Parallel Wavelet Transforms on Multiprocessors

Topic 12: Architectures and Algorithms for Vision and other Senses

Manfred Feil, Rade Kutil, and Andreas Uhl

RIST++ & Department of Scientific Computing
University of Salzburg, AUSTRIA

e-mail: {mfeil,rkutil,uhl}@cosy.sbg.ac.at

Abstract. In this work we discuss several issues relevant for parallel wavelet transforms and their possible implications on the choice of a proper programming paradigm for corresponding multiprocessor implementations. In this context we compare pyramidal wavelet transform, wavelet packet decomposition, and the á trous algorithm.

1 Introduction

Wavelets and the corresponding wavelet transforms have caused a tremendous interest in the areas of imge and signal processing, numerical mathematics, and all types of data analysis applications. However, since not even the term "wavelet" is a very specific one it is obvious that different kinds of wavelet transforms might exist, and they do. Depending on the type of target application one may choose among a multitude of different wavelet transforms, each of them associated with certain advantages and disadvantages. For example, the "fast wavelet transform" (or discrete wavelet transform or pyramidal wavelet decomposition) is very well suited for real-time applications and all kinds of compression purposes, but exhibits a rather low scale-resolution (i.e. a dyadic grid) which makes it not well suited for data analysis purposes.

A vast amount of work has been devoted to implement and analyze parallel algorithms for wavelet transforms. But due to reasons explained above, it is hard to judge the content of a paper devoted to "parallel wavelet transforms". There have been papers devoted to parallel Gabor transform [21, 20], continuous wavelet transform [12, 30, 10], and of course to the fast wavelet transform [33, 28, 24, 16, 11, 17, 18, 5, 15, 23, 7, 2, 25, 14].

In this work we focus onto special problems associated with almost each parallel wavelet algorithm (here we treat pyramidal wavelet decomposition, wavelet packet decomposition, and the á trous algorithm):

- Data decomposition strategies
- Handling of border data

Specifically we investigate the impact of these problems onto the choice of a proper programming paradigm for multiprocessors (i.e. shared memory programming vs. message passing). It is very interesting to see that very different results occur for different types of algorithms.

2 Wavelet transform algorithms

The fast wavelet transform can be efficiently implemented by a pair of appropriately designed Quadrature Mirror Filters (QMF). A 1-D wavelet transform of a signal S is performed by convolving S with both QMF's and downsampling by 2; since S is finite, one must make some choice about what values to pad the extensions with. This operation decomposes the original signal into two frequency-bands (called subbands), which are often denoted coarse scale approximation and detail signal. Then the same procedure is applied recursively to the coarse scale approximations several times (see Fig. 1.a).

The classical 2-D transform is performed by two separate 1-D transforms along the rows and the columns of the image data S, resulting at each decomposition step in a low pass image (the coarse scale approximation) and three detail images (see Fig. 1.b). To be more concise, this is achieved by first convolving the rows of the low pass image S_{j+1} (or the original image in the first decomposition level) with the QMF filterpair G and H (which are a high pass and a low pass filter, respectively), retaining every other row, then convolving the columns of the resulting images with the same filterpair and retaining every other column. The same procedure is applied again to the coarse scale approximation S_j and to all subsequent approximations.

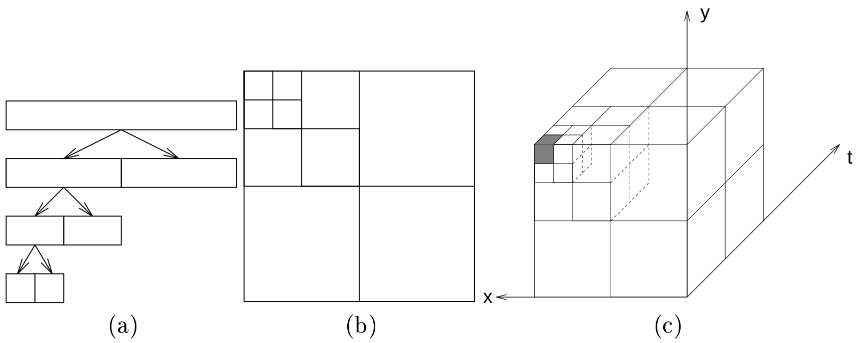


Fig. 1. 1-D (a) and 2-D (b) and 3-D (c) pyramidal wavelet decomposition.

