Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

David Barina · Michal Kula · Pavel Zemcik

Parallel Wavelet Schemes for Images

How to make the wavelet transform friendly to parallel architectures

Received: date / Revised: date

Abstract In this paper, we introduce several new schemes for calculation of discrete wavelet transforms of images. These schemes reduce the number of steps and, as a consequence, allow to reduce the number of synchronizations on parallel architectures. As an additional useful property, the proposed schemes can reduce also the number of arithmetic operations. The schemes are primarily demonstrated on CDF 5/3 and CDF 9/7 wavelets employed in JPEG 2000 image compression standard. However, the presented method is general, and it can be applied on any wavelet transform. As a result, our scheme requires only two memory barriers for 2-D CDF 5/3 transform compared to four barriers in the original separable form or three barriers in the non-separable scheme recently published. Our reasoning is supported by exhaustive experiments on high-end graphics cards.

Keywords Discrete wavelet transforms, Image processing, Parallel architectures

1 Introduction

The two-dimensional discrete wavelet transform (DWT) is a signal-processing transform suitable as a basis for sophisticated compression algorithms. For example, JPEG 2000, an image coding system, is based on such compression technique. This paper focuses on the Cohen–Daubechies–Feauveau (CDF) 5/3 and 9/7 wavelets [1], which are often used for image compression. However, the methods are general, and they are not limited to any specific type of transform. Of course, plenty of other applications are built over the discrete wavelet transform.

The one-dimensional discrete wavelet transform has undergone a gradual development in the last few decades. Probably, the most important advance is the discovery of a factoring algorithm [2] referred to as the lifting scheme. In this context, the discrete wavelet transform or two-band subband filtering can be represented by a polyphase matrix. The lifting scheme algorithm decomposes any wavelet transform with finite filters into a finite sequence of lifting steps, while reducing the number of arithmetic operations. The decomposition corresponds to a factorization of the polyphase matrix filters into elementary matrices. The resulting coefficients of 1-D transform are formed in two subbands. The subbands correspond to low-pass (L) and high-pass (H) filtered subsampled variants of the original signal.

In case of two-dimensional transform [3], one level of the transform can be realized using a separable decomposition scheme. In this scheme, the coefficients are evaluated by successive horizontal and vertical 1-D filtering, resulting in four disjoint groups (LL, HL, LH, and HH subbands). A naive algorithm of 2-D transform computation directly follows the horizontal and vertical filtering loops. As a consequence, the number of elementary polyphase matrices is doubled.

Unfortunately, this separable computation does not reflect the requirements of the parallel architectures where the scheme will need twice as many synchronizations. Such synchronizations often form a bottleneck of the overall calculation. State-of-the-art algorithms fuse the horizontal and vertical loops into a single one, which results in the single-loop approach. However, the number of the elementary polyphase matrices and thus the number of memory barriers remain unaffected.

To solve the outlined issue, we propose several novel spatial lifting structures computing the 2-D discrete wavelet transform with reduced number of memory barriers. These lifting structures are presented in the order in which they were gradually derived. The presented work is accompanied by exhaustive performance experiments.

A typical representative of parallel architectures is the graphics processing unit (GPU) capable of executing a general-purpose program. Actually, this is the architecture used to evaluate the performance of algorithms pre-

Centre of Excellence IT4Innovations
Faculty of Information Technology
Brno University of Technology
Bozetechova 1/2, 612 66 Brno
Czech Republic
E-mail: {ibarina,ikula,zemcik}@fit.vutbr.cz

sented in this paper. We have employed OpenCL language for writing underlying implementations. These implementations were then subject of performance measurements on significant graphics cards of two biggest vendors.

In order to avoid misunderstandings, it should be noted that the schemes presented in this paper do not affect an image compression ratio nor quality. The schemes only affect the speed in which the compression is completed. Since practical applications require a multi-level discrete wavelet decomposition, the question of how to compute this multi-scale pyramid may arise. In this case, the schemes discussed in this paper can simply be applied in a sequence exchanging intermediate results through a off-chip memory (a global memory in the case of GPU). Another possibility is to apply this sequence on blocks exchanging the results using a fast on-chip memory (a local memory on GPU). The latter possibility was used, e.g., in [4; 5] employing the naive algorithm of 2-D transform computation.

The rest of the paper is organized as follows. Section Related Work presents the theory in the necessary level of detail. This theory includes the lifting scheme basics and the spatial lifting structures recently proposed. Subsequent Section Proposed Schemes derives the new spatial lifting structures. Additionally, Section Improvements presents a simple trick proposed in order to reduce the number of arithmetic operations. Section Evaluation and Section Performance offer a thorough performance evaluation. Finally, Section Conclusions summarizes the paper.

2 Related Work

In this paper, the well-known z-transform notation is employed for the description of FIR filters. The transfer function of the FIR filter h_k is a Laurent polynomial defined as

$$H(z) = \sum_{k=k_0}^{k_1-1} h_k \, z^{-k}, \tag{1}$$

where k_0 denotes the smallest and $k_1 - 1$ denotes the largest integer number k for which h_k is non-zero. The degree of a Laurent polynomial $H(z)$ is defined as $|H(z)| = k_1 - k_0 - 1$. Similarly, the transfer function of the two-dimensional FIR filter h_{k_m,k_n} is a bivariate Laurent polynomial defined as

$$H(z_m, z_n) = \sum_{k_m=k_{0,m}}^{k_{1,m}-1} \sum_{k_n=k_{0,n}}^{k_{1,n}-1} h_{k_m,k_n} \, z_m^{-k_m} z_n^{-k_n}, \tag{2}$$

where m refers to the horizontal axis and n to the vertical one. Moreover, to keep consistency with other papers, the $H^*(z_m, z_n) = H(z_n, z_m)$ denotes a filter transposed to the $H(z_m, z_n)$. For simplicity, we have made a small abuse of notation. Instead of the full notation $H(z_m, z_n)$, we only use a shortened labeling, such as H. Finally, we work with 2×2 and 4×4 matrices of Laurent polynomials. These are usually referred to as the polyphase matrices. The 2×2 matrices refers to the 1-D systems, whereas the 4×4 to the 2-D ones. For simplicity, a shortened labeling is used for matrices as well. The superscript T denotes the vector or matrix transposition.

2.1 Discrete Wavelet Transform

The discrete wavelet transform has undergone a gradual development [6] in the last few decades. First, S. Mallat [3] demonstrated the multi-scale wavelet decomposition computed with a pyramidal algorithm based on convolutions with quadrature mirror filters. In detail, the discrete wavelet transform splits the input signal x_k into two components L and H, each subsampled by a factor of 2. Both of these components can be computed by the discrete convolution with two FIR filters $G_0(z)$ and $G_1(z)$ followed by the subsampling. However, such computation is usually not the fastest one. The transform can also be represented by the polyphase matrix [7]. Using this representation, the input signal is initially split into the L, H components. No calculation is performed so far. After such splitting, the DWT

$$\mathbf{y} = \mathrm{M}\,\mathbf{x}. \tag{3}$$

is described by the 2×2 matrix M mapping the initial components

$$\mathbf{x} = \begin{bmatrix} \mathrm{L} & \mathrm{H} \end{bmatrix}^T \tag{4}$$

onto the resulting ones

$$\mathbf{y} = \begin{bmatrix} \mathrm{L} & \mathrm{H} \end{bmatrix}^T. \tag{5}$$

The polyphase matrix is initially assembled as a polynomial matrix

$$\mathrm{M} = \begin{bmatrix} \mathrm{G}_{1,o} & \mathrm{G}_{1,e} \\ \mathrm{G}_{0,o} & \mathrm{G}_{0,e} \end{bmatrix}, \tag{6}$$

where subscript e refers to the even coefficients, whereas o refers to the odd coefficients.

2.2 Lifting Scheme

As a next step, W. Sweldens [2; 8] showed how any discrete wavelet transform can be decomposed into a sequence of simple filtering steps. These steps are referred to as the lifting steps, and the scheme is known as the lifting scheme. The lifting scheme reduces the number of arithmetic operations by up to 50 %. The lifting steps occur in K pairs. The first step is referred to as the predict and the second one to as the update. It may happen that the very first step of the lifting scheme is missing and the sequence of steps starts with the update step. Usually, the very last step has a different form compared to all the others. This one is then called the scaling step.

$$\mathrm{M} = \begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix} \prod_{k=K-1}^{0} \begin{bmatrix} 1 & \mathrm{U}^{(k)} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \mathrm{P}^{(k)} & 1 \end{bmatrix}, \qquad (7)$$

where ζ is a non-zero scaling factor, $\mathrm{P}^{(k)}$ is the kth predict convolution operator, and $\mathrm{U}^{(k)}$ is the kth update convolution operator. In this paper, we focus on a single pair of lifting steps. We thus omit the (k) superscript. We also omit the scaling step, as the application of this step is trivial.

In parallel environments [9], the processing of a single or several adjacent signal samples is mapped to independent processing units, commonly referred to as the threads. To avoid race conditions (the behavior where the output is dependent on the sequence or timing of other threads), the threads must use some type of synchronization method. In this paper, we will consider the use of memory barriers. When we return to the lifting scheme, these barriers are usually required before each of the individual lifting steps. However, certain form of the steps guarantees correctness of the calculation even without using the memory barrier between them. In this paper, the barriers are indicated by the $\big|$ symbol placed in between the steps. For example, $\mathrm{M}_2 \big| \mathrm{M}_1$ denotes a sequence of two steps – the initial M_1 and the subsequent M_2 – separated by the barrier.

The schemes presented above can be extended into two dimensions. The most widely used 2-D extension is Mallat's [3] 2-D decomposition. The transform is defined as the tensor product of 1-D transforms. At each scale of such decomposition, we obtain a quadruple of wavelet coefficients (LL, HL, LH, HH).

2.3 Convolution and Polyphase Schemes

Similarly to the 1-D case, the transform can be computed using the convolution scheme. Considering this case, one needs to convolve the input signal with four 2-D FIR filters. This operation is followed by the subsampling in both dimensions. However, in practical implementations, the subsamplings are built into the convolutions in order to save arithmetic operations. This scheme will further be labeled as Convolution. In this scheme, no barrier is required at all.

