JPEG2000 Hardware Implementation

Procedures and Issues

ALI M. REZA
U.S. Coast Guard Academy
Department of Engineering, Electrical Engineering Section
15 Mohegan Ave., New London, CT 06320
U.S.A.

Abstract: Hardware implementation of JPEG 2000 compression/decompression standard, applicable to the real-time image sequences, is discussed in this article. The general system-level design along with its details are presented. In this work, for clarity and ease of understanding, implementation of lossless and lossy compressions are treated independently. Depending on the application, usually one implementation is preferred over the other and in general there is no need to include both methods on the same hardware platform. For multi-level decomposition, it is assumed that the LL component of the previous stage, i.e., scale-coefficient, is fed back to the same hardware for further decomposition. Lifting approach along with convolution method and polyphase decomposition for implementation of the forward and inverse discrete wavelet transform (DWT) are discussed.

Key–Words: JPEG 2000, Hardware Implementation, Discrete Wavelet Transform, Lossy and Lossless Compression, Lifting Algorithm

1 Introduction

Hardware implementation of JPEG 2000 compression standard for lossy and lossless compressions depend on several different operations. The original image frame, which is assumed to have three color components, is divided into tiles, usually 64×64, and then a given tile component is delivered into the discrete wavelet transform block (DWT). Depending on the number of stages in the DWT, which corresponds to the number of decomposition levels, the LL component, e.g., scale-coefficient, of the previous level is fed back to the DWT block for the next level decomposition. Quantization block is used only during lossy compression and is by-passed in the lossless compression. Each tile is independently quantized and passed to the code-block formation to produce a fixed size rectangular block for coding. Each code-block is decomposed to its bit-plane and passed through the entropy-coding block to produce the compressed bit-stream.

The reverse operation is carried out to recover the original image from the compressed one. In the reverse process, the quantization block simply behaves as a scaling operation to produce the proper scale for a given code. In this case, the inverse DWT (IDWT) is feeding back its reconstructed image, at a given resolution, in order to reconstruct the image at the upper resolution. The number of feedback operation in this case depends on the decomposition level used in the compressed image.

Coding in the JPEG 2000 standard is based on the adaptive arithmetic coding [1]. Encoding is implemented during the compression stage and its inverse operation, referred to as decoding, is implemented during the decompression stage. We have already presented a system level design for implementation of the 2D wavelet transform [2] and the coefficient bit modeling used in the JPEG 2000 standard [3]. Our system level design for the adaptive arithmetic encoding is discussed in [4] and the corresponding discussion for the decoding process is given in [5]. Our main reference in this work is the JPEG 2000 Part I Final Committee Draft Version 1.0 dated 16 March 2000 [6].

There are several real-time video processing hardware implementation that deal with video compression. A programmable and dedicated approach to real-time video processing application is proposed in [7]. This work is not based on JPEG 2000. A hardware architecture dedicated to the macroblock engine of the H.264/AVC video codec standard is presented in [8]. This work is based on H.264 standard and is useful for video processing. A hardware architecture for motion compensated, video frame rate up-conversion application is proposed in [9]. This work is also useful only for video processing. Our implementation is mainly applicable to image sequence processing where correlation between frames are not consid-

ered for compression.

The main procedure, in block diagram form, is shown in Fig. 1. The original image, which may have color components, is divided into tiles, usually 64×64. A given tile is used as input to the discrete wavelet transform block (DWT). Depending on the number of stages in the DWT, which corresponds to the number of decomposition level, the LL component, scale-coefficient, of the previous level is fed back from memory to the DWT block for the next level decomposition.

Quantization block is used only for lossy compression and is bypassed by the lossless compression. Each sub-image of the decomposition is independently quantized and passed on to the code-block formation to produce a fixed-size rectangular block for coding. Each code-block is decomposed to its bit-plane and passed through the entropy-coding block to produce the compressed bit-stream.

The reverse operation is carried out to recover the original image from the compressed one. In the reverse operation the quantization block simply behaves as a scaling operator to produce the proper scale for a given code. In this case, the inverse DWT (IDWT) is recursively feeding back its reconstructed image at a given resolution in order to reconstruct the image at the upper resolution. The number of feedback operation in this case depends on the decomposition level used in the compressed image.

2 Discrete Wavelet Transform (DWT) for Lossless JPEG 2000

In this part, we present the design for implementation of wavelet transform that accommodate JPEG 2000 requirements [10]. The forward discrete wavelet transform is referred to as DWT and its corresponding inverse is denoted by IDWT. Number of stages used for image decomposition is not fixed and is used as a parameter. The wavelet type discussed in this discussion is the one that maps integers to integers with 5/3 biorthogonal filters given in (1).

$$\begin{cases} h_0 = \left\{-\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8}\right\} \\ h_1 = \left\{-\frac{1}{2}, 1, -\frac{1}{2}\right\} \\ g_0 = \left\{\frac{1}{2}, 1, \frac{1}{2}\right\} \\ g_1 = \left\{-\frac{1}{8}, -\frac{1}{4}, \frac{3}{4}, -\frac{1}{4}, -\frac{1}{8}\right\} \end{cases} \tag{1}$$

In this case, based on the lifting algorithm: [11][12][13][14], the filters for forward transform can be implemented by using equations in (2).

$$\begin{cases} y\left(2n+1\right) & = x\left(2n+1\right) - \left\lfloor \frac{x(2n)+x(2n+2)}{2} \right\rfloor; \\ & \text{for } \left\lceil \frac{i_0}{2} \right\rceil - 1 \leq n < \left\lceil \frac{i_1}{2} \right\rceil \\ y\left(2n\right) & = x\left(2n\right) + \left\lfloor \frac{y(2n-1)+y(2n+1)+2}{4} \right\rfloor; \\ & \text{for } \left\lceil \frac{i_0}{2} \right\rceil \leq n < \left\lceil \frac{i_1}{2} \right\rceil \end{cases} \tag{2}$$

In this case, $x(n)$ is the input to the single stage DWT and $y_H(m)$ and $y_L(M)$ are respectively the high and low components of the DWT decomposition. Note that the rate of the input is twice that of the outputs. Similar relations can be derived for the inverse DWT as shown in (3).

$$\begin{cases} y\left(2n+1\right) & = x\left(2n+1\right) - \left\lfloor \frac{x(2n)+x(2n+2)}{2} \right\rfloor; \\ & \text{for } \left\lceil \frac{i_0}{2} \right\rceil - 1 \leq n < \left\lceil \frac{i_1}{2} \right\rceil \\ y\left(2n\right) & = x\left(2n\right) + \left\lfloor \frac{y(2n-1)+y(2n+1)+2}{4} \right\rfloor; \\ & \text{for } \left\lceil \frac{i_0}{2} \right\rceil \leq n < \left\lceil \frac{i_1}{2} \right\rceil \end{cases} \tag{3}$$

In this case, the higher rate signal $x(n)$ is made of its even and odd samples obtained in (3). Note that $\lfloor \cdot \rfloor$ denotes floor operation, whose output is the largest integer that is smaller than or equal to the real number that it operates on. In the case of 2's complement binary numbers, floor operation corresponds to the truncation of the least significant bits. In these equations, i_0 is the index for the first sample and $i_1 - 1$ is the index for the last sample of the input $x(\cdot)$. Also in terms of Lowpass and Highpass components we have the following relationships:

$$\begin{cases} y_H\left(m\right) & = y\left(2n+1\right) \\ y_L\left(m\right) & = y\left(2n\right) \end{cases} \tag{4}$$

The signal is extended in both directions using mirror imaging in both forward and inverse transforms as needed. This is better explained by the signal flow graphs shown in Fig. 2.

Block diagram of this realization, based on storage of intermediate results for both forward and inverse transforms, is shown in Fig. 3. This implementation is sequential. Another method for this operation is based on parallel realization in which intermediate results are recalculated as needed. The parallel version of this realization is shown in Fig. 4.

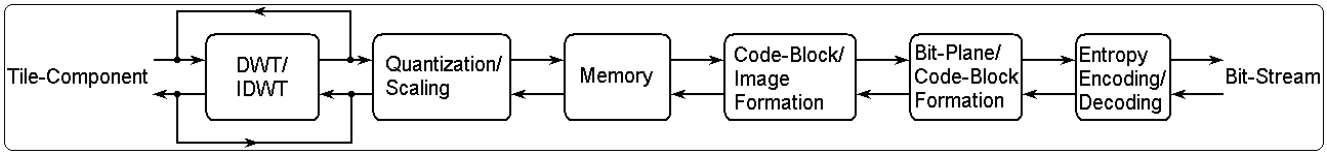

Figure 1: Top-level block diagram for JPEG2000 operation

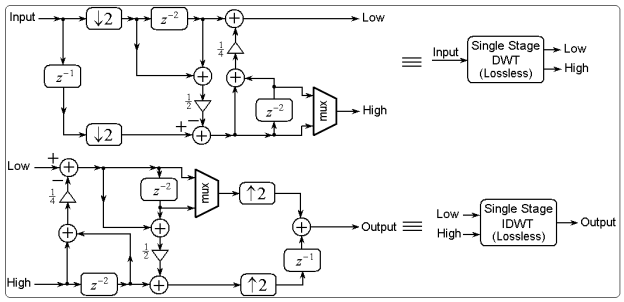

Figure 3: Implementation of (5/3) integer-to-integer wavelet transform based on sequential lifting algorithm.

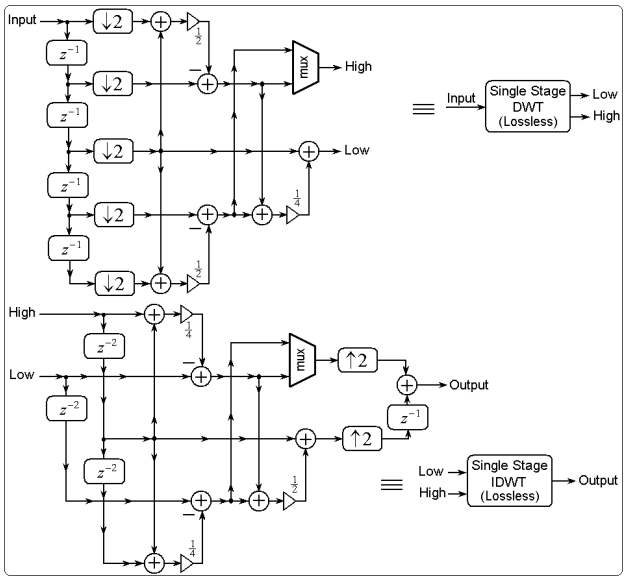

Figure 4: Realization of (5/3) integer-to-integer wavelet transform based on parallel lifting algorithm. In the inverse operation, the output should be settled in a way that samples are properly ordered. That may require repositioning of the unit delay block to the upper path.

