introduced to produce a causal system. We can combine the memory needed for filtering $P_k(z)$ and the delay line z^{-l_k} in between consecutive lifting steps by using the single modified block in Figure 2(c). Figure 2(c) is the exact same structure as the one in Figure 2(b), the only difference is the use of different notation in representing the filtering operation. We switch to a vector notation in Figure 2(c), to facilitate further simplifications in a complete system. The complete analysis system with the use of the cell* in Figure 2(c) is seen in Figure 4. If we combine adjacent delays from two cells into a single delay element we get Figure 5. The quantities γ_k, ϕ_k are defined as:

$$\gamma_k \triangleq \max\{l_k, f_{k-1}\}, k = 0, 1, 2 \ldots, N-1 \tag{10}$$

$$\phi_k \triangleq \max\{g_k, m_k\}, k = 0, 1, 2 \ldots, N-1 \tag{11}$$

In the above equations we assume that:

$$l_k = g_k = 0, \quad \text{for} \quad k \neq 0, 1, 2 \ldots, N_p - 1, \tag{12}$$

and

$$m_k = f_k = 0, \quad \text{for} \quad k \neq 0, 1, 2 \ldots, N_u - 1 \tag{13}$$

The above two conditions are used to avoid special cases in the definitions (10) and (11). The total memory T_S needed for the lifting system in Figure 5 is:

$$T_S = \sum_{k=0}^{N_p-1} \gamma_k + \sum_{k=0}^{N_u-1} \phi_k + \sum_{k=0}^{N_p-1} l_k + \sum_{k=0}^{N_u-1} m_k \tag{14}$$

The above memory does not include the two samples for input and output. The lifting system described in this section is a two input two output system and as such there is also need to buffer the two input samples. In this case the total memory needed to implement the filter bank is $T_A = T_S + 2$. For the case where $l_k = l, g_k = g, k = 0, 1, 2, \ldots, N_p - 1$, $m_k = m, f_k = f, k = 0, 1, 2, \ldots, N_u - 1$ and $N_u = N_p = N$.

$$T_S = N(l + m + f + g) \tag{15}$$

For example for the $9-7$ tap filters:

$$T_S^{(9-7)} = 1 + 1 + 1 + 1 = 4 \tag{16}$$

*By cell we refer to either a prediction or an update step seen in Figure 2

Filter	l_{low}	l_{high}	T_S	$T_A = T_S + 2$	memory savings
$9-7$	9	7	4	6	33.33%
$13-7$	13	7	6	8	38.46%
$9-3$	9	3	4	6	33.33%
$5-3$	5	3	2	4	20.00%
$13-11$	13	11	8	10	9.09%
$5-11$	5	11	7	9	18.18%
$2-6$	2	6	2	4	33.33%
$2-10$	2	10	4	6	40.00%
$2-2$	2	2	0	2	0.00%

Table 1. Memory savings for different filters according to Appendix A

In Appendix A we give several filter coefficients for convolution and lifting implementations as presented in the JPEG2000 standardization meetings [6]. In all cases the memory needed for the lifting structure implementation is significantly smaller than that needed for the straight forward convolution implementation. The total memory needed for the system is denoted by T_S. To this memory we need to add the memory from the two input samples, since the system is a two input two output operator. However, in many cases, depending on our system design, we may not have to account for this memory as part of the WT system memory. Since the input and output memory locations will be provided by another system, for example the image reader or writer. This is the reason we decided not to include the two additional samples of delay in T_S. Still we do include values of T_A.

In Table 1 we present data for nine different filters, the length of the low pass and the high pass filters l_{low}, l_{high}, the memory T_S and the memory T_A, along with the percentage of memory saving for each filter.

3 LIFTING STEPS, FILTER SYMMETRY AND FAST IMPLEMENTATIONS

For a lifting structure to achieve perfect reconstruction no filter symmetry is needed. Because of the structure, arbitrary prediction/update filters can be used and perfect reconstruction will still be guaranteed. However, imposing symmetry helps in reducing numerical operations; as symmetric filters have approximately half the complexity of the equivalent non-symmetric filters of same length.