Moreover, the 2-D transform can be described by the polyphase matrix as well. Using the polyphase representation, the input signal is initially split into the four polyphase components. No calculation is performed so far. Further, the 2-D DWT is described by the 4×4 matrix M mapping the input components

$$\mathbf{x} = \begin{bmatrix} \mathrm{LL} & \mathrm{HL} & \mathrm{LH} & \mathrm{HH} \end{bmatrix}^T \qquad (8)$$

onto the final ones

$$\mathbf{y} = \begin{bmatrix} \mathrm{LL} & \mathrm{HL} & \mathrm{LH} & \mathrm{HH} \end{bmatrix}^T. \qquad (9)$$

Similarly to the 1-D case, this can be written as

$$\mathbf{y} = \mathrm{N}_{\mathrm{P,U}} \big| \mathbf{x}, \qquad (10)$$

where P, U are 1-D predict and update convolution operators. Please notice the included initial barrier. This scheme will further be called as Polyphase.

To define the 2-D polyphase matrices, the predict and update operators must first be migrated into two dimensions. Coupled together with filter transposition defined above, the two-dimensional counterparts of the operators are defined like follows.

$$\begin{bmatrix} \mathrm{P} \\ \mathrm{U} \\ \mathrm{P}^* \\ \mathrm{U}^* \end{bmatrix} = \begin{bmatrix} P(z_m, z_n) \\ U(z_m, z_n) \\ P^*(z_m, z_n) \\ U^*(z_m, z_n) \end{bmatrix} = \begin{bmatrix} P(z_m) \\ U(z_m) \\ P(z_n) \\ U(z_n) \end{bmatrix} \qquad (11)$$

Roughly speaking, the P and U denote the filters oriented along the horizontal axes, whereas the P^* and U^* denote the filters oriented along the vertical one.

2.4 Notation

For readers not familiar with signal-processing notations, a relationship of the block and data-flow diagrams is explained in this section. In this paper, we work with 4×4 matrices of Laurent polynomials, usually referred to as the polyphase matrices, for example, this one

$$\mathrm{T}_{\mathrm{P}}^{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \mathrm{P} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \mathrm{P} & 1 \end{bmatrix}. \qquad (12)$$

Since these matrices define a linear mapping from vectors of form $\begin{bmatrix} \mathrm{LL} & \mathrm{HL} & \mathrm{LH} & \mathrm{HH} \end{bmatrix}^T$ to vectors of the

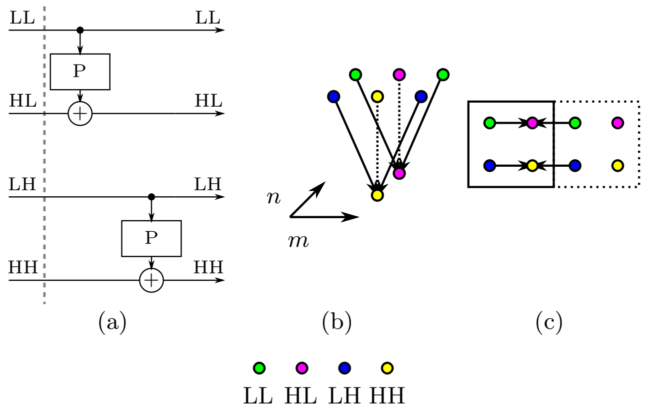


Fig. 1 Different visual representations of the same polyphase matrix.

same form, we can simply illustrate this mapping by the block diagram in Fig. 1(a).

Moreover, the matrices are composed of elementary lifting operators like

$$P(z) \;= -1/2(1+z^{-1}). \tag{13}$$

If we substitute such particular polynomials into the matrix, the mapping gets a specific shape, as illustrated by the dataflow diagram in Fig. 1(b). The solid arrows correspond to multiplication by $-1/2$ along with subsequent summation. The dotted arrows similarly correspond to multiplication by factor of 1, since the matrix T_P^H contains ones on the main diagonal.

For reader's convenience, we use two-dimensional diagrams to illustrate the schemes with CDF 5/3 wavelets. For the example above, such a diagram is shown in Fig. 1(c), whereas the elementary quadruples of coefficients are highlighted by solid and dotted boxes.

2.5 Sweldens Scheme

Following the Mallat's scheme, the predict and update lifting steps are applied in both directions sequentially. This can be classified as a separable scheme. As the convolution is the linear operator, horizontal and vertical steps can be arbitrary interleaved. The baseline formulation of this scheme will be considered as follows. The predict steps are always preceding the update ones. Such separable scheme can be formally described by

$$\mathbf{y} = \mathrm{S}_\mathrm{U}^V \,\big|\, \mathrm{S}_\mathrm{U}^H \,\big|\, \mathrm{T}_\mathrm{P}^V \,\big|\, \mathrm{T}_\mathrm{P}^H \,\big|\, \mathbf{x}, \tag{14}$$

where the individual matrices are defined as follows. Let us mention a short comment on the matrix notation used. For example, the matrix T_P^H is parameterized by the P polynomial. Further in the text, the same matrix appears parameterized by different polynomials, which is completely valid. As it can be expected, the matrix T^H definition is not repeated for such case. For better understanding, the corresponding signal-processing block diagram is shown in Fig. 2. For the CDF 5/3 wavelet, these steps are also graphically illustrated in Fig. 3.

$$\mathrm{T}_\mathrm{P}^H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \mathrm{P} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \mathrm{P} & 1 \end{bmatrix} \tag{15}$$

$$\mathrm{T}_\mathrm{P}^V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \mathrm{P}^* & 0 & 1 & 0 \\ 0 & \mathrm{P}^* & 0 & 1 \end{bmatrix} \tag{16}$$

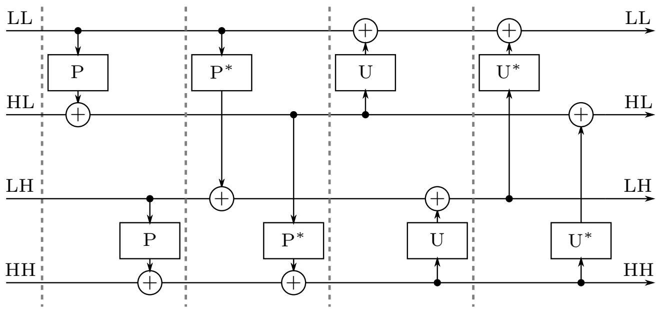


Fig. 2 Block diagram of the Sweldens scheme. The dashed vertical lines indicate barriers. The left half corresponds to the spatial predict operator, whereas the right half to the update one.

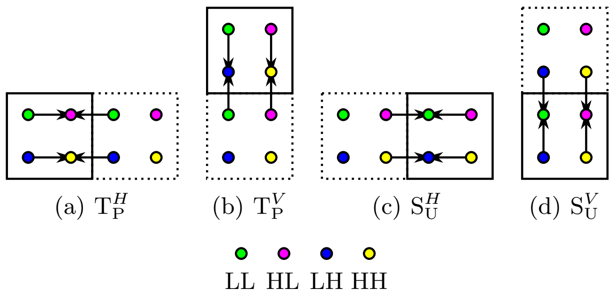

(a) T_P^H
(b) T_P^V
(c) S_U^H
(d) S_U^V
LL HL LH HH

Fig. 3 2-D dataflow diagram, CDF 5/3 wavelet, Sweldens lifting scheme. The displayed part of the calculation results in the coefficients inside of the solid box. The dotted boxes refer to the surrounding threads.

$$\mathrm{S}_\mathrm{U}^H = \begin{bmatrix} 1 & \mathrm{U} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{U} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$\mathrm{S}_\mathrm{U}^V = \begin{bmatrix} 1 & 0 & \mathrm{U}^* & 0 \\ 0 & 1 & 0 & \mathrm{U}^* \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

Please note the barriers in between each of the lifting steps. In total, four barriers are required for each pair of the original 1-D lifting steps. This scheme will further be labeled as Sweldens.

Contemporary approaches on parallel architectures most commonly reflect this separable Sweldens scheme. Exceptionally, the Convolution scheme is employed. Considering the independent horizontal and vertical filtering steps, several different strategies of 2-D DWT implementation can be used. These strategies can be divided into three groups – row–column, block-based, and pipelined methods. The row–column methods process all of the horizontal filtering steps prior to the vertical ones. The row–column method applied on the entire 2-D image was used for instance in [10–16]. In some papers, the transition between the horizontal and vertical stage is accompanied with data transposition. The pipelined methods was used, e.g., in [17] and [18]. These methods uses moving window for the vertical part of the transform. However, the horizontal and vertical parts remain separated. The block-based methods were used, e.g., in [4; 5; 14; 19]. The transform is tiled into blocks, in which the horizontal and vertical processing still remain separated. However, between these parts, the data remain loaded in the local memory (making them faster accessible). For the sake of completeness, some of the works compute an entire [4] or partial [5] multi-scale transform inside the blocks.

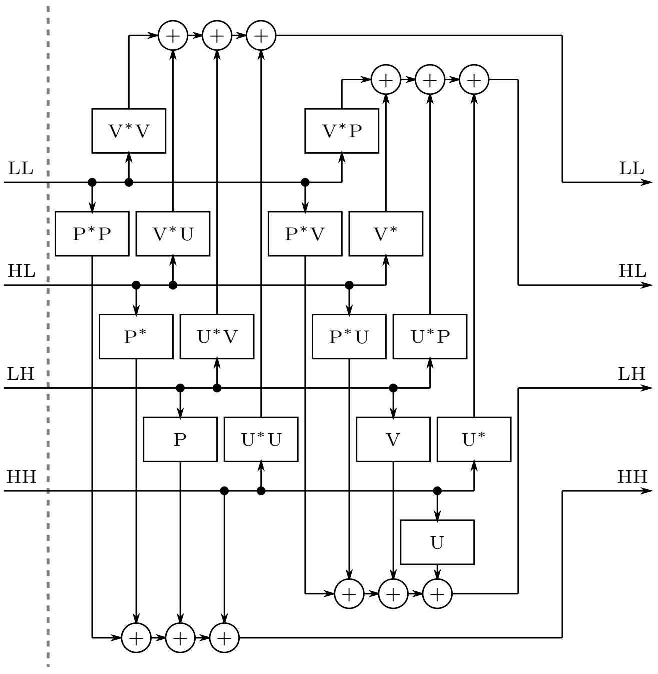
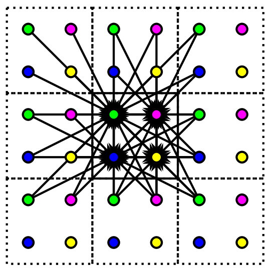


Fig. 4 Block diagram of the Polyphase scheme. The dashed vertical lines indicate implicit barrier.



