Parallel Pipeline Implementation of Wavelet Transforms

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

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: fleum@essex.ac.uk

Abstract

Wavelet transforms have been one of the important signal processing developments in the last decade, especially for applications such as time-frequency analysis, data compression, segmentation and vision. Although several efficient implementations of wavelet transforms have been derived, their computational burden is still considerable. This paper describes two generic parallel implementations of wavelet transforms based on the pipeline processor farming methodology which have the potential to achieve real-time performance. Results show that the parallel implementation of the over-sampled Wavelet Transform achieves virtually linear speedup, while the parallel implementation of the Discrete Wavelet Transform (DWT) also out-performs the sequential version, provided that the filter order is large. The DWT parallelisation performance improves with increasing data length and filter order while the frequency domain implementation performance is independent of wavelet filter order. Parallel pipeline implementations are currently suitable for processing multi-dimensional images with data length at least 512 pixels.

1 Introduction

The theory of wavelets has roots in quantum mechanics and the theory of functions though a unifying framework is a recent occurrence. In the last decade, the application of wavelet theory to a number of areas including data compression [1], image processing [2], time-frequency spectral estimation [3, 4] and applied mathematics [5] has become a significant feature in the literature. Several discrete algorithms have been devised for computing wavelet coefficients. The Mallat algorithm [1, 2] and the 'à trous' algorithm [6] are well established. Shensa [7] originally provided an unified approach for the computation of discrete and continuous wavelet transforms. Efficient implementation of wavelet transforms have been derived based on the Fast Fourier transform (FFT) and short-length 'fast-running FIR algorithms' [10] in order to reduce the computational complexity per computed coefficient. However, the computational burden of the wavelet transform is still large. In order to reduce the time required to calculate the wavelet transform and bring it closer to real-time implementation, this paper suggests the use of pipelined parallel processing. Am alternative medium-grained parallelization for an adaptive transform is given in [11] and fine-grained implementations are surveyed in [12].

2 The Wavelet Transform

Wavelet analysis is performed using a prototype function called a 'wavelet' which has the effect of a bandpass filter. The other bandpass filters (wavelets) are scaled versions of the prototype [9, 8].

The wavelet transform (WT) has been described in a number of different ways. It

can be described as a modification of the short-time Fourier transform (STFT) [8], the decomposition of a signal $s(t)$ into a set of basis functions [3] and as an equivalent of sub-band signal decomposition [13]. The fundamental equation for the wavelet transform is:

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

where $h^*(t)$ is the complex conjugate of the wavelet $h(t)$, a is a term describing scale, b represents a time-shift value, and the term $\sqrt{a}^{-1}$ preserves energy.
There are several types of WT:

- the continuous wavelet transform (CWT) in which the time and time-scale parameters vary continuously;
- the wavelet series (WS) coefficients, in which time remains continuous but time-scale parameters (b, a) are sampled on a 'dyadic' grid [8]; and
- the discrete wavelet transform (DWT), in which both time and time-scale parameters are discrete.

Depending on the application, one of these types of WT may be selected over the others. The CWT is most suitable for signal analysis [4]; WS and DWT have been employed for signal coding applications, including image compression [14], and various tasks in computer vision [1, 2].
Rioul and Duhamel [10] proposed the use of fast convolution techniques (such as the FFT) or 'fast-running FIR filtering' techniques in order to reduce the computational complexity. For the case of large filter lengths, L, with the same order of magnitude as the data length, N, by using the FFT-based technique on the 'à trous' algorithm the computational complexity per computed coefficient is reduced from $2L$ to $4log_2 L$. For the case of short filters where $L << N$,

modest gains in efficiency are obtained. Clearly, the FFT-based fast wavelet transform algorithms provide significant computational savings for large filter lengths. However, DWTs have so far been mostly used with short filters [8]. Furthermore, for a long one-dimensional array or two dimensional data, such as in image processing, the elapsed processing time for wavelet transforms, even in the case of 'fast' FFT-based implementations, is still large.

