DIAGONAL VECTORISATION OF 2-D WAVELET LIFTING

David Barina

Faculty of Information Technology
Brno University of Technology
Czech Republic

Pavel Zemcik

Faculty of Information Technology
Brno University of Technology
Czech Republic

ABSTRACT

With the start of the widespread use of discrete wavelet transform in image processing, the need for its efficient implementation is becoming increasingly more important. This work presents a novel SIMD vectorisation of 2-D discrete wavelet transform through a lifting scheme. For all of the tested platforms, this vectorisation is significantly faster than other known methods, as shown in the results of the experiments.

Index Terms— Discrete wavelet transforms, Image processing

1. INTRODUCTION

The discrete wavelet transform (DWT) [1] is mathematical tool which is suitable to decompose discrete signal into lowpass and highpass frequency components. Such a decomposition can even be performed at several scales. It is often used as a basis of sophisticated compression algorithms.

Considering the number of arithmetic operations, the lifting scheme [2] is currently the most efficient way for computing the discrete wavelet transform. This paper focuses on the CDF (Cohen-Daubechies-Feauveau) 9/7 wavelet [3] which is often used for image compression (e.g., JPEG 2000 standard). Responses of this wavelet can be computed by a convolution with two FIR filters, one with 7 and the other with 9 coefficients. The transform employing such a wavelet can be computed with four successive lifting steps as shown in [2]. Resulting coefficients are then divided into two disjoin groups – approximate and detail coefficients, or L and H subbands. The simple approach of lifting data flow graph evaluation directly follows the lifting steps. This approach suffers with several reads and writes of intermediate results. However, more efficient ways of lifting evaluation [4] [5] exist.

In case of two-dimensional transform, the DWT can be realized using separable decomposition scheme [6]. In this scheme, the coefficients are evaluated by successive horizontal and vertical 1-D filtering resulting in four disjoin groups (LL, HL, LH and HH subbands). A naive algorithm of 2-D DWT computation will directly follow horizontal and vertical filtering loops. Unfortunately, this approach is encumbered with several accesses into intermediate results. The horizontal and vertical loop can be fused into single one yielding into the single-loop approach [7].

In present personal computers, a general purpose microprocessor with SIMD (single instruction, multiple data) instruction set is often found. For example, in x86 architecture, the appropriate instruction set is SSE (Streaming SIMD Extensions). This 4-fold SIMD set fits exactly the CDF 9/7 lifting data flow graph when using the single precission floating-point format.

In this paper, the diagonal vectorisation of wavelet lifting recently published [5] is incorporated into the known single-loop approach. This new implementation is compared to the original one using the vertical vectorisation as well as to the naive approach with separated horizontal and vertical loops. For tested platforms, this new combination is consistently significantly faster than the original approach employing vertical vectorisation.

This paper is focused on the present computers with x86 architecture. All the methods presented in this paper are evaluated using ordinary PCs with Intel x86 CPUs. Intel Core2 Quad Q9000 running at 2.0 GHz was used. This CPU has 32 kiB of level 1 data cache and 3 MiB of level 2 shared cache (two cores share one cache unit). The results were verified on system with AMD Opteron 2380 running at 2.5 GHz. This CPU has 64 kiB of level 1 data, 512 kiB of level 2 cache per core and 6 MiB of level 3 shared cache (all four cores share one unit). Another set of control measurements was done on Intel Core2 Duo E7600 at 3.06 GHz and on AMD Athlon 64 X2 4000+ at 2.1 GHz. These are referred to as alternative platforms. Due to limited space, the details will not be shown here with the exception of a summarizing table. All algorithms below were implemented in C languade using SSE compiler intrinsics.[1] In all cases, 64-bit code compiled using GCC 4.8.1 with `-O3` flag was used.

The rest of the paper is organised as follows. Related Work section discusses the state of the art – especially lifting scheme, vectorisations and 2-D single-loop approach. Single-Loop Approach section focuses on this 2-D approach in more

[1] The code can be downloaded from `http://www.fit.vutbr.cz/research/prod/?id=211`.

