Simple Signal Extension Method for Discrete Wavelet Transform

David Barina, Pavel Zemcik, Michal Kula
Brno University of Technology
Czech Republic
{ibarina,zemcik,ikula}@fit.vutbr.cz

***Abstract*—Discrete wavelet transform of finite-length signals must necessarily handle the signal boundaries. The state-of-the-art approaches treat such boundaries in a complicated and inflexible way, using special prolog or epilog phases. This holds true in particular for images decomposed into a number of scales, exemplary in JPEG 2000 coding system. In this paper, the state-of-the-art approaches are extended to perform the treatment using a compact streaming core, possibly in multi-scale fashion. We present the core focused on CDF 5/3 wavelet and the symmetric border extension method, both employed in the JPEG 2000. As a result of our work, every input sample is visited only once, while the results are produced immediately, i.e. without buffering.**

***Index Terms*—Discrete wavelet transform, lifting scheme**

I. Introduction

The discrete wavelet transform (DWT) is the signal-processing transform suitable as a basis for sophisticated compression algorithms. Particularly, JPEG 2000 is an image coding system based on such compression technique. The transform results in two subbands, corresponding to two FIR filters (a low-pass and high-pass ones). With focus on computing demands, the transform is most commonly calculated using the lifting scheme. This scheme reduces the number of arithmetic operations as compared with the original filter bank. Considering finite signals, some special treatment of signal boundaries is unavoidable in both cases. This induces special prolog and epilog phases, which cause delays at the beginning and end of the processing. Such scenario breaks the ability for a continuous stream processing, especially considering the multi-scale decomposition.

In this paper, we propose a solution consisting of a linear mapping of input coefficients onto output ones. This mapping is built above the lifting scheme. Moreover, the mapping is referred to as the core in the rest of the paper. The key idea is exploiting of the possibility of appropriate adjustments of the core, when it processes close to the signal boundaries. For this reason, we speak about the mutable core. The solution is demonstrated using the CDF 5/3 wavelet (used e.g. in JPEG 2000).

The rest of the paper is organized as follows. Section Related Work reviews the state of the art, especially the lifting scheme. Section Proposed Method proposes the core and the treatment of signal boundaries. The purpose of Section Discussion is to state our interpretations and opinions. Finally, Section Conclusions summarizes the work.

II. Related Work

The discrete wavelet transform has undergone gradual development in the last few decades. As a key advance in image processing, Cohen–Daubechies–Feauveau (CDF) [1] biorthogonal wavelets provided several families of symmetric wavelet bases. Afterwards, W. Sweldens [2]–[4] presented the lifting scheme which sped up such decomposition. Following his work, any discrete wavelet transform can be decomposed into a sequence of simple filtering steps (lifting steps). Readers not closely familiar to the DWT are referred to the excellent book [5] by S. Mallat.

The polyphase matrix [4], [6], [7] is a convenient tool to express the transform filters. The lifting scheme factorizes this matrix into a series of shorter filterings. In detail, the polyphase matrix can be factorized [8], so that

$$P(z) = \prod_{i=0}^{L-1} \left\{ \begin{bmatrix} 1 & S_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T_i(z) & 1 \end{bmatrix} \right\} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}, \quad (1)$$

where K is a non-zero constant, and polynomials $S_i(z), T_i(z)$ represent the individual lifting steps. Focusing on the CDF 5/3 wavelet as an example, the forward transform in can be expressed [4] by the dual polyphase matrix

$$\tilde{P}(z) = \begin{bmatrix} 1 & \alpha\left(1+z^{-1}\right) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta\left(1+z\right) & 1 \end{bmatrix} \begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix}, \quad (2)$$

where α, β are real constants, and the ζ is called the scaling factor. For details about the lifting scheme, see [3], [4]. Unlike a convolution scheme, the lifting allows [9] formulation of the transforms mapping integers to integers.

To keep the total number of wavelet coefficients equal to the number of input samples, symmetric border extension is widely used. The symmetric border extension method assumes that the input image can be recovered outside its original area by symmetric replication of boundary values. A special variant (periodic symmetric extension) of this extension is employed in JPEG 2000 standard.