By analogy to the 2-D case the 3-D wavelet decomposition is computed by applying three separate 1-D transforms along the coordinate axes of the 3-D data. As it is the case for 2-D decompositions, it does not matter in which order the filtering is performed (e.g. a 2-D filtering frame by frame with subsequent temporal filtering, three 1-D filterings along y, t, and x axes, e.t.c.). After one decomposition step we result in 8 frequency subbands out of which only the approximation data (the gray cube in Fig. 1.c) is processed further in the next decomposition step. This means that the data on which computations are performed are reduced to $\frac{1}{8}$ in each decomposition step.

Wavelet packets [32] represent a generalization of the method of multiresolution decomposition and comprise the entire family of subband coded (tree) decompositions. Whereas in the wavelet case the decomposition is applied recursively to the coarse scale approximations, in the wavelet packet decomposition the recursive procedure is applied to all the coarse scale approximations and detail signals, which leads to a complete wavelet packet tree (i.e. binary tree and quadtree in the 1-D and 2-D case, respectively) and more flexibility in frequency resolution. See also Fig. 3 for the 2-D case. The wavelet packet "best basis algorithm" [4] performs an adaptive optimization of the frequency resolution of a complete wavelet packet decomposition tree by selecting the most suitable frequency subbands for signal compression.

Whereas a significant amount of work has been already done concerning parallel algorithms for the fast wavelet transform, only few papers have been devoted to parallel wavelet packet decomposition and its specific features and demands (e.g. approaches for performing the best basis algorithm and the irregular decomposition into such a basis on parallel MIMD [13, 29, 31] and SIMD [9] architectures, application of parallel wavelet packet decomposition in numerics [22]).

The "á trous" algorithm [26, 6] represents a discrete approach to the classical continuous wavelet transform which is used in applications which require a denser time-(scale) plane as obtained with the fast wavelet transform. Examples of practical applications can be found in the field of cosmology for studying the formation of Large Scale Structures in the Universe [19, 27]. Lega et al. [19] describe a parallel SIMD implementation for the CM-200.

The basic idea behind the *á trous* algorithm is the discrete wavelet transform *without* a following decimation step (see Fig. 2.b for a detail image at decomposition level 3). The low-pass filter H is said to be an *á trous* filter if it satisfies

$$h_{2k} = \delta(k)/\sqrt{2}$$

where $\delta_{k,m}$ is the Kronecker delta and $\delta(k) \triangleq \delta_{k,0}$.

Now we proceed from the first approximation $\{c_0(k)\}$ (at scale 0) which is the scalar product of the function $f(x)$ with the scaling function $\phi(x)$ (corresponding to an *á trous* low-pass filter H):

$$c_0(k) = \langle f(x), \phi(x-k) \rangle$$

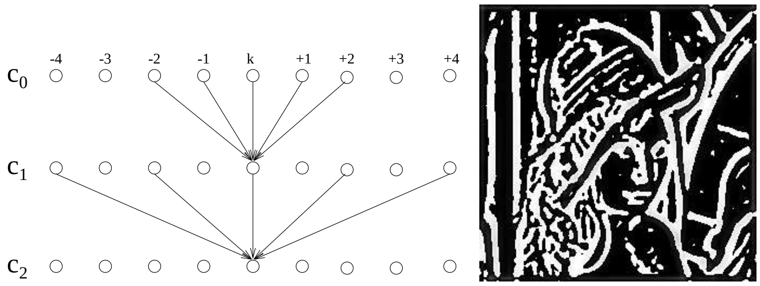
The approximation of $f(x)$ at scale $i \geq 1$ is then performed by the direct "á trous" decomposition:

$$c_i(k) = \frac{1}{2^i} \left\langle f(x), \phi\left(\frac{x-k}{2^i}\right) \right\rangle \tag{1}$$

or according to the well-known *two-scale equation* found in classical wavelet theory [3]($\phi(k) = \sqrt{2} \sum h_l \phi(2k-l)$) by the recursive "á trous" decomposition:

$$c_i(k) = \sum_l h_l c_{i-1}(k + 2^{i-1} l) \tag{2}$$

It is important to notice that the distance between samples increases by a factor 2 from scale $i-1 \quad (i>0)$ to the next one (see Fig.2.a). This fact has severe implications for possible parallelization approaches (see [8]).