3 Hardware Implementation of the Lossless Forward DWT

Hardware implementation that we propose in this part is based on the parallel lifting scheme presented in Fig. 4. This realization is more suitable for hardware implementation mainly due to its timesharing for multiple decomposition level. Since the lossless implementation should properly correspond to its software realization counterpart, we have no choice other than implementing the column processing before the row processing.

When row processing is carried out first and the resultant compressed image is reconstructed by standard software, the result is not perfectly lossless and there will be some rounding error (only on the least significant bit). To avoid this situation and other similar errors, we have decided to use the same implementation approach as carried out in the standard software.

In this case, we assume that the input image is read or arrives in a raster form, one pixel at a time and row by row. The first operation is to properly collect the input data for the first processing unit. This is done in the FDWT Input Interface block shown in Fig. 5.

The size of each buffer is exactly equal to the row size of the corresponding image. The set of five row buffers are considered as a macro block in which the vertical mirror imaging is also incorporated. This part is carried out at the top and the bottom of the input sub-image. The timing and control of this block is discussed in the section related to control signals. This implementation includes the vertical mirror imaging of the input as needed.

Lifting algorithm used in lossless implementation results in a hardware realization with no multipliers. In this case, the required division by 2 and 4 can be realized by binary shift to the right. With reference to Fig. 4, this implementation requires a processing core shown in Fig. 6.

This core, which can be used for all decomposition levels, is always working at the input rate as long as data are coming through it. The way that this core serves all decomposition levels is discussed in the later sections. It should be noted that the same core is used for column processing as is discussed later.

In the next stage, the pixels from rows of the two

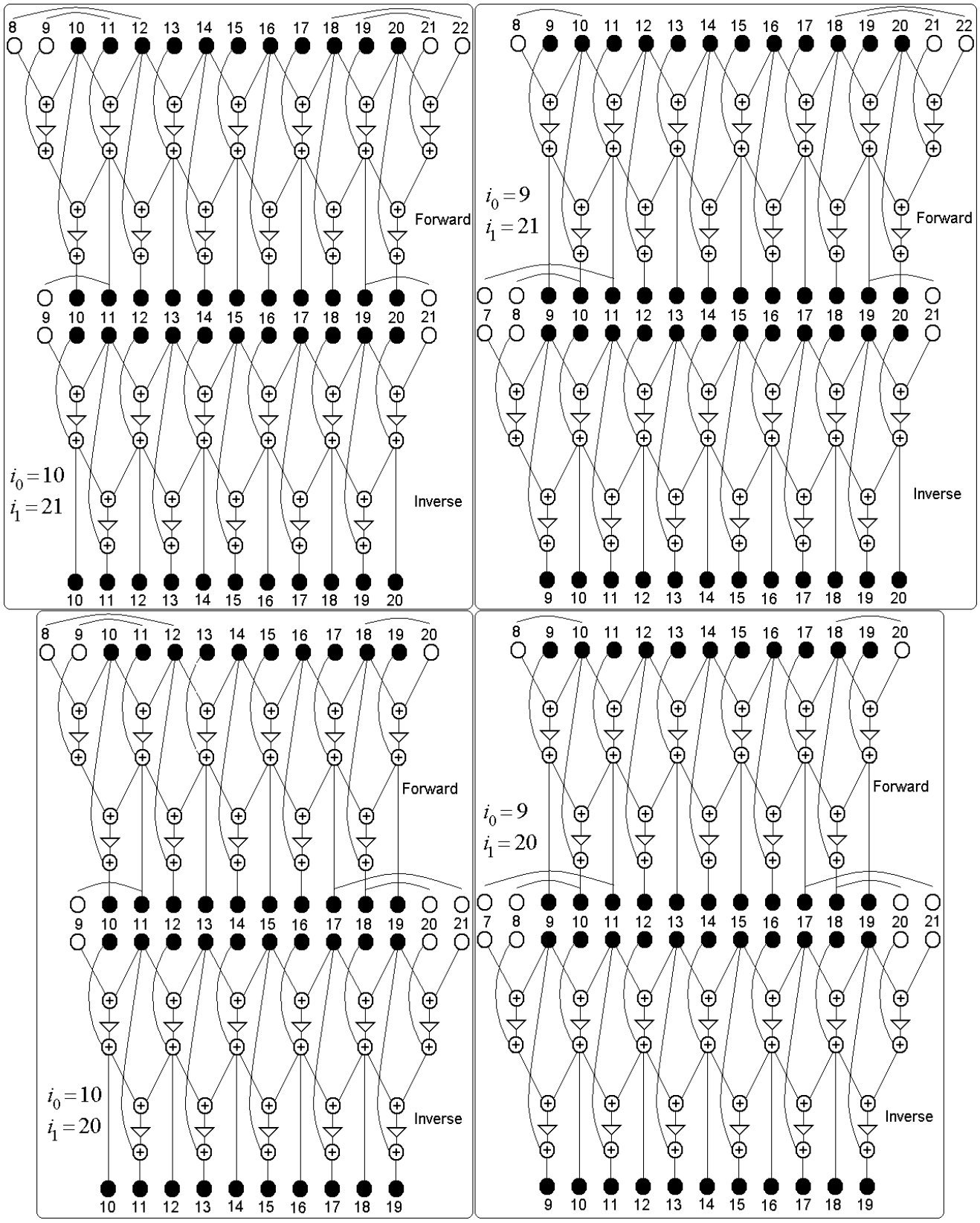

Figure 2: Signal flow graph for four different possible cases shown for both forward and inverse DWT transforms based on lifting implementation of $5/3$ biorthogonal filters. In the forward transform all multipliers in the first set are $1/2$ and in the second set are $1/4$. The reverse is true in the inverse transform.

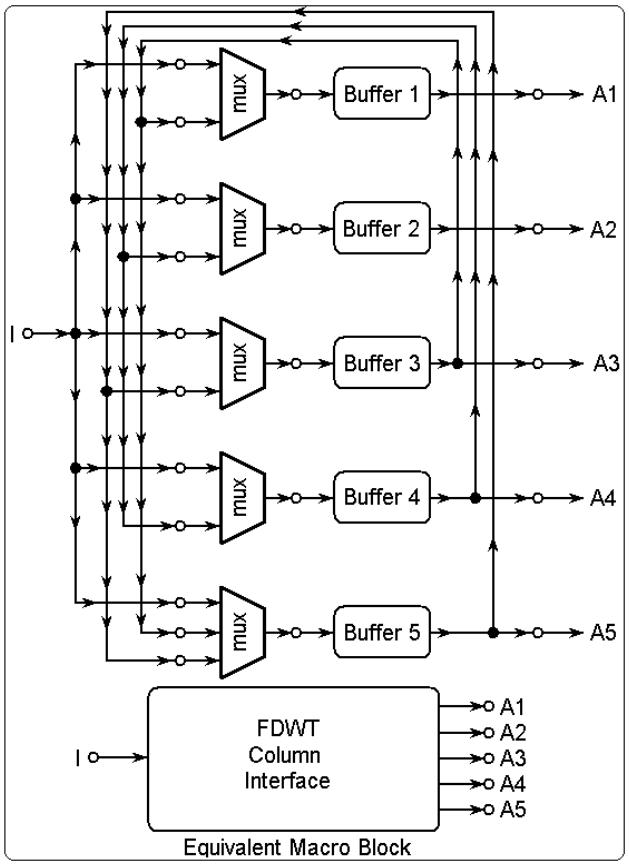

Figure 5: Column Interface block for FDWT with mirror imaging capabilities.

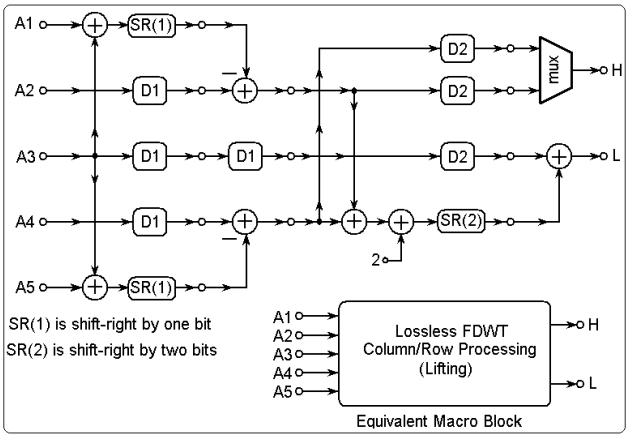

Figure 6: Processing core for the forward DWT based on parallel lifting shown in Fig. 4

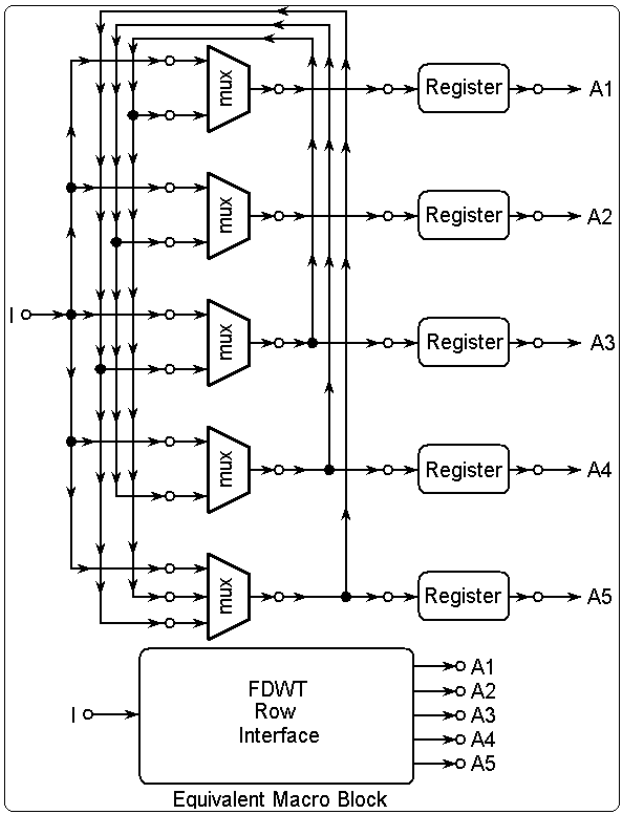

Figure 7: Macro block for Row Interface. This implementation includes the horizontal mirror imaging of the input as needed.

sub-images are independently collected, in a set of five registers for each sub-image, in order to enable the row processing along with the corresponding mirror imaging. This process is carried out in the row interface represented in Fig. 7. The mirror imaging and data flow are controlled by the addressing of MUXs and enable signals of MUXs and Registers.