Consider a symmetric filter $P(z)$:

$$P(z) = \sum_{j=0}^{N-1} p_j(z^{-j} + z^j) \tag{17}$$

a convolution of a signal x_n with the filter will be implemented as follows:

$$y_n = \sum_{i=-\infty}^{\infty} (x_{n-i} + x_{n+i})p_i \tag{18}$$

when we are performing filtering along the vertical direction in two dimensional signals we need to perform all operations in lines instead of samples. In this case we will need to add two lines and multiply the result with the corresponding filter coefficient. Symmetry is a key point to fast implementations of a wavelet transform in software. Moreover we should scan along the image in the horizontal direction as much as possible and try to avoid vertical scanning. The reason is that image data are stored line by line and therefore by scanning in the same order we reduce the memory accesses to a minimum. The key operation needed is the ability to add two lines together and multiply the result by the appropriate coefficient. Adding two lines and multiplying the result by a single number is a highly regular operation, it is highly predictable and does not cause and disturbances in the pipeline of a processor.

The issues of efficient data accesses for complexity reductions did not arise in the JPEG standard since the number of coefficients in a DCT block is very small. Instead, the main focus was on reducing the number of additions and multiplications. In wavelet transforms it is more beneficial to concentrate on efficient memory accesses than on multiplication reductions. For example the 9-7 Daubechies filter bank requires more multiplications per pixel than the DCT transform. Moreover the data manipulations in both cases follow completely different patterns, and more savings are possible by memory re-arrangements than by reducing the number of additions and multiplications.

In all of the above examples the lifting steps were symmetric and were derived from symmetric filters. For symmetric filter banks it is always possible to design lifting structures with symmetric lifting steps [7]. Moreover even length filters correspond to odd length lifting steps, while odd length filters correspond to even length lifting steps.

THEOREM 3.1. *Let $h(z) = (-1)^{\tau_h} h(z^{-1}) z^{\phi_h}$ and $g(z) = (-1)^{\tau_g} g(z^{-1}) z^{\phi_g}$ be the two low pass and high pass filters of an analysis, symmetric, bi-orthogonal filter bank. Every lifting step of the form $p(z) = (-1)^{\tau_h+\tau_g} p(z^{-1}) z^{\frac{\phi_h-\phi_g}{2}}$ applied to the original pair of filters will lead to a new symmetric, bi-orthogonal filter bank.*

Proof:

$$h(z) = (-1)^{\tau_h} h(z^{-1}) z^{\phi_h} \tag{19}$$

$$g(z) = (-1)^{\tau_g} g(z^{-1}) z^{\phi_g} \tag{20}$$

After application of one additional lifting step $p(z)$ the new high pass filter $h_{new}(z)$ will be:

$$h_{new}(z) = h(z) - p(z^2) g(z) \tag{21}$$

The new high pass filter $h_{new}(z)$ will be symmetric if and only if:

$$h_{new}(z) = (-1)^{\tau} z^{\phi} h_{new}(z^{-1}) \Leftrightarrow \tag{22}$$

$$h(z) - p(z^2) g(z) = (-1)^{\tau} z^{\phi} (h(z^{-1}) - p(z^{-2}) g(z^{-1})) \Leftrightarrow \tag{23}$$

$$h(z) \underbrace{(1 - (-1)^{\tau_h+\tau} z^{-\phi_h+\phi})} - g(z) \underbrace{(p(z^2) - p(z^{-2})(-1)^{\tau_g+\tau} z^{-\phi_g+\phi})} = 0 \tag{24}$$

if we set the two terms multiplying $h(z)$ and $g(z)$ equal to zero we will achieve the desired symmetry properties, therefore:

$$1 - (-1)^{\tau_h+\tau} z^{-\phi_h+\phi} = 0, \quad p(z^2) - p(z^{-2})(-1)^{\tau_g+\tau} z^{-\phi_g+\phi} = 0 \Leftrightarrow \tag{25}$$

$$\phi = \phi_n, \quad \tau + \tau_h = even, \quad p(z) = (-1)^{\tau_g+\tau_h} z^{\frac{-\phi_g+\phi_h}{2}} p(z^{-1}) \tag{26}$$

The above theorem is very important since symmetric systems reduce the implementation complexity of a given filter bank by a factor of two. Theorem 3.1 gives us a way to design a symmetric filter bank based on lifting steps, we can start with a small number of lifting steps and keep increasing the number of lifting steps while at the same time preserving the linear phase properties.