The treatment of the symmetric (or any other) border extension requires special prolog and epilog phases. Considering the contemporary solutions, these phases cause increased delays at the beginning and end of the data processing. Consequently, these phases break the stream processing. Although this issue seems insignificant in single-level transform, it becomes a big problem considering the multi-scale processing.

The paper is mainly focused on contemporary processors equipped with some sort of CPU cache. Inherently, the data processing should be performed in a single loop. Excellent introduction to this topic can be found in [10].

Originally, the problem of efficient implementation of the 1-D lifting scheme was addressed in [11] by Ch. Chrysafis and A. Ortega. Their general approach transforms the input data in a single loop. Nonetheless, this is essentially the same method as the on-line or pipelined method mentioned in other papers (although not necessarily using lifting scheme nor 1-D transform). The key idea is to make the lifting scheme causal, so that it may be evaluated as a running scheme without buffering of the whole signal. However, due to borders of the finite signals, some buffering is necessary at least at the beginning and end of the single-loop processing. For 2-D signals, the same idea was also discovered in another papers under various names, e.g. in [12]–[17]. The most sophisticated techniques, that we are aware of, were investigated by R. Kutil in [18]. The author emphasizes that the main issue are the arduous prolog and epilog phases. Many authors [19]–[22] also discovered a similar technique with a slightly different goal (a better utilization of cache lines). This technique is referred to as the aggregation, strip-mine, or loop tiling.

Since this work is based on our previous works in [23]–[25], it should be explained what the difference between this work and the referenced papers is. In [23], we formulated two-dimensional cores using a different notation. However, they were not designed for any effective treatment of signal boundaries. In [24], [25], we have proposed DWT transform engine for JPEG 2000 encoders. Also in this work we have not overcame the problem with effective treatment of signal boundaries. Instead, we have introduced several buffers in order to circumvent the problem.

As it can be expected, we see a gap which can allow for significant simplifications and speedups. Specifically, we propose a solution consisting of a mutable mapping of input coefficients onto output ones.

III. Proposed Method

The section proposes a computation unit built using the lifting scheme technique. The direct consequence of this formulation is the possibility of an elegant treatment of signal boundaries. As mentioned above. the proposes unit is referred to as the core.

In this paragraph, some terminology necessary to understand the following text is clarified. Lag F describes a delay of the output samples with respect to the input samples. The stage is used in the sense of the scheme step, usually the lifting step. In linear algebra, such stage can be described by the linear operator (a matrix) mapping the input vector onto the output vector.

The following part of the section leads to the formulation of the core. For demonstration purposes, only even-length signals are considered. The single level of the discrete wavelet transform decomposes the input signal

$$\left(a^0_{n_0}\right)_{0\le n<N} \tag{3}$$

of size N samples into the resulting wavelet bands

$$\left(d^1_{n_1}\right)_{0\le n_1<N/2}, \qquad \left(a^1_{n_1}\right)_{0\le n_1<N/2}. \tag{4}$$

As a next step, a unit which continuously consumes the input signal a^0 and produces the output a^1, d^1 subbands is proposed. As mentioned above, this unit is referred to as the "core" in this paper. As a consequence of the DWT nature, the core has to consume pairs of input samples. The input signal is processed progressively from the beginning to the end, therefore in a single loop. The corresponding output samples are produced with a lag F samples depending on the underlying computation scheme. The core requires access to an auxiliary buffer B. This buffers hold intermediate results of the underlying computation scheme. The size of the buffer can be expressed as κ coefficients, where κ is the number of values that have to be passed between adjacent cores.

To simplify relations, two functions are introduced given by

$$\Theta(n) = n + F, \text{ and } \Omega(n) = \lceil n/2 \rceil. \tag{5}$$

The function $\Theta(n)$ maps core output coordinates onto core input coordinates with the lag F. The function $\Omega(n)$ maps the coordinates at the input level onto coordinates at the output level with respect to the chosen coordinate system. Note that the $\Omega(n)$ can be defined in many ways. We chose the definition (5) that is compatible with JPEG 2000 standard.

The core transforms the fragment I_n of an input signal onto the fragment O_n of an input signal

$$I_n = \begin{pmatrix} a^0_{\Theta(n)} & a^0_{\Theta(n+1)} \end{pmatrix}^T, \tag{6}$$

$$O_n = \begin{pmatrix} a^1_{\Omega(n)} & d^1_{\Omega(n+1)} \end{pmatrix}^T, \tag{7}$$

while updating the auxiliary buffer.