For each decomposition level, two such interfaces are needed, one for the Low sub-image and another for the High sub-image. Two similar processing units, as shown in Fig. 6, are used to carry out the row processing independently for the two sub-images.

In order to combine these macro blocks for multilevel decomposition, there is a need for proper routing of the inputs and outputs. These operations need input and output interface macro blocks as well as two Row/Column Interface macro blocks. The Interface macro blocks are shown in Fig.8.

In the case of the Input Interface, the rate of incoming data to the Column Interface blocks and the corresponding MUX depends on the row size of the corresponding image. Similarly, in the Column/Row Interface, the data rate to the Row Interface blocks and the corresponding MUX depends on the row size of

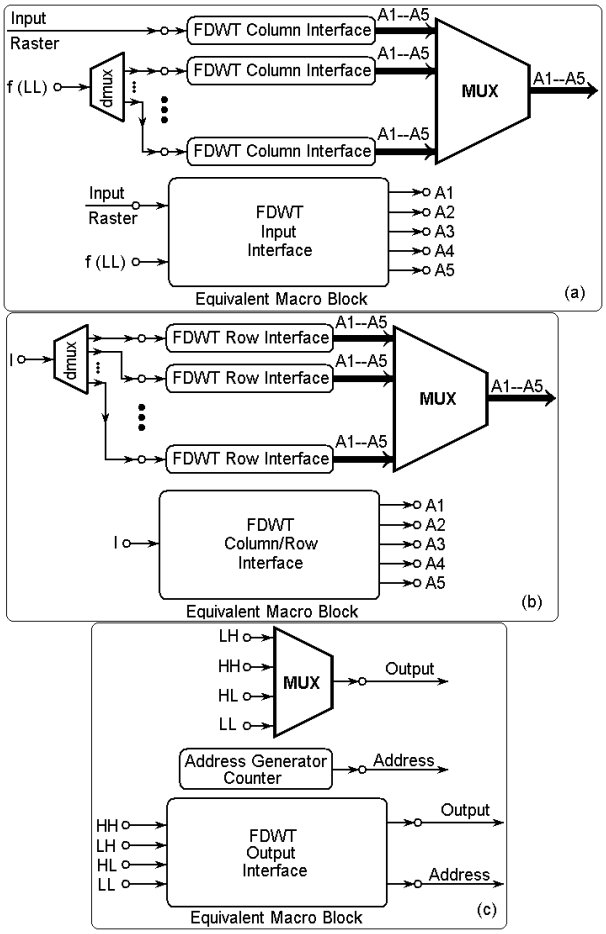

Figure 8: (a) Input Interface, (b) Row/Column Interface, and (c) output interface macro blocks used for multi-level forward DWT decomposition. FDWT Row and Column interface blocks are defined as shown in Fig. 5 and Fig. 7.

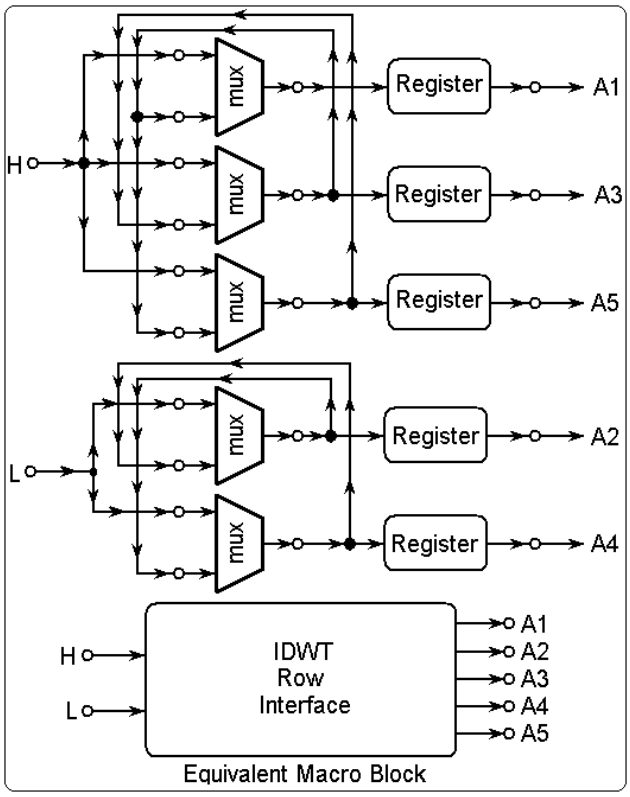

Figure 10: Initial interface before row processing in the inverse DWT.

each corresponding sub-image. Addressing and control signals for these interface macro blocks are discussed in the section corresponding to the signal flow and control signals.

The overall forward DWT implementation, in terms of the defined macro blocks is shown in Fig. 9. In this implementation, the size of the memory used as buffers in all Column Interface blocks is about 10 times the row size of the input image for any number of decompositions. Proper memory design may result in more optimization of the Column Interface by eliminating the need for the MUX in that macro block.

4 Hardware Implementation of the Lossless Inverse DWT

The inverse operation starts in reverse order of the forward transform. The procedure begins with two sets of row processing by initially going through the row interface for mirror imaging as well as proper data flow alignment. This interface is shown in Fig. 10.

After completion of the row processing, the column processing starts. The first block in this stage is the Column Interface block. The structure of row

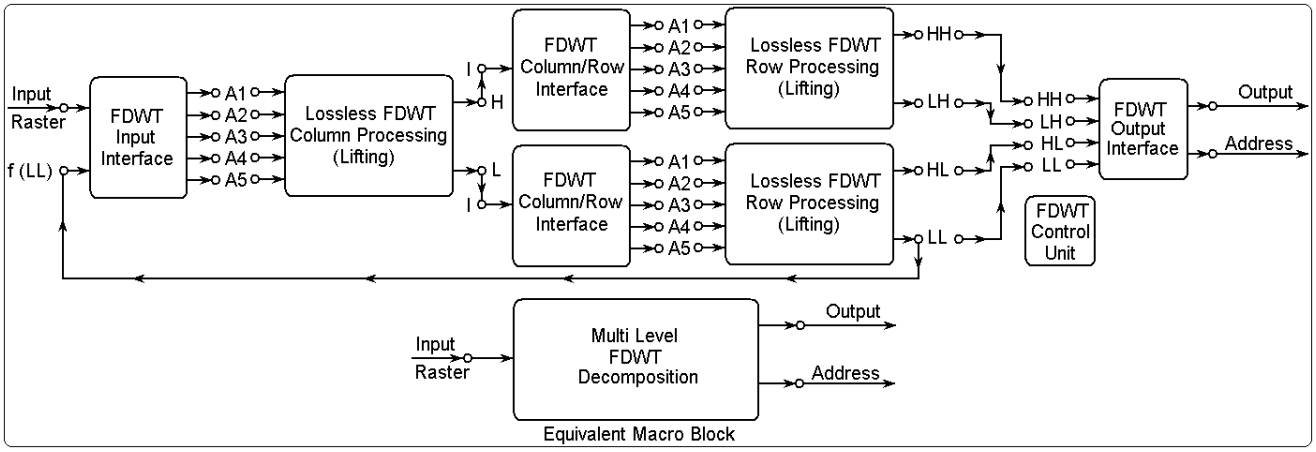

Figure 9: Overall block diagram of the forward DWT based on realization of the parallel lifting algorithm (integer-to-integer 5/3 biorthogonal wavelet transform).

buffers in the Column Interface is a bit different from that of the forward DWT. This structure for a given composition level is shown in Fig. 11.

Similar to the forward transform, the total memory used to handle all decomposition levels is about 10 times the row size of the original image. The vertical mirror imaging is carried out at the top and the bottom of the input sub-images. The timing and control of this block is discussed in the section related to control signals.

The core for the lossless column or row processing for the inverse transform is properly designed to carry out the inverse operation as shown in Fig. 4. This core is working at full rate to service all decomposition levels. The way that data are provided and aligned for processing by this core is controlled by the corresponding interfaces that are designed for column processing and row processing. The structure of this implementation is shown in Fig. 12.

In the inverse transform, the input interface is a little bit more involved. In this case, there are two sets of DWT coefficients that are almost simultaneously synthesized when doing the row processing. The result of this processing is synthesized to generate the higher resolution image. This input interface is shown in Fig. 13.

In this case, the top sub-images consist of low and high DWT coefficients and need to be synthesized in the row direction. It is also assumed that the input sequence is loaded in such a way that for any low coefficient the corresponding high coefficient comes into the interface and properly sent to its corresponding register. Therefore the DWT coefficients are not read in a raster from Memory but they are read in a proper ordering needed for their synthesis.