4 BOUNDARY EXTENSIONS AND LIFTING STEPS

When working with lifting steps at the boundaries of a signal and wanting to take advantage of symmetric extensions, we can extend the original signal in the same way we do for the convolution implementation. Even though this will work and will give the exact same result as the convolution implementation it is preferable not to do so. The reason is that in order to perform symmetric extension along the vertical direction we need to buffer a number of lines equal to half the filter length in a buffer outside our system. This increases significantly our memory requirements and also does not allow us to take advantage of the powerful lifting structures.

A much better solution to the problem is to perform symmetric extension, as we go through each individual lifting step. In this way we can utilize the memory allocated for lifting without a need for any additional memory. For example at the end of the image when we reach a boundary we do not have any lines in the input, so we recycle the lines already stored inside the buffer. We just need to know we have reached the end of the image. The same applies to the beginning of the image. In the beginning of the image we read a line and we copy this line to a location in an internal buffer line. When filtering if we need to go beyond the beginning of the image we just reuse the lines we

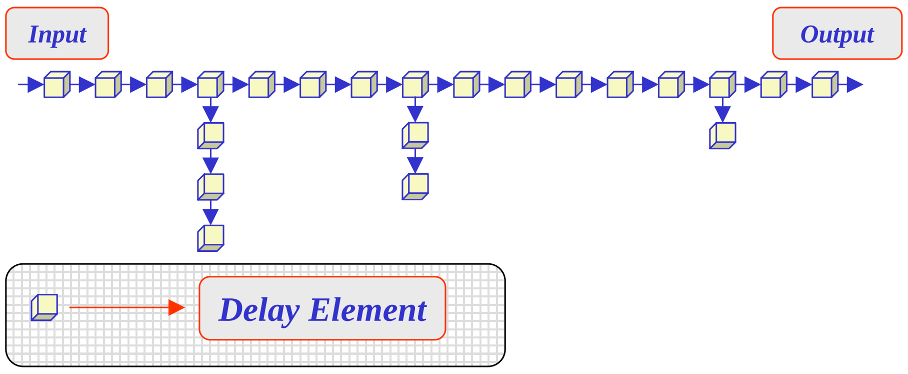

Figure 6. Delay line for one band of an analysis filter bank.

already have in the buffer. This offers a systematic way of dealing with the boundaries since each individual lifting step is responsible for it's own symmetric extension.

A simpler way of looking on this approach is by considering each lifting step as a stand alone system. Consequently, the amount of symmetric extension needed at the boundaries is much smaller than by considering the complete filter bank. Each lifting step performs symmetric extension based on the data it receives from the previous lifting step. Related work has also been independently performed in [8,9] for the purpose of having a parallel implementation of a lifting scheme in a two or more processors, and storing partially updated results.

When working with lifting structures symmetric extension is not the only choice. When working with lifting we can have almost arbitrary boundary conditions without compromising on perfect reconstruction as can be deduced from [2]. We can for example use repetition on each lifting step, so instead of using symmetric extension we can instead just repeat the last sample of our signal. Lifting implementations will always give us perfect reconstruction. The reason we will not elaborate more on this issue is that repetition does not allow us a unified approach in both convolution and lifting implementations of wavelets.

5 SOFTWARE/HARDWARE ARCHITECTURES FOR LIFTING IMPLEMENTATIONS ON IMAGES

The direct lifting algorithm is probably the most efficient approach for filtering in the horizontal direction when working with images. The direct algorithm works in place as described in [2], i.e. we read one line of the original image and we process everything in place. Even symmetric extensions can be handled in place. In the case of symmetric extensions we do not need to extend the input signal instead we can achieve the same effect by using boundary filters. The design of boundary filters is trivial in the case of lifting structures.

