New Non-Separable Lifting Scheme for Images

Michal Kula, David Barina, Pavel Zemcik
Faculty of Information Technology
Brno University of Technology
Czech Republic
e-mail:{ikula,ibarina,zemcik}@fit.vutbr.cz

***Abstract*—We have reduced the number of lifting steps in the calculation of the two-dimensional discrete wavelet transform by factoring the underlying lifting scheme into a new spatial form. Compared with recently proposed non-separable structure, we have reduced also the number of operations. Our scheme is primarily designed for CDF 5/3 and CDF 9/7 wavelets employed in JPEG 2000 image compression standard. In the result, our scheme requires only two steps for 2-D CDF 5/3 transform compared to four steps in the original separable form or three steps in the recent non-separable scheme.**

Keywords-discrete wavelet transform, lifting scheme

I. Introduction

The discrete wavelet transform (DWT) is a signal-processing tool suitable as a basis for sophisticated compression algorithms. Particularly, JPEG 2000 is an image coding system based on such compression technique. JPEG 2000 is the only accepted compression format for Digital Cinema conforming to Digital Cinema Initiatives (DCI) specification.

In this paper, we reduce the number of lifting steps and memory barriers in the lifting scheme upon which DWT is built. This is very convenient for massively-parallel architectures [1] where the lifting steps are evaluated from the inputs to outputs in parallel. On these architectures, a delay of output signals is essentially determined by the number of the synchronization points. In comparison to the recently proposed non-separable lifting scheme [2], we have also reduced the number of arithmetic operations. Our work is directly focused on CDF 5/3 and CDF 9/7 wavelets employed in JPEG 2000 coding system.

II. Lifting Scheme

In this paper, we employ the well-known z-transform notation for the description of FIR filters. The transfer function of the two-dimensional FIR filter h_{k_m,k_n} is defined as

$$H(z_m, z_n) = \sum_{k_m=-\infty}^{\infty} \sum_{k_n=-\infty}^{\infty} h_{k_m,k_n} z_m^{-k_m} z_n^{-k_n}, \quad (1)$$

where m refers to the horizontal axis and n to the vertical one. Moreover, to keep consistency with [3], [4], the $H^*(z_m, z_n) = H(z_n, z_m)$ denotes a filter transposed to the $H(z_m, z_n)$. Due to the limited place, 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.

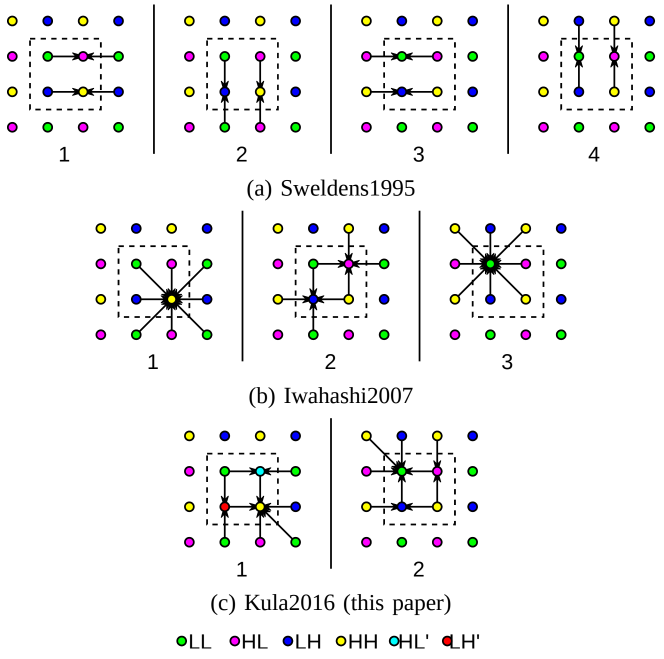


Figure 1. 2-D data-flow graphs of the discussed schemes. The order of the lifting steps is determined by the bottom numbers. The vertical lines indicate necessary memory barriers.