The interface between row processing and col-

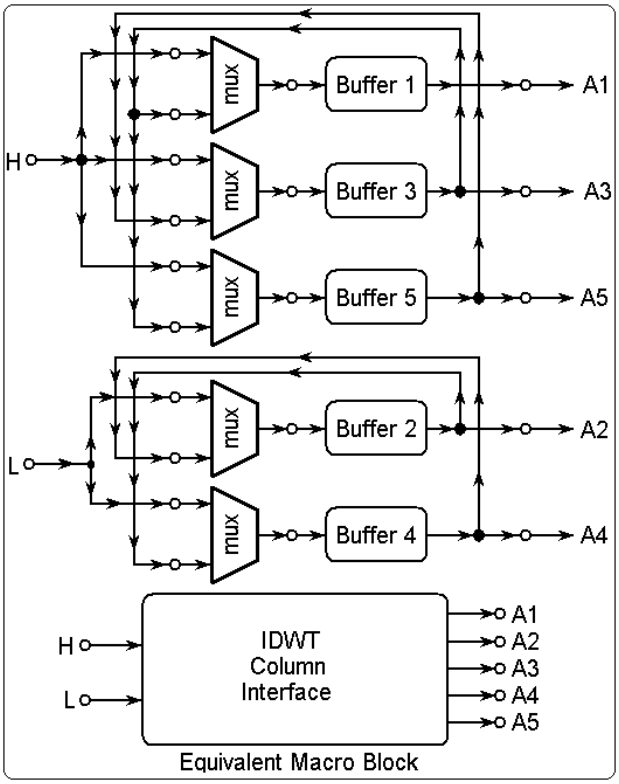

Figure 11: Column Interface macro block used for inverse DWT. In this case, two row buffers are assigned to the Lowpass component of the image and three row buffers are assigned to the corresponding Highpass component the image.

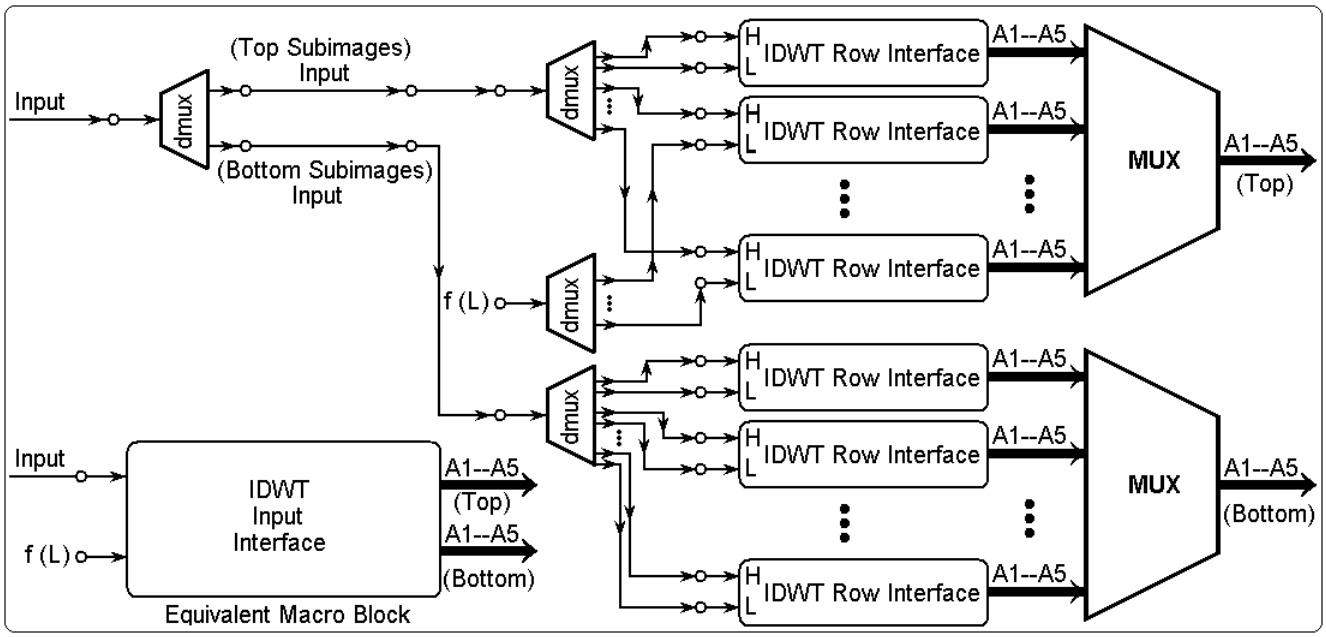

Figure 13: Input interface macro block for inverse DWT. The input properly loads two different sets of IDWT row interfaces. The number of row interfaces in each segment is the same as the decomposition level.

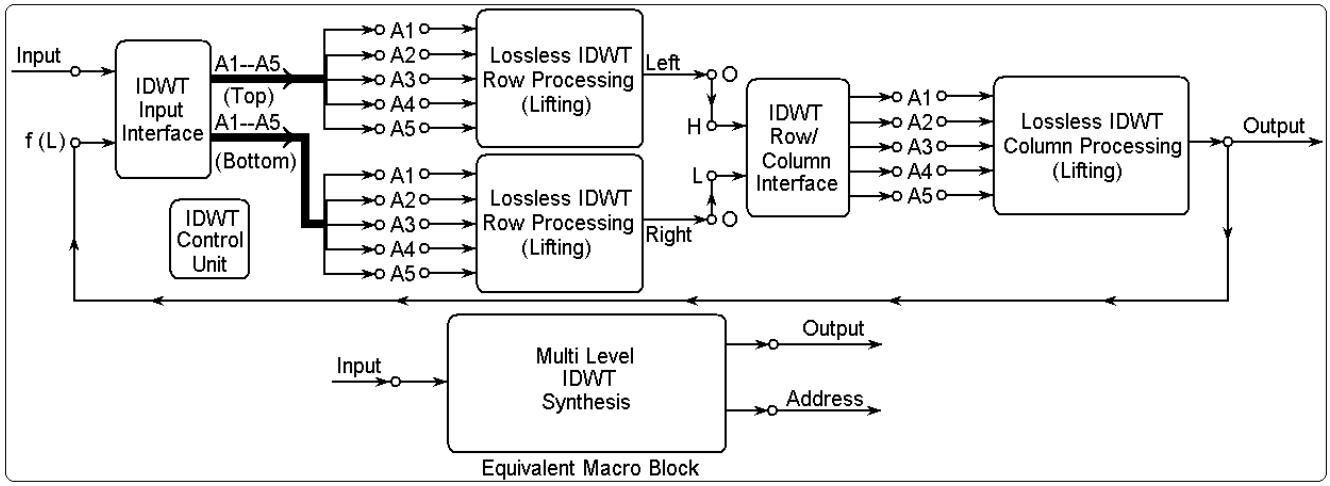

Figure 15: Overall block diagram of the inverse DWT based on realization of the parallel lifting algorithm (integer-to-integer 5/3 biorthogonal wavelet transform).

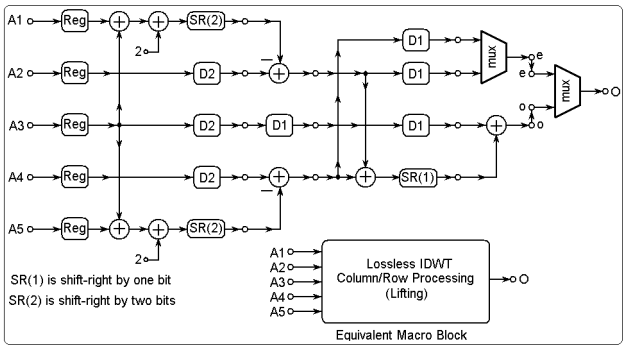

Figure 12: Macro block for column or row processing in the case of the inverse DWT. In this case the final output sequence is prepared by the final switch to properly order the odd and even samples.

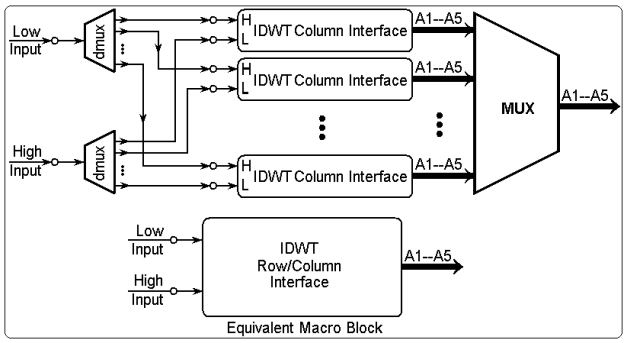

Figure 14: Inverse DWT Row/Column Interface consist of a set of IDWT Column Interfaces. The number of Column Interfaces is equal to the decomposition level.

umn processing consists of a set of Column Interfaces. This interface is referred to as Row/Column Interface and is shown in Fig. 14. The overall inverse transform in terms of defined macro blocks is shown in Fig. 15.

5 Discrete Wavelet Transform (DWT) for Lossy JPEG 2000

In this part, we present system level design for implementation of lossy wavelet transform that accommodate JPEG 2000 requirements. As before, number of stages used for image decomposition is not fixed and is dealt with as a parameter. The wavelet type is restricted to only one type, which is 9/7 biorthogonal wavelet used as standard for lossy DWT in JPEG 2000. In this work we present and use two different formulations for the discrete wavelet transform. One formulation is based on the conventional FIR implementation or convolution and the other one is based on the lifting algorithm. The conventional approach is

Table 1: Standard filter coefficients for (9/7) biorthogonal DWT.

h_0^c	h_1^c	g_0^c	g_1^c
0.0378284555	-0.0645388826	-0.0645388826	-0.0378284555
-0.023849465	0.0406894176	-0.0406894176	-0.023849465
-0.1106244044	0.418092273	0.418092273	0.1106244044
0.3774028556	-0.7884856164	0.7884856164	0.3774028556
0.852698679	0.418092273	0.418092273	-0.852698679
0.3774028556	0.0406894176	-0.0406894176	0.3774028556
-0.1106244044	-0.0645388826	-0.0645388826	0.1106244044
-0.023849465			-0.023849465
0.0378284555			-0.0378284555

based on using filter coefficients provided in Table 1. The lifting implementation is based on equations provided in (5).