This paper describes the parallelisation of two of the most common manifestations of the WT: the DWT used in image processing and data compression, and the oversampled WT used in time-scale spectral estimation.

The parallel environment in which the algorithms were implemented comprised two TMS320C40 (C40) boards with a total of six processors (one dual-C40B board and one quad-C40B module board) [15]. The C40 runs nominally at 50 MHz with six bidirectional ports each capable of communicating at a peak of 20 Mbps.

3 Parallel Implementation of Discrete Wavelet Transform (DWT)

Borrowing from the field of digital filters, e.g. [16], the idea of implementing the DWT by means of a sub-band decomposition has emerged [2, 7, 10]. Sub-band decomposition maps onto a hierarchical structure (Figure 1 [1, 13]) in which half the samples from the previous stage are processed. The output of the tree-like structure is equivalent to a bank of adjacent bandpass filters, spread logarithmically over the frequency, unlike the short-time Fourier transform (STFT) which is equivalent to a linearly-dispersed set of filters [8].

The standard DWT algorithm, implemented directly as a filter bank, is already fast [17, 10]. The DWT is 'fast' as a result of the decomposition of the computation into elementary cells and the sub-sampling operations ('decimation') which occur at each stage. 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. The total complexity of the DWT is bounded by

$$C_{total} = C_0 + \frac{C_0}{2} + \frac{C_0}{4} + \frac{C_0}{8} \cdots < 2C_0 \tag{2}$$

where C_0 is the number of operations per input sample required for the first filter bank [8].

A potential drawback for a straightforward parallelization is that the computational complexity declines exponentially at each stage. However, equation (2) remains the key to parallel implementation of the DWT based on the Pipeline Processor Farm (PPF) methodology [18]. The PPF methodology proceeds from the observation that embedded signal-processing systems with continuous data flow may be characterised as consisting of a series of independent processing stages. It therefore maps the sequential algorithm structure to a generalised parallel architecture based upon a pipeline of stages with well-defined data communication patterns between them. Each stage of the pipeline then exploits parallelism in the most appropriate way, for example data parallelism applied at various levels, algorithmic parallelism, or temporal multiplexing of complete data sets. 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), with each of them containing half as many workers as the previous farm would lead to a totally balanced topology. However, this fine-grained topology would require a large

number of processors and would introduce significant communication overheads. A more practical solution, still based on pipeline processor farming, is the topology presented in Figure 2(a).

This can be regarded as two-stage pipeline processing:

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

According to Equation (2) it can be seen that the total number of computations carried out by the second stage tends to C_0 as the number of octaves analysed increases. Hence, the number of processors for the second farm should be the same as for the first farm in order to have a balanced pipeline.

Figure 2(a) represents a general scalable topology in terms of the number of workers that can be connected to farmers $F1$ and $F2$. However, due to the limited number of processors available in our case, only two workers were employed in the two stages of the pipeline. In this case the topology of Figure 2(a) can be modified slightly, bearing in mind that farmer $F2$ and $F3$ and one of the workers in the second pipeline stage can be implemented on the same processor without affecting the performance. This follows from the following analysis: $F2$ performs only the required permutations of the data outputs received from workers $W21$ and $W22$ after they have finished the octave analysis. Hence, $F2$ is idle during the working time of $W21$ and $W22$. Consequently, the topology of Figure 2(b) was used for our implementation, requiring only five processors.

Figure 3 compares performance of the parallel topology shown in Figure 2(b) with a sequential implementation on a single processor for different data and filter length. The ordinates record 'relative time performance', i.e. all times are on a scale in which sequential processing takes unit time. Timings include data load from the PC host machine, which had local storage facilities, and the overhead from transferring data to worker processors.