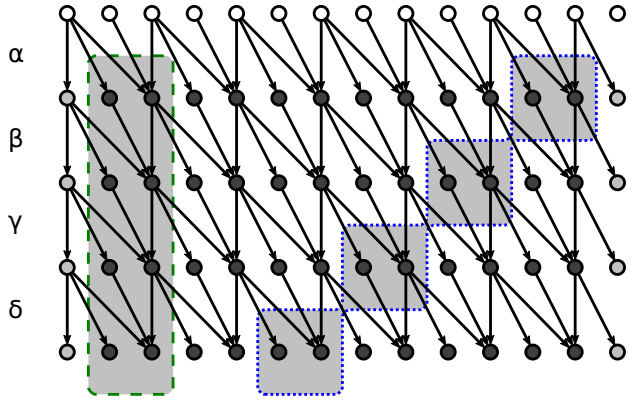

Fig. 1. Complete data flow graph of CDF 9/7 transform. The input signal on top, output at bottom. The vertical (dashed) as well as the diagonal (dotted) vectorisation is outlined.

detail and a fusion with the diagonal vectorisation is proposed here. Diagonal Vectorisation section proposes a combination of the 1-D diagonal vectorisation and the 2-D single-loop approach. Finally, Conclusion section closes the paper.

2. RELATED WORK

According to the number of arithmetic operations, the lifting scheme [2] is today's most efficient scheme for computing discrete wavelet transforms. Any discrete wavelet transform with finite filters can be factored into a finite sequence of N pairs of predict and update convolution operators P_n and U_n. Each predict operator P_n corresponds to a filter $p_i^{(n)}$ and each update operator U_n to a filter $u_i^{(n)}$. These operators alternately modify even and odd signal coefficients. The lifting factorisation is not unique. For symmetric filters, this non-uniqueness can be exploited to maintain symmetry of lifting steps.

In their paper [2], I. Daubechies and W. Sweldens demonstrated an example of CDF 9/7 transform factorisation which resulted into four lifting steps ($N = 2$) plus scaling of coefficients. In this example, the individual lifting steps use 2-tap symmetric filters for the prediction as well as the update. The calculation of the complete CDF 9/7 DWT is depicted in Fig. 1 (coefficient scaling is ommited). The coefficients of the four 2-tap symmetric filter are denoted α, β, γ and δ respectively. During this calculation, intermediate results can be appropriately shared between neighboring pairs of coefficients. This is an in-place implementation, which means the DWT can be calculated without allocating auxiliary memory. Resulting coefficients are interleaved in place of the input signal.

The problem of minimum memory implementations of lifting scheme was adressed in [4] by Ch. Chrysafis and A. Ortega. Their approach is very general and it is not focused on parallel processing. Anyway, this is essentially the same method as the on-line or pipelined computation mentioned in other papers (although not necessarily using lifting scheme nor 1-D transform). Its variation was presented in [7] which is specifically focused on CDF 9/7 wavelet transform. The work was also later extended to [8] where same authors addressed a problem of minimum memory implementation of 2-D transform.

Many papers are focused on 2-D, possibly 3-D [9] transforms. For instance, in [10] S. Chatterjee and Ch. D. Brooks proposed two optimizations of 2-D transform. The first optimization interleave the operations of 1-D transform on multiple columns. The second optimization modifies the layout of the image data so that each sub-band is laid out contiguously in memory.

Furthermore, in [11] authors address the implementation of a 2-D transform focusing on memory hierarchy and SIMD parallelization. Here, the pipelined computation is applied in vertical direction on subsequent rows. The above mentioned work was later extended to [12] where vectorisation using Intel's SSE instructions is proposed on several rows in parallel. The approach they used is simillar to merging vertical and horizontal filtering into single loop as in [7]. In [13], same authors introduced new memory organization for 2-D transforms which is trade-off between the in-place and Mallat (with auxiliary matrix) organization. Then, the authors vectorised transform using approach simillar to one described in [7] where four rows are processed in parallel. In [14] and [15], several techniques for reducing cache misses for 2-D transforms was proposed. Moreover, two techniques to avoid 64K aliasing were proposed in [15].

