NATIONAL INSTITUTE OF ELECTRICITY AND ELECTRONICS

INELEC - BOUMERDES

DEPARTMENT OF RESEARCH

THESIS

Presented in partial fulfillment of the requirement of the

DEGREE OF MAGISTER

In Electronic Systems Engineering

by

Aïcha YAHIAOUI

DESIGN & IMPLEMENTATION OF A PARALLEL WAVELET TRANSFORM ALGORITHM USING UNITY MODEL AND SYSTOLIC ARRAYS

Registration Number 01/ 97

ABSTRACT

In the past two decades, there has been a tremendous progress in the development and analysis of parallel programs for different applications. Since the time required to solve a problem by a traditional computer is significantly reduced when using a parallel computer, we worked on the design and analysis of a parallel algorithm to solve the wavelet transform because the analysis and synthesis using wavelets has became a very useful tool for analyzing non-stationary signals in completion of the windowed Fourier transform which is the most known time-frequency representation.

The wavelet transform is a time-frequency representation describing at the same time the temporal evolution and the frequency content of the signal. Its theory has been used in different domains such as signal processing, analysis of seismo-reflection signals, etc.

In this thesis, we restate the wavelet transform and propose a methodology to transform an initial specification algorithm by a series of refinements into a specification that can be implemented directly on a particular architecture as a mesh SIMD systolic architecture based on transputers using Occam language. The specified algorithm PWT was formally defined using the program notation UNITY. The use of the UNITY model allows a simple description of the parallel algorithm and a well-defined model for analyzing the performance using the *running time* as a performance metric. The running time is expressed as a function of the input size and number of processors used. The merit of such an approach is the fact that programs are viewed like mathematical objects derived straightforwardly from their specifications that can be mapped to a variety of different architectures, in contrast to the *"a posteriori"* verifications commonly found in literature.

The signals analyzed are generated analytically. Mainly, the results of the wavelet transform allow one to visualize the different frequencies composing the signal as well as the instants of their apparitions and to determine for frequency modulated signals, the significant components of the signal contained in every frequency band and the time t which corresponds to the maximum energy which is impossible to do with a frequency analysis only. The PWT algorithm improves the time complexity of the basic algorithm by an exponential factor and allows a good use of the wavelet transform as it is comparable to the direct computation of the transform with a computation reduction time. It is very useful for data compression since data values of the wavelets are obtained through data re-recirculation over the wavelet matrix while the "a trous" algorithm leads to an excessive data redundancy.

Acknowledgments

I wish to express my deepest gratitude to my supervisor D^r Benmohamed-Mahieddine, for his sound advice and patience, and to my co-supervisor, M^r Gougam for initiating this work and encouragements.

I would like to thank especially M^r Aitouche for proofreading many sections of this report and his valuable suggestions.

I thank M^r Sahnoun, M^r Ouzal, M^r Tounsi, M^r Cheref, M^r Dahmas for their various help.

I acknowledge the support of the CST members.

I also thank M^{rs} Zerari for her moral support over the many long years spent together as well as the library staff, M^{iss} Khatir and all INELEC staff.

Acknowledgments

I wish to express my deepest gratitude to my supervisor D^{r} Benmohamed-Mahieddine, for his sound advice and patience, and to my co-supervisor, M^{r} Gougam for initiating this work and encouragements.

I would like to thank especially M^{r} Aitouche for proofreading many sections of this report and his valuable suggestions.

I thank M^{r} Sahnoun, M^{r} Ouzal, M^{r} Tounsi, M^{r} Cheref, M^{r} Dahmas for their various help.

I acknowledge the support of the CST members.

I also thank M^{rs} Zerari for her moral support over the many long years spent together as well as the library staff, M^{iss} Khatir and all INELEC staff.

Dedicaces

A tous ceux qui ont sacrifié leurs vies pour une cause noble,

A tous ceux qui sont morts pour défendre leur principes,

A toi, Ahmed, mort a la fleur de l'âge pour avoir voulu trop vivre,

A toi, Ahcène, qu'on espère revoir un jour malgré une si longue absence,

A ma petite Nawel, qui s'est résignée a accepter une maman trop occupée,

A Nacer, pour ses encouragements de ne jamais baisser les bras,

A Abbés, pour son support moral,

A Kader, pour sa gentillesse,

A tous mes frères et soeurs,

je dédie ce travail.

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 Motivation :

In the past two decades, there has been a tremendous progress in the development and analysis of parallel programs for different applications. The analysis and synthesis using wavelets has became a very useful tool for analyzing non-stationary signals.

Although the idea of wavelets dated back from 1940's [Gabor 46], the use of the analysis is relatively new. It was reactualized in the 1980's [Morlet 83]. Since then, the wavelet analysis has been of most interest due to the possible applications of this theory and its effectiveness in numerous disciplines. The analysis of the signal using wavelets allows the extraction of the pertinent information which depends on the physical nature of the signal: temporal signals (musical sounds, speeches, seismic waves, cardiograms, echograms, etc.) or images. But since the time required to solve a problem by a traditional computer is significantly reduced when using a parallel computer, we worked on the design and the implementation of a parallel algorithm to solve the wavelet transform.

The parallel wavelet transform is an algorithm for computing the continuous wavelet transform after sampling it, in a way that requires less computations than the direct way. The use of a straightforward algorithm leads to a prohibitive computation time, hence the need for a more effective computation procedure. The parallel algorithm is developed using the UNITY algebraic method [Chandy 88], which allows one to clearly understand the model of computation underlying the parallel computer. The first purpose of the proposed algorithm is to compress the data needed for the calculation of the wavelet coefficients. The second purpose is to design a hardware/software solution to compute the wavelet transform, called the PWT algorithm, since application specific integrated circuit (ASIC) designed to implement parallel programs can achieve very high processing rates, and therefore are becoming important. The hardware is represented by systolic array processors and, since systolic arrays have strong connections with formal methods, we derived a concurrent program from its specification using UNITY (Unbounded Nondeterministic Iterative Transformation) formalism. The last aspect of the thesis deals with the analysis of the sequential and the parallel algorithms, namely the complexity measures of each algorithm and the best architecture suited for it.

1.2 The problem :

In 1983, the physicist Jean Morlet [Morlet 83] was the first one to realize the needs of creating another technique for the analysis of transient signals that the Fourier techniques could not handle. He thought of a new method, the wavelet analysis and synthesis. The wavelet analysis is the representation of the signal as a function of time and frequency at the same time. It allows the analyzis of signals that combine phenomena with different scales. The window is no longer fixed but varies by translation and dilation or contraction depending on the signal. He proposed his (complex) wavelet:

$$\psi(t) = \exp\ (i\, c\, \mathrm{t})\ \exp\,(\frac{-\alpha^2\, t^2}{2}) \qquad 1.1$$

where c and α are parameters to be chosen.

He used a basic algorithm to compute the wavelet coefficients S(**b**,**a**) using numerical methods. Daubechies [Daubechies 92] proposed an algorithm presenting all the advantages of wavelet decomposition, using two positive numbers α and β. He used wavelets of frequencies $2^{-(\alpha \times j)}$ where j=...,-2,-1,0,1,2,... The mother wavelet ψ is localized on the interval [-1,1]. The wavelets are localized on intervals of length $2 \times (2^{-(\alpha \times j)})$. Meyer [Meyer 87] working to improve an algorithm created by Morlet and based on the CODES technique (cycle octave data enhancement system), discovered "the orthonormal basis of wavelets". He decomposed the signal using the wavelets $\psi_{j,k}$ in a sum of independent terms. Orthonormal basis means that the decomposition is unique (basis) and the wavelets are independent (orthonormal). Lemarie and Battle [Lemarie 90] discovered new orthonormal basis of wavelets. Barrat and Lepetit [Barrat 90] worked on a new type of analyzing wavelet that allowed a noticeable shortening of the computation time and a larger choice for the analyzing scale. The most known algorithm to compute the continuous wavelet transform is the algorithm 'a trous' by Dutilleux [Dutilleux 89]. Mallat [Mallat 89] used the wavelet analysis in image processing. He developed a pattern matching algorithm which processes the image at different resolutions. He studied the concept of multiresolution decomposition for one and two dimensional signals. He decomposed the signal using orthonormal basis of wavelets. Another algorithm called 'the pyramidal algorithm' based on convolution with 'quadrature mirror filters', and a similar algorithm allowing the reconstruction of the original signal from the wavelet decomposition was developed by Burt and Adelson [Burt 89] and was used by many other researchers.

To derive concurrent programs from their specifications, UNITY formalism was introduced, by Chandy and Misra [Chandy 88]. UNITY program is a formal mathematical model to deal with the representation of algorithms with high parallelization. It was used by Knapp [Knapp 92] in the parallel linear search and the asynchronous fixed point computation algorithms. Thiele [Thiele 92] used UNITY program in his problem: parallel implementation of cellular systems for numerical modeling. UNITY formalism was used by many other researchers in their derivation of parallel programs. Starting from an initial UNITY specification, we proceed in a series of refinement steps until the specification is restrictive enough to be translated directly into UNITY code. UNITY program aims to specify functions in a very formal mathematical way very early in the development phase and to prove that the implementation is correct. UNITY program is the best model to express the parallelism , and the best language to translate the parallelism is the Occam language [Burns 87] because of the concurrence features it represents. Objects from the problem domain are represented by processes that can run concurrently. Transputer [INMOS][1] using a SIMD architecture is the natural machine to implement Occam instructions, since unidirectional synchronous Occam channels can be mapped directly onto the transputer hardware links and processes are used to model the different units.

[1] Inmos is a trademark of the INMOS Group of Companies.

1.3 Thesis structure :

The remaining chapters give details about the theory of wavelets, the decomposition of signals using wavelets, the UNITY model which is the transition state from the sequential algorithm to the parallel one, the parallel architecture using systolic arrays and the implementation of programs written in Occam using transputers [Wexler 89]. We also state the problem and propose a solution for it.

Chapter 2 of this thesis explores the theory of signal processing and the systems that deal with signals. Fourier and wavelet transforms are studied in details and the role and the limits of each is also presented.

In chapter 3, UNITY program, the concurrency extraction tool of the parallel computer, the experimental systems and the different architectures a UNITY program could be mapped to are introduced. The Occam language and its features strongly connected to the transputer that represents the hardware on which the parallel algorithms are going to run, are presented. Finally, to plot the graphs, Matlab[2] is the best system to use and it is briefly introduced in this chapter.

The design and implementation of an improved sequential program making use of the parity of the wavelet transform as well as a parallel program for the wavelet transform application, the PWT, based on UNITY model is the subject of chapter 4. Some of the existing algorithms are briefly presented here. Algorithm complexity of all algorithms are also discussed with a comparison table of their time complexities.

In chapter 5, experimental results obtained by the sequential algorithm as well as the parallel one are presented by means of graphs plotted using Matlab. A description of the experiments and a table comparing the cost of different algorithms are given. A brief discussion of the results and their applications shows the advantages in using the new technique presented in this thesis. A conclusion of what could be the interpretation of the results is the final part of this section.

Finally, a summarize of the work achieved, a comparison to the results obtained with other methods, the benefits from using this new technique and suggested future work are given in chapter 6.

[2] Matlab is a trademark of Mathworks Inc.

CHAPTER 2

Signal Processing Theory

2.0 Introduction :

There are no general methods of analysis nor there are general solutions of all types of signals. We try to design an optimal system to study a certain class of input signals: non-periodical signals. Since the analysis of a given system is facilitated by use of a particular signal representation, we worked out the best formula to represent a signal : the wavelet transform. For this, it is important to include a brief study of signals and their properties in a study of a non-linear system.

Since the wavelet transform is described starting from the Fourier transform which is the most known time-frequency analysis, an introduction to the Fourier transform and particularly the windowed Fourier transform is presented. After defining the wavelet transform, a comparison is made between the two transforms that points out the differences and analogies between them.

So, before proceeding with the design of the system, The Parallel Wavelet Transform, we start first by learning how to analyze it. To provide a basis for a simple design, we develop a mathematical model that represents in the best way the physical model , which concerns the definitions, the notations, the initial conditions, the parameters, etc.

2.1 Signals and systems :

Signals are processed to facilitate the extraction of pertinent informations under the form of reduced characteristic numerical values. The development of signal processing techniques and systems is of great importance. Those techniques usually take the form of a transformation of a signal into another signal that is in some sense more 'desirable'.

2.1.1 Signals and signal processing :

Signals are defined as series of numbers coming from measurements [Meade 91], that conveys information about the state or behavior of a physical system. Signals are represented mathematically as functions of one or more independent variables. For example : a speech signal would be represented mathematically as a function of time, and a picture would be represented as a brightness function of two spatial variables. The two principal problems in signal processing is the ***analysis*** problem which is the evaluation of the response of a signal processing system to a given input to extract information from it or about it and the ***synthesis*** problem which is the design or specification of a processing system which will produce a desired output from a given input. Signals, depending on the characteristics of the time variable *t* which may be *continuous* or *discrete*, are classified into two broad groups:

The first group consists of analog or continuous-time signals and the second group consists of discrete-time signals. The second group is referred to as sequence.

2.1.1.1 Analog or continuous-time signals :

Continuous-time signals can be represented mathematically by a function of a continuous time variable. They are continuous in both space and amplitude. A continuous time signal is not necessary represented by a mathematically continuous function. Example : image, seismic, radar and speech signals. A continuous-time signal with a discontinuous waveform is the pulse function sketched in figure 2.1 and represented by equation 2.1:

$$g(t) = \begin{cases} 1 & \text{if } |t| < \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \qquad 2.1$$

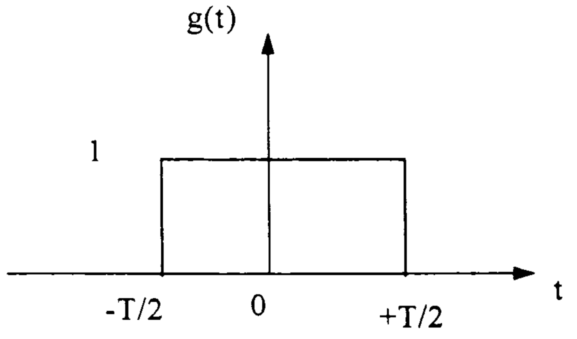


Figure 2.1. A continuous-time signal with a discontinuous waveform.

2.1.1.2 Discrete-time signals :

Discrete-time signals are digital signals that are defined only at a particular set of instants in time which are called *sampling instants* (i.e. the contents of an arithmetic unit in a digital computer). The values of a discrete-time signal x(t) are expressed as a function of the integer variable n. If the sampling interval is T_s, and the discrete signal is defined for t= n.T_s, then the x[n] can be regarded as an ordered sequence of numbers with values x[0], x[1], ..., for n = 0,1,... that is obtained by sampling a continuous-time signal x(t) at equidistant points. The sequence x[n] is defined by:

$$x[n] = x(nT_s)\,, \qquad \text{for } n = 0, \pm1, \pm2, \ldots$$

The signal graph of a discrete-time signal may be represented by figure 2.2:

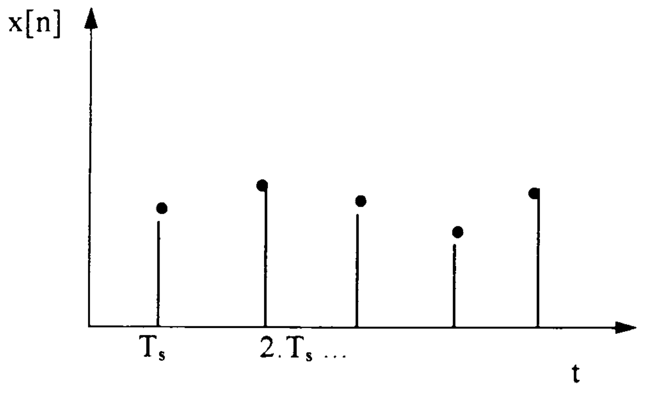


Figure 2.2 Discrete-time signal

2.1.1.3 Representation of signals :

An important aim of signal analysis is to generate models to represent signals in a form that is both understandable and applicable. One way to do this is to represent a signal as a sum of elementary components. So a signal waveform is represented by means of a series of formalized waveforms to make the process of analysis easier. To get a good accuracy, we try to use as many terms as possible. Then we will be able to analyze the signal with some knowledge of its constituent parts. Suppose we have a signal represented by a function f(t) which we wish to represent on a finite interval $[t_1, t_2]$ by a set of n functions $\varphi_1(t)$, $\varphi_2(t)$,..., $\varphi_n(t)$ that are orthogonal on $[t_1, t_2]$.

How should we represent f(t) on $[t_1, t_2]$ in terms of the set of functions$[\varphi_i(t)]_{i=1}^{n}$?

Let us assume a representation or approximation of f(t) by a linear combination of the functions $\varphi_i(t)$ $i = 1,2,\cdots,n$. The representation for f(t) on $[t_1, t_2]$ is of the form :

$$f(t) = c_1\,\varphi_1(t) + c\,\varphi_2(t) + \cdots + c_n.\varphi_n(t).$$

The coefficients c_i can be interpreted as the projection of f(t) in the direction of the function $\varphi_i(t)$. The representation or approximation is then, a decomposition of a function f(t) in terms of the set of functions $\varphi_i(t)\}_{i=1}^{n}$.

2.1.1.4 Signal sampling :

The basis of signal sampling is to represent a continuous-time signal f(t) by a sequence of samples f[n] with values $f(n.T_s)$ [Gabel 73]. f[n] is derived from f(t) by periodic sampling at a frequency $f_s = \frac{1}{T_s}$. The signal is sampled in a way that makes it possible to recover the waveform representing the continuous-time signal f by 'joining up the points' on a graph of the samples. The process of recovering the waveform of the continuous-time signal f(t) from the set of samples is known as interpolation, and a good interpolation depends on the choice of the sampling frequency. If the sampling frequency is chosen adequately, the waveform can be recovered by interpolation, otherwise, if the signal has been sampled at a relatively low rate, vital information is lost about the rapidly changing parts of the signal, and two waveforms can be recovered by the same set of samples giving aliasing signals. To avoid aliased signals, Shannon stated in his sampling theorem that if f(t), the function being sampled, is band-limited, that is, if it contains no frequency components above some bound frequency f_{max}, then it can be completely described by uniformly spaced samples taken at a rate of at least $2.f_{max}$ samples per second. In other words, it is stated that a signal can be sampled at a rate of $f_s \geq 2f_{max}$ samples per second, with no loss of information. This minimum rate of $2.f_{max}$ is commonly known as the Nyquist rate. If the sampling rate is less than the Nyquist rate, the signal is undersampled otherwise it is oversampled.

2.1.2 Systems :

A system maps an input signal to an output signal. A major element in studying signal processing is the analysis, design and implementation of a system that transforms an input signal to a more desirable output signal for a given application.

The representation of a signal is a formal system for making explicit certain types of information, plus the specification of how the system works. The description is the result given by the representation. The systems are of two types : discrete-time systems and continuous-time systems.

2.1.2.1 Discrete-time systems :

A discrete-time system is one in which all or some of the signals in the system are discrete-time signals. A discrete-time system performs numerical operations on one or more input sequences $\{x_n\}$ to produce one or more output sequences $\{y_n\}$.