A set of Daubechies wavelet filters for coefficient lengths from 4 to 40 was implemented [19]. A sequence of five one-dimensional synthesised signal 'frames' (with a frequency response similar to speech) was used as the input signal. Due to memory restrictions on the four C40 module boards, the data length was restricted to a maximum of 1024 points. For filter order four in particular the communication overhead is greater than computation. Where computation is much smaller, e.g. for data length 128, filter order four performs approximately three times more *slowly* than a sequential implementation due to the poor ratio of compute time to communication time. From results in Figure 3 it can be seen that the parallel implementation of the wavelet transform outperforms the sequential algorithm in terms of speed-up for $N \geq 256$ and $L \geq 12$. The best performance is obtained for $N = 1024$ and $L = 40$, in which case a speed-up of almost three is achieved with four worker processors. It is clear from Figure 3 that the performance advantage of the parallel implementation increases compared to a sequential implementation as the data length increases. This finding is important as a wavelet transform of a d-dimensional array is most easily obtained by transforming the array sequentially on its first index (for all values of its other indices), then on its second, and so on. Therefore, this parallelisation of the DWT is suitable for large filter orders which are more 'regular' (preserving

image smoothness [20]), while short filters should be processed sequentially. In computer vision, the use of differing-sized filters, including large-sized filters, stems from [21], though Laplacian of Gaussian filters were used in that instance. Lower-order filters are commonly preferred in image compression because of the reduction in computation but in [14] a filter of order fifteen is employed as part of a biorthogonal coding/decoding scheme, which makes for smoothness and linear phase in the resulting images.

4 Parallel Implementation of Oversampled WT

Because of the reduction in computation the octave band analysis of Section 3 is appropriate for image compression, and has also been used in image enhancement and image fusion [22]. For other purposes, e.g. [23], a less sparse coverage of the frequency range is required. Previously, spectrograms resulting from the STFT have been employed for this purpose. In Figure 4(a) [2, 8, 13] the octave band time-frequency sampling is contrasted to the fuller scheme of Figure 4(b).

Equation (3) is a discrete version of (1), suitable for production of a spectrogram, where a normalized sampling interval is assumed:

$$W[i,a] = \frac{1}{\sqrt{a}} \sum_{n=0}^{N-1} h^* \left[\frac{[n-i]}{a} \right] s[n], \tag{3}$$

with N the number of samples in the signal, and $i = 0, 1, \cdots N-1$. The signal is now oversampled within each octave, unlike the subband decomposition, which is critically-sampled. Additionally, within each of J octaves one can have M voices [10] resulting in an indexing of the scale factor, a, as

$$a = 2^{j+m/M},\ m = 0, 1, \cdots, M-1, \tag{4}$$

where $j = 0, 1, \cdots, J$. The discrete version of the CWT results in a magnitude and phase plot with N samples along the time-shift axis and JM samples along the scale axis. Using Equation (3) to generate a grid with M voices per octave, as in Figure 4(b), requires $2LJM$ multiplications/point and $2MJ(L-1)$ additions/point [10] with the 'à trous' algorithm in which a high- and low- pass filter are employed at each processing stage.

The algorithm in this paper originates from examining the CWT in the frequency domain. For even wavelets, by virtue of the Convolution Theorem and since the Fourier transform is shift invariant

$$F\left[W(b,a)\right] = \frac{1}{\sqrt{a'}} H^*\left(\frac{\omega}{a'}\right) S(\omega),\ a' = a^{-1}, \tag{5}$$

where $F(\cdot)$ is the Fourier operator. The processing of Equation (5) has been analysed previously as in Figure 5 [13].