(a) $\mathrm{N}_{\mathrm{P,U}}$

Fig. 5 2-D dataflow diagram, CDF 5/3 wavelet, Polyphase scheme. The solid box in the middle corresponds to the output coefficients.

Going back to the Polyphase scheme, the polyphase matrix

$$\mathrm{N}_{\mathrm{P,U}} = \begin{bmatrix} \mathrm{V^*V} & \mathrm{V^*U} & \mathrm{U^*V} & \mathrm{U^*U} \\ \mathrm{V^*P} & \mathrm{V^*} & \mathrm{U^*P} & \mathrm{U^*} \\ \mathrm{P^*V} & \mathrm{P^*U} & \mathrm{V} & \mathrm{U} \\ \mathrm{P^*P} & \mathrm{P^*} & \mathrm{P} & 1 \end{bmatrix} \tag{19}$$

can expressed using the auxiliary polynomial $\mathrm{V} = \mathrm{PU} + 1$. The matrix can be obtained as the product of individual matrices of the Sweldens scheme. In this scheme, it is no longer possible to distinguish the vertical and horizontal filtering. Only an initial barrier is required for this scheme. Unfortunately, the number of arithmetic operations has grown in proportion to the square of filter sizes. The corresponding generic signal-processing diagram is shown in Fig. 4. For the CDF 5/3 wavelet, these operations are illustrated in Fig. 5.

2.6 Iwahashi Scheme

Recently, Iwahashi *et al.* [20–22] presented the non-separable lifting scheme, consisting of three spatial lifting steps. As in the previous case, it is not possible to distinguish the vertical and horizontal filtering. The three steps can be described as follows. Initially, a 2-D lifting step leading to the computation of the HH coefficient is performed. This step corresponds to a spatial predict convolution operator. This is followed by parallel computation of the HL and LH coefficients, using the original 1-D predict and update filters. In the third step, the LL coefficient is computed using another 2-D filter. The last step can be understood as a spatial update operator. In the matrix notation, the scheme can be defined as

$$\mathbf{y} = \mathrm{S}^{I}_{\mathrm{U}} \,|\, \mathrm{R}^{I}_{\mathrm{P,U}} \,|\, \mathrm{T}^{I}_{\mathrm{P}} \,|\, \mathbf{x}, \tag{20}$$

where the individual matrices are defined as follows. The signal-processing diagram is shown in Fig. 6. For the CDF 5/3 wavelet, the individual steps are illustrated in Fig. 7.

$$\mathrm{T}^{I}_{\mathrm{P}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathrm{PP^*} & \mathrm{P^*} & \mathrm{P} & 1 \end{bmatrix} \tag{21}$$

$$\mathrm{R}^{I}_{\mathrm{P,U}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \mathrm{P} & 1 & 0 & \mathrm{U^*} \\ \mathrm{P^*} & 0 & 1 & \mathrm{U} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

$$\mathrm{S}^{I}_{\mathrm{U}} = \begin{bmatrix} 1 & \mathrm{U} & \mathrm{U^*} & -\mathrm{UU^*} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

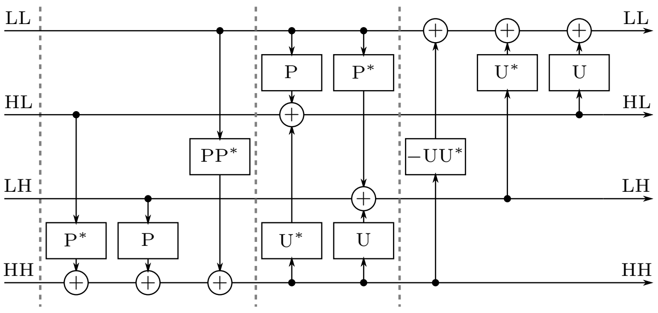


Fig. 6 Block diagram of the Iwahashi scheme. The dashed vertical lines indicate barriers.

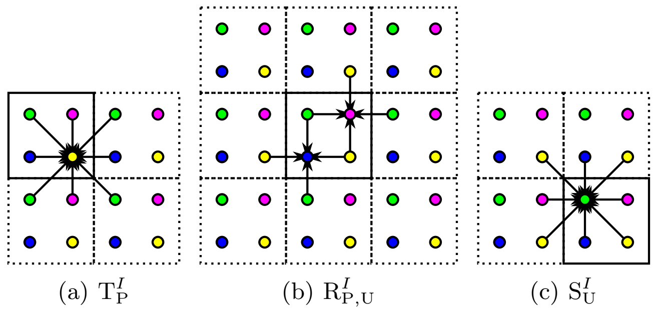


Fig. 7 2-D dataflow diagram, CDF 5/3 wavelet, Iwahashi lifting scheme. The solid box corresponds to the output coefficients.

Three barriers are required in between these steps. As for the Polyphase scheme, the number of arithmetic operations increased proportionally to the square of filter sizes. However, the total number of operations is significantly lower. This scheme will further be labeled as Iwahashi.

When we compare the separable Sweldens and non-separable Iwahashi schemes, some findings becomes obvious at first glance. The number of operations tends to be considerably smaller for the separable case. On the other hand, the number of memory barriers in the non-separable scheme was reduced to 75 % (from four to three barriers). The Polyphase scheme stands apart from these two schemes. It needs only an initial memory barrier. Unfortunately, the number of arithmetic operations is unreasonably large. This is caused by the number of non-zero elements in the corresponding polyphase matrix as well as by the degree of the longest filter V. For clarification, the product of a Laurent polynomial of degree $|P(z)|$ and a Laurent polynomial of degree $|U(z)|$ is a Laurent polynomial of degree $|P(z)| + |U(z)|$. Finally, the Convolution scheme employing four 2-D filters is even worse in terms of the operations. Anyway, only an initial memory is required here as well. Detailed quantitative comparison is provided in Section Evaluation.

When we consider the linearity of the convolution and the dependencies between the individual lifting steps, several gaps can be inferred in the schemes described above. Recombining the operations into a new form could lead to the removal of unnecessary barriers. Actually, exactly this idea is investigated in the following section, in which several novel 2-D schemes are proposed.

Since this work is based on our previous work in [23], it should be explained what the difference between this work and [23] is. In [23], we presented a block-based method employing a scheme foregoing the schemes proposed in this paper. Unlike [23], the schemes presented in this paper are defined by general predict and update operators.

3 Proposed Schemes

In this section, the polyphase matrices, known so far, are reassembled in order to obtain the schemes suitable for parallel architectures. All of the schemes discussed here are general, and they can be used for any discrete wavelet transform. Please note that the contribution of this paper is presented in this section and the following one.

3.1 Explosive Scheme

When we take a detailed look at the original 1-D lifting scheme, a certain pattern can be identified in the predict and update steps. Particularly, the predicts transmit data from L into H samples, whereas the updates transmit data from H into L. The transmission can be viewed from two perspectives – the data flow out from a source component (similarly to an explosion); or the data flow in into a destination component (an implosion). As it can be expected, the Sweldens scheme exactly follows this pattern, since this scheme is a mere extension of 1-D lifting into two dimensions. Roles of source and destination samples properly turn during four lifting steps (horizontal and vertical, predict and update). This procedure can be also seen as a data transmission in direction from LL into HH component (using 1-D predicts), and a transmission from HH into LL one (using updates). The HL and LH components are not relevant in this view. The situation is clearly visible in Fig. 2. In contrast to this scheme, the Iwahashi scheme has a different structure. The leading step transmits data into HH component (using predicts), while the trailing one transmits them into LL one (updates). However, no exclusive source components can be identified in this case. The remaining step in the middle is not relevant. See the block diagram in Fig. 6. Regarding to the perspectives outlined above, the Iwahashi scheme can be classified as an implosive one. However, this is not the only three-step version (two-step scheme is discussed below in the text). Similar scheme can be formulated using data explosions instead of the implosions. Particularly, the LL component spreads the data into its neighborhood during the predict step, whereas the data flow out from the HH component in the update step. No exclusive destination components can be identified here as well.

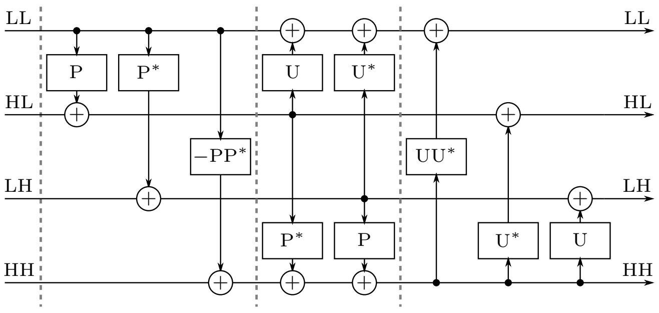
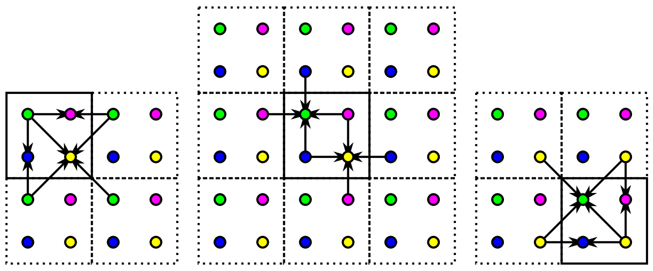


Fig. 8 Block diagram of the Explosive scheme. The dashed vertical lines indicate barriers.