2.1.2.2 Continuous-time systems :

A continuous-time system is one in which both the input and the output are continuous-time signals. Operations performed by a continuous-time system are equations of ordinary algebra and calculus.

2.2 Transforms :

A transform is just a digital filter that simply finds a different way of looking at some problem. To use this filter, three steps are followed :

1) specification of the characteristics required of the filter : they depend on the intended application.
2) design of the filter.
3) implementation of the filter.

All three steps are interleaved, and the key to the analysis of the system is the correct representation of the input signal.

How do they work?

A transformation may just change things so they will appear differently like in image processing or it may break something down into its fundamental parts like in cartography or a piece of music or may extract non-obvious information in non-obvious ways like separating signals from noise. They operate by transforming arrays of numbers into other arrays of numbers. We start off with some pile of numbers, apply some mathematical rules to them (add, subtract, translate, scale, rotate, etc.), and end up with a second pile of numbers. A transform will be chosen depending on the application we have at hand. A transform can be defined as a mathematical operator acting on a signal and verifying the following conditions :

1) conservation of the information.

2) conservation of the energy.

3) inversibility.

2.2.1 Decomposition of functions :

There are an infinite number of decompositions of a function f(t) in terms of elementary functions. The choice of a particular decomposition depends on two factors :

1) The properties of the function f(t) and the interval over which f(t) is to be represented.

2) The properties of the system to be analyzed and the model used to represent the system. The system chosen to analyze the signal depends on the type of the signal to be analyzed. If the system is linear and time-invariant and the exciting function is periodic, the best tool for representing f(t) is the Fourier series. So stationary signals that are statistically invariant over time decompose canonically into linear combinations of sines and cosines waves (e.g. white noise). Non-stationary signals that slowly change with time, decompose into linear combinations of wavelets.

2.2.2 Fourier transform :

The Fourier analysis was the first technique for the analysis of signals and the most known. Since the creation of the Fast Fourier Transform (FFT) algorithm, the Fourier analysis became very important for the analysis of periodic and regular signals.

Suppose that f(t) is a signal bounded on the interval (-a, a) satisfying the following sufficient Dirichlet conditions :

1) f(t) is bounded and of period T.

2) f(t) has at most a finite number of maxima and minima in one period and a finite number of discontinuities.

The above conditions are all we need for engineering applications.

f(t) then can be represented by :

$$f(t) = \sum_{-\infty}^{\infty} F_n \exp\left(j \frac{n}{a} \pi t\right) \quad \text{where} \quad F_n = \frac{1}{2.a} \int_{-a}^{a} f(t) \exp\left(-j \frac{n}{a} \pi t\right) dt$$

In a Fourier series expansion, a periodic function is decomposed in terms of its harmonics that make up the spectrum of f(t). If f(t) is not periodic, and we want to use a representation that is valid on (-a, a), a decomposition of f(t) in terms of a Fourier series is no longer valid and a Fourier transform was used. It is given by :

$$F(\omega) = \frac{1}{\sqrt{2.\pi}} \int_{-\infty}^{\infty} f(t) \exp(-j\omega t)\, dt$$

The Fourier integral decomposes the signal on sinusoidal functions that oscillates indefinitely in time. When added, the effects of those sinusoids cancel in the region of the time axis where the signal is nil. The Fourier transform presents a major inconvenience in the analysis of signals; it is either a time or frequency analysis, but not the two at the same time. However, even with the Fast Fourier transform, the Fourier analysis presents inconveniences which do not allow a correct analysis of all type of signals. Thus, in the spectrum $F(\omega)$, all temporal aspects of the signal disappear; for example, the beginning and the end of a finite signal or the appearance of a singularity. F(ω) gives a representation of the frequency content of f but information concerning time localization of high frequency bursts cannot be read from F(ω). We wish to realize a time-frequency analysis compared to a musical partition where are indicated at the same time the frequency and the duration of the notes. Even for the calculation of the spectrum, to compute F(ω), we have to know f(t) for all real values of t which is impossible since in case of a real time analysis, the signal is processed gradually as the numerical data arrive. It is impossible to know the approached spectrum of a signal whose future is unknown since frequencies with different values can appear. To overcome these technical constraints, the tool was improved. The time localization can be achieved by first windowing the signal and then taking its Fourier transform.

2.2.3 The windowed Fourier transform :

A standard technique for time-frequency localization is the short time Fourier transform. The function f(t) is multiplied with the window function g(t), and the Fourier coefficients are computed by the product f(t)×g(t) given by :

$$F_{win}(\omega,t) = \int f(s)\, g(s-t)\, e^{-i\omega s} ds$$

In the windowed Fourier transform, The g' s are totally concentrated in frequency and totally distributed in time, which means that the transition in the Fourier space gives the maximum information on how the frequencies are distributed but loses all informations concerning time. The function measured by the windowed Fourier transform, $|F_{win}(\omega,t)|^2$ represents the spectogram which gives a coarse time-frequency distribution.

Since the windowed Fourier transform uses a window with fixed length, it is difficult to proceed signals with variable variations; this is the case with speech signals : attack of the note is a very brief phase that contains high frequencies and characterizes the instrument and the player while the rest of the note contains relatively low frequencies. The same holds for phenomena that combine different scales, macroscopic and microscopic.

To summarize, the windowed Fourier transform has the following shortcomings:

a) The choice of the nature of the window and its dimensions are fixed without any possibility of changes.

b) The temporal and frequency resolutions vary inversely one of the other.

c) Poor resolution of the frequency structure at wavelength longer than the window width (i.e. for low frequencies).

d) Poor localization of the high frequencies because the energy is averaged over the window width.

To allow a correct analysis of all types of signals, the wavelet transform is used.

2.3 Theory of wavelets :

Unlike the classic Fourier transformations or the Wigner-ville pseudo representation of the non-stationary signals, wavelets can simply and quickly concentrate on specific portions of a transformation problem, and can do so locally or globally. They also described the distribution of the energy in the time-frequency plane. Both techniques, the windowed Fourier transform and the pseudo time-frequency representation of the non-stationary signal, present the same inconvenience : the nature of the analyzing window which is fixed and cannot be modified during the analysis. The wavelet transform is based on sets of wavelets in the same way that the FFT is based on sine and cosine functions. The theory of wavelets is a mixture of mathematics, scientific computation, and signal processing. The wavelet transform is also a signal processing technique for analyzing signals to solve some application problems. The wavelet analysis is the representation of the signal as a function of time and frequency at the same time. Wavelets are of two types :

1) Time-frequency wavelets.

2) Time-scale or space-scale wavelets.

Time-frequency wavelets are suited for the analysis of quasi stationary signals. Time-scale wavelets are adapted to signals having fractal structure.

2.3.1 The wavelet transform :

Starting from a basic function ψ, called mother wavelet or analyzing wavelet, we construct a family of elementary functions obtained by shifts and dilations from the mother wavelet given by equation (2.1).

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \qquad a > 0, \quad b \in R \qquad 2.1$$

where a is tied to the frequency and b to the time.

($\frac{1}{\sqrt{a}}$ is taken for the normalization that has been chosen so that $\| \psi_{a,b}\| = \| \psi\|$ for all a and b).

The analyzing wavelet characteristics are very different from those of a window. The latter has taken more or less the shape of a square pulse while ψ will be of integral nil and oscillating. ψ and its Fourier transform $\hat{\psi}$ have good localization and thus converge to 0 when t or ω go to infinity.

For time-frequency wavelet, the idea is to divide a wave into segments and keep only one of the segments. The segment represents a piece of a wave or wavelet, which has a beginning and an end. Suppose that s(t) is a temporal signal (signal to be analyzed) and $\psi(t)$ the analyzing wavelet (mother wavelet). To decompose into a wavelet series, the following conditions must be satisfied :

$\psi(t)$ must be admissible.

$\psi(t)$ must be well localized and oscillating.

$\psi(t)$ must be a real valued integrable function.

$\psi(t)$ and s(t) : are of finite energy.

The admissibility condition : $\int_{-\infty}^{+\infty} \psi(t)dt = 0$

The localization condition : $\lim_{|t| \to \infty} \psi(t) \to 0$,

The oscillating condition : $\int_{-\infty}^{\infty} t^{n-1}\, \psi(t)\, dt = 0 \quad \text{for } 0 < n < m$

The integral of $\psi(t)$ must be zero and the same hold for the first m movements of Ψ. Once the mother wavelet $\psi(t)$ has be chosen, it generates the other wavelets. A typical choice for the mother wavelet is the real part of the (complex) Morlet wavelet since it satisfies the above conditions:

$\psi(t) = \exp(i\, c\, t) \times \exp\left(\frac{-\alpha^2\, t^2}{2}\right)$ where c=2 π and α=1.

The other wavelets are defined starting from equation (2.1). Starting with $\psi(t)$, we get the other wavelets by varying ***a*** and ***b***. Figure 2.1 represents the mother wavelet ($a=1$) and high and low frequency wavelets. They keep the same shape but they are stretched ($a>1$) or contracted ($a<1$). For all s(t) such that s(t) has a finite energy, s(t) is a linear combination of the $\psi_{a,b}$'s .

(a) $\psi_{a,b}$ *with a=1,b=0* (b) $\psi_{a,b}$ *with a<1 , b>0* c) $\psi_{a,b}$ *with a>1,b<0*

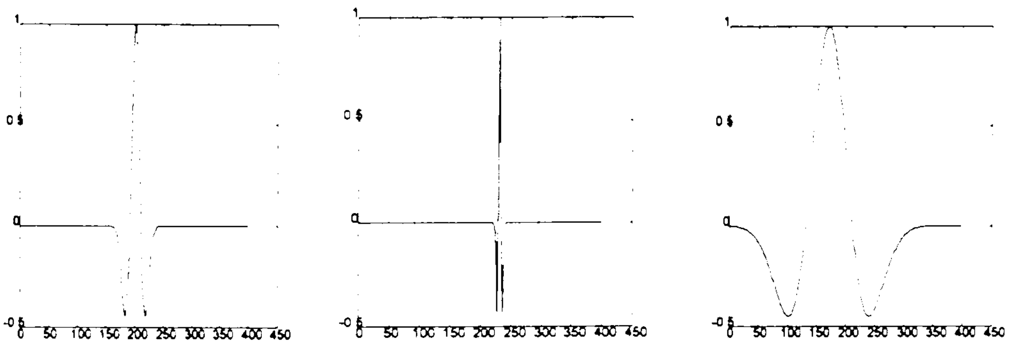


Fig. 2.1 Typical shapes of the real part of the wavelets $\psi_{a,b}$. ((a) represent the mother wavelet, (b) the contracted wavelet and (c) the dilated wavelet).

2.3.2 Different types of wavelet transforms :

There exists many different types of wavelet transforms all starting from the basic formulas :

$$S_{a,b}(t) = \frac{1}{\sqrt{a}}\int s(t)\ \psi^*(\frac{t-b}{a})\ dt \text{ and } T_{m,n}(t) = \frac{1}{a^{\frac{m}{2}}}\int s(t)\ \psi(a_0^{-m}t - nb_0)\ dt$$

where ψ^* is the real part of the Morlet wavelet.

They are divided into two classes, where we distinguish :

a) The Continuous wavelet transform.

b) The Discrete wavelet transform.

The discrete wavelet transforms are divided into two subcategories : the redundant discrete systems or frames and the orthonormal bases of wavelets. In the continuous wavelet transform, the dilation and translation parameters ***a***, ***b*** vary continuously over R (***a*** $\neq$0). The signal s(t) can be reconstructed from its wavelet transform by means of the 'resolution of identity formula' :

$$s(t) = C_\psi^{-1}\iint \frac{da\,db}{a^2}\ (T^{wav}s)(a,b)\psi^*_{a,b} \text{ where } (T^{wav}s)(a,b) = \langle s, \psi_{a,b}\rangle$$

where $\langle s, \psi_{a,b}\rangle$ represents the inner product of s(t) and ψ; s(t) is then a superposition of wavelets. The constant C_ψ depends only ψ and is given by :

$$C_\psi = 2\pi\int_{-\infty}^{+\infty}\left|\hat{\psi}(\omega)\right|^2 |\omega|^{-1} d\omega$$

While in the discrete redundant wavelet transform, the dilation and translation parameters ***a*** and ***b***, both take discrete values where $a = a_0^m$ and $a_0 > 1$.

Different values of m correspond to wavelets of different widths. The discretization of the translation parameter ***b*** should depend on m. :

- narrow wavelets (corresponding to high frequencies) are translated by small steps in order to cover the hole time range.

- wider wavelets (corresponding to low frequencies) are translated by larger steps.

It is difficult here to reconstruct the signal from its wavelet transform.

The orthonormal wavelet basis or multiresolution analysis are obtained for some special choices of ψ, a_0 and b_0. The $\psi_{m,n}$ constitute an orthonormal basis for $L^2(\Re)$ where L^2 represents the set of integrable functions. In the multiresolution approach, successive coarser and coarser approximations to s are written, and at every step, the difference between the approximation with resolution 2^{j-1} and the next coarser level with resolution 2^j , is written as a linear combination of the $\psi_{j,k}$

2.4 Analogies and differences between the windowed Fourier transform and the wavelet transform :

The windowed Fourier transform and the wavelet transform have analogous formulas. One similarity between them, that both take the inner products of s with a family of functions indexed by two labels.

Both the wavelet transform and the windowed Fourier transform can be sampled. The natural sampling used for the window Fourier transform for ω, t is $\omega = n.\omega_0$ and $t = n.t_0$ where ω_0 $t_0 > 0$ are fixed and m and $n \in Z$. On the other hand, the sampling of the dilation parameter for thw wavelet transform is chosen such as $a = a_0^m$ where m $\in$ Z, $a_0 > 1$ and $b = n.b_0.a_0^m$ where n $\in Z$ and $b_0 > 0$, giving the following sampled window and and the sampled wavelet:

$$g_{m,n}(x) = e^{jmw_0 x} g(x - nt_0) \qquad (2.2)$$

$$\psi_{m,n}(x) = a_0^{-\frac{m}{2}} \psi\left(\frac{x - nb_0 a^m{}_0}{a^m{}_0}\right) \qquad (2.3)$$

functions $g_{\omega,t}$ consist of the same envelope function g having the same width regardless of ω translated to the proper time location, and 'filled in' with higher frequency oscillations. In contrast, the time-widths of the $\psi_{a,b}$'s are adapted to their frequency. High frequency $\psi_{a,b}$'s have very narrow time-width while low frequency $\psi_{a,b}$'s are much broader, which makes the wavelet transform better able than the windowed Fourier transform to 'zoom in' on very short-lived high frequency phenomena, such as transient signals.

Conclusion :

The inconvenience of the Fourier transform, is that, more a signal is short in time, more it contains sinusoidal components with significant amplitudes. But conversely, an infinite sinusoidal signal correspond to only one frequency. The disadvantage of both the windowed Fourier transform and the wavelet transform is that they introduce a reference function (e.g. the window function or the wavelet) against which the signal has to be integrated. The Fourier transform is concerned only with stationary signals, while the non-stationary signals contain more informations. It only allows an exclusive analysis: frequential analysis or temporal analysis. It is concerned by the whole signal and not only with portions of it. In some cases, the results of the Fourier transform may be confusing since different signals may have the same spectrum whereas the wavelet transform is adapted to analyze all type of signals and particularly non-periodical signals.

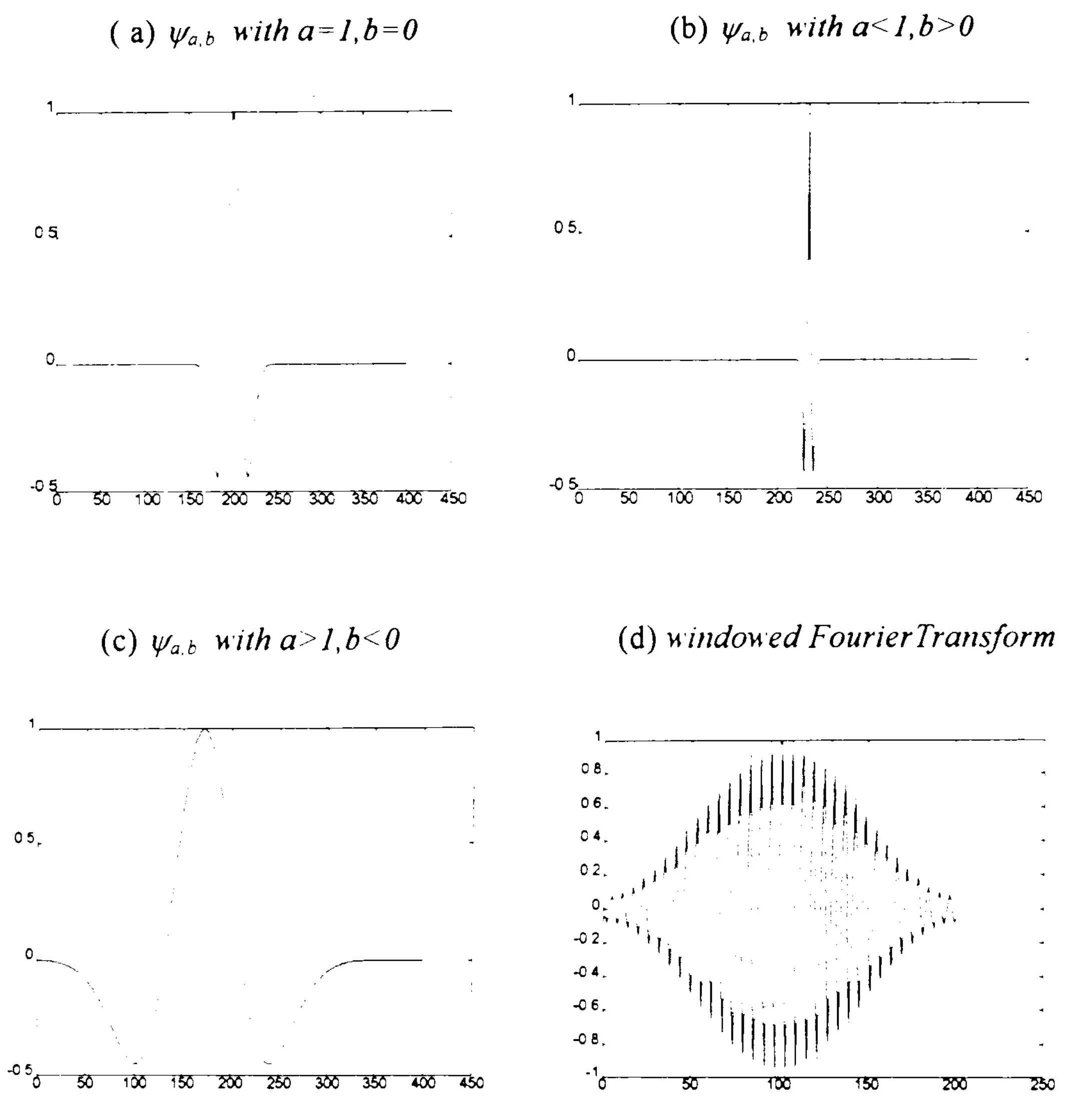


Fig.2.2 In (a),(b),(c) wavelets $\psi_{a,b}$ do not have an envelop while in (d), the windowed Fourier transform $g_{\omega,t}$ has a rigid envelop and the number of oscillations augment with high frequencies.