(a) Decomposition scheme

(b) Detail image at level 3

Fig. 2. Á trous decomposition

The signal difference $\{c_i(k)\} - \{c_{i+1}(k)\}$ contains the information between two scales and is the discrete set associated with the wavelet transform corresponding to $\phi(x)$. The associated wavelet is therefore $\psi(x)$ and it holds:

$$\psi(x) = \phi(x) - \frac{1}{2}\phi\left(\frac{x}{2}\right) \tag{3}$$

In analogy to Eq.1 and 2 the wavelet coefficients $w_i(k)$ are computed directly by:

$$w_i(k) = \left\langle f(x), \psi\left(\frac{x-k}{2^i}\right)\right\rangle$$

or according to Eq.3 recursively by:

$$w_i(k) = c_{i-1}(k) - c_i(k)$$

The á trous algorithm is easily extensible to the two or three dimensional space by replacing the scalar variable k by a corresponding vector $\boldsymbol{k} = (s,t)$ or $\boldsymbol{k} = (s,t,u)$.

3 Data decomposition, border treatment, and programming paradigms

We investigate two different programming paradigms on multiprocessors: shared memory programming and message passing. In shared memory programming a sequential algorithm may be transformed into a parallel one by simply identifying areas which are suitable to be run in parallel i.e. in which no data dependencies exist. Subsequently, only local and shared variables need to be declared and parallel compiler directives are inserted. Therefore, shared memory programming can be performed very quickly. On the other hand, message passing requires an explicit programming of each communication event occurring among processors and is consequently very time demanding. However, message passing programs written in e.g. MPI or PVM may be used without changes on different architectures (no matter if multiprocessors or multicomputers) whereas shared memory programming always uses a native programming language.

Data decomposition seems to be no topic at first for shared memory programming, but the coarse grained programming style required for message passing demands a discussion of data decomposition strategies. Whereas there is nothing to discuss about data decomposition in the 1-D case, different possibilities exist for the 2-D and 3-D cases. The main distinction is among stripe partitioning and checkerboard partitioning (which are the 2-D cases, in the 3-D case simply a third dimension is added). Whereas checkerboard partitioning offers the obvious advantages of minimizing the block-border length at the cost of a larger number of neighbouring blocks, stripe partitioning requires only communication with two direct neighbours.

In the case of wavelet packet decompositions it has turned out [31, 13] that it is advisable to to perform a subband based data decomposition instead of the concepts mentioned before at a certain stage of the computation. This is explained briefly for the 2-D case. Fig.3 shows the data arrangement for levels $j = 0 \dots 2$: at level j we get 4^j subbands, each with 2^{Xmax-j} by 2^{Ymax-j} coefficients.

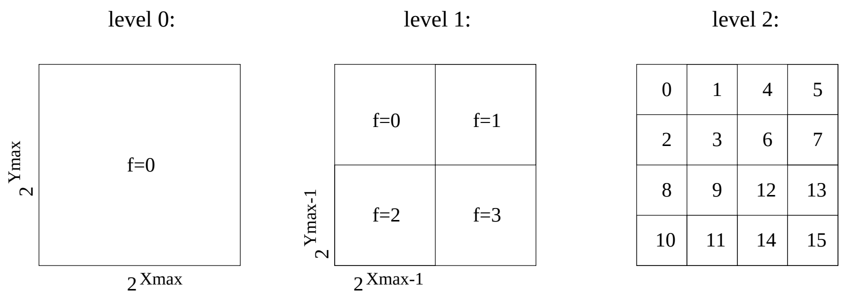


Fig. 3. Data arrangement and labeling for a 2-D wavelet packet decomposition

To do the decomposition in parallel, the data is redistributed according to the subband structure (after an initial stripe or checkerboard distribution - see Fig. 4 on the left side) at that specific decomposition level (denoted "distribution level") where the number of PEs is lower or equal to the number of subbands. Fig.4 shows the data distribution onto 4 PE ($jp = 1$) from level $j = 0$ to 2 (where the date redistribution takes place between level 0 and level 1).