(a) T^E_P (b) $R^E_{P,U}$ (c) S^E_U

Fig. 9 2-D dataflow diagram, CDF 5/3 wavelet, Explosive lifting scheme. The solid box corresponds to the output coefficients.

Again, the step in the middle is not relevant. For further purposes, this newly proposed scheme will be labeled as Explosive. The block diagram is shown in Fig. 8. The steps for the CDF 5/3 wavelet are also illustrated in Fig. 9. Formally, the scheme can be defined as

$$\mathbf{y} = S^E_U \,|\, R^E_{P,U} \,|\, T^E_P \,|\, \mathbf{x}, \tag{24}$$

where the individual matrices follows. Three barriers are required, as in the case of the Iwahashi scheme.

$$T^E_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ P & 1 & 0 & 0 \\ P^* & 0 & 1 & 0 \\ -PP^* & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

$$R^E_{P,U} = \begin{bmatrix} 1 & U & U^* & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & P^* & P & 1 \end{bmatrix} \tag{26}$$

$$S^E_U = \begin{bmatrix} 1 & 0 & 0 & UU^* \\ 0 & 1 & 0 & U^* \\ 0 & 0 & 1 & U \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

3.2 Monolithic Scheme

Motivated by the work of Iwahashi *et al.* [22], we have reorganized the elementary lifting filters in order to remove the middle lifting step. This action consequently reduces the number of memory barriers. As a result, we receive a new two-step non-separable scheme. The first step corresponds to a spatial predict operator. This one is completely responsible for the HH coefficient. In addition, the HL and LH coefficients are partially computed here as well. The second step corresponds to a spatial update. It is responsible for the LL coefficient and completion of the HL and LH ones. Formally, the scheme is defined as

$$\mathbf{y} = S_U \,|\, T_P \,|\, \mathbf{x}, \tag{28}$$

where the S_U and T_P are defined as follows. Moreover, the hypotetical signal-processing diagram is shown in Fig. 10. For the CDF 5/3 wavelet, the scheme is graphically illustrated in Fig. 11.

$$T_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ P & 1 & 0 & 0 \\ P^* & 0 & 1 & 0 \\ PP^* & P^* & P & 1 \end{bmatrix} \tag{29}$$

$$S_U = \begin{bmatrix} 1 & U & U^* & UU^* \\ 0 & 1 & 0 & U^* \\ 0 & 0 & 1 & U \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

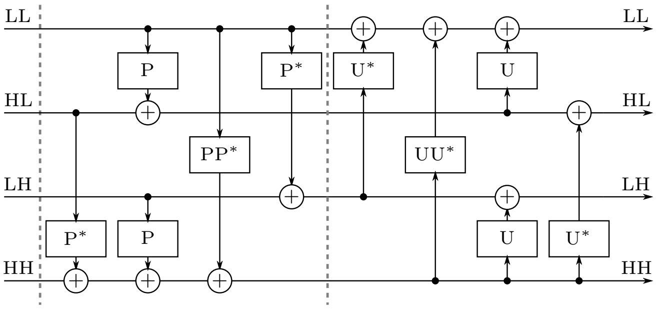
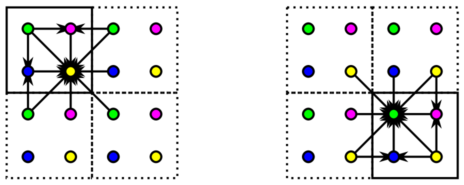


Fig. 10 Block diagram of the Monolithic scheme. The dashed vertical lines indicate barriers. The left half corresponds to the predict operator, whereas the right half to the update.



(a) T_P (b) S_U

Fig. 11 2-D dataflow diagram, CDF 5/3 wavelet, Monolithic scheme. The solid box corresponds to the output coefficients.

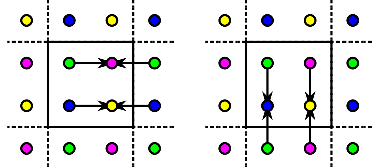
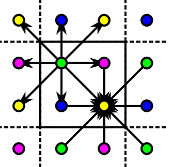
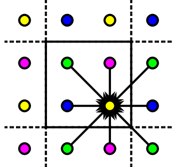
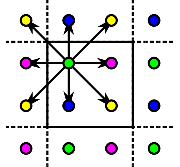
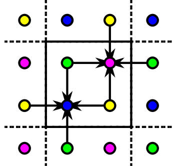
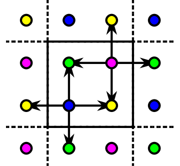
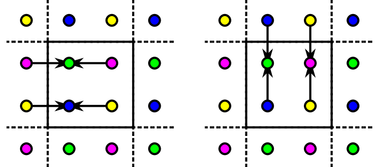
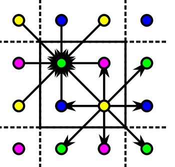
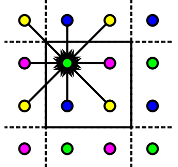
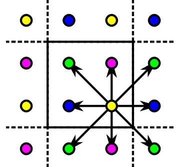
step	Sweldens	Monolithic	Iwahashi	Explosive
predict				
middle				
update				

Table 1 CDF 5/3 wavelet. Shapes of spatial lifting steps for selected schemes. The step in the middle raised from the combination of the original predict and update steps. Illustrative purpose only.

The total number of operations remained the same as for the Iwahashi scheme. However, the number of the explicit barriers has been reduced to only two. This is a crucial contribution of our work. Further in the paper, this scheme will be labeled as Monolithic. One can easily verify the correctness of the proposed scheme by comparing the product $\mathrm{S_U T_P}$ to the matrix $\mathrm{N_{P,U}}$ of the Polyphase scheme.

A comparison of the shapes for selected schemes can be found in Table 1. Regarding the Polyphase scheme, no spatial predict nor update step can be identified in its calculation.

In practical implementations, the formed intermediate coefficients cannot take the same place as the input ones. Otherwise, the race condition occurs. This implies a higher memory consumption compared to the previous schemes. A particular numbers are listed in Table 3.

Two simple observations can be made from the scheme presented so far. The Sweldens scheme requires the lowest number of operations. In contrast to this approach, the non-separable scheme proposed above requires the lowest number of memory barriers. Combining these two observations together, new schemes can be formed. This possibility is investigated below.

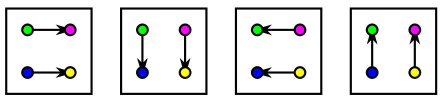
4 Improvements

Additionally, we have made another observation. The operation composed as a product of monomials with the exponent of z_n and z_n being equal to zero (i.e., scalars) never touch the coefficients belonging to the surrounding threads. As the convolution is the linear operation, this monomial can be detached from the original operator and subsequently calculated using the Sweldens scheme. This scheme has a minimal number of arithmetic operations. The rest of the original polynomial shall be computed using different scheme, according to suitability for a particular platform.

In more detail, the original filters were split into two halves as $\mathrm{P} = \mathrm{P_0} + \mathrm{P_1}$, and $\mathrm{U} = \mathrm{U_0} + \mathrm{U_1}$, where $\mathrm{P_0}$ and $\mathrm{U_0}$ are scalars. This is a fundamental step for the following constructions. Now, the scalars $\mathrm{P_0}, \mathrm{U_0}$ can be utilized in the separable Sweldens scheme. This part will never touch the extraneous threads. For a better understanding, see the dataflow diagram in Fig. 12. Conversely, the $\mathrm{P_1}, \mathrm{U_1}$ shall be employed in the Explosive, Iwahashi, Monolithic, or Polyphase scheme in order to minimize the number of required memory barriers. Note that these two schemes can be combined into joint lifting steps. However, such optimization is a simple matter of a specific implementation.

Initially, we have employed the idea described in previous paragraphs in conjunction with the Iwahashi scheme. The resulting scheme is defined as

$$\mathbf{y} = \mathrm{S}^V_{\mathrm{U}_0}\,\mathrm{S}^H_{\mathrm{U}_0}\,\mathrm{S}^I_{\mathrm{U}_1} \,\big|\, \mathrm{R}^I_{\mathrm{P}_1,\mathrm{U}_1} \,\big|\, \mathrm{T}^I_{\mathrm{P}_1} \,\big|\, \mathrm{T}^V_{\mathrm{P}_0}\,\mathrm{T}^H_{\mathrm{P}_0}\mathbf{x}, \tag{31}$$



(a) $\mathrm{T}^H_{\mathrm{P}_0}$ (b) $\mathrm{T}^V_{\mathrm{P}_0}$ (c) $\mathrm{S}^H_{\mathrm{U}_0}$ (d) $\mathrm{S}^V_{\mathrm{U}_0}$

Fig. 12 2-D dataflow diagram, CDF 5/3 wavelet, common steps for all improved schemes.

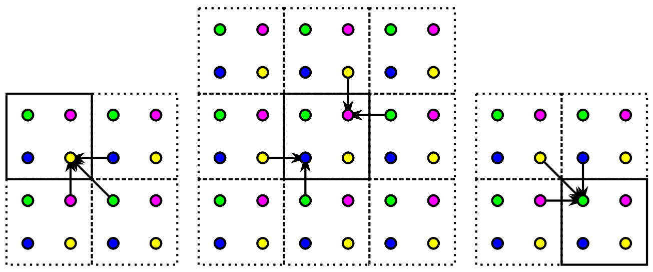

(a) $\mathrm{T}^I_{\mathrm{P}_1}$ (b) $\mathrm{R}^I_{\mathrm{P}_1,\mathrm{U}_1}$ (c) $\mathrm{S}^I_{\mathrm{U}_1}$

Fig. 13 2-D dataflow diagram, CDF 5/3 wavelet, Iwahashi* scheme. The solid box corresponds to the output coefficients.