In the vertical direction special treatment is needed as mentioned above. The best way to implement the lifting structure is to allocate two delay lines one for the even samples and another for the odd samples. Even samples will become low pass coefficients while odd samples will become high pass coefficients. The length of each delay line can be derived from Figure 5. The length for the even samples will be $\sum_i \phi_i + \sum_i l_i$ elements while the delay for the odd samples will be $\sum \gamma_i + \sum_i m_i$. Those will be the total lengths of the delay lines, but the lines will not be "straight", as it can be seen from Figure 6. Each line might have a number of branches equal to the number of lifting steps originating from this line. An example of the whole system with the delay line and filtering can be seem in Figure 7. From Figure 7 one can see that our proposed approach not only minimizes the total memory needed for filtering but also minimizes the delay between input and output. The approach is not only beneficial for software it might also be extremely useful in hardware, since by using this structure we do not need to have multiple accesses to our data, the algorithm becomes one pass unlike the direct implementation which requires a number of passes equal to the number of lifting steps.

6 ASYMPTOTIC MEMORY SAVINGS

Asymptotically, as the filter length grows the amount of memory savings will tend to 50%, this is the exact same savings that we are getting in terms of numerical computations as explained in [7]. The proof of the amount of

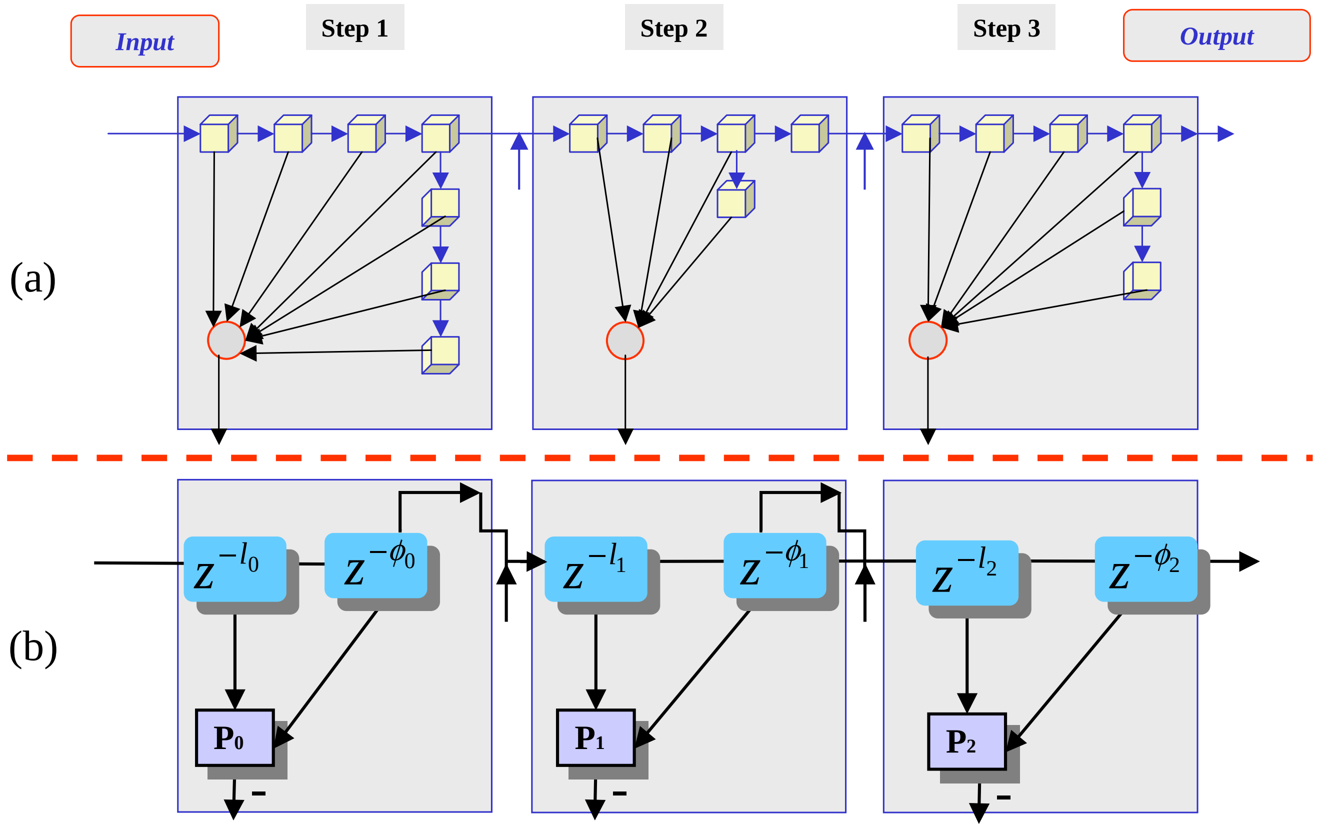

Figure 7. Delay line along with the filtering elements for one band of an analysis filter bank. We have the inner product between the entries in the delay elements and the coefficients of the lifting kernels. The inner product is added or subtracted from the other delay line. Both parts (a) and (b) represent the same system in two different ways. Part (a) is an example of the generic system in part (b). By selecting specific numbers for the parameters of the filter bank we move from (b) to (a).