$$
\begin{cases}
\text{Step1}: y_{2n+1} & \leftarrow x_{2n+1} + \alpha\left(x_{2n} + x_{2n+2}\right); \\
& \text{for } \left\lceil \frac{i_0}{2} \right\rceil - 2 \le n < \left\lceil \frac{i_1}{2} \right\rceil + 1 \\
\text{Step2}: y_{2n} & \leftarrow x_{2n} + \beta\left(y_{2n-1} + y_{2n+1}\right); \\
& \text{for } \left\lceil \frac{i_0}{2} \right\rceil - 1 \le n < \left\lceil \frac{i_1}{2} \right\rceil + 1 \\
\text{Step3}: y_{2n+1} & \leftarrow y_{2n+1} + \gamma\left(y_{2n} + y_{2n+2}\right); \\
& \text{for } \left\lceil \frac{i_0}{2} \right\rceil - 1 \le n < \left\lceil \frac{i_1}{2} \right\rceil \\
\text{Step4}: y_{2n} & \leftarrow y_{2n} + \delta\left(y_{2n-1} + y_{2n+1}\right); \\
& \text{for } \left\lceil \frac{i_0}{2} \right\rceil \le n < \left\lceil \frac{i_1}{2} \right\rceil \\
\text{Step5}: y_{2n+1} & \leftarrow -K y_{2n+1}; \\
& \text{for } \left\lceil \frac{i_0}{2} \right\rceil - 1 \le n < \left\lceil \frac{i_1}{2} \right\rceil \\
\text{Step6}: y_{2n} & \leftarrow (1/K)\, y\,(2n); \\
& \text{for } \left\lceil \frac{i_0}{2} \right\rceil \le n < \left\lceil \frac{i_1}{2} \right\rceil
\end{cases}
\tag{5}
$$

Approximate values of the parameters used in (5) are as follows:

$$
\begin{cases}
\alpha = -1.586\ 134\ 342\ 059\ 924 \\
\beta = -0.052\ 980\ 118\ 572\ 961 \\
\gamma = 0.882\ 911\ 075\ 530\ 934 \\
\delta = 0.443\ 506\ 852\ 043\ 971 \\
K = 1.230\ 174\ 104\ 914\ 001
\end{cases}
\tag{6}
$$

Index i_0 is used to represent the first sample and index $i_1 - 1$ denotes the last sample of the input x_n.

If there is a need to implement DWT filters in a conventional form (using convolution or FIR tap-delays), the filter coefficients are modified to corre-

Table 2: Filter coefficients for (9/7) biorthogonal DWT used in JPEG2000.

h_0	h_1	g_0	g_1
0.02674875741	0.09127176311	-0.09127176311	0.02674875741
-0.01686411844	-0.05754352622	-0.05754352622	0.01686411844
-0.07822326652	-0.59127176311	0.59127176311	-0.07822326652
0.26686411844	1.11508705245	1.11508705245	-0.26686411844
0.60294901823	-0.59127176311	0.59127176311	0.60294901823
0.26686411844	-0.05754352622	-0.05754352622	-0.26686411844
-0.07822326652	0.09127176311	-0.09127176311	-0.07822326652
-0.01686411844			0.01686411844
0.02674875741			0.02674875741

spond exactly to what the lifting approach will provide. This conversion is represented in (7) and the resultant coefficients are given in Table 2.

$$\begin{cases} h_0 = h_0^c \big/ \sqrt{2} \\ h_1 = -h_1^c \cdot \sqrt{2} \\ g_0 = g_0^c \cdot \sqrt{2} \\ g_1 = -g_1^c \big/ \sqrt{2} \end{cases} \tag{7}$$

In this case, x_n is the input to the single stage DWT. The high and low components of the DWT decomposition, represented by $y_H(m)$ and $y_L(m)$ respectively, are given by

$$\begin{cases} y_H(m) &= y_{2n+1} \\ y_L(m) &= y_{2n} \end{cases} \tag{8}$$

Note that the rate of the input is twice that of the outputs. Similar relations can be derived for the inverse DWT as shown in (9). In this case the higher rate signal x_n is made of its even and odd samples obtained in (9).

$$\begin{cases} \text{Step1}: x_{2n} & \leftarrow K y_{2n}; \\ & \text{for } \left\lfloor \frac{i_0}{2} \right\rfloor - 1 \le n < \left\lfloor \frac{i_1}{2} \right\rfloor + 2 \\ \text{Step2}: x_{2n+1} & \leftarrow -(1/K)\, y_{2n+1}; \\ & \text{for } \left\lfloor \frac{i_0}{2} \right\rfloor - 1 \le n < \left\lfloor \frac{i_1}{2} \right\rfloor + 1 \\ \text{Step3}: x_{2n} & \leftarrow x_{2n} - \delta\,(x_{2n-1} + x_{2n+1}); \\ & \text{for } \left\lfloor \frac{i_0}{2} \right\rfloor - 1 \le n < \left\lfloor \frac{i_1}{2} \right\rfloor + 2 \\ \text{Step4}: x_{2n+1} & \leftarrow x_{2n+1} - \gamma\,(x_{2n} + x_{2n+2}); \\ & \text{for } \left\lfloor \frac{i_0}{2} \right\rfloor - 1 \le n < \left\lfloor \frac{i_1}{2} \right\rfloor + 1 \\ \text{Step5}: x_{2n} & \leftarrow x_{2n} - \beta\,(x_{2n-1} + x_{2n+1}); \\ & \text{for } \left\lfloor \frac{i_0}{2} \right\rfloor \le n < \left\lfloor \frac{i_1}{2} \right\rfloor + 1 \\ \text{Step6}: x_{2n+1} & \leftarrow x_{2n+1} - \alpha\,(x_{2n} + x_{2n+2}); \\ & \text{for } \left\lfloor \frac{i_0}{2} \right\rfloor \le n < \left\lfloor \frac{i_1}{2} \right\rfloor \end{cases} \tag{9}$$

Block diagrams of the conventional FIR filtering realization for both forward and inverse DWT are shown in Fig. 16. The same operation is realized through lifting algorithm as shown in Fig. 17. This figure depicts the realization for both forward and inverse DWT. In this writing, the conventional FIR realization is referred to as Convolution approach and the implementation based on the lifting algorithm is referred to as lifting approach.

Efficient implementation of the lossy wavelet transforms along with its corresponding macro blocks are discussed in the following sections. First, we present the forward transform and then we discuss the inverse transform. The control part for this implementation is presented at the end.

6 Hardware Implementation of the Lossy Forward DWT

Hardware realization of two-dimensional wavelet transform requires enough memory to store the intermediate results right after row processing and before the column processing. However, if we use proper row buffering scheme, the column processing can be carried out first and without the necessity of holding the intermediate results, the row processing can follow the column processing. Advantage of this approach is that the input row buffer can be designed independently and separately in a different VLSI chip if needed. This is more desirable for low density FPGAs. When using high density FPGAs the same design along with

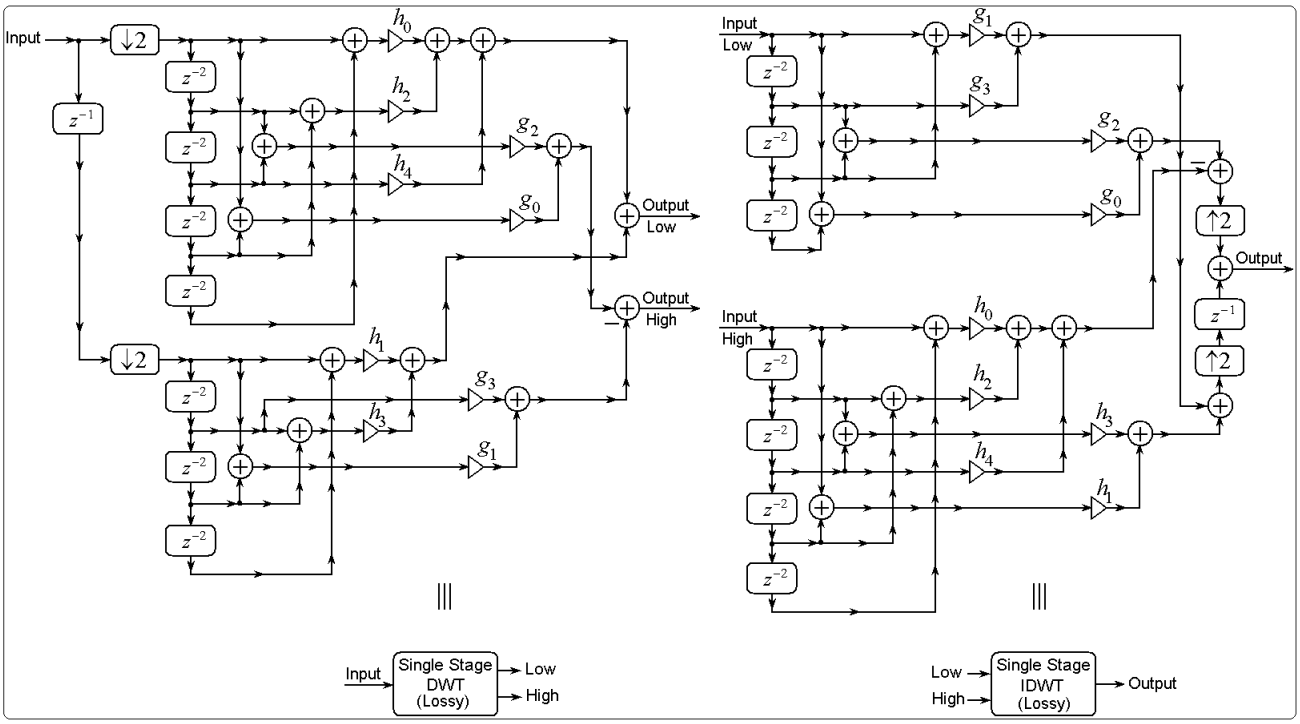

Figure 16: Conventional realization using polyphase implementation.