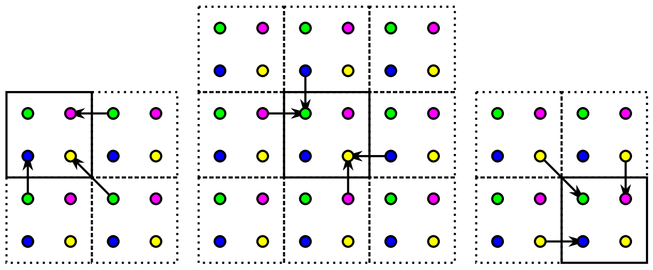

(a) $\mathrm{T}^E_{\mathrm{P}_1}$ (b) $\mathrm{R}^E_{\mathrm{P}_1,\mathrm{U}_1}$ (c) $\mathrm{S}^E_{\mathrm{U}_1}$

Fig. 14 2-D dataflow diagram, CDF 5/3 wavelet, Explosive* scheme. The solid box corresponds to the output coefficients.

where the individual matrices are defined above in the paper. The number of barriers remains the same as for the original Iwahashi scheme. The operations represented by the matrices defined for the Sweldens scheme do not need to be preceded by a barrier. The scheme will be further referred to as Iwahashi*. For the CDF 5/3 wavelet, this scheme is graphically illustrated in Fig. 13.

Similarly, we have employed the same trick in conjunction with the Explosive scheme. This time, the scheme is defined as

$$\mathbf{y} = \mathrm{S}^V_{\mathrm{U}_0}\,\mathrm{S}^H_{\mathrm{U}_0}\mathrm{S}^E_{\mathrm{U}_1} \mid \mathrm{R}^E_{\mathrm{P}_1,\mathrm{U}_1} \mid \mathrm{T}^E_{\mathrm{P}_1} \mid \mathrm{T}^V_{\mathrm{P}_0}\,\mathrm{T}^H_{\mathrm{P}_0}\,\mathbf{x}. \tag{32}$$

Also in this case, the number of barriers remains the same as for the original scheme. Analogously to the previous case, this scheme will be referred to as Explosive*. The dataflow diagram for the CDF 5/3 wavelet is shown in Fig. 14.

As a next step, consider a new construction based on the Monolithic scheme. The same trick can be utilized here as well. In the matrix notation, the newly composed scheme is defined as

$$\mathbf{y} = \mathrm{S}^V_{\mathrm{U}_0}\,\mathrm{S}^H_{\mathrm{U}_0}\,\mathrm{S}_{\mathrm{U}_1} \mid \mathrm{T}^V_{\mathrm{P}_0}\,\mathrm{T}^H_{\mathrm{P}_0}\,\mathrm{T}_{\mathrm{P}_1}\,\mathbf{x}, \tag{33}$$

where the individual matrices are defined above in the text. For the CDF 5/3 wavelet, this scheme is graphically illustrated in Fig. 15. We will label this scheme as Monolithic*.

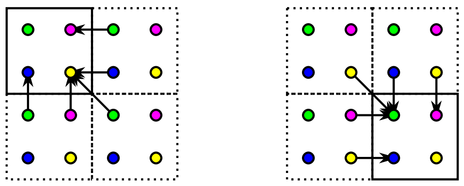

(a) $\mathrm{T}_{\mathrm{P}_1}$ (b) $\mathrm{S}_{\mathrm{U}_1}$

Fig. 15 2-D dataflow diagram, CDF 5/3 wavelet, Monolithic* scheme. The solid box corresponds to the output coefficients.

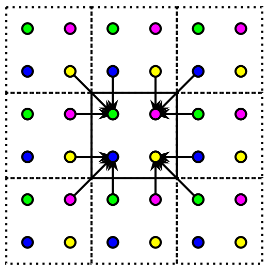

(a) $\mathrm{N}_{\mathrm{P}_1,\mathrm{U}_1}$

Fig. 16 2-D dataflow diagram, CDF 5/3 wavelet, Polyphase* scheme. The solid box corresponds to the output.

The schemes described above are formed such a way that the first lifting step (comprising $\mathrm{P}_1, \mathrm{U}_1$) after the barrier access coefficients of the surrounding threads. The subsequent or preceding steps (comprising $\mathrm{P}_0, \mathrm{U}_0$) read only the local coefficients, which are not accessed by the other threads. Then, the whole sequence can be repeated. Of course, the calculation of transforms consisting of several pairs of lifting steps comprises several such connected schemes.

Finally, we have decided to remove the last explicit barrier, leaving only the initial one in place. The trick lies in the appropriate combination of the Sweldens and Polyphase schemes. This time, the non-separable parts are merged into a joint step $\mathrm{N}_{\mathrm{P}_1,\mathrm{U}_1}$. This step is inherently preceded by a barrier. In case of an initial pair of lifting steps, the barrier at the beginning of the computation is used for this purpose. In more detail, after the input data have been read by each computation unit, the calculations $\mathrm{T}^V_{\mathrm{P}_0}\,\mathrm{T}^H_{\mathrm{P}_0}$ are immediately performed. At this point, the intermediate results can be appropriately shared. This is followed by the initial barrier. Regarding the transforms consisting of several such schemes, the barrier between the connecting schemes is gratefully exploited. In any case, the scheme is thus composed as

$$\mathbf{y} = \mathrm{S}^V_{\mathrm{U}_0}\,\mathrm{S}^H_{\mathrm{U}_0}\,\mathrm{N}_{\mathrm{P}_1,\mathrm{U}_1} \mid \mathrm{T}^V_{\mathrm{P}_0}\,\mathrm{T}^H_{\mathrm{P}_0}\,\mathbf{x} \tag{34}$$

including the discussed barrier. For the CDF 5/3 wavelet, the steps are illustrated in Fig. 16. We will label this scheme as Polyphase*.

For the sake of clarity, the proposed schemes will now be summarized. By reversing the direction of filtering steps in the Iwahashi scheme, the new Explosive scheme was formed. As a next step, the polynomials of the original polyphase matrix were reassembled into a new two-step form. In between the steps, a memory barrier has to be placed. This scheme is denoted as Monolithic. Moreover, the number of arithmetic operations was reduced by splitting the polynomial into two parts. These newly formed polynomials are then employed in appropriate schemes. In this manner, the number of barriers remains unaffected, while the number of operations has been reduced. This simple trick has resulted in the Iwahashi*, Explosive*, Monolithic*, and Polyphase* schemes. Once again, we would like to emphasize that the schemes presented in this paper are general and they are not limited to any specific type of transform.

5 Evaluation

This section analyzes in detail various attributes of the schemes described in the previous sections. Namely, synchronization and memory demands for different wavelets are examined. We realize that such properties do not provide sufficient information on a performance in real environments. For this reason, we are interested in comparing the performance of the discussed schemes on real graphics cards in terms of memory bandwidth in the next section.

The evaluation is presented using the following three wavelets. The first wavelet we have employed is the CDF [1] 5/3 wavelet. This one is used for a lossless compression in the JPEG 2000 compression standard. The lifting scheme is defined by

$$\begin{bmatrix} P(z) \\ U(z) \end{bmatrix} = \begin{bmatrix} -1/2(1+z^{-1}) \\ 1/4(1+z\quad) \end{bmatrix}, \tag{35}$$

and the scaling factor $\zeta = \sqrt{2}$.

As the second wavelet, we have chosen the CDF 9/7 wavelet. In the JPEG 2000 standard, this wavelet is used as a basis for a lossy compression. The underlying scheme is given by

$$\begin{bmatrix} P^{(0)}(z) \\ U^{(0)}(z) \\ P^{(1)}(z) \\ U^{(1)}(z) \end{bmatrix} = \begin{bmatrix} \alpha(1+z^{-1}) \\ \beta(1+z\quad) \\ \gamma(1+z^{-1}) \\ \delta(1+z\quad) \end{bmatrix}, \tag{36}$$

where the $\alpha, \beta, \gamma, \delta$, and the ζ are defined in [2]. Both the CDF wavelets have predict and update convolution operators of degree 1 (two-tap symmetric filters).

The last wavelet included in the comparison is $(4, 4)$ interpolating transform built from the interpolating Deslauriers–Dubuc [8], defined by

$$\begin{bmatrix} P(z) \\ U(z) \end{bmatrix} = \begin{bmatrix} 1/16(z+z^{-2}) - 9/16(1+z^{-1}) \\ 9/32(1+z) - 1/32(z^{-1}+z^{2}) \end{bmatrix}. \tag{37}$$

wavelet	scheme	barriers	operations
CDF 5/3	Sweldens	4	16
	Iwahashi	3	24
	Iwahashi*	3	18
	Explosive	3	24
	Explosive*	3	18
	Monolithic	2	24
	Monolithic*	2	18
	Polyphase	1	63
	Polyphase*	1	23
	Convolution	1	64
CDF 9/7	Sweldens	8	32
	Iwahashi	6	48
	Iwahashi*	6	36
	Explosive	6	48
	Explosive*	6	36
	Monolithic	4	48
	Monolithic*	4	36
	Polyphase	2	126
	Polyphase*	2	46
	Convolution	1	256
DD 13/7	Sweldens	4	32
	Iwahashi	3	64
	Iwahashi*	3	50
	Explosive	3	64
	Explosive*	3	50
	Monolithic	2	64
	Monolithic*	2	50
	Polyphase	1	255
	Polyphase*	1	203
	Convolution	1	256

Table 2 Number of operations and memory barriers examined for various wavelets.

This wavelet is used in Dirac video compression standard. For simplicity, we refer this one to as DD 13/7. The underlying lifting scheme differs from the two previous in employed predict and update convolution operators. These operators now have a degree of 3 instead of 1. Consequently, this difference has resulted in a significantly higher number of arithmetic operations in the case of non-separable filtering steps.

The first examined parameters include the number of arithmetic operations (the scaling steps were omitted) and the number of memory barriers. The schemes presented in this paper can be directly applied on the CDF 5/3 and DD 13/7 transforms, as these comprises only a single pair of lifting steps. The CDF 9/7 transform is computed by two such connected schemes. The comparison is shown in Table 2. Several expectations can be made from the table. On architectures based on serial computation, the schemes should perform accordingly to the number of arithmetic operations. However, on the parallel architectures, the number of employed memory barriers is expected to play an important role. Some of the schemes could benefit from this property.