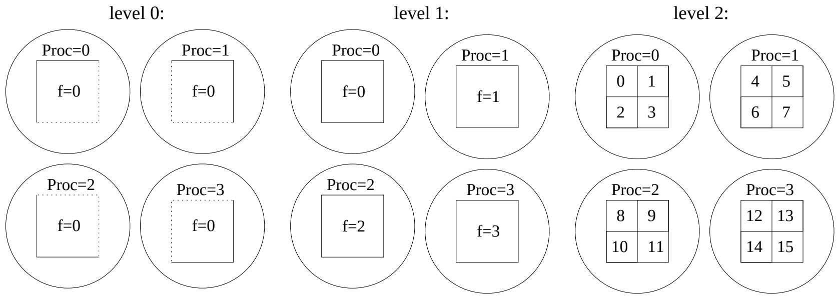


Fig. 4. Repartition of the wavelet packets onto 4 PE

Now we proceed with the discussion of border treatment for all types of wavelet transforms. In order to provide the necessary border data to each PE after an initial data distribution we may distinguish between two approaches for border treatment (compared for 1-D and 2-D fast wavelet transform in [33, 28]):

- *Data swapping* method (also known as *non-redundant data calculation*): each PE computes only non-redundant data and exchanges these results with the appropriate neighbour PE in order to get the necessary data for the next calculation step (i.e. the next decomposition level).
- *Redundant data calculation* approach: in the initialisation step we do not only provide a partial block of the original signal to a PE but broadcast the whole data set to all PE (which limits this approach to moderately sized data sets). Subsequently, in each calculation step each PE computes also redundant data in order to avoid additional communication with neighbour PEs to obtain this data.

See [8] for an exact determination of communication and computation complexity with regard to

- data swapping vs. redundant data calculation and
- data decomposition techniques

for the three types of algorithms considered.

4 Experimental Results

For the 2-D case we employ as test image a 1024×1024 pixel version of the *Lena*-image and perform a complete decomposition (i.e. 10 levels) with Daubechies $W20$ filters. In the 3-D case we use video data with several different dimensions. All the computations have been performed on a SGI POWERChallenge GR (20 IP25 processors at 195 MHz) using either the native shared memory programming language PowerC [1] or a native version of the PVM message passing library (which has been developed for the use on the POWERChallenge Array).

We start with the discussion concerning border treatment in the case of message passing. The 2-D wavelet packet decomposition is the only case where

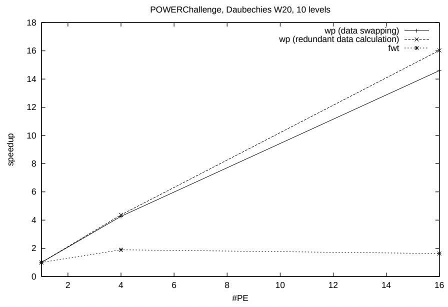


Fig. 5. 2-D wavelet and wavelet packet decomposition (PVM)

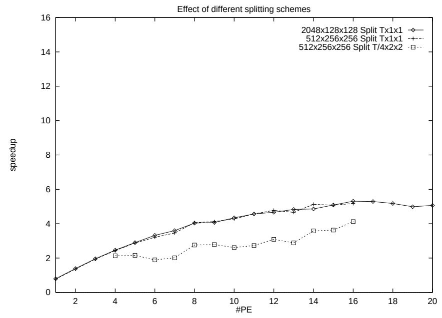
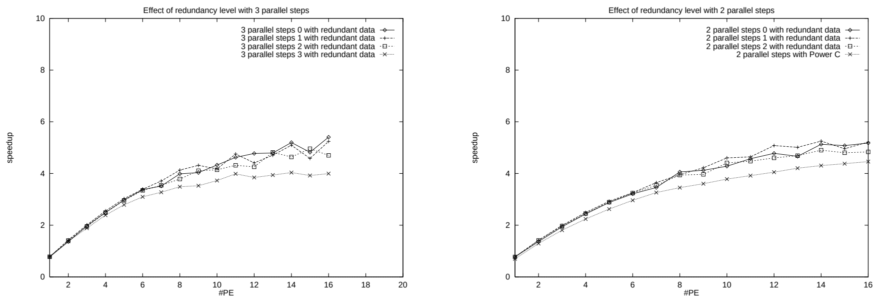


Fig. 6. Data decomposition strategies for 3-D wavelet decomposition

redundant data calculation is superior to data swapping (see Fig. 5 – note that this algorithm (following [13]) is restricted to 4^j processors). For 3-D wavelet decomposition (Fig. 7.a) and the á trous algorithm (Fig. 10) data swapping is clearly superior. The explanation for this effect is that the small amount of total computation in the 2-D wavelet packet decomposition does not justify to invoke the massage passing commands - in this case it is slightly more efficient to do some extra computations. However, almost no difference occurs in the case of 3-D wavelet packet decomposition (Fig. 8.a) – the obvious reason is that only 3 decomposition steps are performed with redundant data calculation or data swapping, the rest of the computation is performed using subband based data distribution. Therefore, the overall amount of the computations where border problems are involved is rather small and consequently it makes no difference which border-treatment approach to choose. This leads us directly to the question about data decompositions.

