Fast Implementation of Discrete Wavelet Transform Based on Pipeline Processor Farming

H. Sava, M. Fleury, A. C. Downton, A. F. Clark

Keywords—

Introduction

Wavelets and Wavelet transforms have been one of the most important developments in image analysis over the last decade, especially for applications such as data compression, segmentation and vision [1,2].

The wavelet transform (WT) has been described in a number of different ways by various authors. It can be described as a modification of the short-time Fourier transform, the decomposition of a signal $s(t)$ into a set of basis functions, and as an equivalent of sub-band signal decomposition[6]. Throughout all these descriptions there is one constant, the equation for the wavelet transform:

$$W(b,a) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} h^* \left(\frac{t-b}{a} \right) s(t) dt \qquad (1)$$

where $h^*(t)$ is the complex conjugate of the wavelet $h(t)$, a is a term describing scale and b represents a time-shift value.

Efficient implementations of wavelet transforms have been derived, based on the FFT and short-length 'fast-running FIR algorithms'[3]. However, for long one-dimensional arrays or two dimensional data, such as encountered in image processing, the time required to calculate wavelet transforms, even in the case of 'fast' FFT-based implementations, is still large.

In order to reduce the time consumption of the wavelet transform and bring it closer to real-time implementation, this paper suggests the use of parallel processing based on the pipeline processor farming (PPF) methodology. The paper is mainly focussed on parallel implementation of the discrete wavelet transform (DWT), which is extensively used in image processing applications. The parallel environment in which the algorithms were implemented comprised two TMS320C40 boards with a total of six processors.

Discrete Wavelet Transform (DWT)

Implementation of the DWT can be introduced by considering sub-band decomposition — the digital filter equivalent of the DWT. The filter bank structure common to both subband decomposition and DWT can be implemented efficiently using a tree structure. The complexity of the DWT is linear in the number of input samples, with a constant factor that depends on the length of the filter.

Department of Electronic Systems Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, U.K, tel: +44 1206 872795, fax: +44 1206 872900, e-mail: herkole@essex.ac.uk

The total complexity of the DWT is bounded by

$$C_{total} = C_0 + \frac{C_0}{2} + \frac{C_0}{4} + \frac{C_0}{8} \cdots < 2C_0 \qquad (2)$$

where C_0 is the operations per input sample required for the first filter bank [5]. The drawback in the DWT implementation is the delay associated with such an iterated filter, which grows exponentially with the number of stages. Equation (2) represents the key for parallel implementation of the DWT based on the Pipeline Processor Farm (PPF) methodology [7]. From Equation (2) it is clear that a pipeline of J stages (*i.e.* J is the number of octaves over which the analysis is performed), each containing half as many workers as the previous farm, would lead to a totally balanced topology. However, this fine-grain topology would require a large number of processors and would introduce significant communication overheads. A more practical solution is the topology presented in Figure 1(a). This can be regarded as three-stage pipeline processing.

- The first farmer is responsible for reading the data and distributing it to the first set of workers which compute the first $N/2$ filter coefficients (i.e., the first filter bank). This requires C_0 operations per input sample in total.
- The second farmer collects the results from the first stage and computes the remainder of the coefficients, by distributing data to the second set of workers.
- The final farmer gathers the overall results from the second stage workers.

From Equation (2) it can be seen that the total number of computations in the second stage tends to C_0 as the number of octaves analysed increases. Hence, since the wavelet transform is regular and static (execution time is independent of the input data), the number of processors for both stages should be equal to achieve a balanced pipeline, as shown in Figure 2(a). As long as the same number of workers is used in each stage however, the overall computational performance of the parallel system can be scaled up incrementally by exploiting the data parallelism within the application.

In practice, due to the limited number of processors available in our development system, two workers were employed in each of the two stages of the pipeline, and farmers 2 and 3 were combined with one of the workers in pipeline stage 2 as shown in Figure 2(b).

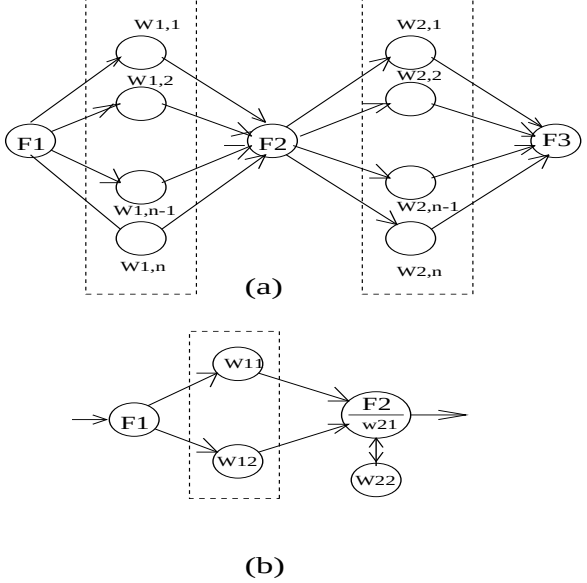

Figure 1. Pipeline architecture of DWT transform.