As can be seen from the referenced table, the Sweldens scheme always leads to the smallest number of operations coupled with the highest number of barriers. The recently proposed Iwahashi scheme reduces the number of barriers by one per one pair of original 1-D lifting steps. Unfortunately, the number of operations is increased at the same time. This increase is particularly noticeable on longer lifting filters, as in the case of DD 13/7 wavelet. The Monolithic scheme further reduces the number of barriers by one per one pair of original steps while keeping the number of operations untouched. In addition to this, the Monolithic* scheme reduces the number of operations. This reduction is most evident on short lifting filters. For instance, in the case of CDF wavelets, the number of operations is reduced to 75 %, whereas in the case of DD 13/7 wavelet, the number of operations is only reduced to 78 %. The number of barriers per one pair of original lifting steps can be even further reduced to a single one by combining all operations into a single step. Such case corresponds to the Polyphase scheme. Unfortunately, the number of operations was increased enormously. For shorter lifting filters, this number can be noticeably reduced using the Polyphase* scheme, in which the number of barriers remains the same. Finally, for lifting factorizations consisting of several pairs of steps, it makes sense to reduce the number of barriers to a single one by using the Convolution scheme. In such case, the number of operations is sadly the highest of all of the schemes.

scheme	barriers	single	double
Sweldens	4	**2**	**3**
Iwahashi	3	3	4
Iwahashi*	3	3	(6) 4
Explosive	3	**2**	**3**
Explosive*	3	**2**	**3**
Monolithic	2	3	6
Monolithic*	2	3	6
Polyphase	**1**	4	(8) 4
Polyphase*	**1**	4	(8) 4

Table 3 Number of memory barriers and local memory cells per quadruple required by the schemes discussed in this paper. Memory cells are given for a single buffering (two barriers) as well as a double buffering (only a single barrier). The numbers in parentheses are valid in the case of connecting schemes. Best features in bold.

scheme	write	read degree-1	read degree-3
Sweldens	$1+4K$	$8K$	$24K$
Iwahashi	$2+4K$	$10K$	$42K$
Iwahashi*	$6K$	$10K$	$42K$
Explosive	$4K$	$10K$	$42K$
Explosive*	$4K$	$10K$	$42K$
Monolithic	$6K$	$10K$	$42K$
Monolithic*	$6K$	$10K$	$42K$
Polyphase	$4K$	$21K$	$117K$
Polyphase*	$4K$	$12K$	$117K$

Table 4 Number of local memory reads and writes for all schemes and wavelets under examination. The K denotes the number of predict/update pairs. The degree-1 polynomials correspond to factorizations of CDF wavelets, whereas degree-3 to DD 13/7.

Other examined parameters included the memory footprint, and number of memory loads/stores. These parameters can be determined from Table 3 and Table 4. All of the numbers are given with respect to the quadruple of coefficients, which usually correspond to a single thread. The number of load (read) operations depends on the length of the lifting operators. For example, the CDF 5/3 and CDF 9/7 factorizations consist of degree-1 convolutional filters. On the contrary, the DD 13/7 consists of degree-3 filters. The number of store (write) operations is independent of the underlying scheme. It may happen that the local memory footprint for the connecting schemes ($K > 1$) differs from the footprint for a single predict/update pair ($K = 1$). These numbers are indicated in the parentheses in Table 3. For clarity, the number of memory barriers is not affected by the improvement proposed in Section Improvements.

6 Performance

To evaluate the considered schemes, we have decided to use high-performance GPUs programmed using the OpenCL framework. In terms of the OpenCL, the schemes are computed using parallel tasks referred to as the kernels. One item from a collection of parallel executions of a kernel is referred to as the work-item or thread. The threads that execute on a single compute unit are grouped into so-called work-groups. The threads in the group execute the same kernel and share local memory. Each work-group can synchronize the threads via memory barriers. Work-groups cannot synchronize with each other. Considering the processing of images, we map overlapping (in order to properly compute the coefficients near tile boundaries) image tiles onto the work-groups. Moreover, each thread is responsible for a single quadruple of transform coefficients (LL, HL, LH, and HH). At the beginning of the computation, the input image is placed in the global memory. The tiles are then transferred into the local memory. After the scheme computation, the resulting coefficients are copied back into the global memory. Such strategy fulfills the definition of a single-loop data processing (therefore without unnecessary data transfers).

One needs to recall that the row–column, block-based, and pipelined methods denote an order in which an entire input image is processed. The row–column approach indicates that all image rows are transformed prior to a transformation of all image columns (or vice versa). Between these two parts, intermediate results are stored in the global memory. However, all of the schemes presented in our paper are implemented as the block-based approaches. This means that the entire input image is split into blocks (the tiles), which are then transformed at once using the local memory for storing the intermediate results. Inside the blocks, some sort of separable scheme can be employed, which essentially corresponds to the row–column approach on a different scale. The block-

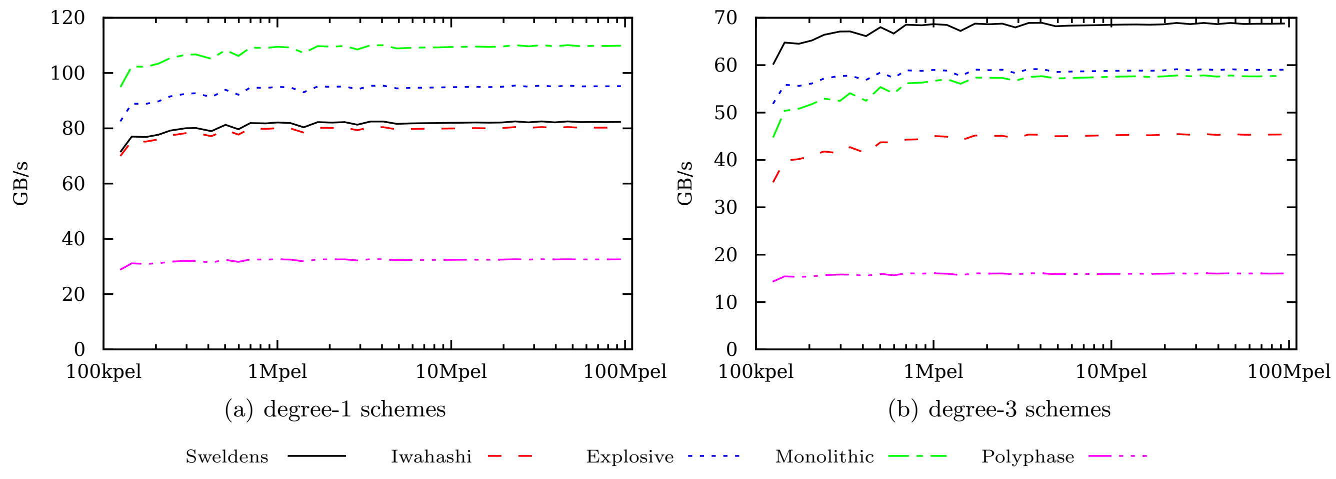


Fig. 17 The baseline schemes on AMD 6970. Evaluation with the degree-1 and degree-3 lifting schemes. Only the performance of a transform code without the memory throughput was measured.

based approaches in various forms were also used in, e.g., [5; 14; 19; 23]. Since the block-based approaches overcome the row–column ones (as shown in [23], or analyzed in [19]), we do not include the classical row–column methods in our performance comparison. Instead, we only compared different schemes employed under the block-based approach. In this context, we would like to make a comment on data transfers between a device and host. Due to the fact that the data are transferred in the same way for all schemes, we measured only a throughput based on a timing of a OpenCL kernels which calculate transforms. Therefore, the transfer times between device and host are not our concern.

The evaluation was performed primarily on two high-end GPUs – AMD Radeon HD 6970 and AMD Radeon HD 5870. Their technical parameters are summarized in Table 5. On both of the cards, variable length VLIW instructions are executed using blocks of 64 threads. In more detail, VLIW instructions can be categorized into several groups (load/store instructions, barrier instructions, control flow instructions and ALU instructions). To utilize whole processing capability, the VLIW instructions should be of maximal length. In other words, as much as possible blocks of independent instructions should be presented in a kernel.

	AMD 6970	AMD 5870
vendor	AMD	AMD
model	Radeon HD 6970	Radeon HD 5870
VLIW length	4	5
multiprocessors	24	20
VLIW processors	384	320
total processors	1 536	1 600
processor clock	880 MHz	850 MHz
performance	2 703 GFLOPS	2 720 GFLOPS
memory	1 GiB GDDR5	1 GiB GDDR5
memory clock	1 375 MHz	1 200 MHz
bandwidth	176 GB/s	154 GB/s
bus width	256-bit	256-bit
local memory	32 KiB	32 KiB

Table 5 Description of the GPUs used for the evaluation.

Several possibilities raised during the implementations of the presented schemes. All of the schemes require several memory cells to interchange the intermediate coefficients. Considering the GPUs, these coefficients can be efficiently stored in the local memory. Unfortunately, it is not possible to rewrite these coefficients using a single memory barrier. As a consequence, two possibilities occur – double buffering using a single memory barrier, and single buffering using two of them. The double buffering increases the memory requirements while maintaining the number of synchronizations. Conversely, the single buffering introduces an addition barrier – separating reading and rewriting of the coefficients. For details, see Table 3. In other words, one can choose whether intermediate results are overwritten in their place using two memory barriers or whether these are written to another location by making use of a single barrier. Moreover, another possibility lies in the method of input and output data delivery. For evaluation purposes, it is possible to completely omit the input and output of data. The transform is not limited by memory bandwidth in this case. For real scenarios, the data can be delivered using the global or texture memory. In our experiments, we chose the latter option.

In the following paragraphs, three fundamental experiments on the described GPUs are presented. The first experiment studies the performance of the baseline schemes mentioned in this paper. The second experiment examines the influence of the improvement proposed in Section Improvements. Finally, the third experiment measures the real performance with CDF 9/7 wavelet and texture memory.