Finally, operations performed inside the core can be described using a matrix C as the relationship

$$\boldsymbol{y} = C\,\boldsymbol{x} \tag{8}$$

of the input vector

$$\boldsymbol{x} = B \parallel I_n \tag{9}$$

with the output vector

$$\boldsymbol{y} = B \parallel O_n, \tag{10}$$

where $\parallel$ denotes the concatenation operator. The (8) is the most fundamental equation of this thesis. In this linear mapping, the matrix C defines the core.

The meaning and the number of individual coefficients in B is not firmly given. The choice of the matrix C involves a degree of freedom of the presented framework. Some notes regarding this choice can be found in Section Discussion.

TABLE I

INDIVIDUAL LINEAR TRANSFORMATIONS INSIDE THE MUTABLE CDF 5/3 CORE. ALSO INVERSE LIFTING STEPS $T^{-1}_{\alpha,\Theta(n)}, S^{-1}_{\beta,\Theta(n)}$ ARE SHOWN. CHANGES ARE DISPLAYED IN DIFFERENT COLOR.

$\Theta(n)$	$T^{-1}_{\alpha,\Theta(n)}$	$S^{-1}_{\beta,\Theta(n)}$	$T_{\alpha,\Theta(n)}$	$S_{\beta,\Theta(n)}$
0	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 2\alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$
1	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2\beta & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$
$\Theta(n)$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$
$\Theta(N-2)$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$
$\Theta(N)$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 2\alpha & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2\beta \\ 0 & 1 & 0 & 0 \end{bmatrix}$

To keep the total number of wavelet coefficients equal to the number of input samples, symmetric border extension is widely used. A particular variant of this extension is employed in JPEG 2000 standard. Please, consult particular details with [26].

This paper describes the core which calculates the CDF 5/3 transform. For the shortest possible lag $F = 1$, it is easy to ensemble the core from (2) as

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \beta & 1 & \beta \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ \alpha & 0 & \alpha & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{11}$$

for

$$\boldsymbol{x} = \begin{bmatrix} B_a & B_d & a^0_{n+1} & a^0_n \end{bmatrix}^T \text{ and} \tag{12}$$

$$\boldsymbol{y} = \begin{bmatrix} B_a & B_d & a^1_{n/2-1} & d^1_{n/2} \end{bmatrix}^T. \tag{13}$$

The B comprises $\kappa = 2$ elements. For the purposes of the discussion, only even-length signals are considered. The core consists of two stages suitable for hardware pipelining.

As mentioned earlier, the core processes the signal in the single loop. The naive way of border handling is described first. Due to the symmetric extension, the core begins the processing at a certain position before the start of the actual signal sequence. Similarly, the processing stops at a certain position after the end of the signal. The samples outside the actual signal are mirrored into the valid signal area. This processing introduces the need for buffering of the input at least at the beginning and the end of the signal. Such buffering breaks the ability of simple stream processing, especially considering the multi-scale decomposition. All approaches referenced in Section Related Work also suffer from this issue.

The situation can be neatly resolved changing the core near the signal border. In more detail, the "mutable" core performs 5 different calculations depending on the position in relation to the input signal. Therefore, the core comprises 2×5 slightly different steps (stages) in total. This can be written in matrix notation as

$$\boldsymbol{y} = S_{\beta,\Theta(n)}\, T_{\alpha,\Theta(n)}\, \boldsymbol{x}, \tag{14}$$

where $T_{\alpha,\Theta(n)}, S_{\beta,\Theta(n)}$ are the linear transformations of the predict and update stages performed at the subsampled output position $\Theta(n)$. These coefficients are generated in $T_{\alpha,\Theta(n)}$ so that these can be used by $T_{\alpha,\Theta(n+2)}$ at the same time at which $S_{\beta,\Theta(n+2)}$ runs. It is essential that the coefficients B_a, B_d are initially set to zero. The output signal is generated with the lag $F = 1$ sample with respect to the input signal. The input a samples outside of the input signal are treated as zeros. Similarly, the output a, d coefficients outside of the output signal are discarded. The following table describes the individual $T_{\alpha,\Theta(n)}, S_{\beta,\Theta(n)}$ transformations. The transform

