Diagonal Vectorisation of 2-D Wavelet Lifting

David Barina and Pavel Zemcik

Faculty of Information Technology of Brno University of Technology
Czech Republic

Introduction

With the start of the widespread use of the discrete wavelet transform in the 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

Wavelet transform

The discrete wavelet transform (DWT) is able to decompose a discrete signal into lowpass and highpass frequency components. DWT is often used as the basis of sophisticated compression algorithms. This is the case of JPEG 2000 and Dirac compression standards in which CDF 9/7 wavelet is employed for lossy compression. Responses of this wavelet can be computed by a convolution with two FIR filters, one with 7 and the other with 9 coefficients.

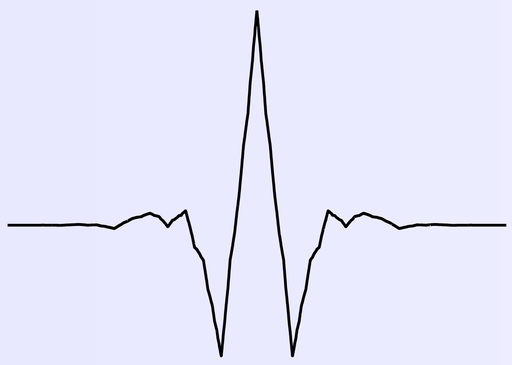

Figure 1: The used CDF 9/7 wavelet.

Single-Loop Approach

R. Kutil considered two nested loops (an outer vertical and an inner horizontal loop) as a single loop over all pixels of the image. Specifically, he merged two vertically vectorized loops into single one. We propose to combine above described approach with our algorithm of diagonal vectorisation. Such a combination allows SIMD vectorization of the algorithm core.

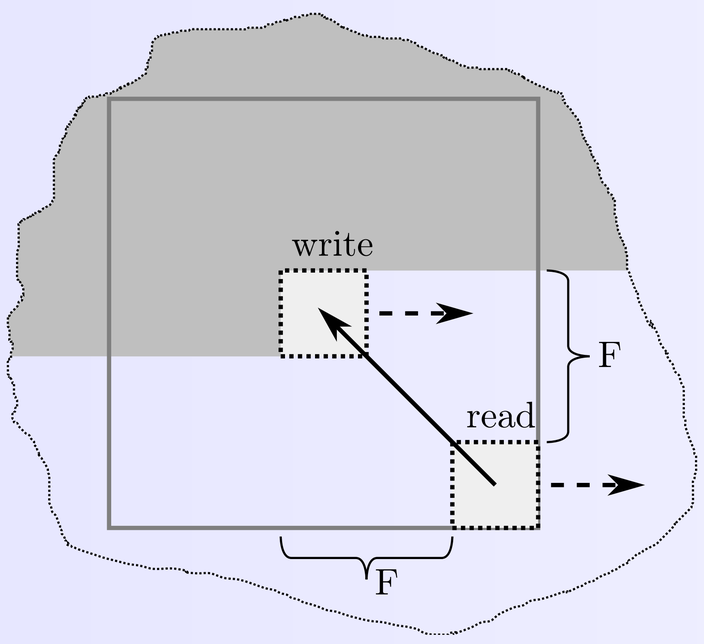

Figure 3: A core of the single-loop approach.

Acknowledgements

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).

Lifting Scheme

According to the number of arithmetic operations, the lifting scheme 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 calculation of the complete CDF 9/7 DWT is depicted below.

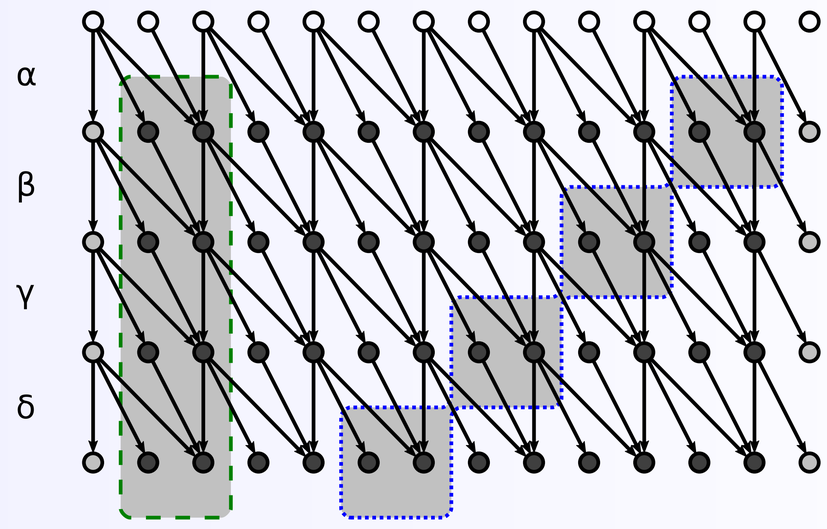

Figure 2: Complete data flow graph of CDF 9/7 transform.

During this calculation, intermediate results can be appropriately shared between neighboring pairs of coefficients. R. Kutil splits lifting data flow graph into vertical areas (see green area in figure). Such a method is called the **vertical vectorisation**. Due to dependencies of individual operations, computations inside these areas cannot be parallelized. Parallel processing using SIMD extensions is achieved by processing four rows of 2-D transform or four adjacent coefficients in one row in parallel. We have presented the **diagonal vectorisation** (the blue area in figure) of this algorithm 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.

Evaluation

All the methods presented in this paper are evaluated using ordinary PCs with Intel x86 CPUs.

Reference platforms 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).

Alternative platforms 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.

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**

The naive algorithm is used as a reference one. All the measurements was performed on 58 megapixel image. All the diagonal implementations were tuned using SSE instructions. In case of vertical implementations, there is no benefit observed from using SSE data types.

Source Code

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