In [16], R. Kutil *et al.* presended SIMD vectorisation of several frequently used wavelet filters. This vectorisation is applicable only on those filters discussed in their paper. Specifically, vectorisation of CDF 9/7 wavelet computed using lifting scheme is vectorised by a group of four successive pairs of coefficients. Unlike a general aproach proposed in [5], their 1-D transform vectorisation handles coefficients in blocks. SIMD vectorisation in [5] processes pairs of coefficients one by one immediately when available (without packing into groups). It is especially useful on systems equipped with small CPU cache.

In [7], R. Kutil splits lifting data flow graph into vertical areas (see green area in Fig. 1). Due to dependencies of individual operations, computations inside these areas cannot be parallelized. However, this approach is advantageous because of a locality of data necessary to compute output coefficients. Parallel processing using SIMD extensions is achieved by processing four rows of 2-D transform or four adjacent coefficients in one row in parallel. In our paper, such a method is called vertical vectorisation. Futher in Kutil's paper, the author focuses on 2-D transform by merging vertical and horizontal passes into a single loop. The issue of cache efficient implementation is also addressed here. To overcome it, the author uses the prime stride between consequent rows of image.

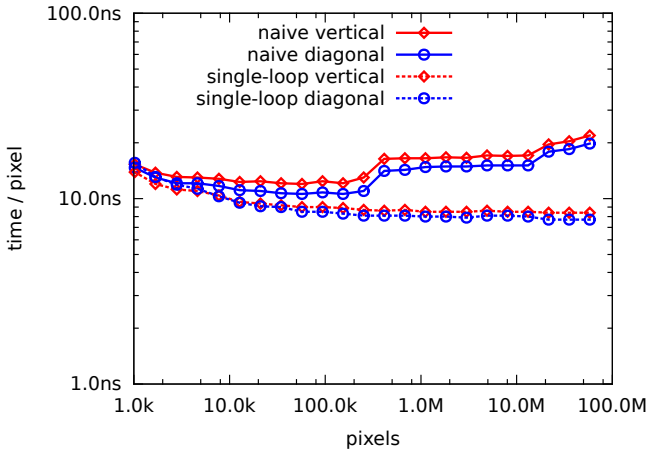

Fig. 2. Comparison of the naive and the single-loop approaches on Intel CPU. Both axes are shown in logarithmic scale.

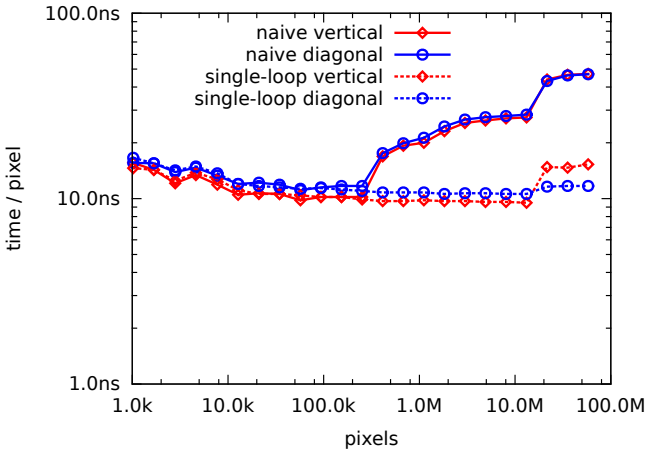

Fig. 3. Comparison of the naive and the single-loop approaches on AMD CPU. Both axes are shown in logarithmic scale.