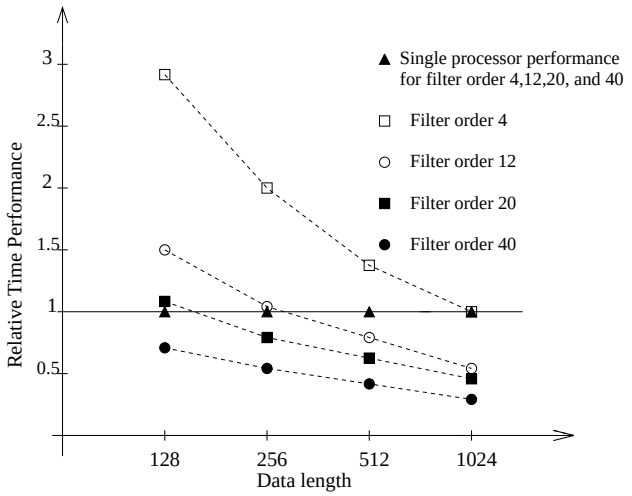

Figure 2. Comparative overall performance of parallel topology and sequential single processor as a function of data and filter length.

A class of Daubechies wavelet filters with coefficient length from 4 to 40 was implemented [8]. A sequence of five one-dimensional synthesised signal 'frames' was used as the input signal. Figure 2 gives the execution time performance of the parallel topology compared with sequential implementation. From these results it can be seen that the parallel implementation of the wavelet transform outperforms the sequential algorithm in terms of execution speed for frame lengths of 256 or more samples and coefficient lengths greater than 12. The best performance is obtained for $N = 1024$ and $L = 40$, in which case a speed-up of almost three is achieved with 4 workers. It is clear from Figure 2 that the performance advantage of the parallel implementation increases as the data length increases. This is particularly important if one bears in mind that a wavelet transform of a d-dimensional array is obtained most easily by transforming the array sequentially on its first index (for all values of its other indices), then on its second, and so on. Thus the approach to parallelisation reported here should be particularly useful in applications such as image compression or computer vision where the amount of data to be processed is substantial, and a significant degree of parallelism is required to obtain real-time performance.

Conclusion

This paper has described a scalable parallel implementation of the discrete wavelet transform based on the pipeline processor farming methodology. Results suggest that, even for small-scale applications with a relatively small number of processors, the parallel implementation out-performs a single processor in execution speed. More importantly, the efficiency of the parallel implementation tends to improve as the data size and filter length increase and the communication to computation ratio reduces, suggesting that parallel implementations of the DWT could be particularly attractive in data-intensive applications such as computer vision, image processing or image coding. Finally, the proposed topology is incrementally scalable, making it possible to utilise exactly the number of processors required to achieve real-time performance for any particular wavelet application.

References

[1] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, pp. 674-693, July 1989.

[2] S. Mallat, "Multi-frequency Channel Decompositions of Images and Wavelet Representation", *IEEE Trans. on Acoustic, Speech, and Signal Processing*, Vol. 37, pp. 2091-2110, 1989.

[3] O. Rioul, M. Vetterli, "Wavelet and Signal Processing", *IEEE Signal Processing Magazine*, pp. 14-38, Oct. 1991.

[4] M. Vetterli, C. Herley, "Wavelets and Filter Banks: Theory and Design", *IEEE Trans. on Signal Processing*, Vol. 40, No. 9, pp. 2207-2232.

[5] O. Rioul, P. Duhamel, "Fast Algorithms for Discrete Continuous Wavelet Transforms", *IEEE Trans. on Information Theory*, Vol. 38, No. 2, pp. 569-586, March 1992.

[6] P. M. Bentley, J.T.E. McDonnell, "Wavelet Transforms: an Introduction", *IEE Electronics and Communication Engineering Journal*, Vol. 6, No. 4, pp. 175-186, 1994.

[7] A. C. Downton, R.W. S. Tregidgo, A. Cuhadar, "Generalised Parallelism for Embedded Vision Applications", in *Parallel Computing: Paradigms and Applications*, A. Y. Zomaya Edt, pp. 553-577, Int. Thomson Computer Press, London, 1996.

[8] W. H. Press, W. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical Recipes in C" *(Second Edition)*, pp. 591-606, Cambridge University Press, Cambridge 1992.