CHAPTER 3

Experimental Systems

3.0 Introduction :

This chapter introduces the different experimental systems that allowed the successful simulation of the PWT. To make the ideas concrete, we introduce the different parallel architectures and the most suited for our application. To program and analyze the parallel program, a formal method, UNITY model, an algebraic *language-based model* underlying the analysis is introduced. The costs of the primitive instructions and a set of rules for computing costs across program expressions are specified in such a model. This formal model allows mapping to different architectures. It also contributes to making a transition from the serial programming and the single processor computer to a parallel computer system. The simulating parallel processor, the transputer and its development system TDS are presented as well as the programming language designed for it, Occam which is explicitly covered.

3.1 Parallel architectures:

Parallel operation implies the use of a number of units operating simultaneously. A fast parallel transformation is produced by the use of a special purpose hardware design based directly on the algorithm form given by the flow diagram (UNITY program in our case) of a fast transformation. The different designs lead each to a family of machines. The essential ones are :

- multiple special purpose functional units.
- associative processors.
- array processors.
- data flow processors.
- functional programming languages.

One of the important criteria to classify parallel architectures is the communication between processes, and the second, the physical nature of the network. Whether we have one stream of instructions or several, and whether we have one stream of data or several, computers will be classified into four categories (Fig. 3.1) :

1) Single Instruction stream, Single Data stream (**SISD**).

2) Single Instruction stream , Multiple Data stream (**SIMD**).

3) Multiple Instruction stream, Single Data stream (**MISD**).

4) Multiple Instruction stream, Multiple Data stream (**MIMD**).

The processors can be connected in linear array, mesh, tree, perfect shuffle, cube, etc.

3.1.1 SISD machine :

A **SISD** machine consists of a single processing unit receiving a single stream of instructions and operating on a single stream of data. In one step during the computation, a datum is obtained from the memory unit on which one instruction sent by the control unit operates. This is the principle of the Von Neumann machine.

3.1.2 SIMD machine :

A **SIMD** machine is an array processor composed of multiple processing units receiving a single stream of instructions and operating on a multiple stream of data. In one step during the computation, the control unit broadcasts an instruction to be executed on local data by all the processor elements (PE). Each PE can make local minor modifications to the broadcast instruction or be programmed to ignore the instruction.

3.1.3 MISD machine :

A **MISD** machine has a multiprocessor structure with N streams of instructions and one stream of data. It is composed of N processors having each its own control unit and sharing a common memory unit. At each step during the computation, each processor according to the instruction received from its own control unit, operates simultaneously with the others upon the same datum received from the shared memory.

3.1.4 MIMD machine :

A **MIMD** machine has also a multiple processor structure with N streams of instructions and N streams of data. In these systems, each processor is fully programmable and can execute its own program. Each processor has its own control unit, its local memory unit and its processing unit. The control unit issues the control of an instruction stream under which each processor operates on a datum received from its own memory. The **MIMD** machines are more flexible than the other classes but their control is much more complex.

3.2 Systolic architectures :

In many digital signal processing applications, large amounts of data to be processed and relatively high operating speeds are needed. Hence, one way of significantly increasing the processing speed is to design application specific integrated circuit (ASIC) for VLSI implementation, and the most appropriate architecture for VLSI implementation is the systolic architecture. The concept of systolic architecture was developed by [Kung 79]. A systolic algorithm [Burns 87] is one that has a collection of identical processes through which data flows. In a pure systolic algorithm, each of the parallel processes executes an identical sequence of instructions. An implementation on a mesh SIMD architecture is well-suited for this type of algorithms. Such an architecture can be represented by a system of transputers where each transputer can hold a process and the data can flow down the links. Systolic comes from the analogy between the circulation of the data flow in the network and the blood in the human body, where the clock insuring the global synchrony constitutes the 'heart of the system'.

3.2.1 SIMD mesh architecture :

A **SIMD** mesh array is a programmable logic array where N processors are arranged into an $m \times m$ array where $N = \sqrt{m}$, giving a mesh network (Fig. 3.2). P(j,k) is the processor in row j and column k, where $0 \leq j \leq m$ and $0 \leq k \leq m$. It is connected to its neighboring processors P(j+1,k), P(j-1,k), P(j,k+1) and P(j,k-1) as shown in (Fig. 3.3).

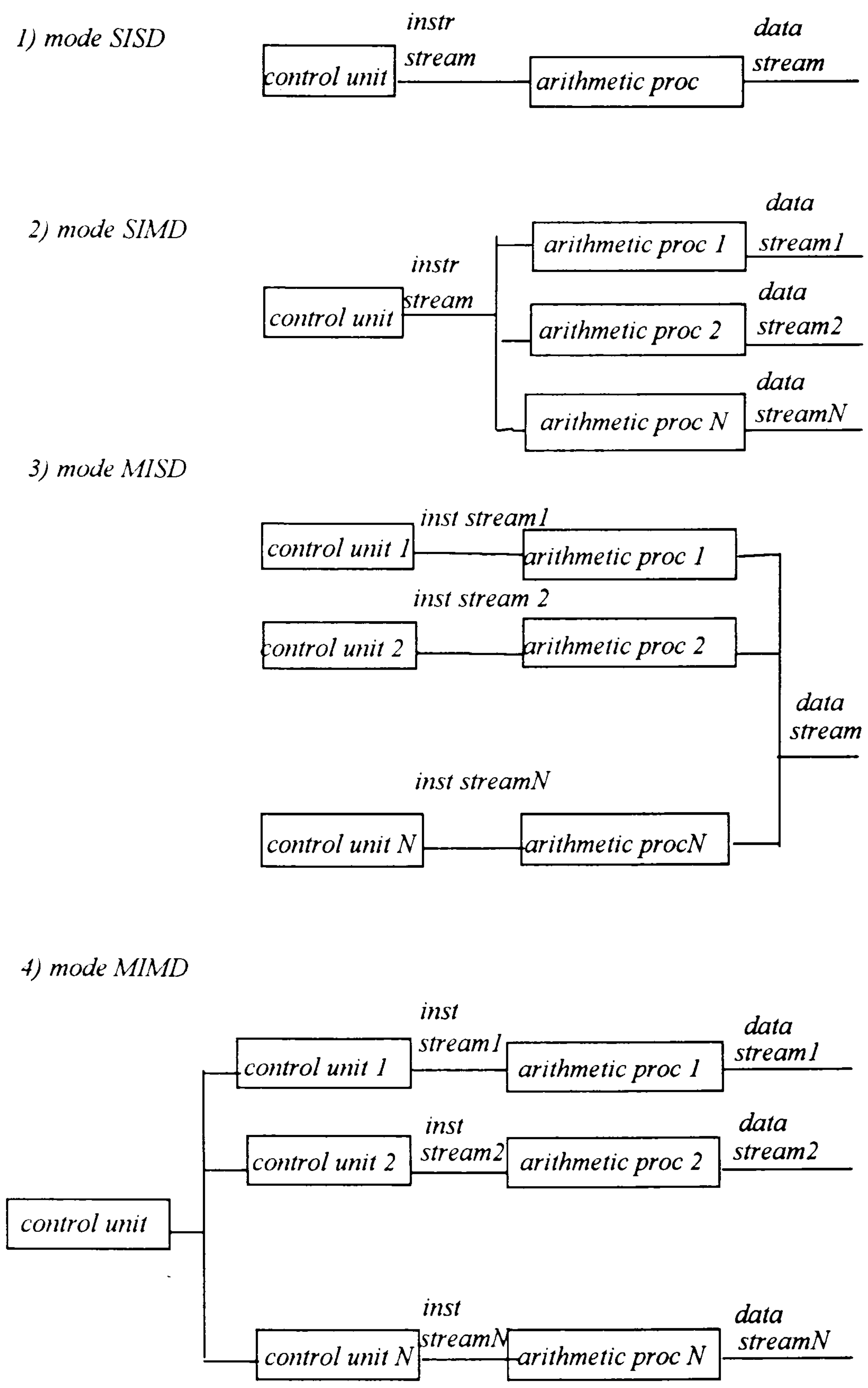


FIG. 3.1 Different types of parallel architectures.

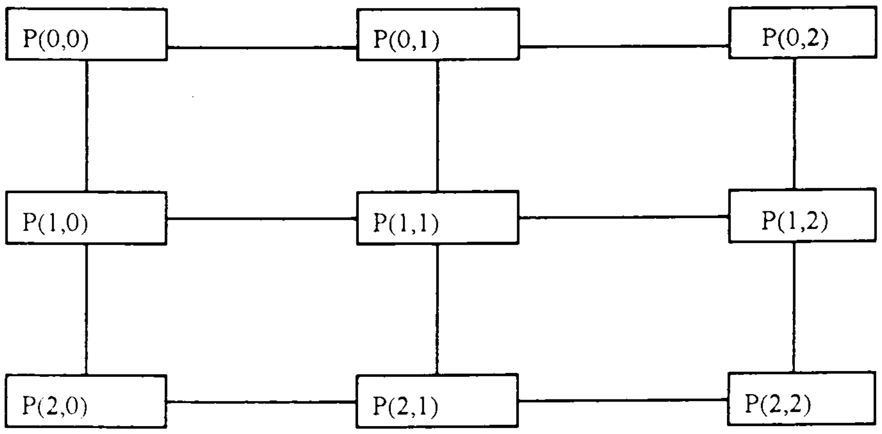


Fig.3.2 A 3x3 mesh array

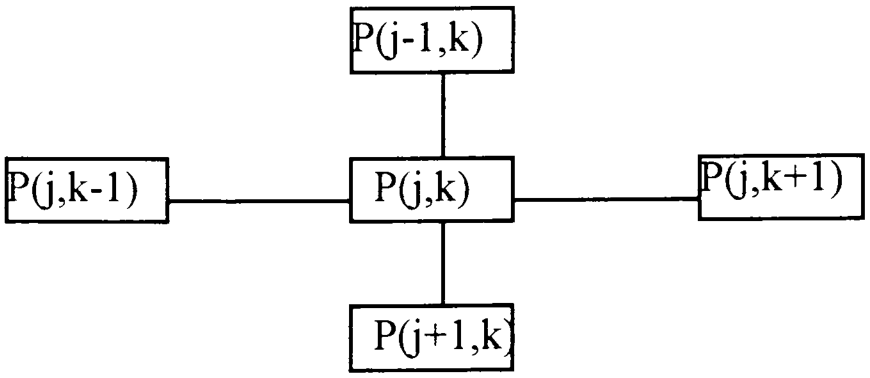


Fig. 3.3 Processors connections.

3.2.2 Systolic arrays :

A systolic array has a SIMD type architecture. It is a simple machine made up of a set of interconnected cells, each capable of performing a simple operation. The cells in a systolic design are typically interconnected as a matrix or a tree. Information between cells flows in a pipelined manner. In one step during the computation, each processor reads data from its input lines (neighbor cells), performs some operations, and writes new data values into the output lines (neighbor cells). Communication with the outside world takes place only with the border cells. Synchronism is achieved by the 'rendez-vous' method : a processor can consume data only when it is produced on its input channels; otherwise, the processor waits. Basically, the processing elements in the systolic array are multiply-add cells.

3.3 Transputers :

The transputer is the first product available which allows the programmer to conceive a MIMD parallel machine of his own without the need to be a computer expert. The systolic architecture can be implemented using transputers. The transputer is a programmable VLSI component designed for the parallel processing, offering high performance and an extreme adaptability for the needs of users. Its internal architecture allows the implementation of the concurrency in a system. A transputer is a microcomputer with its own local memory and can be connected to other transputers.

3.3.1 Transputer architecture :

A transputer (Fig. 3.4) is designed on a single chip containing a 32 bit processor, the T414, a 2 Mbytes of memory, and communication links that provide point to point connection between transputers. Each transputer has four two serial links which can operate at up to 20 Mbits/s, and can be connected to other transputers. Processors are used as standard software and hardware building blocks. The process is the software building block. An interconnected set of processes composes the system. Each process can be regarded as an independent unit of design that can be implemented in hardware. A transputer executing an Occam program can be viewed as a process. It operates as a RISC (reduced instruction set computer) type processor, where most instructions can be executed in a single machine cycle. The transputer is designed to make the use of processes easier and more effective. One process can have at most four channels because only four links can be available for each transputer. A process can communicate only with the processes which exist on tranputers directly connected to it. Transputers present very powerful features which are the communications between system components, the programming language Occam, and the connection of transputers with formal methods. The communication between processes is realized by high-speed point to point serial links. The other major feature is the programming language Occam where Occam channels can be mapped directly onto the transputer hardware links and processes are used to model the different units. The last strong feature of the transputer is its connection with formal methods (UNITY for example).

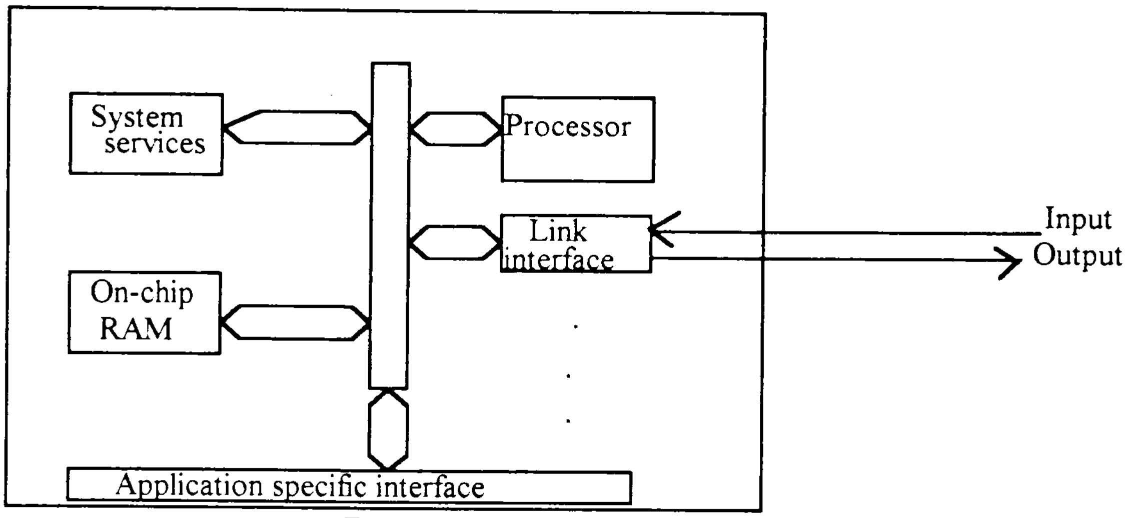


Fig. 3.4 The transputer architecture.

3.3.2 Transputer Development System :

The Transputer Development System or TDS (Fig. 3.5) is an integrated development system used to develop Occam programs for a transputer network. Most of the TDS runs on the transputer board plagued on an IBM PC such as an IMS T414 transputer that has 2 Mbytes of RAM and all the appropriate development software such as the editor and compiler utilities. The server is a program residing on the IBM PC that provides the TDS with access to the terminal and filling system of the IBM PC. Occam programs are entirely edited, compiled and run within the TDS. When a program is run within TDS, it runs in parallel with, and connected to certain components of the TDS such as channels, their protocols and a number of supporting communications of these channels. The channels from.isv and to.isv are used to connect the application process to the server program running on the host IBM PC. The KS and SS channels are running in parallel. They are channels to and from the terminals (keyboard and screen) to enable the application process to communicate data to and from these devices. The monitor procedure contains the interface with the TDS. It supplies the values of keystrokes on the host keyboard to the application while in parallel conveying data and results to the host terminal. Termination of parallel programs are the duty of the programmer. Since there is no control flow, the program terminates when it reaches a fixed point which is a stable state : a state that is repeated for ever and in which all machines are passive. The TDS allows to implement Occam programs since it is the natural machine for them.

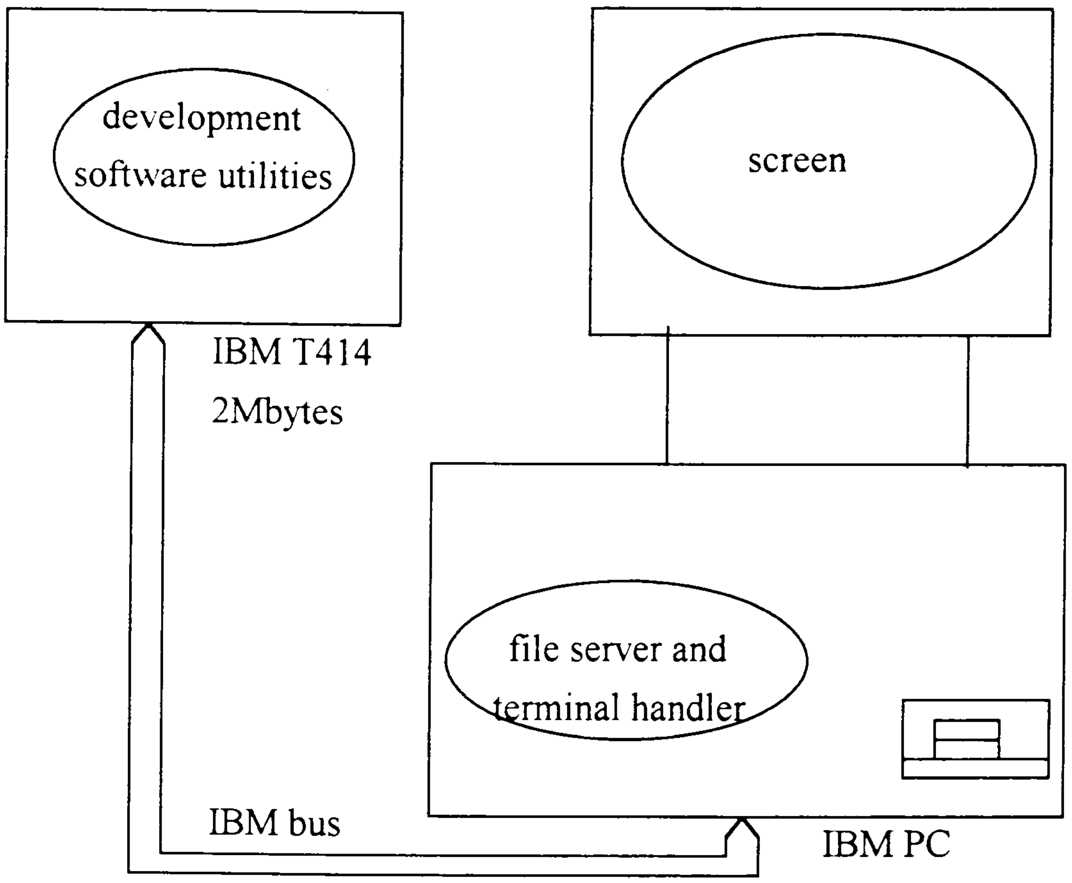

development software utilities
screen
IBM T414
2Mbytes
file server and terminal handler
IBM bus
IBM PC

Fig. 3.5 The Transputer Development System

3.4 Occam model :

Occam is the native language of the transputer. The name Occam comes from William of Occam's. It is issued from CSP(communicating sequential processes) [Hoare 85] which is a concurrency representation language. According to this model, a parallel system is composed of a set of concurrent processes representing system objects linked together by a means of channels. So an Occam program represents just an expression of an algorithm of a solution of a problem and Occam is the natural language of the transputer that provides a simple means of specifying concurrency, creation and replication of processes. Logical gates are modeled by processes while connecting wires are modeled by channel instances. So an electronic circuit will be modeled by processes for gates and channels for connecting wires.