memory savings is essentially the same as the one presented in [7] for the number of numerical operations.

7 CONCLUSIONS AND FUTURE WORK

In this paper we described how to use a lifting structure as a causal system. We explained why lifting implementations proposed up to now cannot be causal. By forcing a lifting implementation to be causal we achieved a substantial amount of memory savings. Memory savings also translate to speed advantages on a general purpose processor. We gave a procedure to move from a lifting structure of a given length to a lifting structure of larger length, while preserving symmetry. Symmetry is of fundamental importance in reducing computational complexity. We described a process to apply symmetric extension on the bounds of a signal for the case of lifting filtering without the need for additional memory, we achieved this by extending the signal on each individual lifting step instead of the whole original signal. The effect is the same as the extension of the whole signal.

There are a lot of open problems in this area. It will be of great practical interest to investigate design algorithms to derive a symmetric lifting structure from a symmetric filter bank. It would also be of interest to design filter banks directly in the lifting form. An even more detailed description of lifting implementations in causal form, in a step-by-step approach would be of interest along with description of computational tricks. This could serve as a reference for software and hardware implementations.

A Example of Lifting Structures

In this section we present a number of different filter banks along with the amount of memory needs for each one. We apply the analyses of the previous sections to derive the parameters of the proposed implementations. We give the convolution kernel, as well as the coefficients of the lifting factorization. Most of the filters were obtained from [6].

A.1 13-7 Filter Bank

$$\begin{aligned} L_{(13,7)} &= \frac{1}{2^9}\left[\begin{array}{ccccccccccccc} -1 & 0 & 18 & -16 & -63 & 144 & 348 & 144 & -63 & -16 & 18 & 0 & -1 \end{array}\right] \\ H_{(13,7)} &= \frac{1}{2^4}\left[\begin{array}{ccccccc} 1 & 0 & -9 & 16 & -9 & 0 & 1 \end{array}\right] \end{aligned} \tag{27}$$

$$P_0^{(13,7)} = \frac{1}{2^4}\left[\begin{array}{cccc} -1 & 9 & 9 & -1 \end{array}\right], \qquad U_0^{(13,7)} = \frac{1}{2^5}\left[\begin{array}{cccc} -1 & 9 & 9 & -1 \end{array}\right] \tag{28}$$

$$N_u = N_p = 1, \quad l_0 = 2, \quad g_0 = 1, \quad m_0 = 1, \quad f_0 = 2, \quad \gamma_0 = 2, \quad \phi_0 = 1 \tag{29}$$

$$T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 6 \tag{30}$$

$$T_A = T_S + 2 = 8 \tag{31}$$

A.2 9-7 Filter Bank

$$\begin{aligned} L_{(9,7)} &= \frac{1}{2^6}\left[\begin{array}{ccccccccc} 1 & 0 & -8 & 16 & 46 & 16 & -8 & 0 & \end{array}\right] \\ H_{(9,7)} &= \frac{1}{2^4}\left[\begin{array}{ccccccc} 1 & 0 & -9 & 16 & -9 & 0 & 1 \end{array}\right] \end{aligned} \tag{32}$$

$$P_0^{(9,7)} = \frac{1}{2^2}\left[\begin{array}{cccc} -1 & 9 & 9 & -1 \end{array}\right], \qquad U_0^{(9,7)} = \frac{1}{2^2}\left[\begin{array}{cc} 1 & 1 \end{array}\right] \tag{33}$$

$$N_u = N_p = 1, \quad l_0 = 2, \quad g_0 = 1, \quad m_0 = 0, \quad f_0 = 1, \quad \gamma_0 = 1, \quad \phi_0 = 1 \tag{34}$$