The discrete wavelet transform has undergone a gradual development in the last few decades. Initially, Daubechies [5] constructed orthonormal bases of compactly supported wavelets. Subsequently, Cohen *et al.* [6] developed several families of symmetric biorthogonal wavelet bases referred to as CDF biorthogonal wavelets. In addition to such wavelets, Mallat [7] demonstrated the wavelet representation of images computed with a pyramidal algorithm based on convolutions with quadrature mirror filters. Finally, Sweldens [8], [9] presented the lifting scheme which sped up such decomposition. He showed how any discrete wavelet transform can be decomposed into a sequence of simple filtering steps.

When we combine this scheme with Mallat's 2-D decomposition, we obtain a quadruple of wavelet coefficients (LL, HL, LH, HH). Now we focus on CDF 5/3 wavelet using a factorization as specified by JPEG 2000 standard. In this case, the two-dimensional transform employing such wavelet consists of two horizontal and two vertical lifting steps. For better understanding, this scheme is graphically illustrated in Fig. 1a (referred to as

Sweldens1995). The order of these steps is limited, but not strictly fixed. In this paper, we consider these steps as (1) horizontal step resulting into HL, HH; (2) horizontal step into LL, LH; (3) vertical step into LH, HH; and (4) vertical step into LL, HL. Considering the parallel evaluation and the data dependencies, a memory barrier have to be inserted between each two steps. These barriers form the major bottleneck of the overall computation. The entire 2-D transform is composed of 16 arithmetic (multiply-accumulate) operations per output quadruple of coefficients contained in four lifting steps. Excluding the implicit barriers at the beginning and at the end of the scheme, three memory barriers are required in between these steps in total. As the scheme uses two-tap FIR filters

$$\begin{bmatrix} \mathrm{P} \\ \mathrm{U} \end{bmatrix} = \begin{bmatrix} P(z) \\ U(z) \end{bmatrix} = \begin{bmatrix} \alpha(1+z \quad) \\ \beta(1+z^{-1}) \end{bmatrix}, \tag{2}$$

the most complex operation is calculated over three operands. Note that α coefficient is used in step 1 and 3, β in step 2 and 4. The scheme for CDF 9/7 comprises two such connected transforms. Note that the schemes presented in this paper works as well for asymmetric filters.

Since the birth of the lifting scheme, its efficient realization has been studied in many papers focused on various platforms. Chrysafis *et al.* [10] modified the 1-D scheme in the way that it enabled serial (online, pipelined) signal processing. Their approach was extended into 2-D in many papers, e.g. [11]. So far, the 2-D image must be processed twice – once in the horizontal and once in the vertical direction. In [11], the author focused on the 2-D transform in that he fused the vertical and horizontal pass into a single loop. However, it is still possible to identify the vertical and horizontal filtering steps.

Recently, Iwahashi *et al.* [2]–[4] presented the non-separable lifting scheme employing genuine spatial filtering steps

$$\begin{bmatrix} \mathrm{P} \\ \mathrm{P}^* \\ \mathrm{PP}^* \end{bmatrix} = \begin{bmatrix} \alpha(1+z_m) \\ \alpha(1+z_n) \\ \alpha^2(1+z_m+z_n+z_m z_n) \end{bmatrix}, \tag{3}$$

$$\begin{bmatrix} \mathrm{U} \\ \mathrm{U}^* \\ \mathrm{UU}^* \end{bmatrix} = \begin{bmatrix} \beta(1+z_m^{-1}) \\ \beta(1+z_n^{-1}) \\ \beta^2(1+z_m^{-1}+z_n^{-1}+z_m^{-1}z_n^{-1}) \end{bmatrix}. \tag{4}$$

In this scheme, it is no longer possible to distinguish the vertical and horizontal filtering. In their construction, the authors derived the non-separable 2-D scheme for CDF 5/3 and subsequently CDF 9/7 transforms. As an initial step of CDF 5/3 transform, the input signal is split into quadruples (LL, HL, LH, HH).