3.4.1 Notion of processes :

The process is the privileged software tool to manage the processor time. A process like a classical sub-program, executes a certain number of operations. Instead of being executed sequentially, processes are simply juxtaposed in an application program. The transputer implementation of Occam instruction allocates a variable to represent each channel; this variable being shared by the process which sends output over the channel, and the process which takes input from the channel.

3.4.2 Information about processes :

To analyze a given source program, the processor assigns unique sequential numbers for identification to every process and channel. For each process the following informations are extracted :

1) The process name which is not unique .

2) The total number of channels required by each process. For each channel, the channel name, the direction (in or out channel), the type (vector or not), the name of the process connected through this channel and the channel number of partner process are extracted.

3.4.3 Information about processors :

To identify each transputer in the machine and its links, unique sequential numbers are given, and link connections to other transputers are also described. All information concerning processes are extracted from user programs.

3.4.4 Format of an Occam program :

The structure of an Occam program is indicated by successive indentations from left to right (two spaces by indentation). The rules of indentation should be strictly respected. Each process in a program must start on a new line and must respect the indentation rules above :

– in case of a syntactical construct, the keyword should be on the first line of the construct, followed by the options when needed. The associated processes to the construct start on a new line with an indentation of two spaces (one process per line).

– in case of a process doted with a guard, the guard occupies the first line and the process indented by two spaces, starts on the second line.

– for a conditional process, the condition is written on the first line and the associated process starts on the following line with indentation of two spaces.

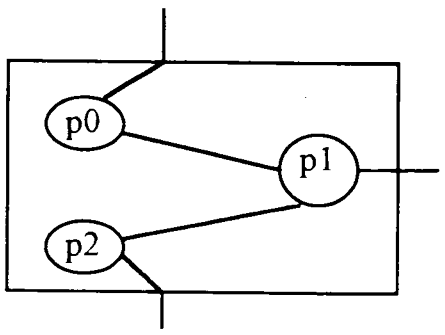
Communication processes are denoted by a channel name, an optional ! or ? and the name of what is to be transferred.

```
 in ? var
out ! expr
```

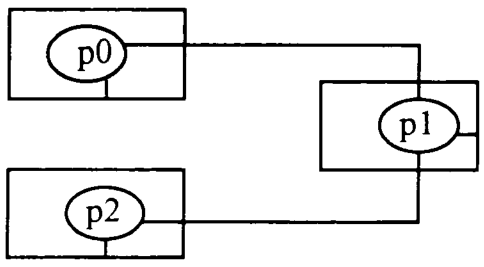
The first represents input process and the second, output process. **Ch.a** represents an internal process.

3.4.5 Process mapping with Occam :

Since Occam is the native language of the transputer, it provides powerful programming constructs which allow the parallelism of an algorithm to be described : independent processes represent blocks of code exchanging information via inter-process communication channels. Two processes can either be executed in parallel on different transputers or concurrently within the same transputer (time-sharing). When a program has to run on a multitransputer system, allocations of process's channels to hardware links of transputers must explicitly be described. For programs running on one transputer, only one process can be allocated to one transputer. But three processes can be mapped onto one transputer or distributed over three transputers (Fig. 3.6). Since a transputer does not have routing ability, a process can only communicate with the processes which exist on transputers directly connected to it.



Three processes on one transputer.



The same processes distributed over three transputers

Fig. 3.6 Mapping processes onto one or several transputers

3.5 Unity program :The parallel model

In this section, we present UNITY model for designing the parallel wavelet transform since UNITY is the object that converts a sequential algorithm to a parallel one in a straightforward manner starting from the specification of the problem and ending with a parallel program. An important aspect of UNITY is that the model is defined directly in terms of language constructs rather than trying to appeal to any intuition of a machine. The model is a virtual one for which running times for various physical machines can be given.

UNITY stands for : Unbounded Nondeterministic Iterative Transformation. UNITY is a computational model added to a proof system. It is made of three parts : specification, design, verification. UNITY is not a programming language but adopts the minimum machinery to illustrate ideas about programming. A UNITY program does not specify (on which processor in a multiprocessor system) **WHERE** an assignment is to be executed. It does not specify to **WHICH** process an assignment belongs. It does not specify **HOW** assignments are to be executed or **HOW** an implementation may halt a program execution (problem of termination).

UNITY separates concerns between : **what** and **when** on one hand and **where** and **how** on the other hand. **What** is specified in a program. **When, Where, How** are specified in a mapping. A mapping is the description of how programs are to be executed on a target machine. In UNITY, mappings are more important and more complex than in traditional programs. UNITY programs are mapped to programs in the conventional programming languages running on a given architecture.

3.5.1 UNITY program structure :

In this section, the grammar for the model is introduced that defines a minimum set of rules to specify the programming language. The notation is described using Backus Normal Form (BNF).

```
program     Program       program-name
              declare       declare-section
              always        always-section
              initially     initially-section
              assign        assign-section
            End
```

program-name : any string of text.

declare-section : variables and their types.

always-section : define certain variables as functions of others.

initially-section : define initial values of some of the variables.

assign-section : contains a set of assignment statements.

3.5.2 How do we built a UNITY program :

A program using UNITY notation is presented in a top-down design manner. First we start with the problem specification which is the starting point. Second, we transform the initial specification by a series of top-down refinements into one specification that can be implemented straitghforwardly. At each stage of the top-down refinement, we prove that the correctness of the solution is preserved. During these refinements, new program variables may be added to the specification, invariants may be introduced. The program proper is only obtained in the last step of the development process. UNITY is in the spirit of 'VERIFYING A PROGRAM BEFORE IT IS WRITTEN'. Correctness and termination are treated separately by UNITY.

3.5.3 Correctness :

The proof of correctness depends crucially on a notion of weak fairness : every statement in the program is executed infinitely often.

3.5.4 Termination :

The program terminates when it reaches a fixed point which is a stable state: a state that is repeated forever and in which all machines are passive. UNITY allows to concentrate on solving the problem at hand and treats the problem of termination/detection as a separate concern to be addressed by the implementation of a UNITY program.

3.5.5 Formal link between program design and implementation :

In UNITY, the basic structure consists of a set of statements defining relationships between the variables of the program. Two basically different notations are used : the equation scheme and the assignment scheme.

3.5.5.1 The equation scheme :

The program is essentially a set of equations separated by || for parallel or by □ for sequential. It only defines dependencies between the variables.

Example I :

```
<|| i, j : i = 1 Λ 1 < = j <= 2 :: x[i, j] = x[0,j] >
```

In the conventional way : we put x[1,1] = x[0,1] and || x[1,2]=x[0,2].

Example II :

```
x = abs(y)
x =  y  if  y >= 0 ~
    -y  if  y < 0
```

In an equation scheme, a variable appears at most once on the left side of an equation. Any variable appearing on the right-hand side of an equation is defined earlier in the ordering.

3.5.5.2 The assignment scheme :

The syntax of the assignment scheme is identical to that of the equation scheme after replacing = by :=. The interpretation leads directly to a parallel algorithm for executing the program.

Example I :

```
<|| i : 0<= j <= n :: a[ i ]:= b[ i ] >
```

it means : a[0]:=b[0] || a[1]:=b[1] || ,.., || a[n]:=b[n].

Example II :

```
x =  -1  if y < 0
      0  if y = 0
      1  if y > 0
```

is translated in assignment scheme as follows :

```
x := -1  if  y < 0 ~
      0  if  y = 0 ~
      1  if  y > 0
```

3.6 Matlab :

MATLAB [Shahian 93] stands for *matrix laboratory*. Matlab is a very powerful tool that engrosses a set of algebraic mathematics applied to a variety of fields. It is a high-performance software package for scientific and engineering numeric computations. Numerical analysis, matrix multiplication, signal processing and graphics are treated by MATLAB in an easy-to-use environment where problems and solutions are expressed in a mathematical way. Matlab can be used by itself to solve a certain number of problems or incorporated with other texts to improve the understanding of some topics or to interpret some results or to draw some graphs that other software cannot realize. Matlab allows not simply to get the answers but rather to understand how to get them through examples by use of its help function. It also provides the underlying mathematical theory and the computational algorithms for some important applications.

3.6.1 Program utilities :

Matlab is both an environment and a programming language. It supplies tools for creating functions and programs. The programs are stored under the name of M-files. Matlab language allows the user to build his own reusable tools or use application toolboxes already build by others and already existing as software ready to use to facilitate the solution of different application problems. The facilities offered by Matlab made it an easy to understand and to use tool.

3.6.2 Extending memory :

The amount of memory available for Matlab can be extended by using the window's virtual memory capabilities. Optimizing the memory is done through the use of a swap file. Matlab under window offers the ability to windows to "swap" automatically unused pages of memory out to the disk, so that they are freed up to be re-used. Smartdrive, a disk caching utility provided with windows can considerably improves the performance of the hard drive.

Conclusion :

The definitions of the different systems we will be using to design and implement the different algorithms and programs, allow a perfect ordering of the different steps we will go through to achieve our goals. The advantage of using a virtual model based on a set of processes connected through a network is a performance model that can be mapped onto various real machines. The study of the systolic SIMD mesh architecture showed that it is the most adapted for the PWT application because of its implementation on transputers and translation to Occam language. Matlab offers facilities to draw different graphs for the interpretation of results.

CHAPTER 4

The Parallel Wavelet Transform Algorithm-Design and Implementation

4.0 Introduction :

The aim of this chapter is to design and implement an algorithm to achieve the potential benefit which can be obtained by applying parallelism and pipelining computations to solve the wavelet transform and decide on the best architecture among the different architectures on which a UNITY program can be mapped: a highly parallel architecture based on a mesh SIMD systolic array. Starting from the basic algorithm, we will go through some algorithms (algorithm 'a trous', recurrent algorithm, etc.), analyzing their time complexities and comparing them. Using a straightforward algorithm leads to a prohibitive amount of data to be generated, and hence to an excessive computation time. More effective computation procedures are needed to reduce the amount of data and the computation time. A new design is presented based on UNITY model. In the design of the Parallel Wavelet Transform, algorithm called PWT, we start by stating the problem specification and from there, by a series of refinements, end up with a parallel program that can be directly translated in Occam language and mapped to a systolic array architecture using transputers. The PWT is presented as a series of simple, independent modules where each module can be simply verified using a set of input data. A digital simulation on the TDS system was undertaken, since digital simulations are now frequently used to analyze designs in an attempt to identify potentially costly design errors before the hardware is built. The testbed system components and experimental factors of the PWT algorithm are given by Figure 4.1. Analyzing a given algorithm is the process of determining how good is that algorithm which could be translated by how fast, how expensive to run and how efficient is in its use of the available resources. When analyzing an algorithm, we evaluate it using the standards metrics which are the *running* time, the number of processors, the cost, and the speedup it produces [AKL 89].

The most important measure in evaluating a parallel algorithm is its running time which is the time elapsed from the moment the algorithm starts to the moment it terminates. Thus, the running time is the time taken by the algorithm to solve a problem on a parallel computer. It is also equal to the time elapsed between the moment the first processor starts computing and the moment the last processor ends computing. Generally, the time required to solve a computational problem is evaluated by counting the number of basic operations or steps executed by the algorithm in the worst case. An expression is derived describing the number of steps as a function of the input size. The definition of a step depends on the theoretical model of computation we are using. A step corresponds to a basic operation such as add, compare, swap, that requires a constant number of time units or *cycles* to be done on a SISD computer. Whereas, for a parallel algorithm, it is the running time that is evaluated and which is usually obtained by counting two kinds of steps: *computational* steps and *routing* steps. An arithmetic or logic operation performed on a datum within a processor is defined as a *computational* step while the traveling of a datum from one processor to another via the shared memory or through the communication network is defined as a *routing* step. For a problem of size *n*, *t(n)* denotes a function of *n* that defines the parallel worst case running time of an algorithm. The other measure in evaluating a parallel algorithm is the number of processors it requires to solve a problem which is also a function of the input size and is denoted by *p(n)*. Then, the cost *c(n)* of a parallel algorithm, a function of *n*, is defined as the product of the two metrics *p(n)*, *t(n)*. We summarize the different definitions in table 4.1

running time = termination time - starting time (experimental metric).
running time = counting of computational steps and routing steps (analytical metric).
Cost = parallel running time × number of processors used.
$speedup = \frac{\text{worst-case running time of fastest known sequential algorithm for problem}}{\text{worst-case running time of parallel algorithm}}.$

table 4.1 Evaluating criteria for a parallel algorithm.

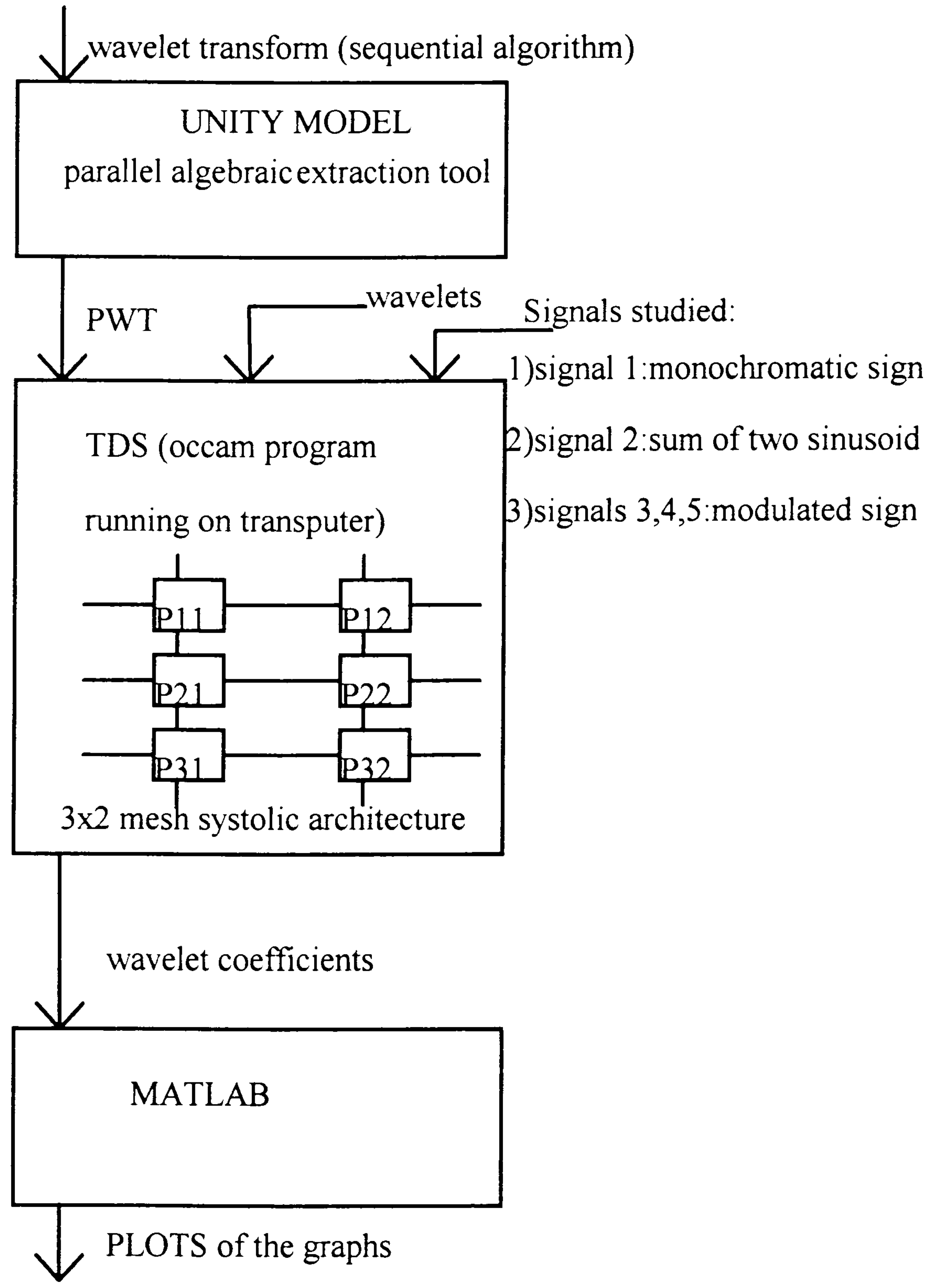

wavelet transform (sequential algorithm)
UNITY MODEL
parallel algebraic extraction tool
PWT
wavelets
Signals studied:
1)signal 1:monochromatic sign
2)signal 2:sum of two sinusoid
3)signals 3,4,5:modulated sign
TDS (occam program
running on transputer)
P11
P12
P21
P22
P31
P32
3x2 mesh systolic architecture
wavelet coefficients
MATLAB
PLOTS of the graphs

Figure 4.1 Overall structure of the PWT.

4.1 The basic algorithm :

Since the time-frequency analysis decomposes the signal in both the time domain and the frequency domain, we are looking for the frequencies that compose the signal, the time they were emitted and their lasting time (where each frequency starts and how long it lasts).

The wavelet transform of s(t) was given by equation (4.1):

$$S(b,a) = \frac{1}{\sqrt{a}} \int_0^\infty s(t)\ \psi^*(\frac{t-b}{a})\ dt \qquad (4.1)$$

where ψ^* denotes the complex conjugate of ψ. ***a*** is tied to the frequency and ***b*** to the time.

To compute the coefficients, the real part of the (complex) Morlet wavelet is often used. It was given by equation (1.1):

When s and ψ are sampled with a sampling period authorized by Shannon's theorem, the continuous equation (4.1) becomes equation (4.3):

$$S(iT_s,a) = \frac{T_s}{\sqrt{a}} \sum_n \psi^*(\frac{(n-i)\ T_s}{a})\ s(n\ T_s) \qquad (4.3)$$

where $\frac{1}{T_s}$ is the sampling frequency.

Values of ***a*** that increases by power of 2 are often used (a=a_0, 2a_0, 4a_0 ...) in order to exploit domains of extended frequencies. For each value of ***a***, corresponding to one frequency (*f=1/a*), the coefficients S(b,a) are computed. When changing the value of ***a***, the wavelet ψ is dilated or contracted and new coefficients are calculated. The wavelet coefficients given by equation (4.3) are computed by effectuating the inner product between s(n.T_s) and ψ(k.T_s). The computation of the coefficients can be implemented by a simple Finite Impulse Filter like shown by Figure 4.2.

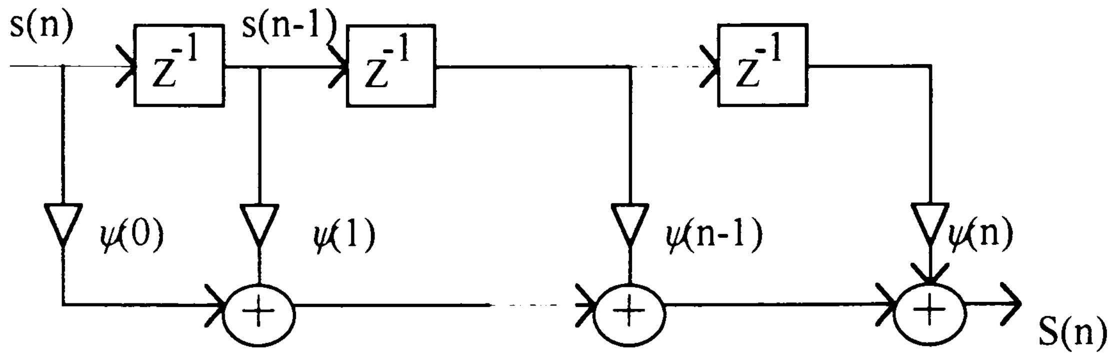


Fig. 4.2 Finite Impulse Filter.