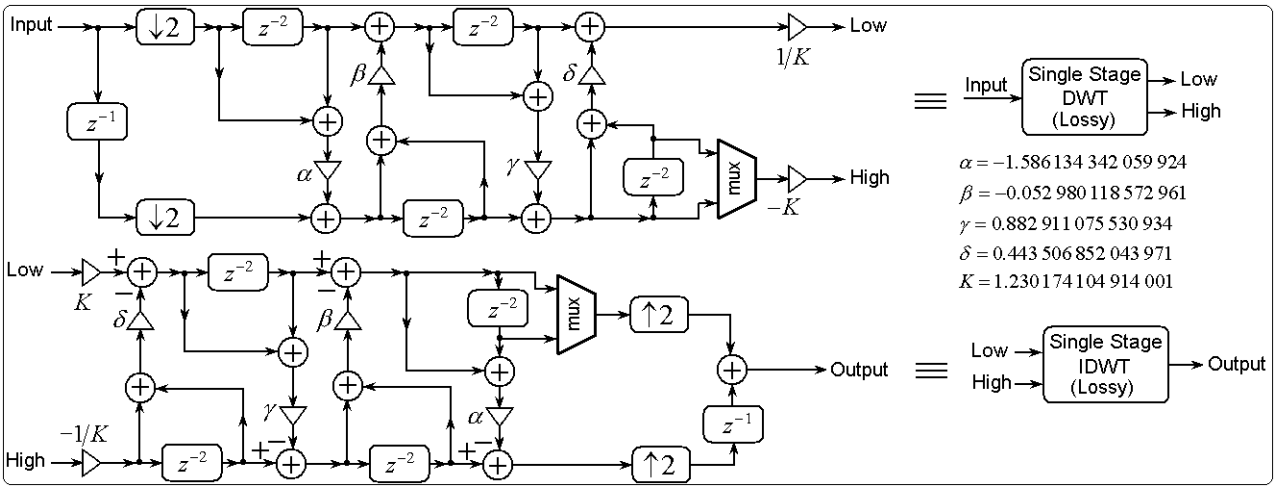

Figure 17: Implementation of (9/7) biorthogonal wavelet transform based on lifting algorithm.

the input buffer can be put into a single chip and there is no need to modify the design that is presented in this work. In terms of the memory requirement, both approaches require same amount of memory.

For efficient implementation of the lossy forward transform, we recommend to apply column transformation first and then transform each row by utilizing input row buffer. In this case the buffer will include the original data and does not encounter bit-growth as drastically as it would if we had to process rows first. In terms of making the mirror image of data in all directions, we have found that the best approach suitable for the column processing is the conventional approach. In this case the number of buffers would be kept at its minimum. Lifting approach can be efficient only if we use mirror imaging outside the chip. In this report we only present the more efficient conventional approach for both column processing as well as row processing.

Hardware implementation based on the conventional approach requires a set of nine input buffers properly configured for convolution filtering. The structure of the input buffer is shown in Fig. 18. In this structure, the mirror imaging is carried out with proper control of the MUX units in the design. The mirror imaging is considered for both top and bottom part of the image input. Depending on the indexes of the first row (top) and the last row (bottom), the mirror imaging is carried out for four or three rows. The detail of this structure and the way the control is organized is discussed in the section on control unit.

The processing core that should come right after the row buffers consist of multipliers and adders as shown in Fig. 19. This core, which can be used for all decomposition levels, is always working at the input rate as long as data are coming through it in real time. The way that this core serves all decomposition levels is discussed in the later section on control.

Row processing can immediately start on the output of the processed columns. The processing core in this case is exactly the same as that of column processing. The interface between the output of the processed columns and the input of the row processing is shown in Fig. 20. In this case, the realization is similar to that of row buffers but instead of each buffer we have a register.

In order to combine these macro blocks for multilevel decomposition, there is a need for proper routing of the inputs and outputs. These operations are carried out by input and output interface macro blocks as shown in Fig. 21.

The overall forward DWT implementation, in terms of the defined macro block is shown in Fig. 22.

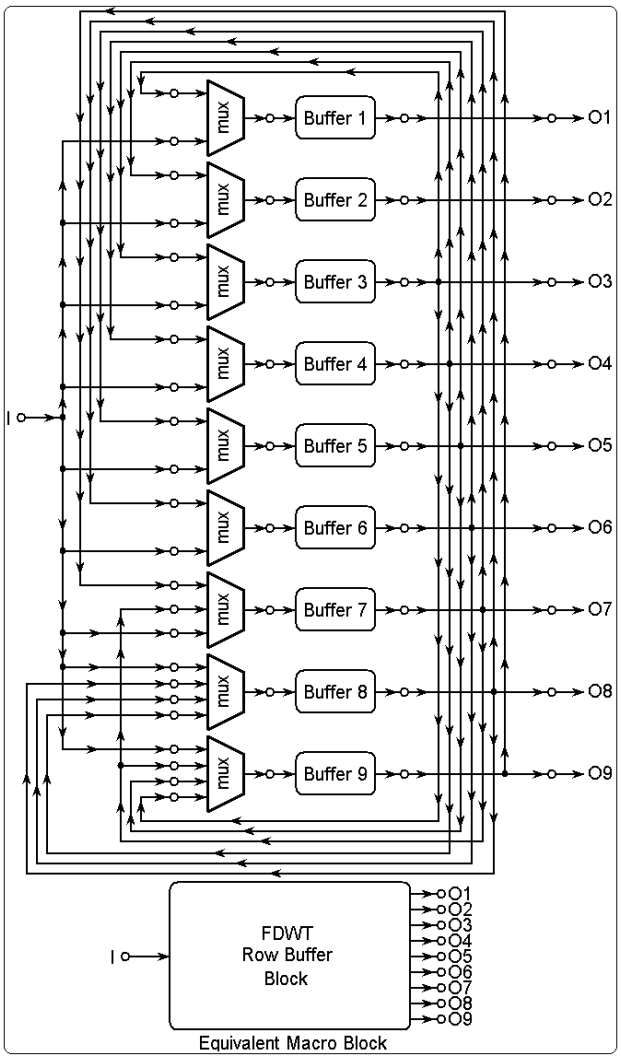

Figure 18: Detailed block diagram of the input buffer for forward DWT. The corresponding equivalent macro block is also shown. With proper control of MUX units, the mirror imaging is carried out.

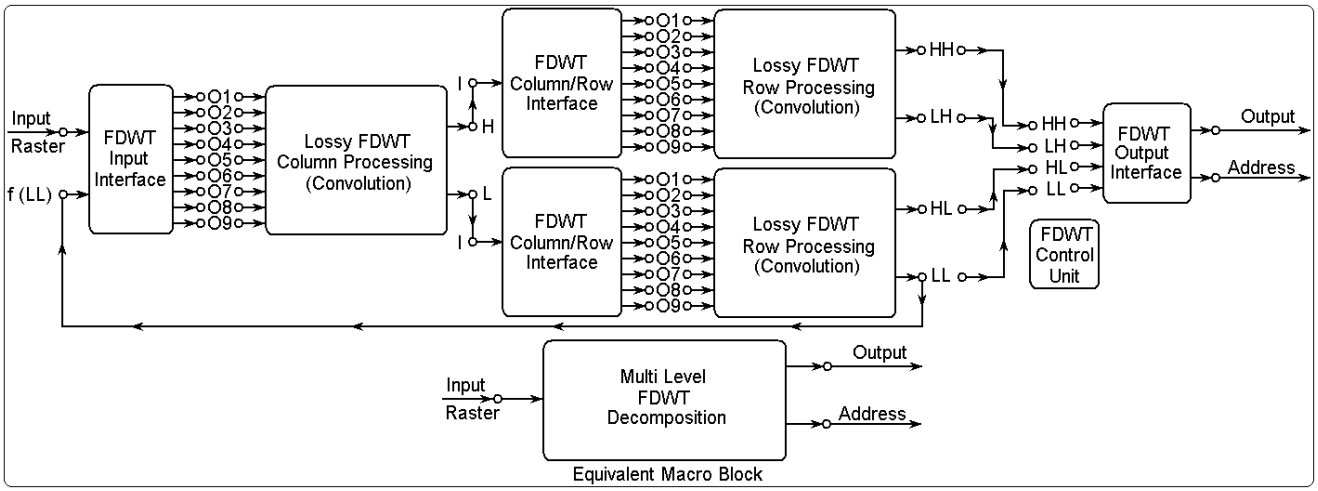

Figure 22: Overall block diagram of the forward DWT based on realization of the conventional convolution algorithm for (9/7) biorthogonal wavelet transform.

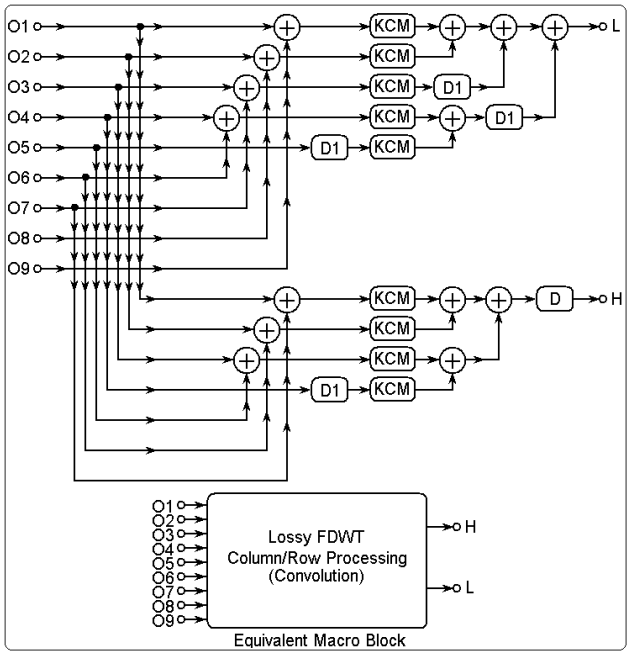

Figure 19: The main processing macro block for the lossy forward DWT.

7 Hardware Implementation of the Lossy Inverse DWT

The inverse operation starts with a set of row buffers. The structure of row buffers in this case is a bit different from that of the forward DWT. This structure for a given number of decomposition level is shown in Fig. 23.

Similarly, the core for the lossy column or row processing for the inverse transform is properly designed to carry out the corresponding operation as shown in Fig. 16. The structure of this implementation is shown in Fig. 24. The interface between column and row processing is similarly designed for inverse transform as shown in Fig. 25.