$$T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 4 \tag{35}$$

$$T_A = T_S + 2 = 6 \tag{36}$$

A.3 9-3 Filter Bank

$$\begin{aligned} L_{(9,3)} &= \frac{1}{2^7}\left[\begin{array}{ccccccccc} 3 & -6 & -16 & 38 & 90 & 38 & -16 & -6 & 3 \end{array}\right] \\ H_{(9,3)} &= \frac{1}{2}\left[\begin{array}{ccc} -1 & 2 & -1 \end{array}\right] \end{aligned} \tag{37}$$

$$P_0^{(9,3)} = \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right], \qquad U_0^{(9,3)} = \frac{1}{2^6}\left[\begin{array}{cccc} -3 & 19 & 19 & -3 \end{array}\right] \tag{38}$$

$$N_u = N_p = 1, \quad l_0 = 1, \quad g_0 = 0, \quad m_0 = 1, \quad f_0 = 2, \quad \gamma_0 = 1, \quad \phi_0 = 1 \tag{39}$$

$$T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 4 \tag{40}$$

$$T_A = T_S + 2 = 6 \tag{41}$$

A.4 5-3 Filter Bank

$$L_{(5,3)} = \frac{1}{2^3}\left[\begin{array}{ccccc} -1 & 2 & 6 & 2 & -1) \end{array}\right], \qquad H_{(5,3)} = \frac{1}{2}\left[\begin{array}{ccc} -1 & 2 & -1 \end{array}\right] \tag{42}$$

$$P_0^{(5,3)} = \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right], \qquad U_0^{(5,3)} = \frac{1}{2^2}\left[\begin{array}{cc} 1 & 1 \end{array}\right] \tag{43}$$

$$N_u = N_p = 1, \quad l_0 = 1, \quad g_0 = 0, \quad m_0 = 0, \quad f_0 = 1, \quad \gamma_0 = 1, \quad \phi_0 = 0 \tag{44}$$

$$T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 2 \tag{45}$$

$$T_A = T_S + 2 = 4 \tag{46}$$

A.5 13-11 Filter Bank

$$\begin{aligned} L_{(13,11)} &= \frac{1}{2^{10}}\left[\begin{array}{ccccccccccccc} -3 & 0 & 22 & 0 & -125 & 256 & 724 & 256 & -125 & 0 & 22 & 0 & -3 \end{array}\right] \\ H_{(13,11)} &= \frac{1}{2^8}\left[\begin{array}{ccccccccccc} -3 & 0 & 25 & 0 & -150 & 256 & -150 & 0 & 25 & 0 & -3 \end{array}\right] \end{aligned} \tag{47}$$

$$P_0^{(13,11)} = \frac{1}{2^8}\left[\begin{array}{cccccc} 3 & -25 & 150 & 150 & -25 & 3 \end{array}\right], \qquad U_0^{(13,11)} = \frac{1}{2^2}\left[\begin{array}{cc} 1 & 1 \end{array}\right] \tag{48}$$

$$N_u = N_p = 1, \quad l_0 = 3, \quad g_0 = 2, \quad m_0 = 0, \quad f_0 = 1, \quad \gamma_0 = 3, \quad \phi_0 = 2 \tag{49}$$

$$T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 8 \tag{50}$$

$$T_A = T_S + 2 = 10 \tag{51}$$

A.6 5-11 Filter Bank

$$\begin{aligned} L_{(5,11)} &= \frac{1}{2^3}\left[\begin{array}{ccccc} -1 & 2 & 6 & 2 & -1) \end{array}\right] \\ H_{(5,11)} &= \frac{1}{2^7}\left[\begin{array}{ccccccccccc} -1 & 2 & 7 & 0 & -70 & 124 & -70 & 0 & 7 & 2 & -1 \end{array}\right] \end{aligned} \tag{52}$$

$$P_0^{(5,11)} = \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right], \quad U_0^{(5,11)} = \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right], \quad P_1^{(5,11)} = \frac{1}{2^4}\left[\begin{array}{cccc} -1 & 1 & 1 & -1 \end{array}\right] \tag{53}$$