Initially, the values of the wavelets are computed separately like shown in chapter 2 and fed as data, as well as the values of the sampled signal, the sampling period, an initial frequency **f** corresponding to the maximum frequency and the number of octaves (each octave corresponds to one searched frequency). s(k) represents the kth sample of the signal and $\psi(k,v)$ the kth sample of the analyzing wavelet corresponding to the searched frequency ($\mathbf{f}/2^v$).The corresponding flowchart is shown in Figure 4.3.

Algorithm complexity :

This algorithm leads to a very heavy load. Going through the loops in Figure 4.3, to compute one coefficient S(i,a), N multiplications and (N-1) additions are needed. To compute all the coefficients, the operation has to be repeated for the number M of octaves. The algorithm is of order $O(N^2xM)$ where N is the size of the input data.

4.2 The algorithm ' a trous' :

It was proposed by [Dutilleux 89]. We start always from the basic formula of the wavelet transform given by equation (4.1). Rather than using a continuous wavelet, dilating it and then sampling it, a wavelet sampled at a fixed number of points was used. The dilation operates then on the sequence of samples. Dilating by a factor of 2, means inserting a zero between every 2 samples in the sequence. But this is a poor representation of the analyzing wavelet. A more effective way is to replace the zeros by values. Performing a linear interpolation between the original samples is a simple and more effective solution. A linear interpolator F and a preintegrator F_1 are defined as follow and represented by Figure 4.4 and Figure 4.5 respectively:

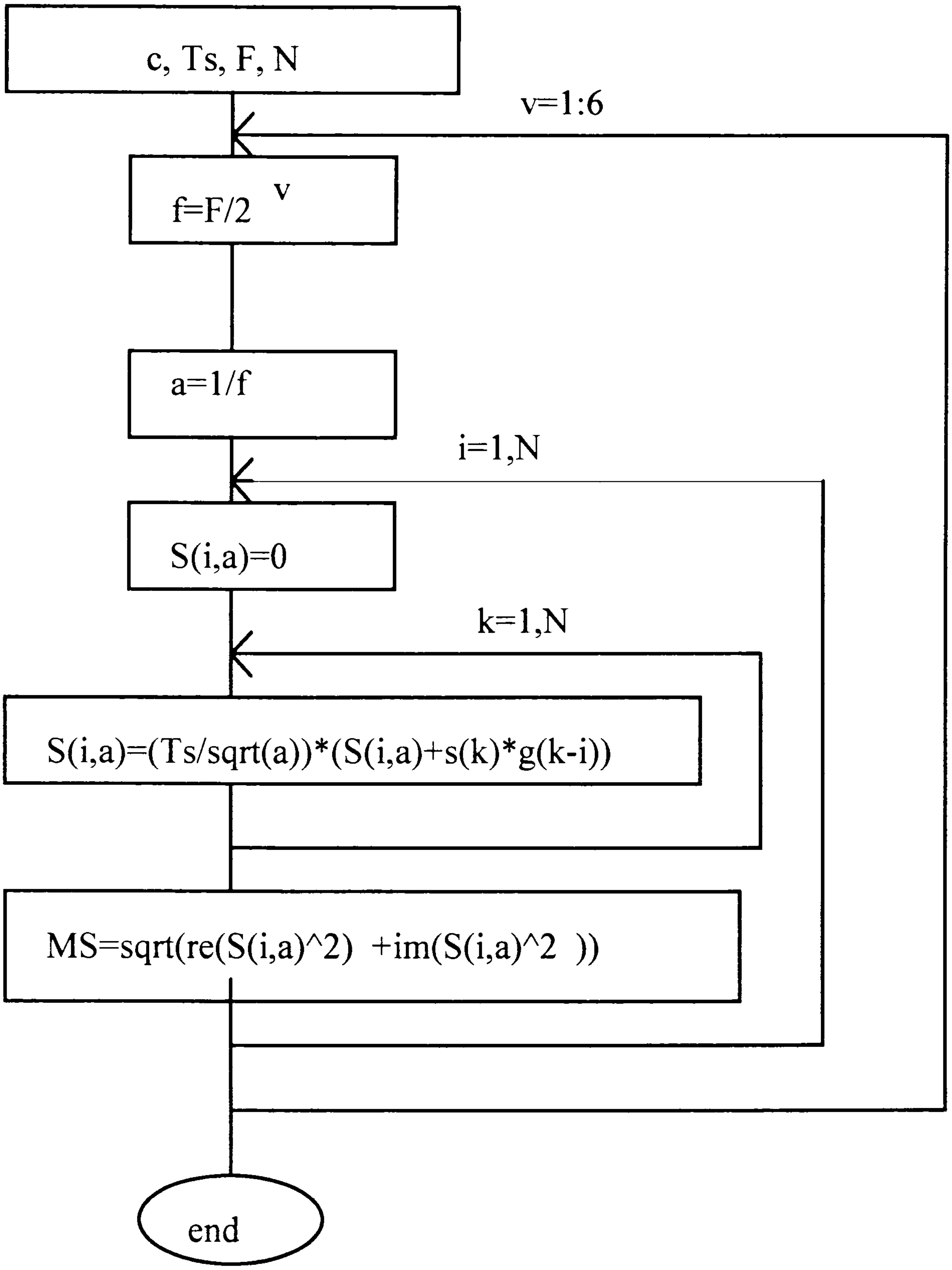

c, Ts, F, N
v=1:6
f=F/2 v
a=1/f
i=1,N
S(i,a)=0
k=1,N
S(i,a)=(Ts/sqrt(a))*(S(i,a)+s(k)*g(k-i))
MS=sqrt(re(S(i,a)^2) +im(S(i,a)^2))
end

Fig. 4.3 Flowchart of the basic algorithm.

$F=\frac{1}{2}[\psi(n)-\psi(n-1)]$ and $F_1=1-TD_2F$.

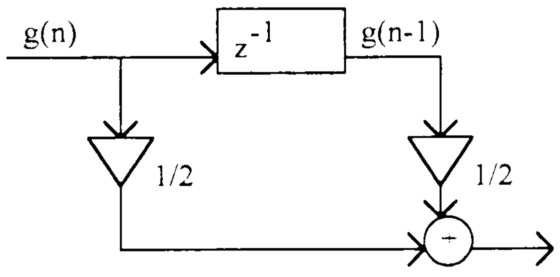


Fig. 4.4 Linear interpolator F

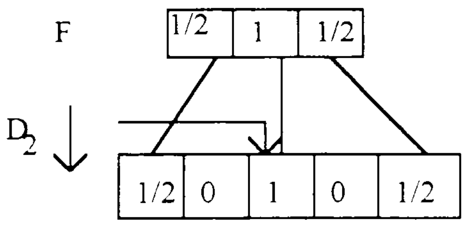


Fig. 4.5 Preintegrator F_1

The convolution with the wavelet ψ at the scale a=2n is equivalent to the product of the preintegrating filters with dilated version of the wavelet. The sequence s(n) is decimated and the final result is interpolated to get the initial dimensions. This algorithm is implemented by a parallel convolver shown in Figure 4.6. This block-structured model produces interesting results but the computation load is still very heavy.

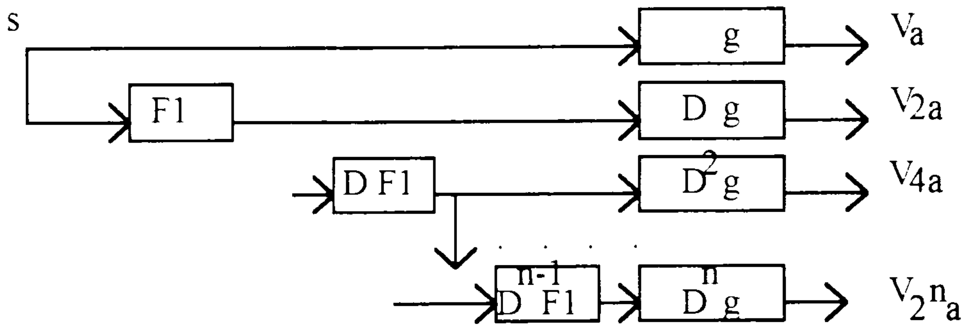


Fig. 4.6 Parallel convolver.

Algorithm complexity :

The number of operations (multiply-adds) for a single convolution using the basic algorithm is $|g_2^n|=|g_1|2^n$ where $|g|$ represents the complexity corresponding to the convolution product by the wavelet. The algorithm 'a trous' yields $|KO^n g|=|g|+n(1+|F|)$, where KO^n g is the convolution with the wavelet g and n is the number of the octave. So the exponential growth in n is reduced to a linear one.

4.3 The recurrent algorithm :

The recurrent algorithm was proposed by [Barrat 90]. It was implemented to improve the time complexity of the wavelet transform. Starting always with equation (4.1), and for practical reasons, sampling the signal and the wavelet, yields to equation (4.3). Instead of using as mother wavelet, the Morlet wavelet, a recursive function was used. The choice is given by :

$h(t) = (1 + c'|t|)\, e^{-\sigma|t|} e^{ic't}$ where c' = 5 and σ = 1.5.

The Z-transform of h(ξt) is calculated. This function is then decomposed into two contributions :

$$\begin{aligned}
h_+(\xi t) &= (1+\sigma\xi t)\, e^{-\sigma\xi t} e^{ic'\xi t} && \mathit{for}\ t \geq 0 \\
&= 0 && \mathit{for}\ t < 0 \\
h_-(\xi t) &= 0 && \mathit{for}\ t \geq 0 \\
h_-(\xi t) &= (1-\sigma\xi t) e^{\sigma\xi t} e^{ic'\xi t} && \mathit{for}\ t < 0
\end{aligned}$$

$$H_+ = \frac{1 + a_1 z^{-1}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

We can see that we can have
$$H_- - H_+^*(z^{-1}) - h(0) = \frac{a_1' z + a_2' z^2}{1 + b_1^* z + b_2^* z^2}$$

where :
$$\begin{aligned}
a_1 &= (\sigma\xi T - 1)\, e^{-\xi(\sigma - ic')T} \\
b_1 &= -2\, e^{-\xi(\sigma - ic')T} \\
b_2 &= e^{-2\xi(\sigma - ic')T} \\
a_1' &= (a_1^* - b_1^*) \\
a_2^* &= -b_2^*
\end{aligned}$$

Then, the wavelet coefficients are computed by the following method :

$$\begin{aligned}
S_+(k,\xi) &= s(k) + a_1 s(k-1) - b_1 S_+(k-1) - b_2 S_+(k-2) \\
S_-(k,\xi) &= a_1' s(k+1) + a_2' s(k+2) - b_1^* S_-(k+1) - b_2^* S_-(k+2) \\
S(k,\xi) &= \sqrt{\xi}\,[S_-(k,\xi) + S_+(k,\xi)]
\end{aligned}$$

If the computations of S_- and S_+ could be done in parallel, a factor of 2 could be gained in computation time.

Algorithm complexity :

The time complexity of the recursive implementation doesn't depend on the number of voices n. In general, for the complex calculation of $S(k,\xi)$, we have :

22 multiplications and 21 additions which gives a complexity of order 22.

Thus, the global complexity of the recurrent algorithm is of order $nN \cdot h_1$ where N is the number of samplings, n the number of the octave, h_1 the complexity of the recurrent algorithm. This method allows to compute all the wavelet coefficients at the same time.

4.4 The Parallel Wavelet Transform Algorithm :

Increasing demands for high speed in many real time applications have stimulated the development of a number of new parallel algorithms. To optimize the algorithm, we try to combine both sequential and parallel programming techniques like shown in the following figure (Fig . 4.7); since it is usually possible for a given problem to adjust the processing time per communication by the use of a combination of parallel and sequential algorithm.

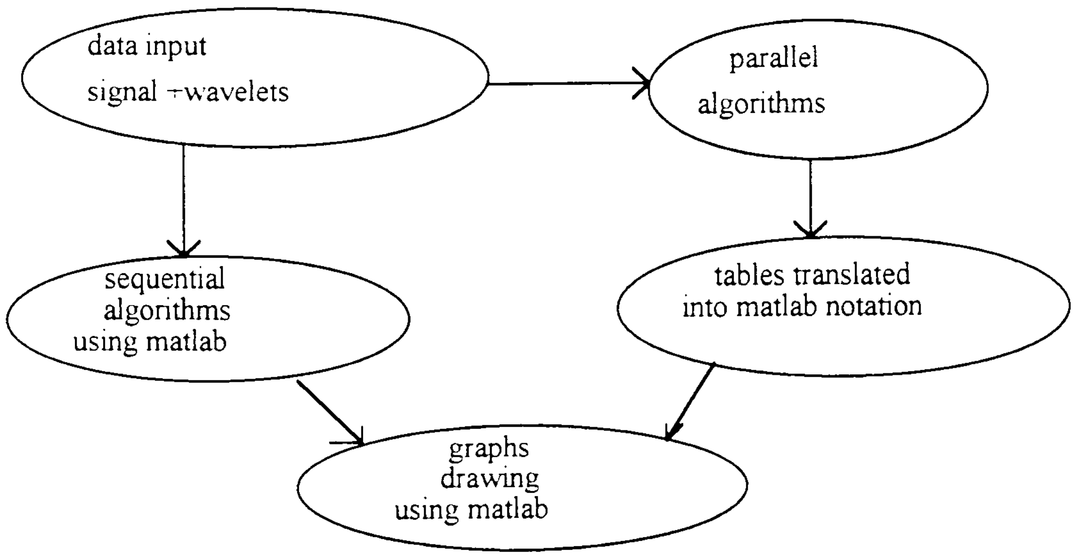


Fig.4.7 Combination of sequential and parallel algorithms.

4.4.1 The sequential algorithm :

We want to analyze functions of a given signal s(t) in terms of wavelets obtained by shifts and dilations from a suitable basic wavelet called the mother wavelet. Suppose that s(t) is a temporal signal (signal to be analyzed) and $\psi(t)$ is the analyzing wavelet (mother wavelet depending on two parameters *a* and *b* where *a* is tied to the frequency and *b* to the time). To decompose into a wavelet series, ψ (t) must be admissible, well localized, oscillating, and a real valued function ($\psi \in L^2(R)$); ψ (t) and s(t) are of finite energy. Once the mother wavelet ψ (t) has been chosen, it generates the other wavelets. A typical choice for the mother wavelet is the real part of the Morlet wavelet or its conjugate (eq.4.2). Starting with $\psi(t)$, we get the other wavelets $\psi_{a,b}$ (t) by varying ***a*** and ***b*** like stated in section (4.1).

For all s(t) such that s(t) has a finite energy, s(t) is a linear combination of the $\psi_{a,b}$'s:

$$s(t) = \frac{1}{a^2}\sum_a \sum_b S_{a,b}(t)\ \psi(t)$$

where $S_{a,b}$(t) is given by equation (4.1).

Suppose that s and ψ are scaled with a period T authorized by Shannon's theorem. $S_{a,b}$ is sampled by setting b and t as follow : $b=iT$ and $t=nT$, giving equation (4.3).
Developing, we get the different values of S for i=1,2,.....,N and n=1,2,...,N where N is the number of samples and ***a*** the octave.

$$S_{i,a} = S(iT,a) = \frac{T}{\sqrt{a}}[\psi(-i\frac{T}{a})s(1T) + \psi(-(i-1)\frac{T}{a})s(2T) + \ldots + \psi(0\frac{T}{a})s(iT)]$$

To get all coefficients, we compute the inner product of the signal s with the translated versions of the wavelets (varying b). We get the following scheme representing all coefficients .

$$S_{1,a} = S(1T,a) = \frac{T}{\sqrt{a}}[\psi(0\frac{T}{a})s(1T) + \psi(1\frac{T}{a})s(2T) + \cdots + \psi((N-1)\frac{T}{a})s(NT)]$$

$$S_{2,a} = S(2T,a) = \frac{T}{\sqrt{a}}[\psi(-1\frac{T}{a})s(1T) + \psi(0\frac{T}{a})s(2T) + \cdots + \psi((N-2)\frac{T}{a})s(NT)]$$

.

.

$$S_{N,a} = S(NT,a) = \frac{T}{\sqrt{a}}[\psi(-N\frac{T}{a})s(1T) + \psi(-(N-1)\frac{T}{a})s(2T) + \cdots + \psi(0\frac{T}{a})s(NT)]$$

The signal is swept by all the wavelets. The other coefficients are calculated in the same way for different values of ***a***. The final matrix for the coefficients of the wavelet transform is as follow :

$$S(kT,a) = \begin{bmatrix} S_{11} & . & . & . & S_{1M} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ S_{N1} & . & . & . & S_{NM} \end{bmatrix}$$

So the wavelet analysis is the representation of the signal as a function of time and frequency at the same time. Suppose that the signal is sampled on N points , and it is generated for M different frequencies. Using a basic algorithm, for one frequency we need to generate a matrix $\psi(N,N)$, with **NxN** entries. If N=300 and M=6 then 9000 values are needed to compute one coefficient. To compute all the coefficients, we need 6×9000 values , which takes a lot of memory area. We remark that the matrix presents a lot of symmetry, since the function representing the analyzing wavelet in this case is an even function : all values on the diagonals are identical. Making use of this property, we elaborate the following algorithm:

1) Compute the values of the wavelet for half the interval which corresponds to either calculating the values of the row or the column.

2) Generate the other half of the wavelet by symmetry.

3) The wavelet matrix for one octave is reduced to one column vector. The wavelet matrices for all octaves are reduced to one matrix *ψ(N,M)* where each column vector represents one frequency wavelet matrix.

4) By circulating those values we compute all the transform coefficients.

$$\psi_{n,m}(t) = \begin{bmatrix} \psi(1,f_1) & \ldots & \psi(1,f_6) \\ \psi(2,f_1) & \ldots & \psi(2,f_6) \\ \psi(3,f_1) & \ldots & \psi(3,f_6) \\ \psi(4,f_1) & \ldots & \psi(4,f_6) \\ \psi(5,f_1) & \ldots & \psi(5,f_6) \end{bmatrix}$$

The improved version of the sequential algorithm makes use of two sums taking the diagonals as symmetric lines :

$$S(kT,a) = \frac{T}{\sqrt{a}}\left[\sum_{m=1}^{k-1} s(m)\psi((k-m+1)T,a) + \sum_{m=k}^{N} s(m)\psi((m-k+1)T,a)\right] \tag{4.4}$$

As said before, it takes into account the fact that all the values of the diagonals are the same. So we need only to generate the values of the first row or first column for each wavelet *ψ*, and by permutating these values of *ψ* we obtain the values of all the wavelets. The algorithm making use of this scheme is given in Figure 4.8.