In the inverse transform, the input interface is somewhat more involved. In this case, there are two sets of DWT coefficients that are almost simultaneously synthesized when doing the column processing. The result of this processing is synthesized to generate the higher resolution image. This input interface is shown in Fig. 26. The overall inverse transform in terms of defined macro blocks is shown in Fig. 27.

8 Conclusion

We have presented a system level design for hardware realization of JPEG 2000 compression and decompression standards. We have considered both lossless and lossy compressions and dealt with each case independently. In implementation of the discrete wavelet transform, we have considered realization based on lifting algorithm as well as convolution and polyphase methods. It was shown that implementation based on

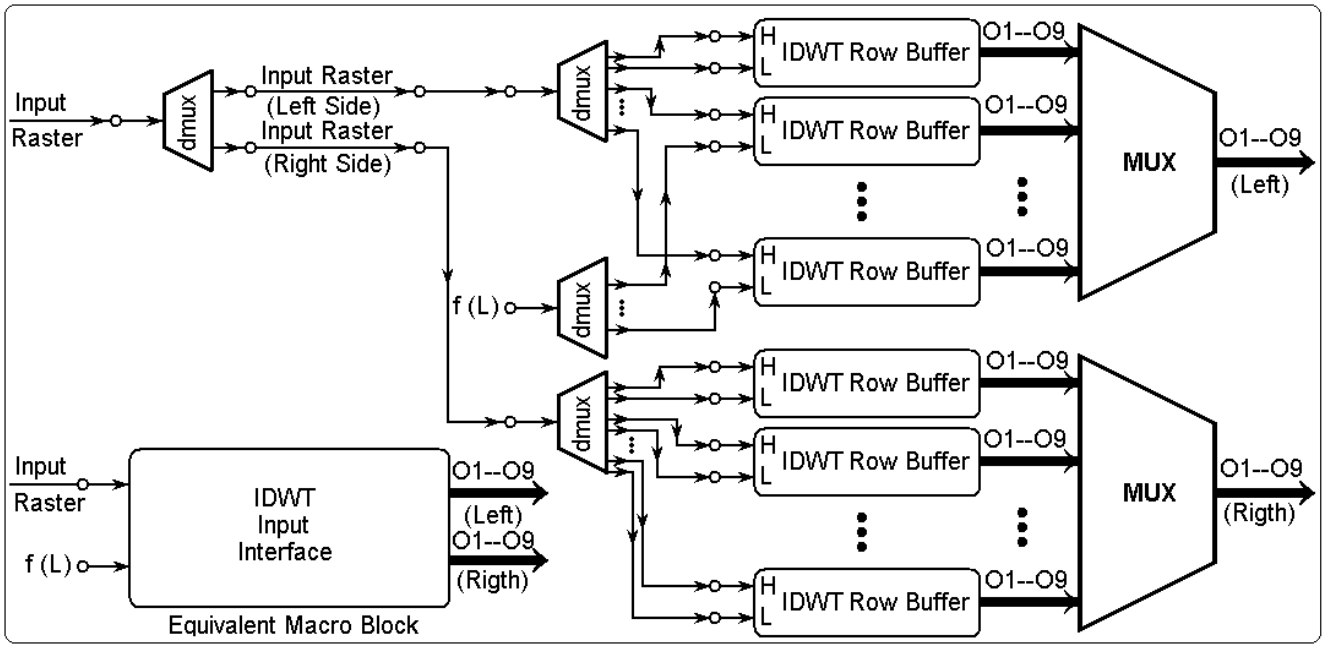

Figure 26: Input interface macro block for inverse DWT. The input raster is properly load two different sets of IDWT row buffers. The number of row buffers in each segment is the same as the decomposition level.

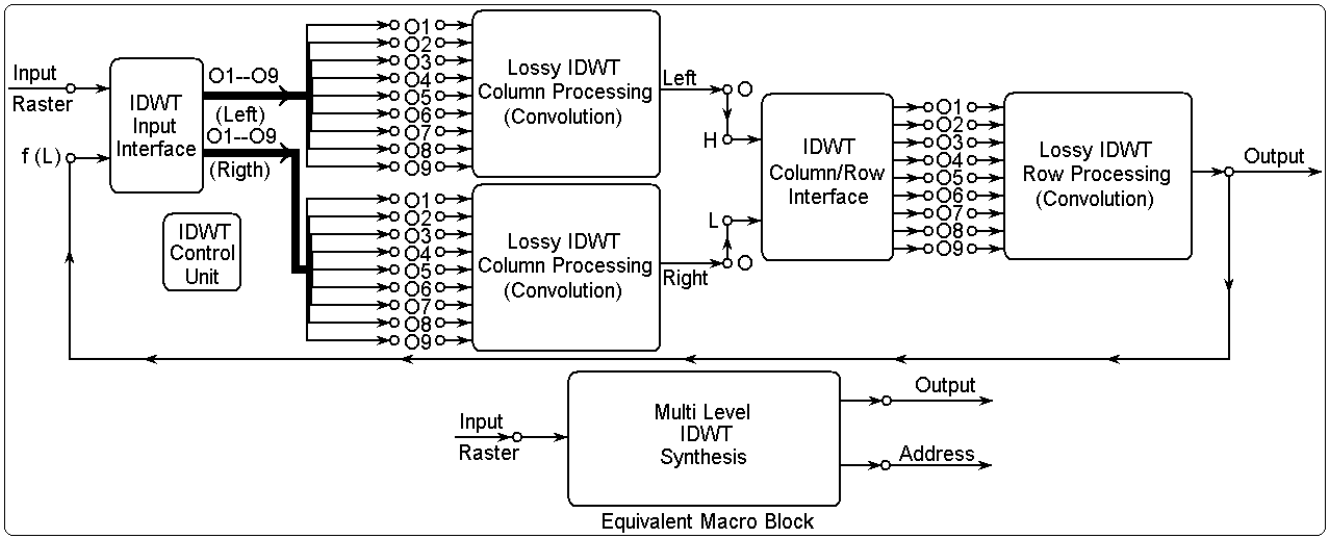

Figure 27: Overall block diagram of the inverse DWT based on realization of the convolution algorithm (9/7 biorthogonal wavelet transform).

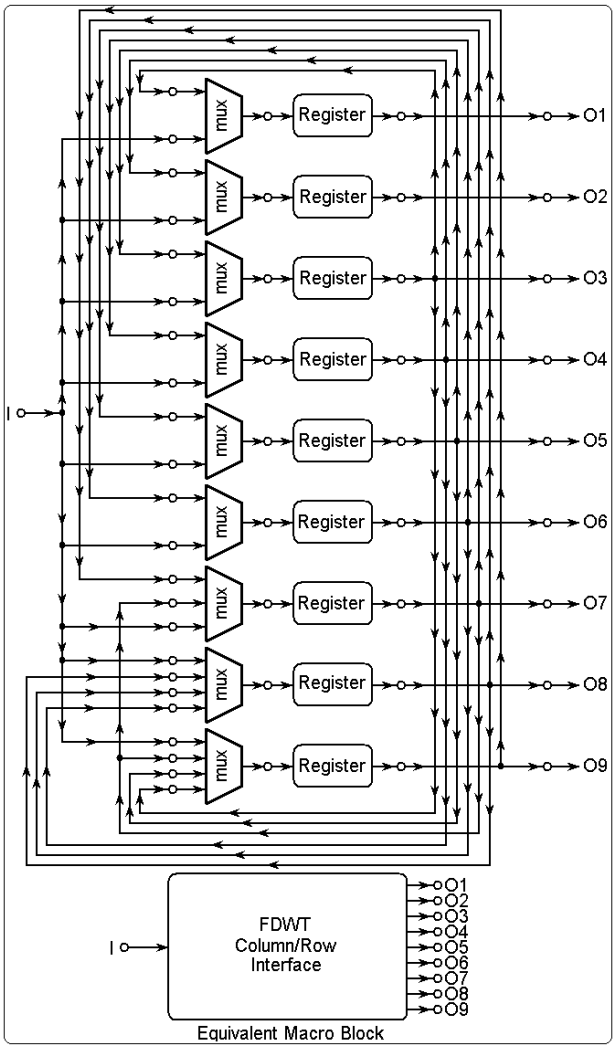

Figure 20: The interface macro block between column and row processing. Horizontal mirror imaging is conducted on this core as needed. For each decomposition levels there is a need for two of these blocks, one for Lowpass and the other one for Highpass outputs.

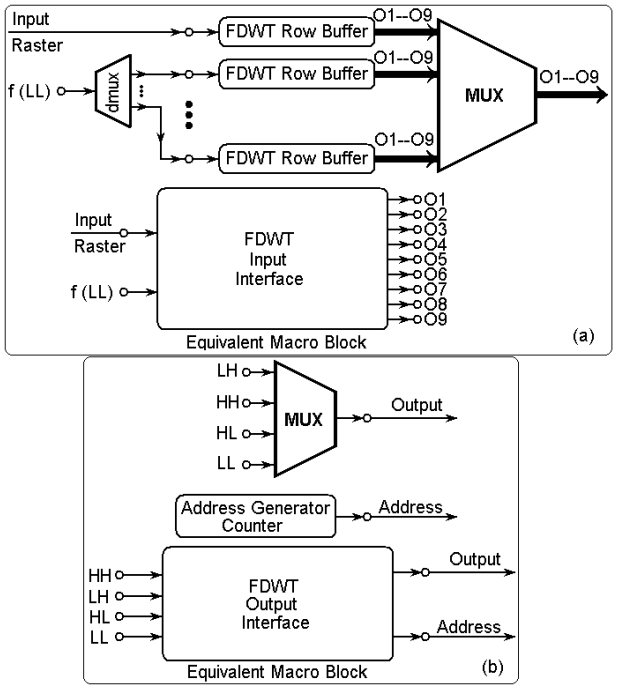

Figure 21: (a) Input and (b) output interface macro blocks used for multi-level forward DWT decomposition. In the FDWT input interface the row buffers are defined as shown in Fig. 18.

lifting algorithm is not always advantages in terms of hardware. We have presented each system level block and provided enough details for hardware implementation.

This implementation is mainly intended for video processing where each frame is required to be processed independently. This might have applications in medical imaging systems as well as digital cinema.

References:

[1] Marpe, D.; Schwarz, H.; and Wiegand, T., "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, 2003, pp. 620636.