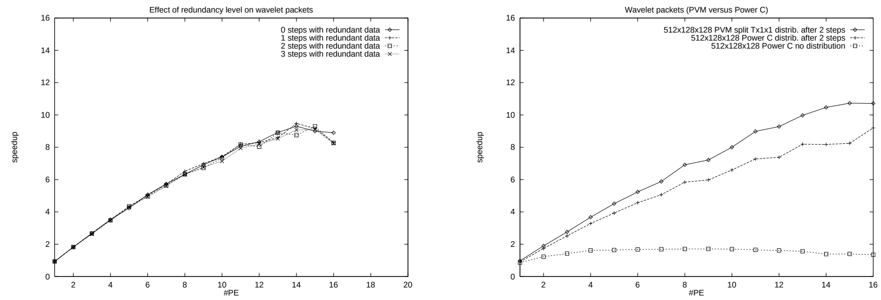


(a) Data swapping vs. redundant data comp.

(b) PVM vs. PowerC

Fig. 7. 3-D pyramidal wavelet decomposition

Fig. 6 shows that for a 3-D wavelet decomposition it is advantageous to use the stripe partitioning scheme. It is shown very clearly that the 3-D analogue to the checkerboard partitioning scheme performs much worse (which is due to the higher number of communication invokements caused by the higher number of neighbouring blocks as compared to the stripe partitioning). The advantage of subband based parallelization for wavelet packet decomposition is dramatically shown in Figs. 8.b and 9 – in both cases (3-D and 2-D) almost no speedup is achieved when “no distribution” (i.e. no subband based distribution) is performed, whereas monotonically increasing and significant speedup is achieved with subband based data distribution.



(a) Data swapping vs. redundant data comp.

(b) PVM vs. PowerC

Fig. 8. 3-D wavelet packet decomposition

Now let us proceed to the question whether message passing or shared memory programming is the better way for parallel wavelet transforms on multiprocessors. The answer is very clear for 2-D wavelet transform: when comparing Figs. 5 and 9 we immediately see that the message passing algorithm hardly reaches any speedup whereas its shared memory counterpart almost reaches linear speedup. The reason is (similar to the explanation concerning border treatment in the 2-D wavelet packet case) that the amount of computation involved does not justify to invoke communication procedures, whereas the simple structure of the data flow in the algorithm is ideally suited for shared memory programming. Considering the results for 3-D wavelet decomposition the situation changes (see Fig. 7.b). We clearly see that the algorithm implemented in PowerC exhibits slightly worse scalability as compared to the PVM case(s). This trend is continued with much higher significance for 2-D and 3-D wavelet packet decomposition (Figs. 9, 5 and 8.b). In both cases the PowerC algorithm where simply the loops corresponding to data rows or slices are distributed do not reach any speedup (“no distribution”). If for shared memory programming the subband based data distribution concept is implemented as well (which is fairly complicated to do) we obtain a considerable speedup, but still lower as compared to the message passing case (e.g. for 2-D wavelet packet decomposition speedup 8.8 vs. 16 with 16 processors, in the 3-D case speedup 9 vs. 11).

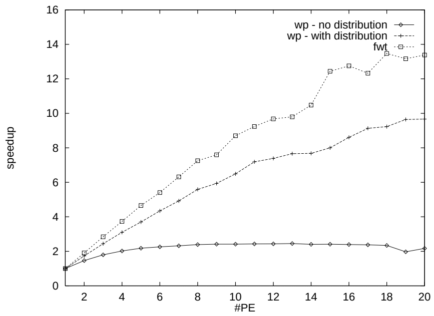


Fig. 9. 2-D wavelet and wavelet packet decomposition (PowerC)

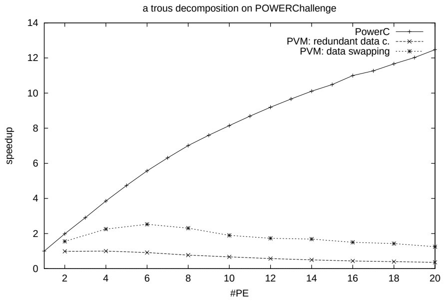


Fig. 10. 2-D á trous decomposition

A very different situation may be observed for the á trous algorithm (Fig. 10). Whereas we do not reach speedup higher than 3 using message passing, speedup close to linear is achieved with shared memory programming. This effect is on the one hand due to the high amount of computation involved in the á trous algorithm (which allows the high speedup), on the other hand due to the complicated border treatment (see section 2 – this causes the bad performing message passing approach).