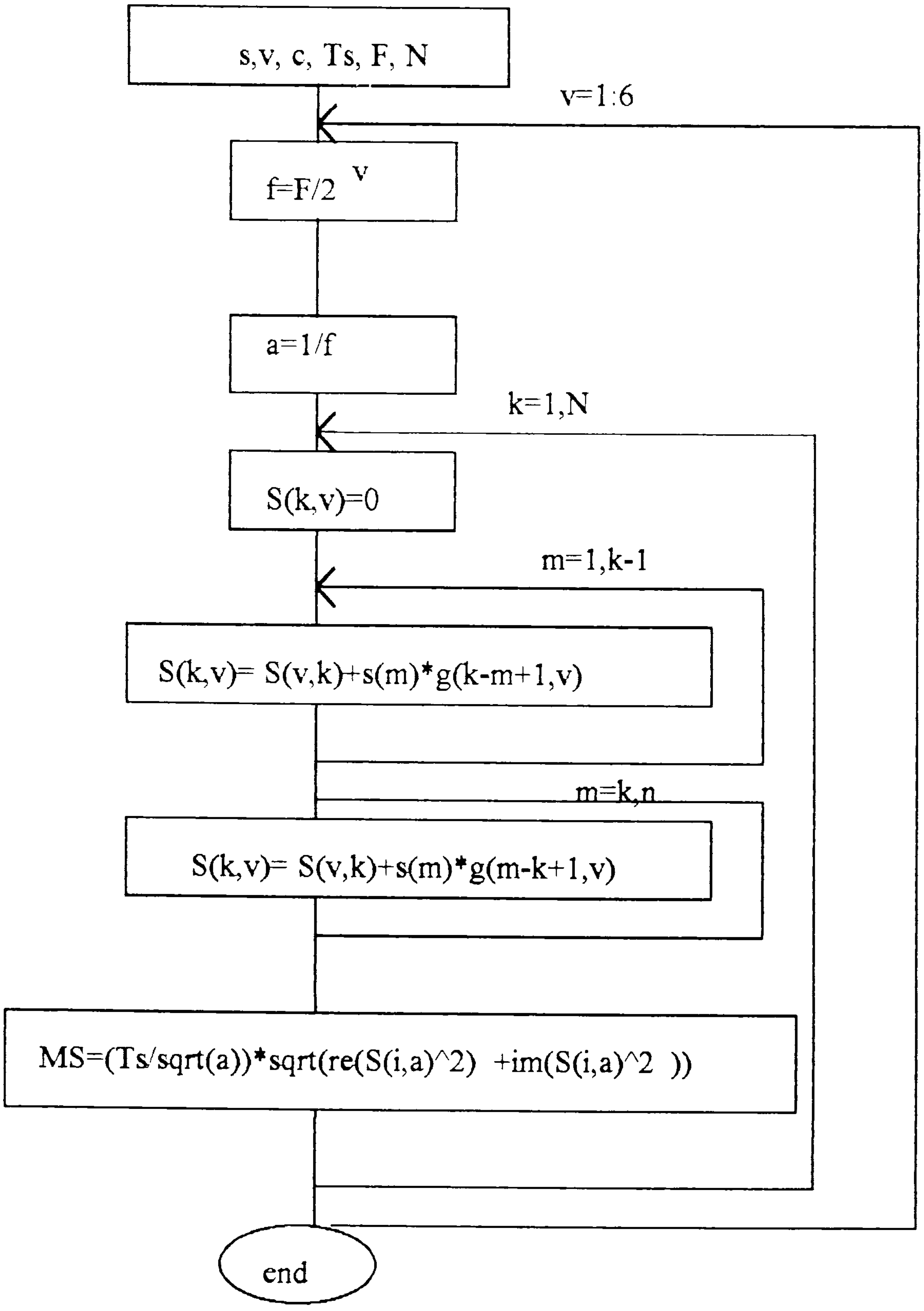

s,v, c, Ts, F, N
v=1:6
f=F/2 v
a=1/f
k=1,N
S(k,v)=0
m=1,k-1
S(k,v)= S(v,k)+s(m)*g(k-m+1,v)
m=k,n
S(k,v)= S(v,k)+s(m)*g(m-k+1,v)
MS=(Ts/sqrt(a))*sqrt(re(S(i,a)^2) +im(S(i,a)^2))
end

Fig. 4.8 Flowchart of the sequential algorithm.

Illustration:

Let N=5 be the number of sampling points M=6 the number of frequencies. We want to decompose the signal s(t) that is sampled as follow : $s = (s_1 \quad s_2 \quad s_3 \quad s_4 \quad s_5)$.

Using a basic algorithm, for each frequency f such that $a = \frac{1}{f}$, one wavelet $\psi_{i,j}$ is generated. Six matrices of the same form as the following are generated for the wavelets.

$$\psi(kT,a_i) = \begin{bmatrix} \psi_{1,1} & \psi_{1,2} & \psi_{1,3} & \psi_{1,4} & \psi_{1,5} \\ \psi_{1,2} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} & \psi_{1,4} \\ \psi_{1,3} & \psi_{1,2} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} \\ \psi_{1,4} & \psi_{1,3} & \psi_{1,2} & \psi_{1,1} & \psi_{1,2} \\ \psi_{1,5} & \psi_{1,4} & \psi_{1,3} & \psi_{1,2} & \psi_{1,1} \end{bmatrix}$$

The wavelet coefficients are computed based on equation (4.3). Developing, we obtain for k=1,2 :

$$S(1.T,a) = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{bmatrix} \begin{bmatrix} \psi_{1,1} & \psi_{1,2} & \psi_{1,3} & \psi_{1,4} & \psi_{1,5} \\ \psi_{1,2} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} & \psi_{1,4} \\ \psi_{1,3} & \psi_{1,2} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} \\ \psi_{1,4} & \psi_{1,3} & \psi_{1,2} & \psi_{1,1} & \psi_{1,2} \\ \psi_{1,5} & \psi_{1,4} & \psi_{1,3} & \psi_{1,2} & \psi_{1,1} \end{bmatrix}$$

$$S(2.T,a) = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{bmatrix} \begin{bmatrix} \psi_{1,2} & \psi_{1,3} & \psi_{1,4} & \psi_{1,5} & \psi_{1,1} \\ \psi_{1,1} & \psi_{1,2} & \psi_{1,3} & \psi_{1,4} & \psi_{1,5} \\ \psi_{1,2} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} & \psi_{1,4} \\ \psi_{1,3} & \psi_{1,2} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} \\ \psi_{1,4} & \psi_{1,3} & \psi_{1,2} & \psi_{1,1} & \psi_{1,2} \end{bmatrix}$$

And so on for all the other coefficients.

Based on equation 4.4, and making use of the steps described for the sequential algorithm, we get the following scheme:

$$S(1\,T,a)=\begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{bmatrix}\begin{bmatrix} \psi(1,f_1) & \psi(1,f_2) & \psi(1,f_3) & \psi(1,f_4) & \psi(1,f_5) & \psi(1,f_6) \\ \psi(2,f_1) & \psi(2,f_2) & \psi(2,f_3) & \psi(2,f_4) & \psi(2,f_5) & \psi(2,f_6) \\ \psi(3,f_1) & \psi(3,f_2) & \psi(3,f_3) & \psi(3,f_4) & \psi(3,f_5) & \psi(3,f_6) \\ \psi(4,f_1) & \psi(4,f_2) & \psi(4,f_3) & \psi(4,f_4) & \psi(4,f_5) & \psi(4,f_6) \\ \psi(5,f_1) & \psi(5,f_2) & \psi(5,f_3) & \psi(5,f_4) & \psi(5,f_5) & \psi(5,f_6) \end{bmatrix}$$

$$S(2,T,a)=\begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{bmatrix}\begin{bmatrix} \psi(2,f_1) & \psi(2,f_2) & \psi(2,f_3) & \psi(2,f_4) & \psi(2,f_5) & \psi(2,f_6) \\ \psi(1,f_1) & \psi(1,f_2) & \psi(1,f_3) & \psi(1,f_4) & \psi(1,f_5) & \psi(1,f_6) \\ \psi(2,f_1) & \psi(2,f_2) & \psi(2,f_3) & \psi(2,f_4) & \psi(2,f_5) & \psi(2,f_6) \\ \psi(3,f_1) & \psi(3,f_2) & \psi(3,f_3) & \psi(3,f_4) & \psi(3,f_5) & \psi(3,f_6) \\ \psi(4,f_1) & \psi(4,f_2) & \psi(4,f_3) & \psi(4,f_4) & \psi(4,f_5) & \psi(4,f_6) \end{bmatrix}$$

And so on. The hole matrix for the transform coefficients is computed as $S(N,M)=s(N,N)\times\psi(N,M)$

$$S_{11}=s_1\psi(1,f_1)+s_2\psi(2,f_1)+s_3\psi(3,f_1)+s_4\psi(4,f_1)+s_5\psi(5,f_1)$$
$$S_{21}=s_1\psi(2,f_1)+s_2\psi(1,f_1)+s_3\psi(2,f_1)+s_4\psi(3,f_1)+s_5\psi(4,f_1)$$

...

...

$$S_{56}=s_1\psi(5,f_6)+s_2\psi(4,f_6)+s_3\psi(3,f_6)+s_4\psi(2,f_6)+s_5\psi(1,f_6)$$

$$S(kT,a)=\begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} & S_{15} & S_{16} \\ S_{21} & S_{22} & S_{23} & S_{24} & S_{25} & S_{26} \\ S_{31} & S_{32} & S_{33} & S_{34} & S_{35} & S_{36} \\ S_{41} & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\ S_{51} & S_{52} & S_{53} & S_{54} & S_{55} & S_{56} \end{bmatrix}$$

Making use of equation (4.4) and to improve the time complexity of the above algorithms, we generated a parallel algorithm based on UNITY program and mapped to Occam model.

4.4.2 The Parallel algorithm :

There are basically three approaches that can be considered when designing parallel application programs [May 87]. The first approach is the dataflow decomposition where the parallelism is obtained by decomposing the algorithm into a number of smaller, simpler components which can be executed in parallel. The parallelism arises directly from the algorithm. In this scheme, component processes could be written concurrently and the program could be executed on a pipeline of processors. The second approach is the data structure decomposition where the parallelism is obtained by distributing the data to be processed between a number of processors in a way that preserves the geometrical structure of the data. The third approach is the processor farm scheme where a number of processors are used to process data farmed out by a controlling processor.

We propose a parallel algorithm called *Parallel Wavelet Transform* (PWT) where the two sums are computed in parallel by $N \times M$ processors, where N is the number of samples and M the number of voices (frequencies). The scheme used to obtain the fine grid parallelism is the dataflow parallelism; the modules are written to run as concurrent processes on a pipeline of processors. For the calculations of the wavelet transform using the parallel algorithm, all rows are processed in parallel as well as columns. To transform the sequential algorithm into a parallel algorithm, we make use of UNITY model which is the model underlying the parallel computer.

4.4.2.1 Specification of the PWT : A first solution

We derive a concurrent program from its specification using UNITY formalism. Starting from an initial UNITY specification, we proceed in a series of refinement steps until the specification is restrictive enough to be translated directly into UNITY code. Let $S(b,a)$ be the wavelet transform. Replace it by its sampled version $S(iT,a)$ using UNITY notation.

$$S(iT,a) = \langle + j :\ 0 \le j < N :: s(j) \times \psi(\frac{(j-i)T}{a}) \rangle$$

4.4.2.2 Refinement specification :

A refinement of this specification: let k represents the row of the (k,i)th element to calculate. We partition the elements of the signal s into 2 sets s_1 and s_2 according to the kth element of s.

$$S_1(iT,a)=\langle +j:0\leq j<i::s_1(j)\times\psi((j-i+1)T,a)\rangle$$

$$S_2(iT,a)=\langle +j:i\leq j<N::s_2(j)\times\psi((i-j+1)T,a)\rangle$$

For all i, $0 \leq i < N$

$$\begin{aligned}S(iT,a)&=\langle +j:0\leq j<N::s(j)\times\psi(iT,a)\rangle\\&=\langle +j:0\leq j<i::s_1(j)\times\psi((j-i+1)T,a)\rangle+\\&\quad\langle +j:i\leq j<N::s_2(j)\times\psi((i-j+1)T,a)\rangle\end{aligned}$$

Programs are derived into equational scheme before transformed into multiple assignment statements.

4.4.2.3 Derivation of the program from the specification :

Program P_1 is derived from the definition of S into equational scheme.

```
Program P1 { PWT transform }
    {i , j , k are quantified : 0 ≤ i < N, 0 ≤j < N, 0 ≤ k < M}

    always

    < || i, k :: answer(iT,k) = <+ j :: s[j] × ψ[j , k]>>

end {P1}.
```

Program P_1 is the sequential version and can be executed in $O(N^2xM)$ on one processor or in O(logN) time on $O(N^2xM)$ processors where each S(iT,a) could be computed using N processors, or in O(N) on O(NxM) processors. If one processor is assigned to compute one S(iT,a), a deadlock problem may arise: s(i) and ψ (j,k) can be accessed by an arbitrary number of processors. The constraint appropriated for pipelining and systolic architectures is that a variable may appear at most once in the right side of any statement. In a systolic array, each processor has connections to a small number of neighbors with which it can communicate data values. The 1st refinement is the relax of the access requirement to the data by creating a variable S where the sum of the 1st j terms is cumulated before passing it to the next processor.

Program P_2

```
  {i , j , k are quantified : $0 \le i < N$, $0 \le j < N$, $0 \le k < M$}
  always
  < || i , k :: S[ 0 , i , k ] = 0 >
  [] < [] j :: < || i , k :: S[j+1 , i , k] = S[j , i , k] + s[ i , j ] * $\psi$[j , k]>>
  [] < || i , k :: answer [i , k] = S[N , i , k] >
end {$P_2$}.
```

The data access problem is still existing. The 2nd refinement tries to solve the data access by passing the values of s after being used, horizontally to the P(j,k+1) processors to be used, and sums of the first j terms are computed and passed vertically to the P(j+1,k) processors (Fig.4.9). The values of ψ are put in matrix GWAVE.

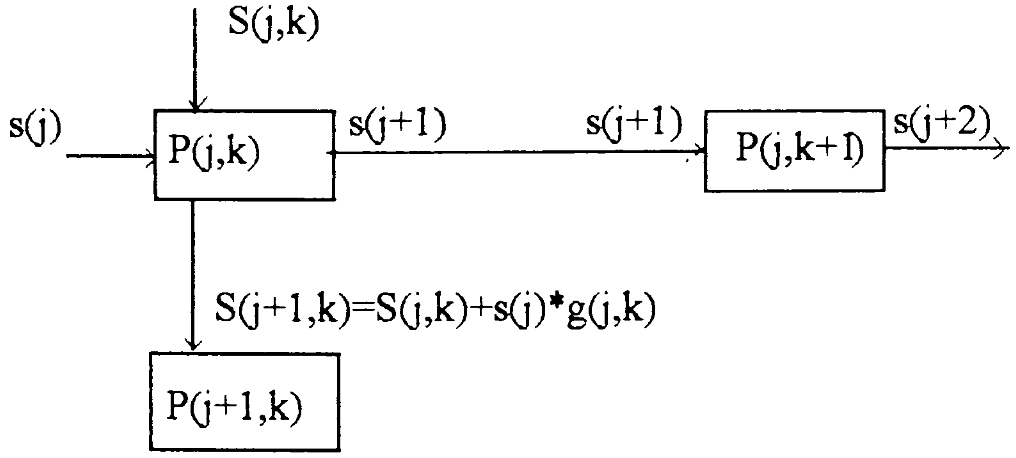


Fig.4.9 communication between processors.

```
Program P_3
 Declare
 NS : array [0..N , 0..n-1 , 0..m-1] of integer
 WE : array [0..N , 0..n-1 , 0..m-1] of integer
always
{i , j , k are quantified : 0 ≤ i < N, 0 ≤j < N, 0 ≤ k < M}
 < [] i , k :: NS [0 , k , i] = 0 >
[]< [] i , j :: WE[j , 0 , i] = s[i , j] >
[]<[]i,j,k::NS[j+1,k,i]=NS[j,k,i]+WE[j,k,i]*GWAVE[j,k]|| WE[j,k+1,i]=WE[j,k,i]>
[] < [] i , k :: answer[i , k] = NS[N , k , i] >
end { P_3 }.
```

The third refinement is to combine all equations into equations suitable for parallel synchronous execution. Time variables t(j,k,i) will be associated to NS(j,k,i) and WE(j,k,i). The t(j,k,i) is the step number at which NS(j,k,i) and WE(j,k,i) are to be computed. In program P_4 the values of s are put in matrix XVEC.

```
Program P_4
{i , j , k are quantified : 0 ≤ i < N, 0 ≤j < N, 0 ≤ k < M}
  always
  < [] t : 0 ≤ t < N + N + M : :
  < || i , k : t = i + k :: NS[0 , k , i] = 0  >
    || < || i , j : t = i + j :: WE[j , 0 , i] = XVEC[i , j]
    ||<|| i,j,k:t=i+j+k+1::NS[j+1,k,i],WE[j,k+1,i]=
         NS[j,k,i]+WE[j,k,i]*GWAVE[j,k],WE[j,k,i]>
 || < || i , k : t = N + i + k + 1 : : ANSWER [i , k] = NS[N, k , i] >>
end { P_4}.
```

Program P_4 has N+N+M equations. For each value of t, $0 \le t \le N+N+M$, we have one equation. A refinement of program P_4 gives program P_5 where operational behavior of processors in a synchronous architecture is better captured. P_5 is in assignment scheme. Variables NS(j,k), WE(j,k) local to P(j,k) whose i^{th} values are the same as NS(j,k), WE(j,k) as program P_4 .

```
Program $P_5$
Declare t : integer
initially t = 0
{i , j , k are quantified : $0 \le i < N$, $0 \le j < N$, $0 \le k < M$}
< || i , k : t = i + k : : NS[0 , k] : = 0 >
|| < || i , j : t = i + j : : WE[j , 0] : = XVEC[i , j] >
|| <|| i,j,k:t=i+j+k+1::NS[j+1,k],WE[j,k+1]:=NS[j,k]+WE[j,k]*GWAVE[j,k],WE[j ,k]>
    || < || i , k : t = N+ i + k + 1 : : ANSWER [i , k] = NS[N , k ] >
    || t : = t + 1 if t < 2 * N + M

end { $P_5$ }.
```

A variable indexed (j,k) is read only by processor (j,k) which is the limited data access constraint. This program is translated in Occam language.

4.4.2.4 Implementation on a systolic array–Synchronous mode :

A systolic algorithm is designed to be executed on a systolic array. The data access is restricted to one processor per data in each step which is appropriate for pipelining and systolic networks. In one computation step, each processor reads data from its input lines, performs some computations and writes new data values into the output lines. The algorithm is implemented on a grid of N×M synchronous processes arranged in a mesh configuration to compute the N×M transform coefficients. Mesh rows are numbered from 1 to N and mesh columns from 1 to M. The wavelets $\psi_{j,k}$ are local to the processors while the values of the signal are fed into the boundary processors in the leftmost column as shown in Figure 4.10 for N=3 and M=4. Initially the topmost row receives a stream of zeros corresponding to the cumulating sums S. To ensure that s_{ij} meets the $\psi_{j,k}$ in processor P(j,k) at the right time, row i of s falls one time unit behind row (i-1) for $2 \le i \le 3$. At the beginning of the ith step, processor (j,k) computes $S(j,k)+s(j)\times\psi_{j,k}(f_i)$ and assigns it to S(j+1,k) and passes *s(j)* to *s(j+1)*. The wavelet transform coefficients are available from the (N,k)th processors, on the bottommost row as shown in Figure 4.10. In this program, the data is pipelined in both horizontal and vertical directions. It flows from processor P(j,k) to its neighboring processors. The synchronism is achieved by the 'rendez-vous' method: a processor can consume data only when it is produced on its input channels, otherwise the processor waits. The UNITY program was translated to Occam language and simulated on transputers using a SIMD architecture. The simulation was made successful because of the structure of Occam: objects are simulated by processes that can run concurrently and communications are simulated by channels.