$$\begin{aligned} &N_u = 2, \quad N_p = 1, \quad l_0 = 1, \quad g_0 = 1, \quad m_0 = 0, \quad f_0 = 0, \quad l_1 = 2, \quad g_1 = 1, \\ &\gamma_0 = 1, \quad \phi_0 = 1, \quad \gamma_1 = 2 \end{aligned} \tag{54}$$

$$T_S = l_0 + l_1 + m_0 + \gamma_0 + \phi_0 + \gamma_1 = 7 \tag{55}$$

$$T_A = T_S + 2 = 9 \tag{56}$$

A.7 2-6 Filter Bank

$$L_{(2,6)} = \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right], \qquad H_{(2,6)} = \frac{1}{2^3}\left[\begin{array}{cccccc} 1 & 1 & -8 & 8 & -1 & -1 \end{array}\right] \tag{57}$$

$$P_0^{(2,6)} = \begin{array}{c} 1 \end{array}, \quad U_0^{(2,6)} = \frac{1}{2}, \quad P_1^{(2,6)} = \frac{1}{2^2}\left[\begin{array}{ccc} -1 & 0 & 1 \end{array}\right], \tag{58}$$

$$\begin{aligned} &N_u = 2, \quad N_p = 1, \quad l_0 = 0, \quad g_0 = 0, \quad m_0 = 0, \quad f_0 = 0, \quad l_1 = 1, \quad g_1 = 1, \\ &\gamma_0 = 0, \quad \phi_0 = 0, \quad \gamma_1 = 1 \end{aligned} \tag{59}$$

$$T_S = l_0 + l_1 + m_0 + \gamma_0 + \phi_0 + \gamma_1 = 2 \tag{60}$$

$$T_A = T_S + 2 = 4 \tag{61}$$

A.8 2-10 Filter Bank

$$\begin{aligned} L_{(2,10)} &= \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right] \\ H_{(2,10)} &= \frac{1}{2^7}\left[\begin{array}{cccccccccc} -3 & -3 & 22 & 22 & -128 & 128 & -22 & -22 & 3 & 3 \end{array}\right] \end{aligned} \tag{62}$$

$$P_0^{(2,10)} = \;1\;, \quad U_0^{(2,10)} = \frac{1}{2}, \quad P_1^{(2,10)} = \frac{1}{2^6}\left[\begin{array}{ccccc} -3 & 22 & 0 & -22 & 3 \end{array}\right] \tag{63}$$

$$\begin{aligned} &N_u = 2, \quad N_p = 1, \quad l_0 = 0, \quad g_0 = 0, \quad m_0 = 0, \quad f_0 = 0, \quad l_1 = 2, \quad g_1 = 2, \\ &\gamma_0 = 0, \quad \phi_0 = 0, \quad \gamma_1 = 2 \end{aligned} \tag{64}$$

$$T_S = l_0 + l_1 + m_0 + \gamma_0 + \phi_0 + \gamma_1 = 4 \tag{65}$$

$$T_A = T_S + 2 = 6 \tag{66}$$

A.9 2-2 Filter Bank (Haar)

$$L_{(2,2)} = \frac{1}{2}\left[\begin{array}{cc} 1 & 1 \end{array}\right], \qquad H_{(2,2)} = \left[\begin{array}{cc} -1 & 1 \end{array}\right] \tag{67}$$

$$P_0^{(2,2)} = \;1\;, \qquad U_0^{(2,2)} = \frac{1}{2} \tag{68}$$

$$N_u = N_p = 1, \quad l_0 = 0, \quad g_0 = 0, \quad m_0 = 0, \quad f_0 = 0, \quad \gamma_0 = 0, \quad \phi_0 = 0 \tag{69}$$

$$T_S = l_0 + l_1 + m_0 + \gamma_0 + \phi_0 = 0 \tag{70}$$

$$T_A = T_S + 2 = 2 \tag{71}$$

A.10 9-7 Filter Bank Daubechies (floating point)