The FFT of the wavelet is pre-calculated. Any suitable FFT is possible but if $N = 2^n$ then a split-radix FFT, which uses a four-way butterfly for odd terms of a radix-2 transform, has minimal time complexity. The overall complexity for real data when performed sequentially is given as $[6 + (n-3) + 2/N]\, MJ$ multiplications/point and $[6 + (3n-5) + 4/N]MJ$ additions/point [10, 13], assuming three real multiplications and adds for each complex multiplication. The FFT-based algorithm is selected not only because it results in reduced complexity compared with the direct implementation given by Equation (3), but also because it represents a straightforward parallel implementation as shown in Figure 6.
Processor 1 reads the data and estimates the signal's FFT. The second layer of processors can be increased to as many as JM. Each processor performs a convolution between the signal's FFT and respective wavelet's FFT and then calculates the resulting IFFT in order to generate the corresponding wavelet transform coef-

ficients. It should be noted that the wavelet coefficients for each worker can be pre-calculated and stored in a look-up table by each of these processors. The third-layer processor is responsible for arranging the data in the proper order and storing them in the appropriate format.

Due to the number of processors available, the topology investigated in this paper included up to four processor in the second layer. Hence, the analysis was carried out for four octaves or two octaves with two voices. A stream of five one-dimensional 'frames' of synthetic data was used as the input signal. A Morlet wavelet (frequency-modulated Gaussian) was used because of its direct scale/frequency property[1], making it pertinent to time-frequency analysis [13]. For the architecture of Figure 6 with four worker processors a speed-up of 4.32 was obtained compared to the sequential implementation. The wavelet length is not included as a parameter in this case because the algorithm uses zero padding on the sides of scaled wavelets to match the data length in order to perform the convolution in the frequency domain. Hence, the time performance is independent of wavelet length.

This finding suggests that the efficiency of the parallel implementation in this case is high, and that parallel implementation will therefore be especially useful in the analysis of large number of octaves and many voices. Finally, if it is required to achieve balance with a large farm of workers, the time performance can be further improved by parallel implementation of the FFT computation stage using data-farming techniques [24].

[1] A Gaussian is the only waveform that fits Heisenberg's inequality with equality.

5 Conclusion

This paper describes two parallel implementations of the wavelet transform based on pipeline processor farming. Results suggest that with a small number of processors parallel implementations will out-perform single processor sequential solutions, provided that in spatial-domain parallelisations the filter order is large. In the case of the DWT, the parallel performance improves as the data size and filter length increase, which implies that DWT parallel implementations could be of particular value in applications such as computer vision, image processing or data compression. In the case of the finer–sampled WT, a highly efficient parallel implementation is achieved and further performance improvement can be obtained by parallelising the initial signal FFT as well as the second stage handling of the convolution and inverse FFTs.

Acknowledgement

This work was carried out under EPSRC research contract GR/K40277 'Parallel software tools for embedded signal-processing applications' as part of the EPSRC PSTPA (Portable Software Tools for Parallel Architectures) directed programme.

References

[1] Mallat, S.: 'A theory for multiresolution signal decomposition: The wavelet representation', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1989, **11**, (7), pp. 674–693

[2] Mallat, S.: 'Multi-frequency channel decompositions of images and wavelet representation', *IEEE Trans. on Acoustic, Speech, and Signal Processing*, 1989, **37**, pp. 2091–2110

[3] Kronland-Martinet, R., Morlet, J., Grossmann, A.: 'Analysis of sound patterns through wavelet transforms', *Int. J. of Pattern Recognition and Artificial Intelligence*, 1987, **1**, pp. 273–302

[4] 'Time-frequency and wavelet analysis', *IEEE Eng. in Medicine and Biology Mag.*, Special Issue, M. Akay (Edt), March/April 1995, **14**, (2)

[5] Lemarie, P. G., Meyer, Y.: 'Ondelettes et bases Hilbertiennes', *Revista Matematica Ibero Americana*, 1986, **2**

[6] Holschneider, M., Kronland-Martinet, R., Tchamitchian, Ph.: 'A real-time algorithm for signal analysis with the help of the wavelet transform'. Wavelets, Time-Frequency Methods and Phase Space, J.M. Combes, A. Grossmann, Ph. Tchamitchian, Eds, (Springer, IPTI, Berlin, 1989), pp. 286–297