Algorithm complexity :

It takes 2*i+j+N-2 steps for s_3, to reach P(i,j). To reach P(N,M) which is the last processor to terminate, 2*N+M-2 steps are required to compute the product. Program P_5 can be executed in O(N+N+M) time on O(NxM) parallel synchronous processors. *t* starts at 0. At each step number, a number of operations is executed. The sums are computed in $t \leq N+N+M$. The overall program is then executed in $t \leq N+N+M$. The program has been implemented using the Transputer Development System (TDS3). An example of the timing of a program where N=3 and M=2 is shown in Figure 4.10.

4.4.2.5 Running programs within TDS :

The parallel wavelet transform runs within TDS which is an integrated development system used to develop Occam programs for a transputer network. Most of the TDS runs on the transputer board plagued on an IBM PC such as an IMS T414 transputer that has 2 Mbytes of RAM and all the appropriate development software such as the editor and compiler utilities. The "server" is a program residing on the IBM PC that provides the TDS with access to the terminal and filling system of the IBM PC. Occam programs are entirely edited, compiled and run within the TDS. When a program is run within TDS, it runs in parallel with, and connected to certain components of the TDS such as channels, their protocols and a number of supporting communications of these channels. The channels from.isv and to.isv are used to connect the application process PWT to the server program running on the host IBM PC. The KS and SS channels are running in parallel. They are channels to and from the terminals (keyboard and screen) to enable the PWT process to communicate data to and from these devices. The monitor procedure contains the interface with the TDS. It supplies the values of keystrokes on the host keyboard to the application while in parallel conveying data and results to the host terminal. Termination of parallel programs are the duty of the programmer. Since there is no control flow, the program terminates when it reaches a fixed point which is a stable state : a state that is repeated forever and in which all machines are passive. Three programs are written to read the DOS files corresponding to the signal values and the wavelet values. The multiplier is the parallel application program to compute the coefficients. A fourth program is written to output the file obtained by the parallel process as a DOS file to be converted to MATLAB notation to plot the graphs. The application, the keyboard.handler and the screen.handler run in parallel. The block diagram giving the PWT and the server is given by Figure 4.9.

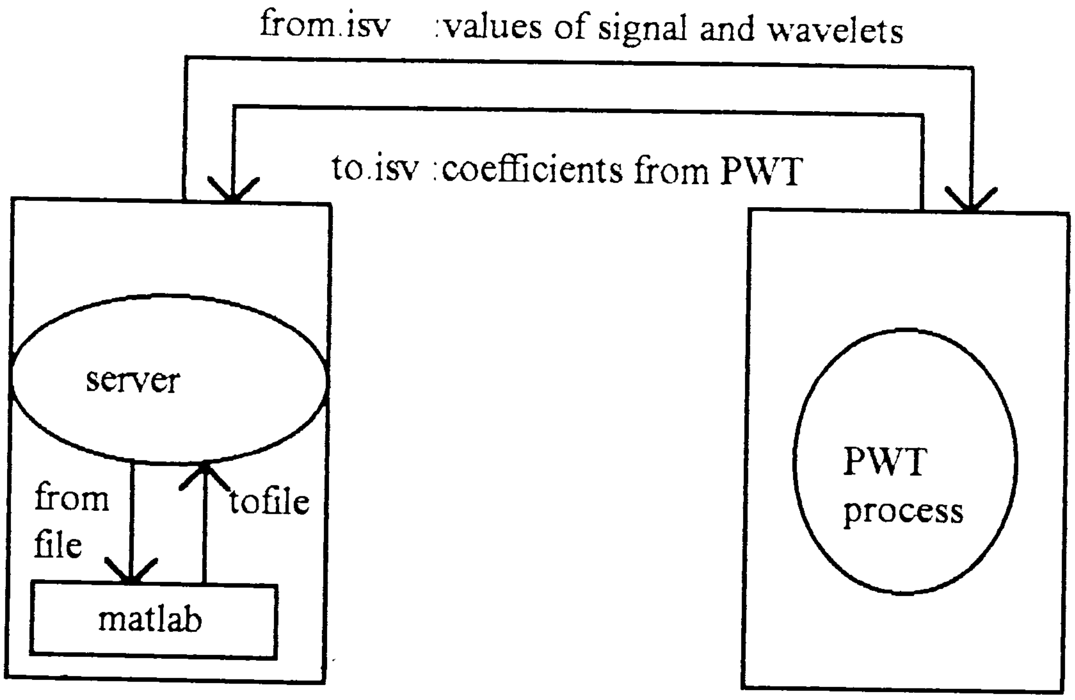
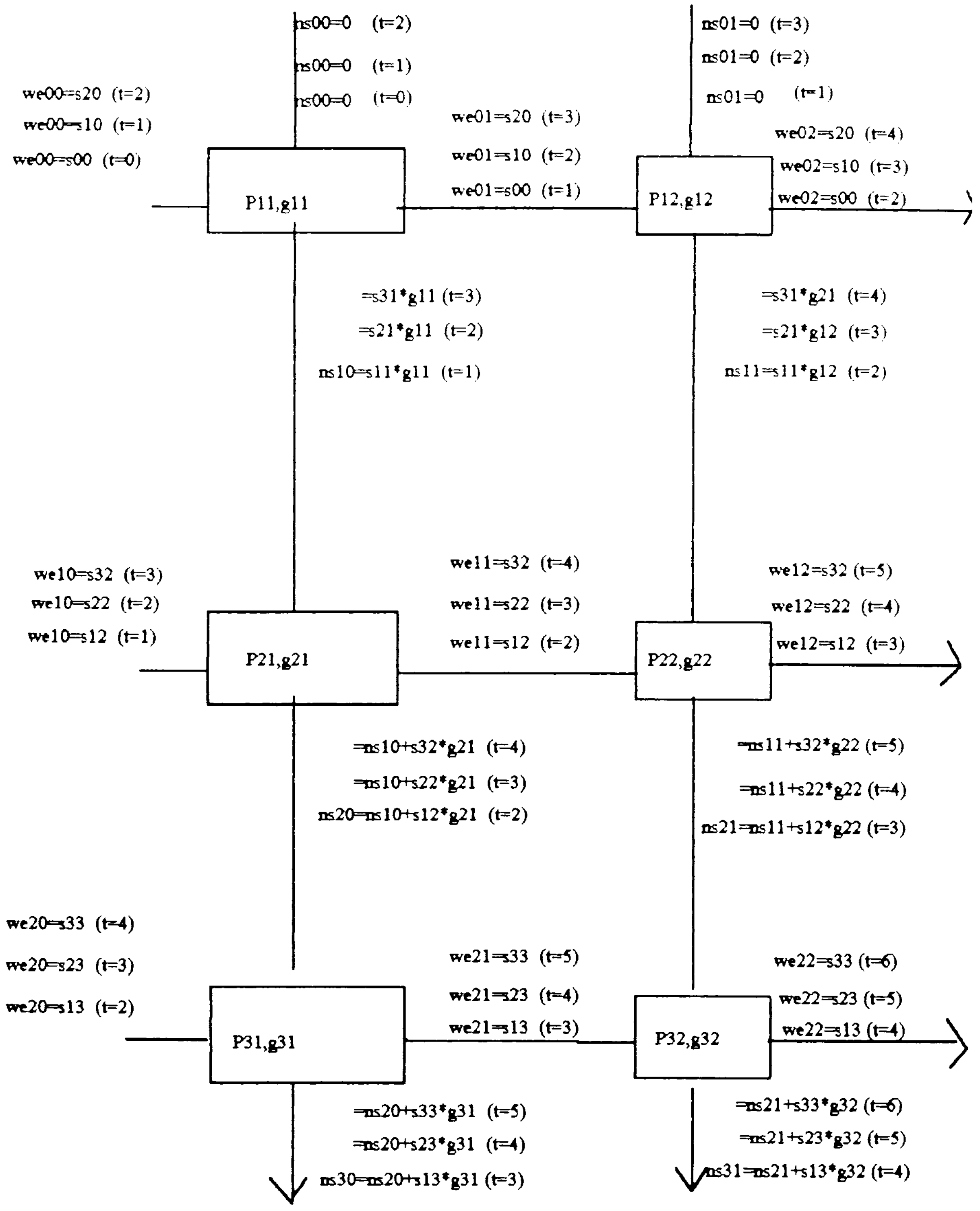

from.isv :values of signal and wavelets
to.isv :coefficients from PWT
server
from file
tofile
matlab
PWT process

Fig.4.9 Block diagram of the PWT on the TDS



A00 =ns30=s11*g11+s12*g21+s13*g31 (t=4)   A01=ns31=s11*g12+s12*g22+s13*g32 (t=5)

A10=ns30=s21*g11+s22*g21+s23*g31 (t=5)   A11=ns31=s21*g12+s22*g22+s23*g32 (t=6)

A20=ns30=s31*g11+s32*g21+s33*g31 (t=6)   A21=ns31=s31*g12+s32*g22+s33*g32 (t=7)

Fig. 4.10 A timing of a program implemented on a 3x2 mesh architecture.

The main program and the subprograms are as follow :

... libraries !the libraries describes the interfaces available to user programs.

... declarations !declaration types of the variables and initial values.

... procedure read signal ! inputs the DOS file containing the signal values.

... procedure read wave ! inputs the DOS file containing the wavelet values.

... procedure write coefficients ! writes the file obtained by the PWT as a DOS file.

... procedure multiply ! computes coefficients

... procedure sinkV !sink vertical: outputs the values of the signal inputted on the w.e channel.

... procedure sinkH ! sink horizontal: outputs results (coefficients) on the n.s channel.

... PWT !the main program using the different processes.

... go to TDS !return to the development system after execution of the program.

The main program gives the arrangement of the procedures.

```
SEQ
    ... set initial values.
    read.signal (from.isv, to.isv, keyboard, screen, XVEC) !reads the signal
                                   values
    read.wave (from.isv, to.isv, keyboard, screen, GWAVE) !reads the wave
                              values
    SEQ
        ... initialize matrix coeff to zero
        ...  compute matrix
        write.coeff(from.isv, to.isv, keyboard, screen, ANSWER)
```

The first procedure reads the values of the signal generated by MATLAB.

```
Proc read.signal (CHAN OF SP from.isv, to.isv, CHAN OF KS keyboard,
                 CHAN OF SS screen, [ ]REAL32 sig)
    ... declarations
      SEQ
         PAR
             so.keystream.from.file.(from.isv, to.isv, keyboard,
                                     'c:\matlab\bin\sig.dat', bres)
             ss.scrstream.sink(echo)
             ... body of procedure
  ... test for errors
:
```

The second procedure reads the values of the wavelets generated by MATLAB.

```
Proc read.wave(CHAN OF SP from.isv, to.isv, CHAN OF KS keyboard, CHAN OF SS
                 screen, [ ]REAL32 wave, INT n, m)
    ... declarations
      SEQ
       PAR
          so.keystream.from.file.(from.isv, to.isv, keyboard, 'c:\matlab\bin\wave.dat',
bres)
             ss.scrstream.sink(echo)
        ... body of procedure
  ... test for errors
  :
```

The third procedure writes the wavelet transform coefficients to a file into MATLAB.

```
Proc write.coeff (CHAN OF SP from.isv, to.isv, CHAN OF KS keyboard, CHAN OF
                                SS screen, [ ]REAL32 coeff, INT n, m)
   ... declarations
     SEQ
       PAR
          ... body of procedure
          so.scrstream.to.file (from.isv, to.isv, tofile,
                                  'c:\matlab\bin\coeff.dat', bres)
          ks.keystream.sink (keyboard)
 ... test for errors
:
```

The fourth procedure is the main procedure. It computes the coefficients in parallel.

```
Proc mult (VAL REAL32 aij, CHAN OF REAL32 n,s,w,e)
  REAL32 result, bij :
  SEQ
    PAR
      n ? result !the cumulative sum is stored in result which enters from the north
channel.
    w ? bj ! the values bj of the signal are entering from the east channel.
    result := result + (aij * bj) ! the coefficient is calculated here.
     PAR
      s ! result    ! the partial sum is outputted south of the channel.
     e ! bj !the value of the signal is outputted on the west channel.

:
```

Procedure sink horizontal : to output the values of the signal on [illegible] [illegible]tmost channel.

```
Proc sinkH (CHAN OF REAL32 e)

    REAL32 h :

    SEQ

        e ? h  ! the values of the signal are sinked on the east.we[illegible] [illegible]nel.

  :
```

Procedure sink vertical : to output the final results.

```
Proc sinkV (CHAN OF REAL32 s, REAL 32 R)

    REAL32  v :

    SEQ

        s ? v  ! the final values of the coefficients are sinked verti[illegible]

        R := v ! the value of v goes to R to be stored in the matrix [illegible]

  :
```

4.4.2.6 Comparison of the algorithms: basic, recurrent, 'a trous', PWT

The comparison of the PWT algorithm and the other algorithms in terms of *running time* is depicted in table 4.2. The time complexity is defined as the running time executed by the algorithm in the worst case (parallel running time: number of parallel *cycles* where each instruction is performed in the earliest possible cycle). The PWT algorithm is more performing than the basic algorithm and as performing as the recurrent algorithm for the same kind of applications. We see that the exponential growth in N of the basic algorithm is reduced to a linear one. It is highly efficient and flexible. With reasonable memory requirements, the algorithm attempts to avoid the repeated calculations of the wavelets and the frequent access to critical regions unlike the algorithm 'a trous'. The PWT running time is very reduced since many processors are executing at the same time which reduces the running time of the algorithm.

Compared to the mentioned above algorithms, the PWT presents different aspects like parallelization that was used for the first time for the continuous wavelet transform using Occam language and transputers.

Algorithm	running time (analytic definition) N : number of samples M : number of octaves
Basic Algorithm	$O(N^2 x M)$
Algorithm 'a trous'	$2^{M-1}.N.\lvert g_1 \rvert$, $\lvert g_1 \rvert=100$, g_1 : complexity of the convolution product.
Recurrent Algorithm	$M.N. \lvert h_1 \rvert$, $\lvert h_1 \rvert=50$, h_1 : complexity of the recurrent algorithm.
PWT Algorithm	$O(N+N+M)$

Table 4.2. Comparison table of the running time of the algorithms used.

Conclusion:

The problem was to describe an algorithm which allows a given signal to be decomposed in an optimal way as a linear combination of judiciously chosen time-frequency atoms. To optimize the design, we used an architecture that is systolic and exploits pipelining and parallelism in order to obtain high speed and throughput. We made the analysis simple: the steps of the algorithm are easy to follow. The analytic approach provides a simple framework for the evaluation and explanation of the design. The PWT algorithm is presented as a series of simple, independent modules, which are highly portable. For each module, simple verification tests can be used. The PWT uses a lot of parallelism. All coefficients are computed at the same time in a pipelined manner. This processing method permits a processing element column to perform parallel computation on the data, and then recirculating it to compute new values for the coefficients. So, the approach consists of the reordering/permutation of the data instead of recalculating it whenever needed. Data is compressed by a factor of 1/N compared to the basic algorithm. The new design offers a fully pipelined high performance circuit that is very suitable to VLSI implementation.

CHAPTER 5

Experimental Results

5.0 Introduction :

In this chapter, some signals to be tested on the PWT algorithm are presented. The signals used are signals generated analytically. A presentation and a discussion of the results obtained shows the usefulness of the wavelet transform and the effectiveness of the PWT algorithm. Finally, a table to compare the performance of the PWT algorithm with some other algorithms treating the same subject is included.

5.1 Description of the signals :

Different signals were simulated to validate the PWT algorithm. They were either generated in the experimental contexts described above or obtained using numerical methods. The *first* signal studied is a *monochromatic signal*, sum of 3 sinusoids with different frequencies starting and ending at different times. The *second* signal is also a *monochromatic signal* composed of the sum of two sinusoids whose frequencies are very close. The *third*, *fourth* and *fifth* signals are *frequency modulated signals* following different modulation laws. The *first* among the last *three* signals follows an increasing hyperbolic law, while the modulation law of the *second* is decreasing hyperbolic and the modulation of the *third* is linear. In all cases, the modulation law is given by v_i (t).

5.2 Results obtained:

We present examples of decomposition of different signals. The signals are based on the use of synthetic data not corrupted by noise. The wavelet used is the Morlet wavelet. The analyzed signal is drawn next to the representation using wavelets: in abscissa, the position parameter b related to time and in ordinate the voices related to frequencies. The calculations were realized on an IBM PS/1 using TDS3 for the parallel computations and MATLAB to plot graphs. The signals as well as different graphs representing them are given below. For all signals, the localization parameter b varies from 0 to NT, a, the scaling parameter corresponds to the inverse of the frequency. For the *first* signa*l* the initial frequency is 16 KHz. To Each voice v, corresponds f_i= 16000/2^v Hz where 0 <v <7. The *second* signa*l* corresponds to a signal composed of the sum of two sinusoids. The frequencies f_1 and f_2 are very close. The *third* and *fourth* signals represent frequency modulated signals. They follow a hyperbolic law (increasing for the former and decreasing for the latter). The 5th signal represents another frequency modulated signal. The signal here follows a linear law. The *sixth* signal is a superposition of different signals. To obtain a positive, easy to present results, we compute the modulus of the wavelet transform called scalogram of the analyzed signal s(t). The wavelet coefficients are represented by their scalograms. The different graphs drawn represent:

1) The signal to be analyzed.
2) Its Fourier spectrum.
3) Its wavelet bidimentional representation or its modulation law (for FM signals).
4) A 3-D plot of the wavelet decomposition or its wavelet decomposition (for FM signals) .

Table 5.1 shows the simulation parameters used in the decomposition of the different signals.

Parameter	Meaning
N	number of samples
T_s	sampling period
F	maximum frequency
f_v	frequency corresponding to the octave v
v_0	average frequency
M	number of octaves
v	the octave
a	parameter related to frequency
b	position parameter

Table 5.1: Simulation parameters.

First signal :

The signal analyzed here is a monochromatic signal, sum of three sinusoids given by:

$$s(t)=Y(t-83T)\cos(2\pi f_0 t)+Y(t-170T)\cos(4\pi f_0 t)+Y(t-250T)\cos(8\pi f_0 t)$$

Y: step function, $N=400, T=\frac{1}{32\,10^3}$

$f_v=\frac{F}{2^v}, v\in[1,6], F=16000Hz$, $f_0=1000$ Hz

The graphs representing this decomposition are shown by Figure 5.1.

Second signal :

The signal analyzed is composed of the sum of two sinusoids. It is given by:

$$s(t)=\cos(2\pi f_1 t)+\cos(2\pi f_2 t)$$

where $f_1=1000$ Hz and $f_2=2000$ Hz.