This is the popular Daubechies 9-7 filter bank:

$$\begin{aligned} L_{(9,7)} &= [\; 0.026749 \quad -0.016864 \quad -0.078223 \quad 0.266864 \quad 0.602949 \\ &\qquad 0.266864 \quad -0.078223 \quad -0.016864 \quad 0.026749 \;] \\ H_{(9,7)} &= [\; -0.045636 \quad 0.028772 \quad 0.295636 \quad -0.557543 \quad 0.295636 \quad 0.028772 \quad -0.045636 \;] \end{aligned} \tag{72}$$

$$\begin{aligned} P_0^{(9,7)D} &= -1.586134342\left[\begin{array}{cc} 1 & 1 \end{array}\right], \quad U_0^{(9,7)D} = -0.05298011854\left[\begin{array}{cc} 1 & 1 \end{array}\right] \\ P_1^{(9,7)D} &= 0.8819110762\left[\begin{array}{cc} 1 & 1 \end{array}\right], \quad U_1^{(9,7)D} = 0.4435068522\left[\begin{array}{cc} 1 & 1 \end{array}\right] \end{aligned} \tag{73}$$

$$N_u = N_p = 1, \quad l_0 = 2, \quad g_0 = 1, \quad m_0 = 0, \quad f_0 = 1, \quad \gamma_0 = 1, \quad \phi_0 = 1 \tag{74}$$

$$T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 4 \tag{75}$$

$$T_A = T_S + 2 = 6 \tag{76}$$

A.11 13-7 Filter Bank CRF

$$
\begin{aligned}
L_{(13,7)} &= \frac{1}{2^8}\left[\ -1 \quad 0 \quad 14 \quad 16 \quad -31 \quad -80 \quad 164 \quad -80 \quad -31 \quad 16 \quad 14 \quad 0 \quad -1\ \right] \\
H_{(13,7)} &= \frac{1}{2^4}\left[\ -1 \quad 0 \quad 9 \quad 16 \quad 9 \quad 0 \quad -1\ \right]
\end{aligned} \tag{77}
$$

$$
P_0^{(13,7)CRF} = \frac{1}{2^4}\left[\ -1 \quad 9 \quad 9 \quad -1\ \right], \qquad U_0^{(13,7)CRF} = \frac{1}{2^4}\left[\ -1 \quad 5 \quad 5 \quad -1\ \right] \tag{78}
$$

$$
N_u = N_p = 1, \quad l_0 = 2, \quad g_0 = 1, \quad m_0 = 1, \quad f_0 = 2, \quad \gamma_0 = 2, \quad \phi_0 = 1 \tag{79}
$$

$$
T_S = l_0 + m_0 + \gamma_0 + \phi_0 = 6 \tag{80}
$$

$$
T_A = T_S + 2 = 8 \tag{81}
$$

REFERENCES

1. R. Calderbank, I. Daubechies, W. Sweldens, and B. L. Yeo, "Losless Image Compression using Integer to Integer Wavelet Transforms," in *International Conference on Image Processing (ICIP), Vol. I*, pp. 596–599, IEEE Press, 1997.
2. W. Sweldens, "The Lifting Scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.* **3**(2), pp. 186–200, 1996.
3. C. Chrysafis, *Wavelet Image Compression Rate Distortion Optimizations and Complexity Reductions.* PhD thesis, University os Southern California, January 2000.
4. R. L. Claypoole, R. G. Baraniuk, and R. D. Nowak, "Adaptive Wavelet Transforms via Lifting," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, (Seattle, WA), 1998.
5. R. Claypoole, G. Davis, W. Sweldens, and R. Baraniuk, "Nonlinear Wavelet Transforms for Image Coding," in *Proc. of 31th Asilomar Conf. on Signals, Systems and Computers*, 1997.
6. C. Chui, J. Spring, and L. Zhong, "Integer Wavelet Transforms," Tech. Rep. ISO/IEC JTC/SC29/WG1 N769 Document, Geneva, Teralogic Inc., March 1998.
7. I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps"," *J. Fourier Anal. Appl.* **4**(3), pp. 245–267, 1998.
8. W. Jiang and A. Ortega, "Parallel Architecture for the Discrete Wavelet Transform based on the Lifting Factorization," in *in Proc of SPIE in Parallel and Distributed Methods for Image Processing III*, (Denver, CO), July 1999.
9. W. Jiang and A. Ortega, "Efficient Discrete Wavelet Transform Architectures Based on Filterbank Factorizations," in *in Intl. Conf. on Image Processing*, (Kobe, Japan), October 1999.