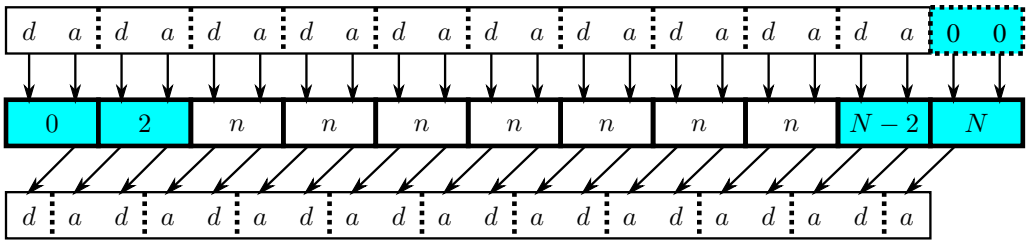

Fig. 1. Signal processing using the mutable core. The input position on the original location n is shown. The first and last two cores (highlighted) differ from the others.

is defined using the α, β constants. Table I enumerates the individual core stages. In addition, Figure 1 illustrates their usage. The very first and very last cores access outside the signal. The input samples already split into a, d subbands. As a result, the signal is transformed without buffering, possibly on a multi-scale basis.

IV. Discussion

So far, only the CDF 5/3 wavelet has been discussed as an illustrative example. However, the presented computation scheme is general.

One can identify the core in arbitrary underlying lifting schemes. However, its implementation can be obscure in some cases mainly due to the increasing number of intermediate results in auxiliary buffers. For his reason, a lifting factorization employing steps in the form of degree-1 filters is a proven choice. Many such factorizations of various wavelets have been presented in the literature, e.g. [4]. Moreover, the symmetric filters with lengths $2x \pm 1$, as is case of the CDF 5/3 and 9/7 filters, can be implemented through some sequence of lifting steps having this particular form. See [26] for details.

In parallel environments, the presented scheme can be "unwrapped" to fit the number of computing units (threads). Instead of passing the intermediate results through the buffer B, an direct exchange of them then arises. Unfortunately, this exchange introduces the need for a synchronization barrier, that can be a bottleneck of the processing.

The multi-scale decomposition can be easily constructed by chaining the cores into a series. In this case, only the a coefficients are linked to the next decomposition level. Each subsequent decomposition level causes additional delay F at the half sampling rate.

Considering multi-dimensional signals, multi-dimensional cores can be constructed as the tensor product of 1-D cores. For example, the 2-D core consumes the 2×2 fragment of the input signal a^0 and immediately produces the four-tuple (a^1, h^1, v^1, d^1) of resulting subbands.

The core C can also be internally reorganized in order to minimize some of the resources. In [23], we demonstrated this property on FPGA where the minimization of the core latency had a direct impact on the utilization of flip-flop (FF) circuits and look-up tables (LUT).

The construction of the matrices themselves is governed by simple rule. When the coefficient outside a valid signal sequence would be accessed, the zero is put into the matrix at the corresponding place. Additionally, the symmetrization is obtained by placing the factor of 2 at desired locations at the same moment. In this paper, we have discussed only the symmetric signal extension. Nevertheless, the matrices for any other extension could be constructed. For zero-padding, the factor of 2 should be eliminated everywhere in Table I.

One can simply verify the correctness of the scheme presented by its unwrapping. The data-flow graph obtained shows the mirroring at the beginning and end of the signal.

In conclusion, this paper presents nice idea to take of boundary effects in DWT processing. Usually some form of wrap-around or replication is necessary to take care of the image boundaries. But we have proposed a slightly different way of handling this which then makes efficient parallel or pipelined processing possible.

V. Conclusions

We have focuses on treatment of signal boundaries for computing of the discrete wavelet transform. In the state-of-the-art methods, the treatment of signal boundaries is performed in a complicated way. This way causes buffering of the signal at least at the beginning and end of data.

In this paper, we have overcome this issue. This was accomplished using the proposed mutable core which performs the transform in a single loop. Using this core, the complete transform can be evaluated as a running scheme. In other words, no buffering is required anywhere. The resulting coefficients are exactly the same as with the original lifting scheme.