$$\begin{aligned} \mathrm{x} &= [\mathrm{X}_{00} \quad \mathrm{X}_{10} \quad \mathrm{X}_{01} \quad \mathrm{X}_{11}]^T \\ \mathrm{y} &= [\mathrm{LL} \quad \mathrm{HL} \quad \mathrm{LH} \quad \mathrm{HH}]^T \end{aligned} \tag{5}$$

Then, spatial lifting steps leading to the calculation HH coefficients are performed. This is followed by parallel computation of the HL and LH coefficients. In the third step, the LL coefficient is updated. Formally, these steps are described as $\mathrm{y} = N_3 N_2 N_1 \mathrm{x}$, where $\mathrm{N}_3\mathrm{N}_2\mathrm{N}_1$ are compressed into the matrix

$$N = \begin{bmatrix} 1 & \mathrm{U} & \mathrm{U}^* & -\mathrm{UU}^* \\ \mathrm{P} & 1 & 0 & \mathrm{U}^* \\ \mathrm{P}^* & 0 & 1 & \mathrm{U} \\ \mathrm{PP}^* & \mathrm{P}^* & \mathrm{P} & 1 \end{bmatrix}. \tag{6}$$

Note that the compressed notation is incorrect, however, used by the authors of [2]–[4]. The scheme is graphically illustrated in Fig. 1b (referred to as Iwahashi2007). As in the original scheme, a memory barrier must be inserted between each two steps. As the result, such scheme consists of 24 arithmetic operations in three lifting steps separated by two explicit memory barriers. The most complex operation is calculated over 9 operands which leads to a performance issue. (This is because the number of operands is proportional to the data path with the maximum delay.) Again, the scheme for CDF 9/7 comprises two such connected transforms.

Since this work is based on our previous work in [13], it should be explained what the difference between this work and previous work is. In [13], we have presented a block-based method intended for graphics cards employing a scheme foregoing the scheme proposed in this paper. In this paper, we further reduce the number of arithmetic operations.

III. Proposed Method

Motivated by the work of Iwahashi *et al.* [4], we have reorganized the elementary lifting filters in order to obtain a highly parallelizable scheme. The main purpose of this modification is to minimize the number of memory barriers that slow down the calculation. As a result, we get several 2-D FIR filters.

The scheme we formed is composed of elementary filters $\mathrm{P} = \mathrm{P}_0 + \mathrm{P}_1$, and $\mathrm{U} = \mathrm{U}_0 + \mathrm{U}_1$ given by

$$\begin{bmatrix} \mathrm{P}_0 \\ \mathrm{P}_1 \\ \mathrm{U}_0 \\ \mathrm{U}_1 \end{bmatrix} = \begin{bmatrix} P_0(z_m, z_n) \\ P_1(z_m, z_n) \\ U_0(z_m, z_n) \\ U_1(z_m, z_n) \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha \\ \beta \\ \beta \end{bmatrix} \begin{bmatrix} 1 \\ z_m \\ 1 \\ z_m^{-1} \end{bmatrix}, \tag{7}$$

where α, β denote filter parameters.

The filters above are assembled into more complex operations. Our scheme consists of two halves between which a memory barrier is placed. The first half of the scheme uses the filters derived from $P(z_m, z_n)$. Similarly, the second half uses the filters derived from $U(z_m, z_n)$.

$$\begin{bmatrix} \mathrm{P} \\ \mathrm{P}_0 \\ \mathrm{P}_0^* \\ \mathrm{P}_1 \\ \mathrm{P}_1^* \\ \mathrm{P}_1\mathrm{P}_1^* \end{bmatrix} = \begin{bmatrix} \alpha(1+z_m) \\ \alpha \\ \alpha \\ \alpha z_m \\ \alpha z_n \\ \alpha^2 z_m z_n \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \mathrm{U} \\ \mathrm{U}_0 \\ \mathrm{U}_0^* \\ \mathrm{U}_1 \\ \mathrm{U}_1^* \\ \mathrm{U}_1\mathrm{U}_1^* \end{bmatrix} = \begin{bmatrix} \beta(1+z_m^{-1}) \\ \beta \\ \beta \\ \beta z_m^{-1} \\ \beta z_n^{-1} \\ \beta^2 z_m^{-1} z_n^{-1} \end{bmatrix} \tag{9}$$