[7] Shensa, M.: 'The discrete wavelet transform: Wedding the À trous and Mallat algorithms', *IEEE Trans. on Signal Processing*, 1992, **40**, (10), pp. 2464–2482

[8] Rioul, O., Vetterli, M.: 'Wavelet and signal processing', *IEEE Signal Processing Mag.*, Oct. 1991, pp. 14–38

[9] Vetterli, M., Herley, C.: 'Wavelets and filter banks: Theory and design', *IEEE Trans. on Signal Processing*, 1992, **40**, (9), pp. 2207–2232

[10] Rioul, O., Duhamel, P.: 'Fast algorithms for discrete continuous wavelet transforms', *IEEE Trans. on Information Theory*, 1992, **38**, (2), pp. 569–586

[11] Uhl, A., Bruckmann, A.: 'Double tree wavelet compression on parallel MIMD computers'. $6^{\underline{th}}$ Int. Conf. on Image Processing and its Applications, IEE Conf. Pub. 443,1997, pp. 179-183

[12] Chakrabarti, C., Vishvanath, M. 'Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers', *IEEE Trans. on Signal Processing*, 1995, **43**, (3), pp. 759-771

[13] Bentley, P. M., McDonnell, J. T. E.: 'Wavelet transforms: An introduction', *IEE Elect. and Comm. Eng. J.*, 1994, **6**, (4), pp. 175–186

[14] Antonini, M., Barlaud, M., Daubechies, I., Mathieu, P., "Image coding using wavelet transforms', *IEEE Trans. on Image Processing*, **1**, 1992, (2), pp. 205–220

[15] 'DPC/C40B Technical Reference Manual', LSI. Ltd., Epinal Way, Loughborough, UK, January 1993.

[16] Smith, M. J. T., Barnwell, T. P.: 'Exact reconstruction techniques for tree-structured subband coders', *IEEE Trans. Acoust., Speech, & Sig. Proc.*, 1986, **34**, (3), pp.434–440

[17] Ramstad, T. A., Saramäki, T.: 'Efficient multi-rate realisation for narrow transition-band FIR filters'. IEEE Int. Symp. Cir. and Systems, 1988, pp. 2019–2022

[18] Downton, A. C., Tregidgo R. W. S., Çuhadar, A.: 'Generalised parallelism for embedded vision applications'. Parallel Computing: Paradigms and Applications, A. Y. Zomaya (Edt), Int. Thomson Computer Press, London, 1996, pp. 553–577

[19] Press, W. H., Teukolsky, W. A., Vetterling, W. T., Flannery, B. P.: 'Numerical Recipes in C',(Cambridge University Press, Cambridge 1992), pp. 591-606

[20] Cohen A., Ryan, R. D.: 'Wavelets and multiscale signal processing', (Chapman & Hall, London, 1995)

[21] Marr, D., Hildreth. E.: 'Theory of edge detection', *Proc. of the Roy. Soc. of London B*, 1980, **207**, pp. 187-217

[22] Castleman, K. R.:'Digital image processing',(Prentice Hall, Englewood Cliffs, NJ, 1996)

[23] Williams, W. J., Zaveri, H. P., Sackellares, J. C.:'Time-frequency analysis of electrophysiology signals in epilepsy', *IEEE Eng. in Medicine and Biology Mag.*, March/April 1995, pp. 133–143

[24] Fleury, M., Clark, A. F.: "Parallelising 2-D frequency transforms in a flexible manner', submitted to *IEE Part I (Vision, Image and Signal Processing)*, 1996