Implementation of 2-D or 3-D DWT was also studied on modern programmable GPUs (graphics processing units), mostly using the lifting scheme. When implemented in GPU's fragment shaders, the filter bank scheme often outperforms lifting scheme filtering for shorter wavelets, as it was reported in [17]. The authors of [18] compared previous approach with lifting and convolution implementation of various integer-to-integer wavelet transform using CUDA in favor of the lifting scheme.

3. SINGLE-LOOP APPROACH

This section focuses on the single-loop 2-D approach in more detail and proposes a fusion with the diagonal vectorisation. Such a combination allows SIMD vectorization of the algorithm core. For some platforms, this is significantly faster than the original single-loop approach.

A useful implementation of the vertical vectorisation of 1-D wavelet lifting was described in [7]. We have presented the diagonal vectorisation (the blue area in Fig. 1) of this algorithm in [5] which allows the use of SIMD processing without grouping coefficients into blocks. These two vectorisations can be naively used for 2-D transform by both vertical and horizontal filtering. It is called a naive approach in this paper. Performance evaluation of these methods is depicted in Fig. 2 and Fig. 3 (the naive curves). The performance degradation between 100 k and 1 M pixels on Intel platform is caused by deteriorated level 1 data cache hit/miss ratio. On AMD, the performance degradation between 100 k and 1 M as well as between 10 M and 100 M is caused by decreasing level 2 cache hit/miss ratio. See [19] for more details. In order to avoid doubts about possible caching issues in the naive implementation, it should be noted that these are avoided using the prime stride between consequent rows of image.

In [7], R. Kutil considered two nested loops (an outer vertical and an inner horizontal loop) as a single loop over all pixels of the image. He called it single-loop approach. Specifically, he merged two vertically vectorized loops into single one. Since the output of horizontal filtering is the input of vertical one, he uses the output of the first filtering immediately when it is available.

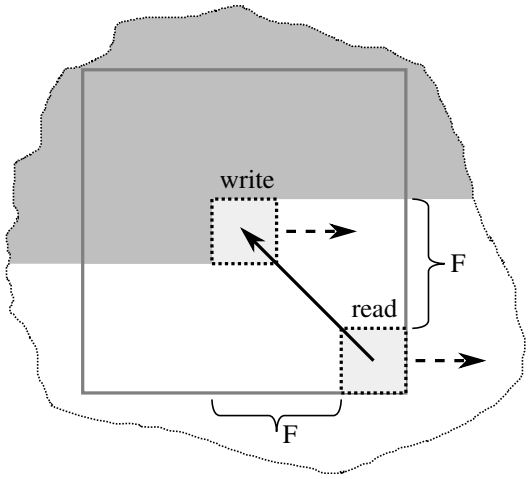

Fig. 4. A core of the single-loop approaches. Already processed area is highlighted in gray. An outer rectangle is the support for 2×2 coefficients in the middle (labelled write).

One step of the vertical vectorisation requires two values (a pair) to perform an iteration (see dataflow graph in Fig. 1). Thus, this algorithm needs to perform two horizontal filterings (on two consecutive rows) at once. For each row, a low-pass and a high-pass coefficient is produced, which makes 2×2 values in total. The image processing by this core is outlined in Fig. 4 where F indicates a lag of the core. Vertical

vectorisation algorithm passes four values from one iteration to the other in horizontal direction for each row (8 in total). In vertical direction, the algorithm needs to pass four rows between iterations. The length of coresponding prolog as well as epilog phases are $F = 4$ coefficients. The situation is illustrated in Fig. 5.