Finally, our scheme is composed of four operators referred to as S_1 to S_4. Between the second S_2 and third S_3 operator, the memory barrier must be inserted in order to properly exchange intermediate results. Thus, S_1 and S_2 form the first lifting step and S_3 and S_4 form the second one. Note that it is also possible to rewrite our scheme using six operators instead of four. It would be also possible to rewrite the scheme with just two operators, however, it is not possible to capture a retention of intermediate results in such case. Additionally, our scheme requires the induction of two auxiliary variables (the intermediate results) per each quadruple of coefficients LL, HL, LH, and HH. These auxiliary variables are denoted as HL′, LH′. Their initial as well as final values are not important. The scheme $\mathrm{y} = S_4\,S_3\,S_2\,S_1\mathrm{x}$ describes the relation between input x and output y vectors. Note that in practical realizations, each single computing unit (e.g., thread) can be responsible of one such vector. The vectors are given by

$$\begin{aligned} \mathrm{x} &= [\mathrm{X}_{00} \quad \mathrm{X}_{10} \quad \mathrm{X}_{01} \quad \mathrm{X}_{11} \quad \mathrm{HL}' \quad \mathrm{LH}']^{\mathrm{T}}, \\ \mathrm{y} &= [\mathrm{LL} \quad \mathrm{HL} \quad \mathrm{LH} \quad \mathrm{HH} \quad \mathrm{HL}' \quad \mathrm{LH}']^{\mathrm{T}}. \end{aligned} \tag{10}$$

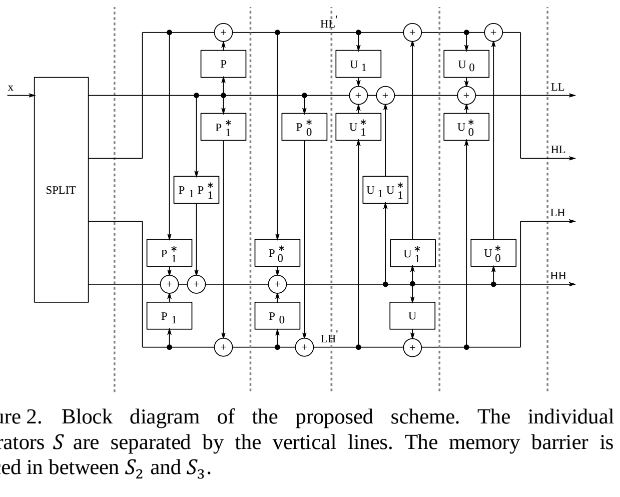


Figure 2. Block diagram of the proposed scheme. The individual operators S are separated by the vertical lines. The memory barrier is placed in between S_2 and S_3.

Regarding this notation, the individual steps are defined as follows. For better understanding, the hypothetical signal-processing block diagram of this scheme is shown in Fig. 2. In addition, the operations are graphically illustrated in Fig. 1c (referred to as Kula2016). The figure shows that our scheme is based on the Iwahashi2007 scheme. However, unlike the Iwahashi2007, we have broken the middle step into two parts and merged these parts into remaining two steps. Note that operators S_1 and S_2 are represented by the first lifting step and operators S_2 and S_3 by the second one. One can easily verify the correctness of our scheme by comparing the top left 4×4 submatrix of $S_4S_3S_2S_1$ (which corresponds to LL, HL, LH, and HH) with the matrix $N_3N_2N_1$.

TABLE I. PARAMETERS OF THE DISCUSSED 2-D LIFTING SCHEMES VALID FOR CDF 5/3 WAVELET.