Though simple to use and fast to implement the shared memory programming approach does not always lead to satisfying results as compared to the message passing algorithm. Whereas very well suited for pyramidal wavelet decomposition and the á trous algorithm the performance of a straightforward shared memory programming wavelet packet decomposition is extremely poor. Even if the subband based distribution concept is employed (which requires profund algorithmic knowledge and a significantly higher implementation effort) message passing still remains clearly superior.

5 Conclusion

In the context of parallel wavelet transforms we have discussed border treatment and data decomposition strategies and their possible implications on the choice of a proper programming paradigm for corresponding multiprocessor implementations. We have seen that pyramidal wavelet transform, wavelet packet decomposition, and the á trous algorithm exhibit very different requirements and properties concerning these topics and need to be treated quite differently in order to obtain optimal performance on a multiprocessor.

Acknowledgements

The first and second author are supported by the Austrian Science Fund FWF, project no. P11045-ÖMA.

References

1. Barr E. Bauer. *Practical Parallel Programming.* Academic Press, New York, NY, USA, 1992.
2. C. Chakrabarti and M. Vishvanath. Efficient realizations of the discrete and continous wavelet transforms: From single chip implementations to mappings on SIMD array computers. *IEEE Transactions on Signal Processing*, 3(43):759–771, 1995.
3. C.K. Chui. *An Introduction to Wavelets.* Academic Press, San Diego, 1992.
4. R.R. Coifman and M.V. Wickerhauser. Entropy based methods for best basis selection. *IEEE Transactions on Information Theory*, 38(2):719–746, 1992.
5. C.D. Creusere. Image coding using parallel implementations of the embedded zerotree wavelet algorithm. In B. Vasudev, S. Frans, and P. Sethuraman, editors, *Digital Video Compression: Algorithms and Technologies 1996*, volume 2668 of *SPIE Proceedings*, pages 82–92, 1996.
6. P. Dutilleux. An implementation of the algorithme á trous to compute the wavelet transform. In J.M. Combes, A. Grossmann, and Ph. Tchamitchian, editors, *Wavelets: Time-Frequency Methods and Phase-Space*, pages 298–304. Springer-Verlag, Berlin, 1989.
7. T. El-Ghazawi and J. LeMoigne. Multi-resolution wavelet decomposition on the MasPar massively parallel system. *ISCA*, 1(1):24–30, 1994.
8. M. Feil. Wavelet transforms on parallel architectures. Master's thesis, University of Salzburg, August 1998.
9. M. Feil and A. Uhl. Wavelet packet decomposition and best basis selection on massively parallel SIMD arrays. In *Proceedings of the International Conference "Wavelets and Multiscale Methods" (IWC'98), Tangier, 1998.* INRIA, Rocquencourt, April 1998. 4 pages.
10. M. Feil, A. Uhl, and M. Vajtersic. Continous wavelet transform on massively parallel arrays. In E.H. D'Hollander, G.R. Joubert, F.J. Peters, U. Trottenberg, and R. Völpel, editors, *Parallel Computing: Fundamentals, Applications and New Directions*, number 12 in Advances in Parallel Computing, pages 207–210. North Holland, 1998.
11. J. Fridman and E.S. Manolakos. On the scalability of 2D discrete wavelet transform algorithms. *Multidimensional Systems and Signal Processing*, 8(1–2):185–217, 1997.
12. I. Gertner, R. Peskin, and S. Walther. Parallel computation of the continous wavelet transform. In *Adaptive Signal Processing*, volume 1565 of *SPIE Proceedings*, pages 414–422, 1991.
13. E. Goirand, M.V. Wickerhauser, and M. Farge. A parallel two-dimensional wavelet packet transform and some applications in computing and compression analysis. In R. Motard and B. Joseph, editors, *Applications of Wavelet Transforms in Chemical Engineering*, pages 275–319. Kluwer Academic Publishers Group, 1995.
14. M. Hamdi. Parallel architectures for wavelet transforms. In *Proceedings of the 1993 Workshop on Computer Architectures for Machine Perception*, pages 376–384. IEEE Computer Society, 1993.