List of Figures

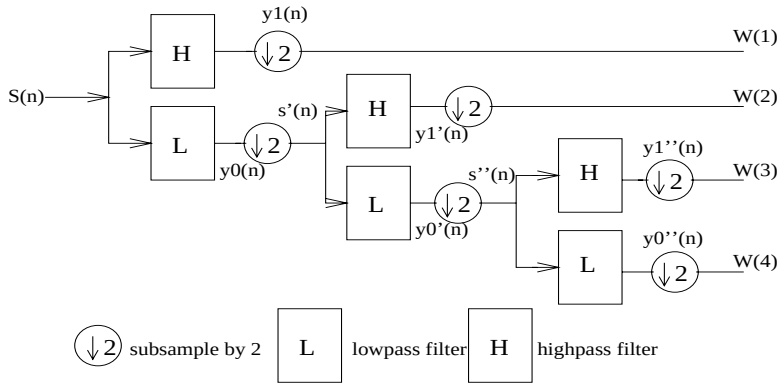

Figure 1: Block diagram of the DWT implemented as a sub-band decomposition using a filter bank tree

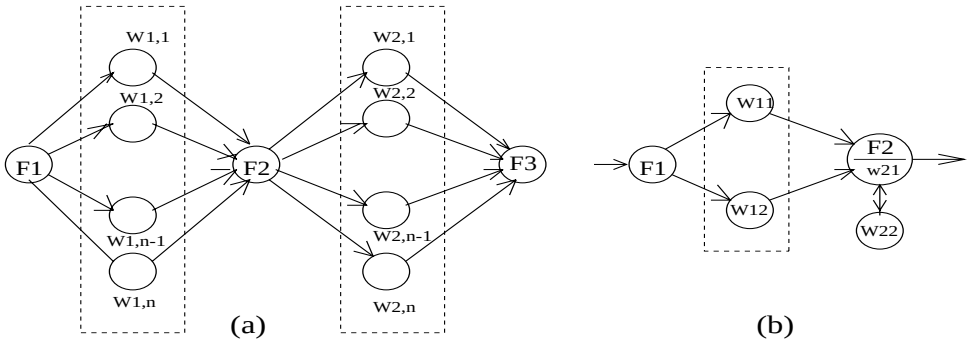

Figure 2: Pipeline architecture of DWT transform

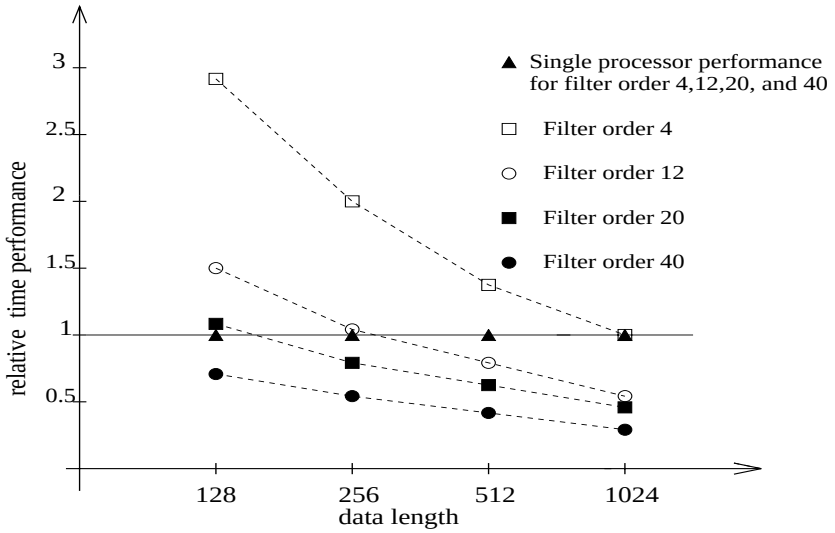

Figure 3: Comparative overall performance of parallel topology and sequential single processor as a function of data and filter length