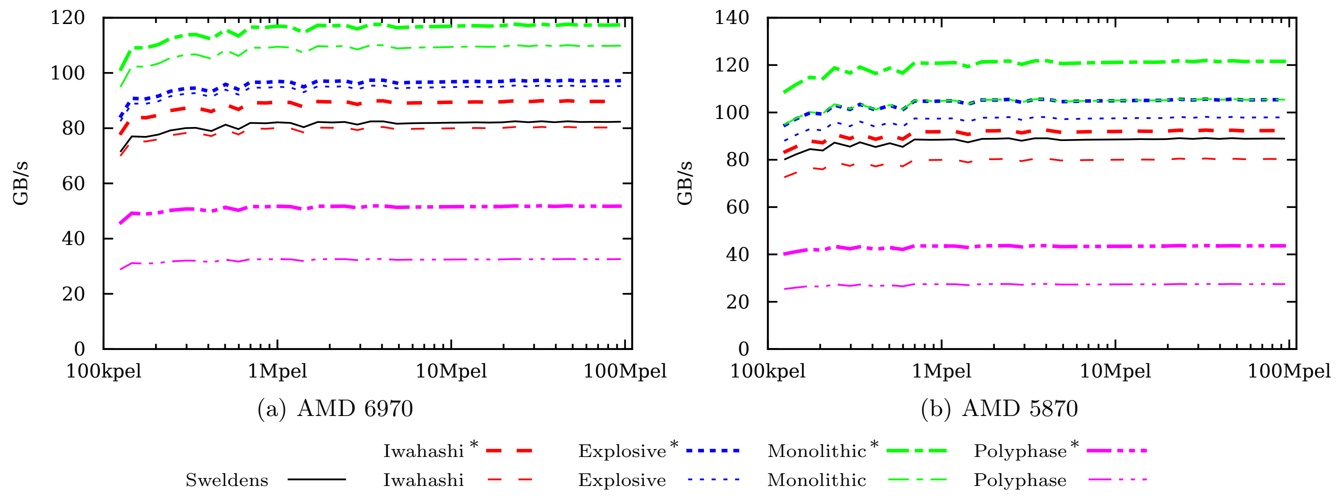


Fig. 18 The schemes on AMD 6970 and AMD 5870. Evaluation with the degree-1 schemes. Only the performance of a transform code without the memory throughput was measured.

In the first experiment, the performance of the baseline schemes (without improvements proposed in Section Improvements) was examined. The measurements were conducted on the AMD 6970 card with two different lifting scheme shapes (degree-1 and degree-3 operators). Only the transform performance was measured, without the influence of memory throughput. The presented results are the average of ten measurements. The results are shown in Fig. 17. One can easily observe a different behavior for short and long lifting operators. For the short operators, the reduction in the number of lifting steps clearly improves the performance. The situation actually corresponds directly to the number of memory barriers. Conversely, in the case of the long operators, the situation is tilted in favor of the number of arithmetic operations. Note that the horizontal axes are in a logarithmic scale. The vertical axes express the transform throughput in GB/s (gigabytes per second).

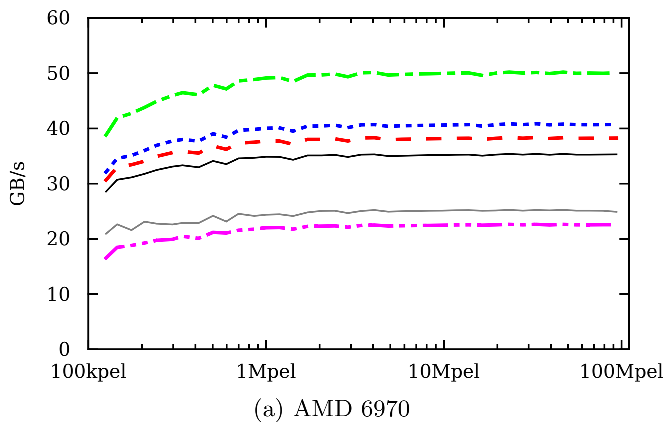
In the second experiment, the contribution of the improvements proposed in Section Improvements was examined. The measurements were performed on both of the cards under the evaluation. This time we have focused on the degree-1 schemes only. As in the previous case, only the transform performance was measured using the average of ten measurements. The results are shown in Fig. 18. As expected, the improvements slightly increase the transform performance. However, the order of the schemes still corresponds to the number of memory barriers. Several schemes perform even worse than the original separable Sweldens scheme – namely, the original Iwahashi and both Polyphase schemes. It is not surprising for the original Polyphase scheme, as this one exhibits quite a high number of operations and load instructions (see Table 4 and Table 2). In case of Polyphase* scheme, the decisive factor was the number of load instructions coupled with a high local memory footprint (see Table 3).

scheme	throughput AMD 6970	throughput AMD 5870
Monolithic*	117.426	121.579
Monolithic	109.865	105.407
Explosive*	97.214	105.344
Explosive	95.263	97.877
Iwahashi*	89.748	92.288
Sweldens	82.336	88.924
Iwahashi	80.284	80.283
Polyphase*	51.776	43.619
Polyphase	32.593	27.462

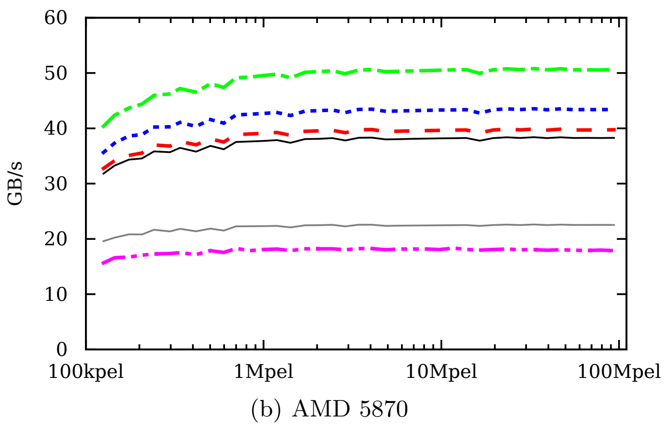
Table 6 The degree-1 schemes on AMD 6970 and AMD 5870. The performance of a transform code without the memory throughput is listed. Values given in GB/s at the end of plots in Fig. 18.

A little surprising is the situation regarding the original Iwahashi scheme. In this case, the scheme contains a relatively high number of operations, wherein there is no additional advantage. For convenience, the values at the end of plots in Fig. 18 are listed in Table 6.

In the last experiment, we were interested in a real performance. This experiment was performed on both of the cards with CDF 9/7 wavelet. The input as well as output raster were supplied by the texture memory. This time, we show only the improved schemes as these always outperform the original ones. The results are shown in Fig. 19. The horizontal axes are in a logarithmic scale, and the vertical ones express the total throughput (limited by the memory). The Convolution and Polyphase schemes exhibit a significantly worse performance, according to the number of operations. In contrast to this, the other schemes perform better as compared to the original separable implementation. More specifically, the Monolithic and Explosive schemes have the very best performance. This fact corresponds to the reduction of the number of steps (and thus the memory barriers).



(a) AMD 6970



(b) AMD 5870

Fig. 19 The improved schemes on AMD 6970 and AMD 5870. Evaluation with the CDF 9/7 wavelet and texture memory was performed.

	AMD 5870			AMD 6970		
scheme	CDF 5/3	CDF 9/7	DD 13/7	CDF 5/3	CDF 9/7	DD 13/7
Sweldens	32.59	32.69	40.00	40.53	40.25	48.50
Iwahashi	34.88	35.90	50.12	41.33	42.33	61.80
Iwahashi*	32.00	34.29	45.83	42.11	40.75	57.00
Explosive	36.88	39.32	49.57	45.14	46.64	60.47
Explosive*	33.79	33.33	55.56	42.42	42.21	65.42
Monolithic	38.18	39.67	51.59	48.57	47.76	64.44
Monolithic*	38.62	37.36	55.79	48.39	44.58	69.49
Polyphase	43.44	43.49	37.76	52.57	54.30	47.21
Polyphase*	31.50	32.43	33.38	41.88	40.58	41.91
Convolution	—	73.79	—	—	83.95	—

Table 7 ALU packing percentage for AMD 6970 and AMD 5870.

The schemes presented in this paper were also subject of examination at other graphics cards under various scenarios. Note that a link to the results is below. Specifically, we tackled these additional cards – NVIDIA Titan X, AMD Fury X, NVIDIA 580, and AMD 290X. Obviously, the proposed non-separable schemes presented in this paper do not exhibit the best performance in all cases. This is especially true for a lifting factorizations employing a longer convolution operators, as is the case of the DD 13/7 wavelet. On the other hand, the proposed schemes seems to be the proven choice for VLIW architectures combined with a short lifting operators, e.g., the CDF 5/3 and CDF 9/7 wavelets.

This point needs to be explained in detail. In general, the number of memory accesses, instruction dependencies, as well as barriers, decreases the ALU utilization, which then degrades the performance. Unlike other architectures, AMD VLIW architectures pack multiple independent instructions into VLIW bundles. Thus, amount and dependencies of instructions between each two neighboring barriers play a significant role in terms of performance. In other words, the number of barriers in VLIW architectures plays a stronger role than in other architectures. Indeed, on the AMD VLIW architectures, code profiling showed that memory barriers limit an average length of VLIW instructions (ALU packing percentage in Table 7), which degrades the performance. The ALU packing percentage refers to the percent of cores in the VLIW processor that are being utilized. On the other (non-VLIW) architectures, the number of local memory accesses (see Table 4) and the number of arithmetic instructions (see Table 2) play a major role. On such architectures, the ALU packing is not measurable due to an absence of the VLIW bundles. For comparative purposes, the ALU packing percentage can be understood as 100 % for all schemes on these architectures.