15. M. Holström. Parallelizing the fast wavelet transform. *Parallel Computing*, 21(11):1837–1848, 1995.
16. C. Koc, G. Chen, and C. Chui. Complexity analysis of wavelet signal decomposition and reconstruction. *IEEE Trans. on Aereospace and Electronic Systems*, 30(3):910–918, July 1994.
17. D. Krishnaswamy and M. Orchard. Parallel algorithm for the two-dimensional discrete wavelet transform. In *Proceedings of the 1994 International Conference on Parallel Processing*, pages III:47–54, 1994.
18. G. Lafruit and J. Cornelius. Parallelization of the 2D fast wavelet transform with a space-filling curve image scan. In A.G. Tescher, editor, *Applications of Digital Image Processing XVIII*, volume 2564 of *SPIE Proceedings*, pages 470–482, 1995.
19. E. Lega et al. A parallel algorithm for structure detection based on wavelet and segmentation analysis. *Parallel Computing*, (21):265–285, 1995.
20. M. Misra and T. Nichols. Computation of 2-D wavelet transforms on the CM-2. In *Proceedings of the International Conference on Applications in Parallel and Distributed Computing*, 1994.
21. M. Misra and V.K. Prasanna. Parallel computation of 2-D wavelet transforms. In *Proc. of the 11th IAPR Int. Conference on Pattern Recognition*, volume IV, pages 111–114. IEEE Comput. Soc. Press, 1992.
22. L. Bacchelli Montefusco. Parallel numerical algorithms with orthonormal wavelet packet bases. In C.K. Chui, L. Montefusco, and L. Puccio, editors, *Wavelets: Theory, Algorithms and Applications*, pages 459–494. Academic Press, San Diego, 1994.
23. J.N. Patel, L.H. Jamieson, and A.A. Khokhar. On the scalability of 2-D wavelet transform algorithms on fine-grained parallel machines. In *Proceedings of the 1996 International Conference on Parallel Processing*, 1996.
24. J.N. Patel, A.A. Khokhar, and L.H. Jamieson. Scalability of 2-D wavelet transform algorithms: analytical and experimental results on coarse-grain parallel computers. In *Proceedings of the 1996 IEEE Workshop on VLSI Signal Processing*, pages 376–385, 1996.
25. M.M. Pic and H. Essafi. Wavelet transform on Connection Machine and SYMPATI 2. *International Journal of Modern Physics C*, 4(1):97–103, 1993.
26. M.J. Shensa. Wedding the à trous and Mallat algorithms. *IEEE Trans. on Signal Process.*, 40(10):2464–2482, 1992.
27. E. Slezak, A. Bijaoui, and G. Mars. Identification of structures from galaxy counts: Use of the wavelet transform. In *Astron. Astrophys.*, volume 22, pages 301–316, 1990.
28. S. Sullivan. Vector and parallel implementations of the wavelet transform. Technical report, Center for Supercomputing Research and Development, University of Illinois, Urbana, 1991.
29. A. Uhl. Adapted wavelet analysis an moderate parallel distributed memory MIMD architectures. In A. Ferreira and J. Rolim, editors, *Parallel Algorithms for Irregulary Structured Problems*, volume 980 of *Lecture Notes in Computer Science*, pages 275–284. Springer, 1995.
30. A. Uhl. Vector and parallel wavelet transforms for the analysis of time varying signals. In R.S. Schreiber et al., editors, *Proceedings of the seventh SIAM conference on parallel processing for scientific computing*, pages 9–14, 1995.
31. A. Uhl. Wavelet packet best basis selection on moderate parallel MIMD architectures. *Parallel Computing*, 22(1):149–158, 1996.
32. M.V. Wickerhauser. *Adapted wavelet analysis from theory to software*. A.K. Peters, Wellesley, Mass., 1994.

33. M-L. Woo. Parallel discrete wavelet transform on the Paragon MIMD machine. In R.S. Schreiber et al., editor, *Proceedings of the seventh SIAM conference on parallel processing for scientific computing*, pages 3–8, 1995.