[2] Reza, Ali M. and Turney, Robert D., "FPGA Implementation of 2D Wavelet Transform," Proceedings of the Thirty Third Annual Asilomar Conference on Signals, Systems, and Computers. October 1999, Pacific Grove, California.

[3] Reza, Ali M., "System level design of the coding and modeling of the adaptive arithmetic coding used in the JPEG 2000," International Journal

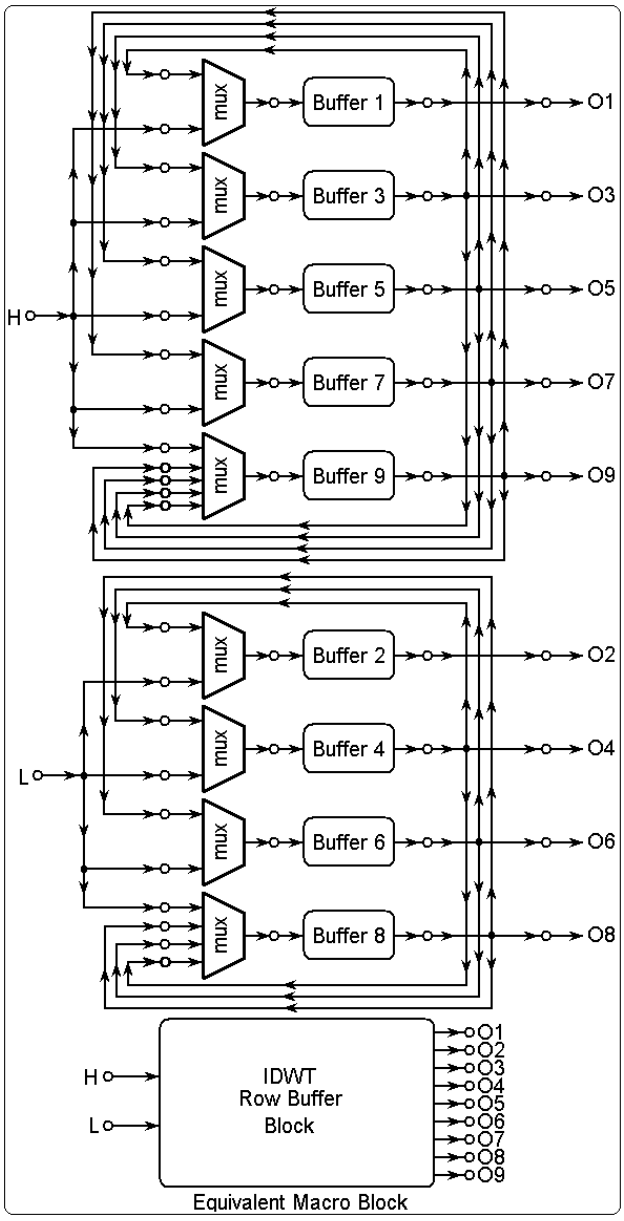

Figure 23: Row buffer macro block used for inverse DWT. In this case, two row buffers are assigned to the Lowpass component of the image and three row buffers are assigned to the Highpass component.

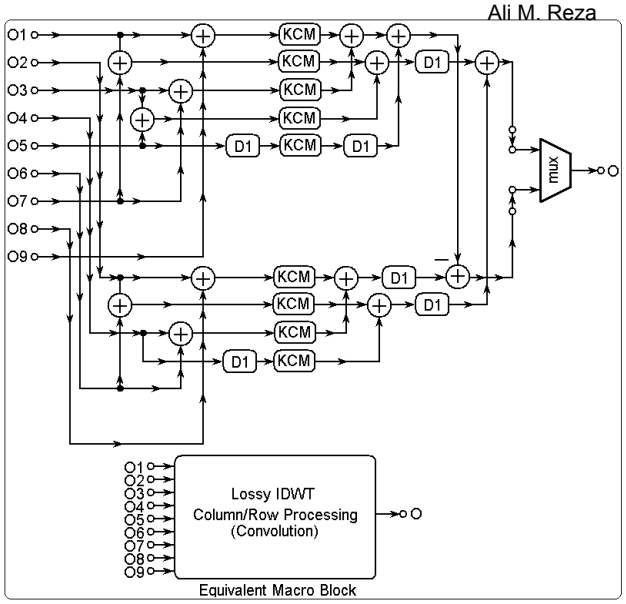

Figure 24: Macro block for column or row processing in the case of the inverse DWT. In this case the final output sequence is prepared by the final switch to properly order the odd and even samples.

of Information Engineering (IJIE), Vol. 2, No. 3, September 2012, PP. 86-96.

[4] Reza, Ali M., "System Level Design of the Adaptive Arithmetic Encoding Used in the JPEG 2000 Standard," International Journal on Electrical Engineering and Informatics, Vol. 5, No. 1, March 2013.

[5] Reza, Ali M., "System Level Design of the Adaptive Arithmetic Decoding Used in the JPEG 2000," Submitted to International Journal on Electrical Engineering and Informatics.

[6] Coding of Still Pictures: JBIG&JPEG, "JPEG 2000 image coding system," JPEG 2000 Final Committe Draft Version 1.0, ISO/IEC JTC1/SC29 WG1, JPEG 2000 Editor Martin Boliek, Coeditors: Charilaos Christopoulos, and Eric Majani, March 16, 2000.

[7] Ahmad, A.; Loo, K. K.; and Cosmas, J., "VLSI Architecture Design Approaches for Real-Time Video Processing." WSEAS Transactions on Circuits and Systems, Vol. 7, No. 8, August 2008, pp. 855-868.

[8] Atitallah, A. Ben; Loukil, H.; Kadionik, P.; and Masmoudi, N., "Advanced Design of TQ/IQT Component for H.264/AVC Based on SoPC Validation." WSEAS Transactions on Circuits and Systems, Vol. 11, No. 7, July 2012, pp. 211-223.

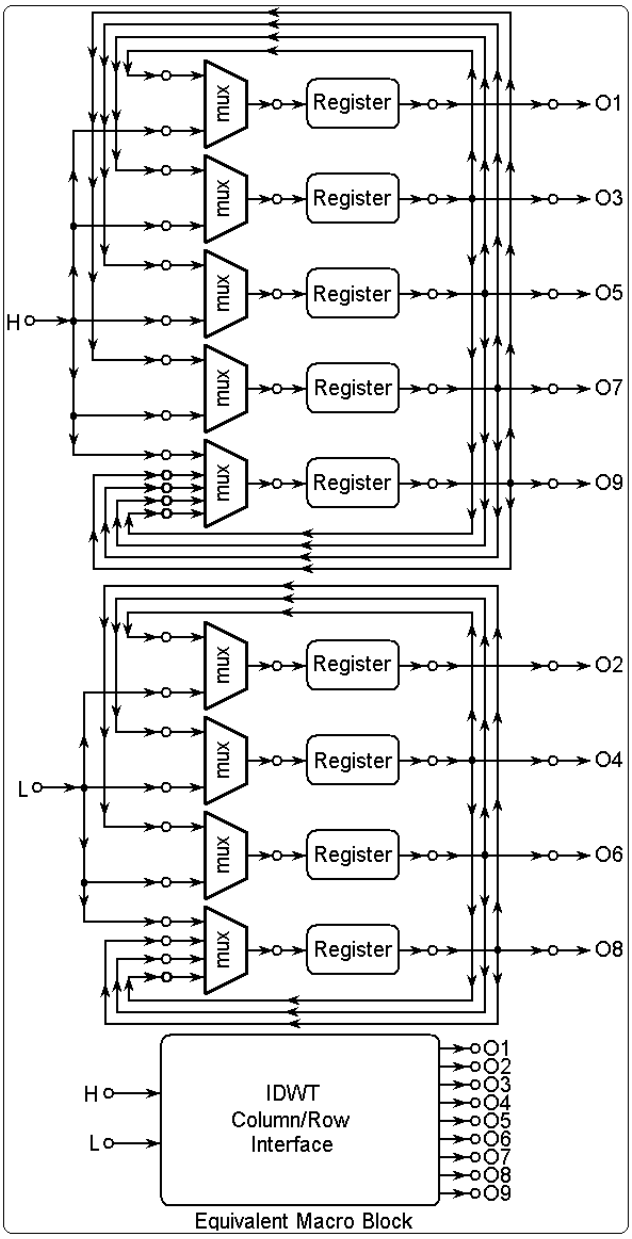

Figure 25: Interface between column and row processing in the inverse DWT.

[9] Ho, Huong, "A Hardware Architecture for Motion Compensated Video Frame Rate Up-Conversion." WSEAS Transactions on Circuits and Systems, Vol. 11, No. 2, February 2012, pp. 43-55.

[10] ISO/IEC JTC 1/SC 29/WG 1 N1890, Date: 25 September 2000, ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8) Coding of Still Pictures JBIG JPEG, Joint Bi-level Image Joint Photographic Experts Group Experts Group, TITLE: JPEG 2000 Part I Final Draft International Standard (corrected and formatted), SOURCE: ISO/IEC JTC1/SC29 WG1, JPEG 2000 Editor Martin Boliek, Co-editors Charilaos Christopoulos, and Eric Majani, PROJECT: 1.29.15444 (JPEG 2000).

[11] Sweldens, W., Construction and Applications of Wavelets in Numerical Analysis. Ph.D. Thesis, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, 1994.

[12] Sweldens, W., "The lifting scheme: A custom-design construction of biorthogonal wavelets," Appl. Comput. Harmon. Anal., 3(2):186-200, 1996.

[13] Daubechies, I. and Sweldens, W., "Factoring wavelet transforms into lifting steps," Technical report, Bell Laboratories, Lucent Technologies, 1996.

[14] Calderbank, R., Daubechies, I., Sweldens, W., and Yeo, B.-L., "Wavelet transforms that map integers to integers," Appl. Comput. Harmon. Anal., 3:127-153, 1996.

DISCLAIMER AND NOTE

The views expressed herein are those of the author and are not to be construed as official or reflecting the views of the Commandant, the U.S. Coast Guard, the Department of Homeland Security, or any agency of the U.S. Government.