It should also be interesting to show another measure provided by an OpenCL profiler. In the first instance, consider AMD 5870 card. Such implementations, in which threads need to store less than 5 coefficients (20 bytes) into a local memory, exhibit an occupancy 100 %, as can be seen in Table 8. In detail, 256 threads in work group × 6 work groups result in occupancy 1536 of 1536 threads. This is valid for all these implementations with the exception of Polyphase scheme, in which the occupancy is limited by the number of vector registers, due to an

	AMD 5870			AMD 6970		
scheme	CDF 5/3	CDF 9/7	DD 13/7	CDF 5/3	CDF 9/7	DD 13/7
Sweldens	100.00	100.00	100.00	95.24	95.24	95.24
Iwahashi	100.00	100.00	100.00	95.24	95.24	95.24
Iwahashi*	100.00	* 83.33	100.00	95.24	95.24	95.24
Explosive	100.00	100.00	100.00	95.24	95.24	95.24
Explosive*	100.00	100.00	100.00	95.24	95.24	95.24
Monolithic	* 83.33	* 83.33	* 83.33	95.24	95.24	95.24
Monolithic*	* 83.33	* 83.33	* 83.33	95.24	95.24	95.24
Polyphase	** 83.33	* 50.00	100.00	95.24	* 57.14	95.24
Polyphase*	100.00	* 50.00	100.00	95.24	* 57.14	95.24
Convolution	—	100.00	—	—	95.24	—

Table 8 GPU occupancy measurement for AMD 6970 and AMD 5870. The numbers indicate a percentage. Explanation: * is limited by a local memory (LDS), ** by registers (VGPR).

optimizing compiler. For AMD 6970, due to the use of 256 threads in work-groups and due to maximal number 1344 of threads in multiprocesors, implementations exhibit only an occupancy 95.24 % (256 threads in work group × 5 work groups = 1280 of 1344). On the other hand, considering implementations in which threads need to store less than 7 coefficients (28 bytes) into a local memory, the occupancy is not limited by a size of a local memory.

In summary, we can conclude that the reduction in lifting steps can improve performance, at least on some platforms. This is documented by measurements in Figs. 17, 18, and 19. It turned out, however, that such optimization makes sense only for a short lifting operators (exemplary, degree-1 lifting filters).

For the sake of completeness, it should be noted that the improvement proposed in Section Improvements can be also applied on the Convolution. Doing so, the scheme achieves a slightly better performance. However, we understand the Convolution scheme as the reference method. For this reason, we leave it unimproved. Eventually, the proposed improvement makes no sense in conjunction with the Sweldens scheme.

All the source codes used in this article together with all the results are available in a repository on the website of the authors' affiliation.[1]

7 Conclusions

In this paper, we have proposed several non-separable lifting schemes for the calculation of the discrete wavelet transform. The proposed schemes produce exactly the same results as the commonly used separable lifting scheme. Using our schemes, the transform can be computed in a smaller number of steps. On parallel architectures, this property has resulted in a smaller number of synchronizations.

Namely, we have proposed two-step 2-D lifting scheme compatible to the commonly used four-step separable one. Unlike the separable scheme, the proposed scheme consists of spatial predict and update operators. Since the number of the lifting steps was halved, our scheme reduces also the number of memory barriers, which form a major bottleneck on parallel architectures. In addition, we have proposed the three-step scheme reducing the memory access overhead. For a moment, let K denote the number of predict-update pairs. In absolute numbers, the original separable scheme requires to write $1 + 4K$ coefficients per predict/update pair, whereas our three-step scheme requires $4K$ coefficients only. Additionally, the proposed two-step scheme requires three memory cells per thread, whereas the proposed three-step scheme requires two cells only (same as the separable scheme). Finally, we have proposed an improvement usable for all non-separable scheme, including the already known ones. This improvement significantly reduces the number of arithmetic operations. More specifically, the original non-separable schemes require 24 arithmetic operations for CDF 5/3 wavelet, whereas the improved variants require 18 operations only for the same case. Even greater savings are achieved in the case of a non-factorized polyphase matrix (same as the convolution for the CDF 5/3 wavelet). In this case, the proposed improvement reduces the number of operations from 63 to 23. All of the proposed schemes are general and can be used in conjunction with any discrete wavelet transform.

The proposed schemes were subject to performance measurements. In experiments on the two selected high-end GPUs (AMD Radeon HD 6970 and 5870), the proposed schemes outperform all the others for short lifting filters. This includes the well-known CDF 5/3 and CDF 9/7 wavelets, employed, e.g., in JPEG 2000 compression standard.

Future work, we would like to do, consists of extensions to multi-dimensional systems, and extensions to another subband transforms.

Acknowledgements This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science – LQ1602.

[1] `http://www.fit.vutbr.cz/research/prod/?id=483`

References

1. Cohen A, Daubechies I, Feauveau JC (1992) Biorthogonal bases of compactly supported wavelets. Communications on Pure and Applied Mathematics 45(5):485–560, DOI 10.1002/cpa.3160450502
2. Daubechies I, Sweldens W (1998) Factoring wavelet transforms into lifting steps. Journal of Fourier Analysis and Applications 4(3):247–269, DOI 10.1007/BF02476026
3. Mallat S (1989) A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(7):674–693, DOI 10.1109/34.192463
4. Matela J (2009) GPU-based DWT acceleration for JPEG2000. In: Annual Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, pp 136–143
5. Arguello F, Heras DB, Boo M, Lamas-Rodriguez J (2012) The split-and-merge method in general purpose computation on GPUs. Parallel Computing 38(6–7):277–288, DOI 10.1016/j.parco.2012.03.003
6. Mallat S (2009) A Wavelet Tour of Signal Processing: The Sparse Way. With contributions from Gabriel Peyre, 3rd edn. Academic Press
7. Strang G, Nguyen T (1997) Wavelets and Filter Banks. Wellesley-Cambridge Press
8. Sweldens W (1996) The lifting scheme: A custom-design construction of biorthogonal wavelets. Applied and Computational Harmonic Analysis 3(2):186–200
9. Rauber T, Runger G (2013) Parallel Programming: for Multicore and Cluster Systems. Springer, DOI 10.1007/978-3-642-37801-0
10. Franco J, Bernabe G, Fernandez J, Ujaldon M (2011) The 2D wavelet transform on emerging architectures: GPUs and multicores. Journal of Real-Time Image Processing 7(3):145–152, DOI 10.1007/s11554-011-0224-7
11. Tenllado C, Lario R, Prieto M, Tirado F (2004) The 2D discrete wavelet transform on programmable graphics hardware. In: Visualization, Imaging and Image Processing Conference 2004, pp 808–813
12. Tenllado C, Setoain J, Prieto M, Pinuel L, Tirado F (2008) Parallel implementation of the 2D discrete wavelet transform on graphics processing units: Filter bank versus lifting. IEEE Transactions on Parallel and Distributed Systems 19(3):299–310, DOI 10.1109/TPDS.2007.70716
13. Franco J, Bernabe G, Fernandez J, Acacio M (2009) A parallel implementation of the 2D wavelet transform using CUDA. In: 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp 111–118, DOI 10.1109/PDP.2009.40
14. Blazewicz M, Ciznicki M, Kopta P, Kurowski K, Lichocki P (2012) Two-dimensional discrete wavelet transform on large images for hybrid computing architectures: GPU and CELL. In: Euro-Par 2011: Parallel Processing Workshops, LNCS, vol 7155, Springer, pp 481–490, DOI 10.1007/978-3-642-29737-3_53
15. Galiano V, Lopez O, Malumbres M, Migallon H (2011) Improving the discrete wavelet transform computation from multicore to GPU-based algorithms. In: Proceedings of the 11th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE), pp 544–555
16. Galiano V, Lopez O, Malumbres M, Migallon H (2013) Parallel strategies for 2D discrete wavelet transform in shared memory systems and GPUs. The Journal of Supercomputing 64(1):4–16, DOI 10.1007/s11227-012-0750-5
17. van der Laan W, Roerdink JBTM, Jalba A (2009) Accelerating wavelet-based video coding on graphics hardware using CUDA. In: Proceedings of 6th International Symposium on Image and Signal Processing and Analysis (ISPA), pp 608–613, DOI 10.1109/ISPA.2009.5297658
18. van der Laan WJ, Jalba AC, Roerdink JBTM (2011) Accelerating wavelet lifting on graphics hardware using CUDA. IEEE Transactions on Parallel and Distributed Systems 22(1):132–146, DOI 10.1109/TPDS.2010.143
19. Song C, Li Y, Guo J, Lei J (2014) Block-based two-dimensional wavelet transform running on graphics processing unit. IET Computers Digital Techniques 8(5):229–236, DOI 10.1049/iet-cdt.2013.0141
20. Iwahashi M (2007) Four-band decomposition module with minimum rounding operations. Electronics Letters 43(6):27–28, DOI 10.1049/el:20073479
21. Iwahashi M, Kiya H (2010) A new lifting structure of non separable 2D DWT with compatibility to JPEG 2000. In: Acoustics Speech and Signal Processing (ICASSP), pp 1306–1309, DOI 10.1109/ICASSP.2010.5495427
22. Iwahashi M, Kiya H (2013) Non separable two dimensional discrete wavelet transform for image signals. In: Discrete Wavelet Transforms – A Compendium of New Approaches and Recent Applications, InTech, DOI 10.5772/51199
23. Kula M, Barina D, Zemcik P (2016) Block-based approach to 2-D wavelet transform on GPUs. In: International Conference on Information Technology – New Generations (ITNG), Springer, pp 643–653, DOI 10.1007/978-3-319-32467-8_56

Author Biographies

David Barina received the PhD degree at the Faculty of Information Technology, Brno University of Technology, Czech Republic. He is currently a member of the Graph@FIT group at the Department of Computer Graphics and Multimedia at FIT, Brno University of Technology. His research interests include wavelets and fast algorithms in signal and image processing.

Michal Kula received the MS degree at the Faculty of Information Technology, Brno University of Technology, Czech Republic. He is currently a PhD student and member of the Graph@FIT group at the Department of Computer Graphics and Multimedia at FIT, Brno University of Technology.

Pavel Zemcik received his PhD degree from the Faculty of Electrical Engineering and Computer Science, Brno University of Technology, Czech Republic. He works as a full professor, dean and member of the Graph@FIT group at the Department of Computer Graphics and Multimedia at FIT, Brno University of Technology. His interests include acceleration of computer vision and graphics algorithms, programmable hardware and also applications.