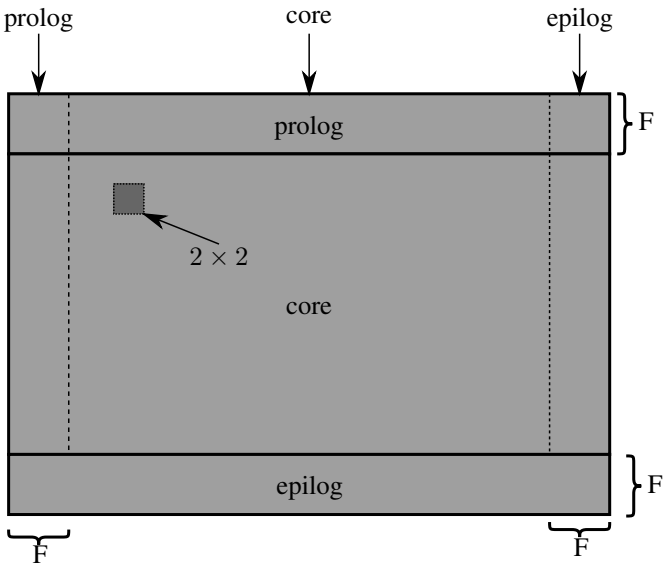

Fig. 5. Simplified view of the single-loop approach showing the prolog and epilog phases. The length F of these phases is 4 (vertical vectorisation) or 10 (diagonal one).

4. DIAGONAL VECTORISATION

We propose to combine above described approach with our algorithm of diagonal vectorisation presented in [5]. In this case, single-loop pass over all pixels of the image remains. In contrast to Kutil's work, we employed diagonal vectorisation in the core of our algorithm. As in the previous case, the diagonal vectorisation uses two values to perform one iteration. Thus, also this algorithm needs to perform two horizontal filterings at once. The kernel has also size of 2×2 values in total. In the horizontal direction, diagonal vectorisation passes four values from one iteration to the other for each row. However, the algorithm needs to pass 12 rows between iterations in vertical direction. The lengths of coresponding prolog and epilog phases are $F = 10$ coefficients now. Unlike the vertical vectorisation, the diagonal one can be SIMD-vectorised in 1-D and thus also in 2-D case. Finally, a large part of the 2×2 diagonally vectorised kernel is written using SSE instructions which perform 4 operation in parallel. A disadvantage is the $3\times$ more memory occupied by row buffers.

Both of the described approaches require complicated treatment of image border areas using several different combinations of 1-D prolog and epilog phases. It makes their implementation very tedious. The proposed single-loop approach using diagonal vectorisation was compared to the original one with vertical vectorisation. The execution times are plot in Fig. 2 and Fig. 3. They also show the naive implementations using separated horizontal and vertical filtering. All the diagonal implementations were tuned using SSE instructions. In case of vertical implementations, there is no benefit observed from using SSE data types.

Reference platforms

	Intel		AMD	
algorithm	time	speedup	time	speedup
naive vertical	21.9	1.0	47.1	1.0
naive diagonal	19.8	1.1	46.9	1.0
single-loop vertical	8.4	2.6	15.3	3.1
single-loop diagonal	**7.7**	**2.8**	**11.7**	**4.0**

Alternative platforms

	Intel		AMD	
algorithm	time	speedup	time	speedup
naive vertical	19.4	1.0	154.0	1.0
naive diagonal	17.7	1.1	152.3	1.0
single-loop vertical	6.2	3.1	20.4	7.5
single-loop diagonal	**5.7**	**3.4**	**17.0**	**9.1**

Table 1. Performance evaluation of both of the vectorisations and both of the approaches. The time is given in nanoseconds per one pixel.

Both of the above approaches (naive, single-loop) in combination with vectorisations (vertical, diagonal) are compared in Table 1. The naive algorithm is used as a reference one. All the measurements was performed on 58 megapixel image.

5. CONCLUSION

We have presented a novel approach to 2-D wavelet lifting scheme reaching speedup at least $4.0\times$ on AMD and $2.8\times$ on Intel platform for large data. This approach fuses single-loop approach and diagonal vectorisation. This proposed fusion allows for SIMD-vectorisation of newly formed 2-D image processing core while preserving its minimum memory requirements.

Further work could explore behavior of proposed approaches on the other architectures (vector processors, many-core systems) or with other wavelets (different lifting factorisations).

Acknowledgement

This work has been supported by the IT4Innovations Centre of Excellence (no. CZ.1.05/1.1.00/02.0070) and the EU FP7-ARTEMIS project IMPART (grant no. 316564).