The future work we would like to do includes an extension to more complicated lifting factorizations, and a hardware implementation of the core proposed.

Acknowledgement

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] 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. doi:10.1002/cpa.3160450502

[2] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Wavelet Applications in Signal and Image Processing III*, A. F. Laine and M. Unser, Eds. Proc. SPIE 2569, 1995, pp. 68–79.

[3] ——, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186–200, 1996.

[4] 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. doi:10.1007/BF02476026

[5] S. G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way. With contributions from Gabriel Peyre.*, 3rd ed. Academic Press, 2009.

[6] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[7] M. Vetterli, J. Kovacevic, and V. Goyal, *Foundations of Signal Processing*. Cambridge University Press, 2014.

[8] R. E. Blahut, *Fast Algorithms for Signal Processing*. Cambridge University Press, 2010.

[9] A. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, 1998. doi:10.1006/acha.1997.0238

[10] U. Drepper, "What every programmer should know about memory," Red Hat, Inc., Tech. Rep., Nov. 2007.

[11] 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. doi:10.1117/12.408615

[12] D. Chaver, C. Tenllado, L. Pinuel, M. Prieto, and F. Tirado, "2-D wavelet transform enhancement on general-purpose microprocessors: Memory hierarchy and SIMD parallelism exploitation," in *High Performance Computing — HiPC 2002*, ser. Lecture Notes in Computer Science, S. Sahni, V. K. Prasanna, and U. Shukla, Eds. Springer, 2002, vol. 2552, pp. 9–21. doi:10.1007/3-540-36265-7_2

[13] A. Shahbahrami and B. Juurlink, "A comparison of two SIMD implementations of the 2D discrete wavelet transform," in *Proc. 18th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, Nov. 2007, pp. 169–177.

[14] A. Shahbahrami, "Improving the performance of 2D discrete wavelet transform using data-level parallelism," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, Jul. 2011, pp. 362–368. doi:10.1109/HPCSim.2011.5999847

[15] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 378–389, 2000. doi:10.1109/83.826776

[16] J. Oliver, E. Oliver, and M. P. Malumbres, "On the efficient memory usage in the lifting scheme for the two-dimensional wavelet transform computation," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, Sep. 2005, pp. I–485–8. doi:10.1109/ICIP.2005.1529793

[17] D. Barina and P. Zemcik, "Vectorization and parallelization of 2-D wavelet lifting," *Journal of Real-Time Image Processing (JRTIP)*, in press. doi:10.1007/s11554-015-0486-6

[18] 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. doi:10.1109/PDP.2006.14

[19] P. Meerwald, R. Norcen, and A. Uhl, "Cache issues with JPEG2000 wavelet lifting," in *Visual Communications and Image Processing (VCIP)*, ser. SPIE, vol. 4671, 2002, pp. 626–634.

[20] D. Chaver, C. Tenllado, L. Pinuel, M. Prieto, and F. Tirado, "Vectorization of the 2D wavelet lifting transform using SIMD extensions," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2003, p. 8. doi:10.1109/IPDPS.2003.1213416

[21] 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. doi:10.1109/ICME.2002.1035902

[22] J. Tao and A. Shahbahrami, "Data locality optimization based on comprehensive knowledge of the cache miss reason: A case study with DWT," in *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, Sep. 2008, pp. 304–311. doi:10.1109/HPCC.2008.7

[23] D. Barina, M. Musil, P. Musil, and P. Zemcik, "Single-loop approach to 2-D wavelet lifting with JPEG 2000 compatibility," in *Workshop on Applications for Multi-Core Architectures (WAMCA)*. IEEE, 2015, pp. 31–36. doi:10.1109/SBAC-PADW.2015.10

[24] D. Barina, O. Klima, and P. Zemcik, "Single-loop software architecture for JPEG 2000," in *Data Compression Conference (DCC)*, 2016, pp. 582–582. doi:10.1109/DCC.2016.19

[25] ——, "Single-loop architecture for JPEG 2000," in *International Conference on Image and Signal Processing (ICISP)*, ser. Lecture Notes in Computer Science (LNCS), vol. 9680. Springer, 2016, pp. 346–355. doi:10.1007/978-3-319-33618-3_35

[26] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 2002.