Scheme	Steps	Operations	Operands	Memory
Sweldens1995	4	16	3	4
Iwahashi2007	3	24	9	4
Kula2016 (this paper)	2	18	4	6

Compared with [4], the total number of operations has been reduced from 24 to 18 for the CDF 5/3 wavelet. The calculation of CDF 9/7 transform comprises two such connected transforms between them another barrier is placed. In total, such a calculation contains three explicit memory barriers.

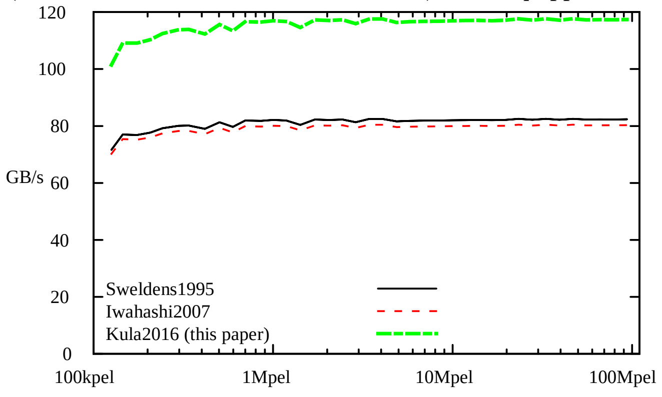
IV. EVALUATION

Quantitative comparison for CDF 5/3 wavelet of all the methods discussed is provided in Table I. The columns describe: number of lifting steps, number of arithmetic operations, maximum number of operands per the lifting step result (the complexity of steps), and number of memory cells per the coefficient quadruple (inclusive). For CDF 9/7 wavelet, the number of lifting steps and thus the number of operations must be doubled. In general, the schemes can be used for any lifting factorization with two-tap filters.

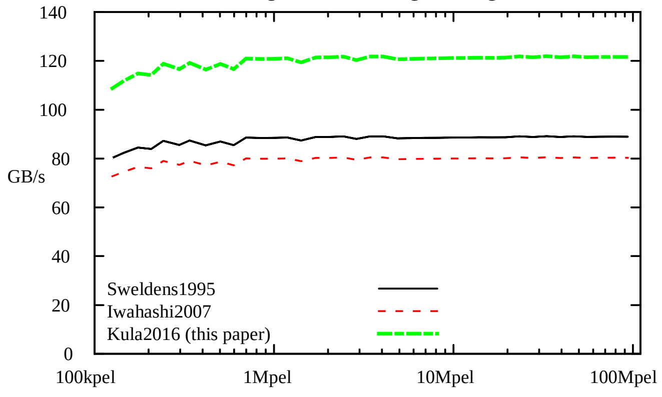
The original Sweldens1995 scheme provides the best choice in terms of arithmetic operands as well as their complexity. However, it requires three explicit synchronization points (memory barriers) for CDF 5/3 wavelet. This can be an issue for parallel processing. The recently proposed Iwahashi2007 scheme uses the highest number of operations of all schemes. On the other hand, it requires only two synchronizations for CDF 5/3 wavelet and does not need any additional memory. In numbers, this scheme reduces the number of lifting steps to 75 %. Finally, the proposed Kula2016 scheme provides a trade-off in the number of operations. Moreover, for CDF 5/3 wavelet, only one barrier is needed for its realization. In comparison to the original scheme, this scheme reduces the number of lifting steps to 50 % only.

To evaluate the proposed scheme, we decided to use high-performance GPUs programmed using the OpenCL framework. Considering the image processing, we map overlapping image tiles onto work-groups. Each thread is responsible for a single quadrature of transform coefficients (LL, HL, LH, and HH). At the beginning of the computation, the input image is placed into global memory. The tiles are then transferred into local memory. After the computation, the results are copied back into the global memory.