6. REFERENCES

[1] Stéphane Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way. With contributions from Gabriel Peyré.*, Academic Press, 3 edition, 2009.

[2] Ingrid Daubechies and Wim Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[3] A. Cohen, Ingrid 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.

[4] Christos Chrysafis and Antonio Ortega, "Minimum memory implementations of the lifting scheme," in *Proceedings of SPIE, Wavelet Applications in Signal and Image Processing VIII*, 2000, vol. 4119 of *SPIE*, pp. 313–324.

[5] David Bařina and Pavel Zemčík, "Minimum memory vectorisation of wavelet lifting," in *Advanced Concepts for Intelligent Vision Systems (ACIVS)*. 2013, Lecture Notes in Computer Science (LNCS) 8192, pp. 91–101, Springer London.

[6] Stéphane G. 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.

[7] Rade 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.

[8] Christos Chrysafis and Antonio Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 378–389, 2000.

[9] Rade Kutil and Andreas Uhl, "Hardware and software aspects for 3-D wavelet decomposition on shared memory MIMD computers," in *Parallel Computation. Proceedings of ACPC'99*. 1999, vol. 1557 of *Lecture Notes on Computer Science*, pp. 347–356, Springer.

[10] Siddhartha Chatterjee and Christopher D. Brooks, "Cache-efficient wavelet lifting in JPEG 2000," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2002, vol. 1, pp. 797–800.

[11] Daniel Chaver, Christian Tenllado, Luis Pinuel, Manuel Prieto, and Francisco Tirado, "2-D wavelet transform enhancement on general-purpose microprocessors: Memory hierarchy and SIMD parallelism exploitation," in *High Performance Computing HiPC 2002*, Sartaj Sahni, Viktor K. Prasanna, and Uday Shukla, Eds., vol. 2552 of *Lecture Notes in Computer Science*, pp. 9–21. Springer, 2002.

[12] Daniel Chaver, Christian Tenllado, Luis Pinuel, Manuel Prieto, and Francisco Tirado, "Wavelet transform for large scale image processing on modern microprocessors," in *High Performance Computing for Computational Science VECPAR 2002*, José M. L. M. Palma, A. Augusto Sousa, Jack Dongarra, and Vicente Hernández, Eds., vol. 2565 of *Lecture Notes in Computer Science*, pp. 549–562. Springer, 2003.

[13] Daniel Chaver, Christian Tenllado, Luis Pinuel, Manuel Prieto, and Francisco 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.

[14] Peter Meerwald, Roland Norcen, and Andreas Uhl, "Cache issues with JPEG2000 wavelet lifting," in *Visual Communications and Image Processing (VCIP)*, 2002, vol. 4671 of *SPIE*, pp. 626–634.

[15] Asadollah Shahbahrami, Ben Juurlink, and Stamatis Vassiliadis, "Improving the memory behavior of vertical filtering in the discrete wavelet transform," in *Proceedings of the 3rd conference on Computing frontiers (CF)*. 2006, pp. 253–260, ACM.

[16] Rade Kutil, Peter Eder, and Markus Watzl, "SIMD parallelization of common wavelet filters," in *Parallel Numerics '05*, 2005, pp. 141–149.

[17] Christian Tenllado, Javier Setoain, Manuel Prieto, Luis Pinuel, and Francisco Tirado, "Parallel implementation of the 2D discrete wavelet transform on graphics processing units: Filter bank versus lifting," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 3, pp. 299–310, 2008.

[18] Wladimir J. van der Laan, Andrei C. Jalba, and Jos B. T. M. Roerdink, "Accelerating wavelet lifting on graphics hardware using CUDA," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 132–146, 2011.

[19] Ulrich Drepper, "What every programmer should know about memory," Red Hat, online at URL: `http://www.akkadia.org/drepper/cpumemory.pdf`, accessed: 2014-05-23, 2007.