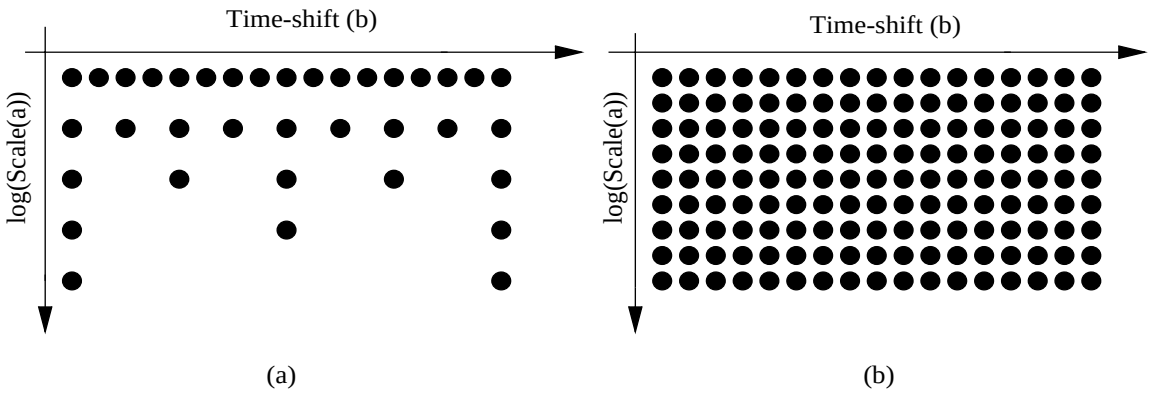

Figure 4: Sampling of the time-scale plane: (a) DWT, (b) finer sampling in scale and no sub-sampling

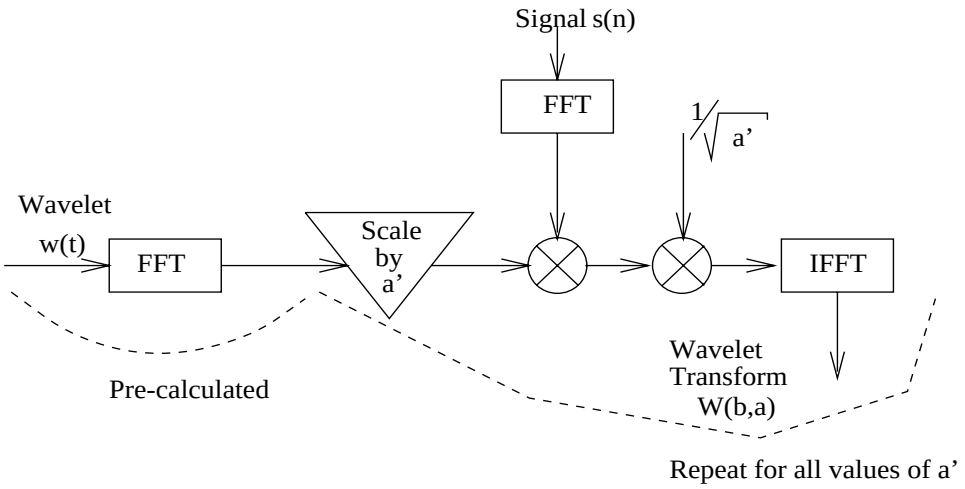

Figure 5: Block diagram of the FFT-based fast WT

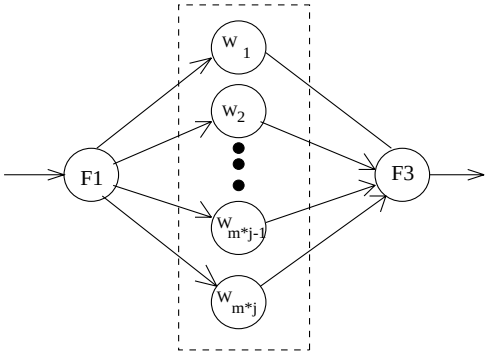

Figure 6: Pipeline architecture implementation of finer sampling in scale of WT