The evaluation was performed on two high-end GPUs (AMD Radeon HD 6970 and HD 5870), both equipped with



(a)AMD6970



(b)AMD5870

Figure 3. A transform performance (without the memory throughput) on AMD 6970 and AMD 5870.

1 GB GDDR5 memory. The AMD Radeon HD 6970 contains 1 536 processors (384 VLIW4 processors) clocked at 880 MHz (memory at 1 375 MHz). The AMD Radeon HD 5870 comprises 1 600 processors (320 VLIW5) clocked at 850 MHz (memory 1 200 MHz). On both of the cards, variable length VLIW instructions are executed using blocks of 64 threads.

We have examined the performance of the schemes. 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. 3. The horizontal axes are in a logarithmic scale and the vertical ones express the pure transform throughput. The Iwahashi2007 scheme performs even worse than the original separable Sweldens1995 scheme. It is not surprising, as the scheme exhibits highest number of operations.

V. Conclusions

We have factored the 2-D lifting scheme of CDF 5/3 wavelet into two spatial lifting steps. Our scheme reduces the number of lifting steps and memory barriers. In general, the barriers are the major bottleneck of parallel computations. Compared to recently proposed non-separable structure, our scheme also reduces the number of arithmetic operations. The scheme for CDF 9/7 can be obtained as two such connected schemes.

A key idea behind the factorization is to group corresponding one-dimensional lifting steps into joint two-dimensional non-separable form. Future work, we would like to do, comprises adaptation to other wavelets, possibly in more dimensions.

Acknowledgment

This work was supported by the TACR Competence Centres project V3C – Visual Computing Competence Center (no. TE01020415), and 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.

References

[1] T. Rauber and G. Runger, Parallel Programming: for Multicore and Cluster Systems. Springer, 2013.

[2] M. Iwahashi, “Four-band decomposition module with minimum rounding operations,” Electronics Letters, vol. 43, no. 6, pp. 27–28, 2007.

[3] M. Iwahashi and H. Kiya, “A new lifting structure of non separable 2D DWT with compatibility to JPEG 2000,” in Acoustics Speech and Signal Processing (ICASSP), 2010, pp. 1306–1309.

[4] ——, “Non separable two dimensional discrete wavelet transform for image signals,” in Discrete Wavelet Transforms – A Compendium of New Approaches and Recent Applications. InTech, 2013.

[5] I. Daubechies, “Orthonormal bases of compactly supported wavelets,” Communications on Pure and Applied Mathematics, vol. 41, no. 7, pp. 909–996, 1988.

[6] A. Cohen, I. Daubechies, and J.-C. Feauveau, “Biorthogonal bases of compactly supported wavelets,” Communications on Pure and Applied Mathematics, vol. 45, no. 5, pp. 485–560, 1992.

[7] S. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 674–693, 1989.

[8] W. Sweldens, “The lifting scheme: A custom-design construction of biorthogonal wavelets,” Applied and Computational Harmonic Analysis, vol. 3, no. 2, pp. 186–200, 1996.

[9] I. Daubechies and W. Sweldens, “Factoring wavelet transforms into lifting steps,” Journal of Fourier Analysis and Applications, vol. 4, no. 3, pp. 247–269, 1998.

[10] C. Chrysafis and A. Ortega, “Minimum memory implementations of the lifting scheme,” in Proceedings of SPIE, Wavelet Applications in Signal and Image Processing VIII, ser. SPIE, vol. 4119, 2000, pp. 313–324.

[11] S. Chatterjee and C. D. Brooks, “Cache-efficient wavelet lifting in JPEG 2000,” in Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), vol. 1, 2002, pp. 797–800.

[12] R. Kutil, “A single-loop approach to SIMD parallelization of 2-D wavelet lifting,” in Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2006, pp. 413–420.

[13] M. Kula, D. Barina, and P. Zemcik, “Block-based approach to 2-D wavelet transform on GPUs,” in International Conference on Information Technology – New Generations (ITNG). Springer, 2016, pp. 643–65