The graphs representing this decomposition are shown by Figure 5.2.

Third and fourth signals :

The following are modulated signals. For a frequency modulated signal, the energy is distributed according to the modulation rule: increasing or decreasing modulation rule. Let s(t) be the following frequency modulated signal given by:

$$s(t)=\cos(\frac{2\pi\upsilon_0}{\alpha}\times\log\left|1+\alpha.t\right|)$$

where the frequency law is given by:

$$\upsilon_i(t)=\frac{\upsilon_0}{1+\alpha.t}$$

where υ_0 represents an average frequency and α is given by equation(5.7) and equation(5.8

$$\alpha = \frac{-F}{(F+\upsilon_0)T} \quad \text{if the law of modulation is increasing .} \qquad (5.7)$$

$$\alpha = \frac{-F}{(\upsilon_0 - F)T} \quad \text{if the law of modulation is decreasing.} \qquad (5.8)$$

N=400, F=80 Hz, t=2 s and ν_0 = 40 Hz.

The decomposition of s(t) is shown in Figure 5.3, and in Figure 5.4 for increasing and decreasing modulation laws respectively.

Fifth signal:

Another frequency modulated signal is a one whose modulation law is linear and is represented by : $s(t) = \cos(2\pi(\nu_0 t + \frac{Ft^2}{2T}))$

and $$\nu_i(t) = \nu_0 + \frac{Ft}{T}$$

N=400, F=80 Hz, T=2 s, and

Figure 5.5 represents the different graphs for s(t).

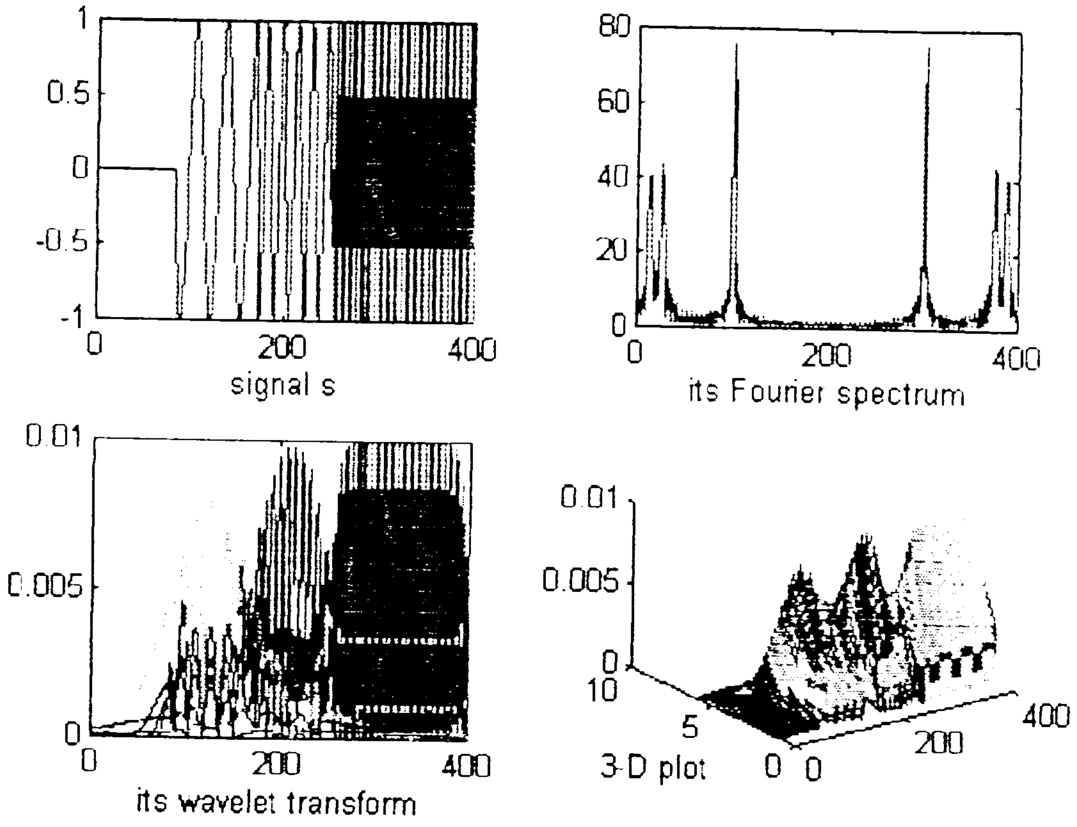


Fig. 5.1 Decompositon of a monochromatic signal.

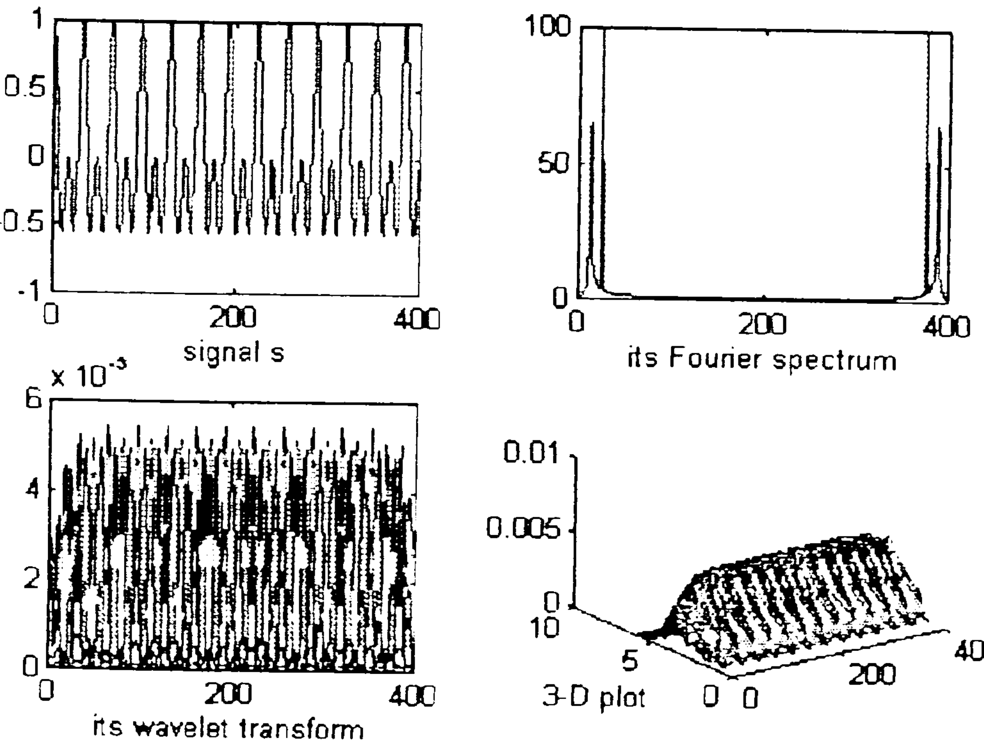


Fig. 5.2 Decomposition of a signal sum of two sinusoids.

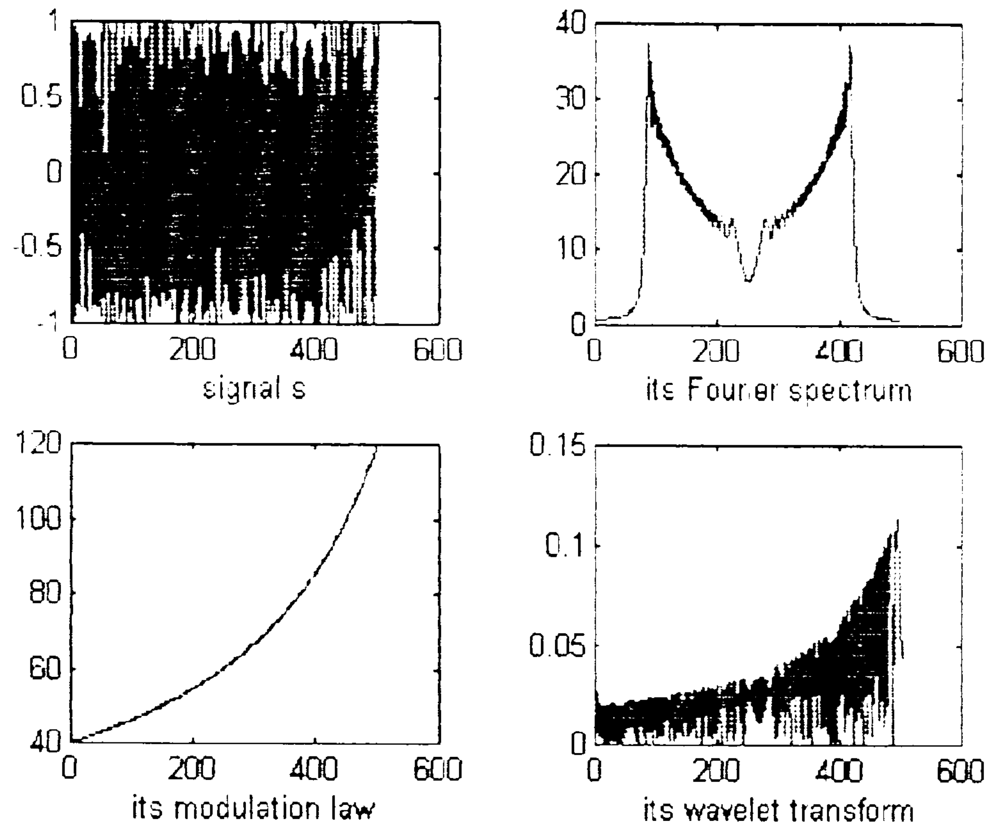


Fig. 5.3 Decomposition of a signal whose modulation law is increasing.

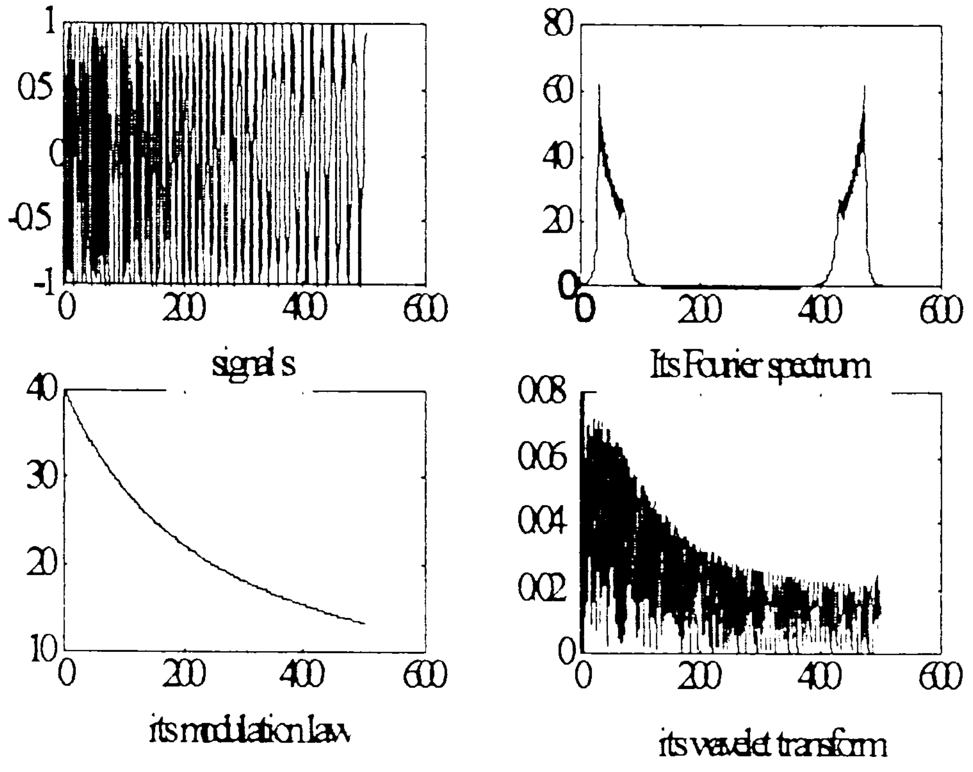


Fig. 5.4 Decomposition of a signal whose modulation law is decreasing.

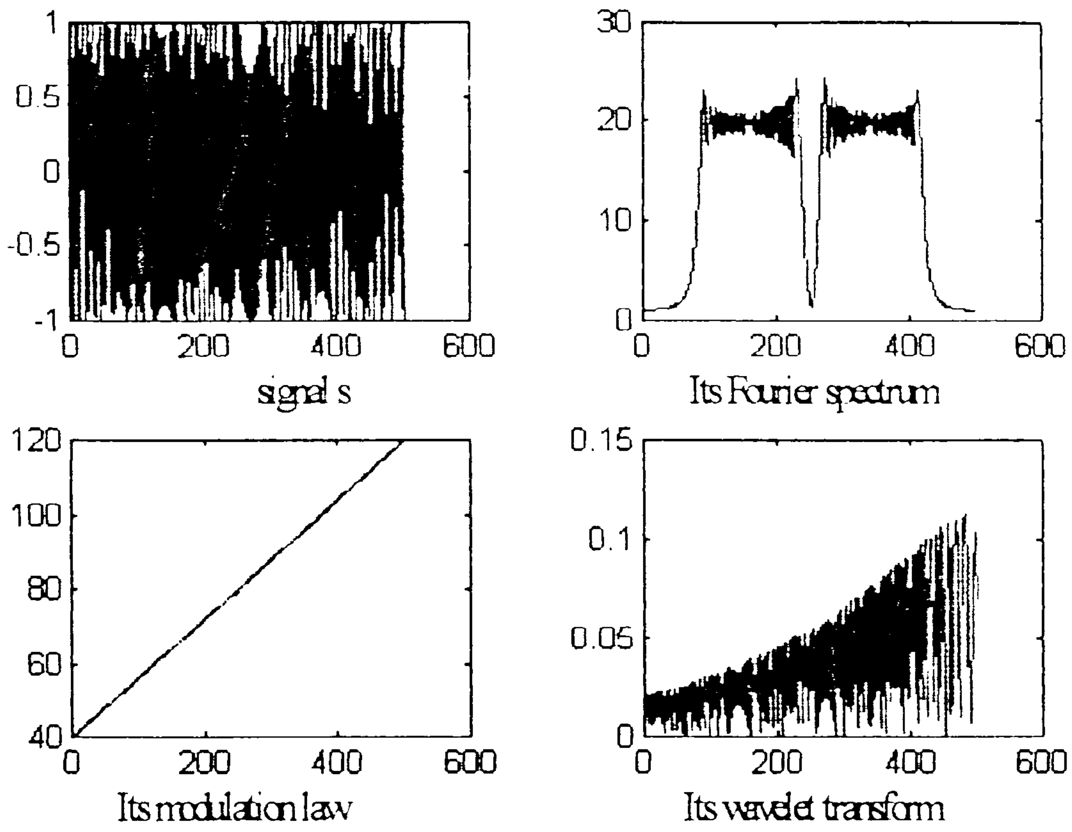


Fig. 5.5 Decomposition of a signal whose modulation law is linear.

5.3 Discussion :

Figure 5.1 represents the wavelet analysis of a monochromatic signal composed of three sinusoids. This decomposition gives the energy distribution of the signal in the time-frequency plane and the distribution of the frequencies along the time-axis. (i.e. when each frequency starts and how long it lasts). The wavelet coefficients represents the three fundamental frequencies of the signal (1 MHz, 2 MHz, 4 MHz). The temporal starting of each sinusoidal function is indicated by a cone and the peaks correspond to the different frequencies. These different forms with different dimensions informs on the energy contribution of each sinusoidal component. We also see the transition phenomena represented by the convergence of the cones that points out to this phenomena.

Figure 5.2 corresponds to a signal composed of the sum of 2 sinusoids. We get then a shuffling phenomena : at certain time the two waves are superposed and at others, they annihilate mutually. This phenomena is used in acoustic field by musicians to tune their instruments. In fact when the two ropes vibrate at very close frequencies, we can clearly hear the variations of the sound intensity. When they are tuned and vibrate at the same frequency, this phenomena disappear. Thus, the wavelet transform reacts very similarly to the ear. The wavelet transform is very small at the time the two waves cancel out and becomes more important when they are superposed. If the frequencies of the two sinusoids get away, they will appear at different highs in the time-frequency plane. The decomposition for monochromatic signals gives the energy distribution of the frequencies along the time axis. In this representation, for each fundamental frequency, the shuffling phenomena corresponds to the amplitude of the sinusoid.

Figures (5.3), (5.4), (5.5) give the distribution of the energy in the time-frequency domain. We remark that the energy distribution of a signal obeys to the law that describes the evolution of the signal in the time-frequency plane. Parseval's theorem verifies the conservation of the energy even after the transformation. In the time-frequency representation, we have two degrees of freedom to estimate qualitatively the energy distribution. According to the frequency axis :

– for a fixed t, we have the evolution of the energy.

According to the time axis :

– for a given frequency, we get the energy distribution as a function of time.

This type of analysis is used in the field of petroleum, cardiograms, echograms, etc. The time-frequency analysis allows one to observe the nature of the signal whose parameters are not known (rate of amplitude, frequency modulation, etc.).

As a conclusion, we can say that one of the major advantages of the time-frequency analysis is to allow a global and a local analysis of the signal at the same time, i.e., a panoramic analysis and a detailed analysis. We remark that the scalogram, by its positive character can assume the energy density role, which is not the case for a number of time-frequency representations (i.e. Wigner-ville representation which is a pseudo-representation of the energy density since it can be negative). More, the scalogram is a real number, which can be translated by a visualization of the energy distribution.

The wavelet transform is an easy to understand time-frequency representation compared to others that are more complex.

The implementation presented improves the time complexity of the basic algorithm by an exponential factor, as well as the algorithm 'a trous' and the recurrent algorithm like shown by the following (table 5.1). The results are exact and do not admit any approximations due to the truncation of the wavelet since the wavelet is taken on the same number of samplings as the signal unlike the algorithm 'a trous'.

One important part of studying algorithms is the analyzing performance that is used to predict how the *running time* of an algorithm [Blelloch 96] grows as a function of the input size. To analyze the performance, UNITY model was needed to estimate the costs. UNITY model is a virtual model that can be mapped to several architectures. The performance is calculated in terms of the number of *instruction cycles* a computation takes which correspond to the *running time* of the algorithm and is expressed as a function of the input size and the number of processors used. Speed-up and efficiency can also be defined in terms of running time and number of processors in order to evaluate an algorithm. One of the most important criteria in systolic array design is the number of processors of the array. The number of processors must be carefully chosen in order to achieve the maximum speed-up with the least number of processors. The number of processors is minimized because processors occupy precious resources such as silicon area on the chip of the array.

In general, the time required to solve a problem is obtained by counting the number of basic operations or *steps* executed by the algorithm in the worst case. While the *running time* for a sequential algorithm is in terms of *computational* steps, the *running time* for a parallel algorithm is obtained by counting *computational* and *routing* steps. The *running times* of the different algorithms is summarized in the following table. Referring to table 4.1 and replacing the parameters by values, we obtain table 5.1 to show the costs of the different algorithms for different values of the octave v. The analytic formulas giving the time complexity of the different algorithms were defined in *section 4.2* for the algorithm 'a trous', in *section 4.3* for the recurrent algorithm and in *section 4.4* for the PWT.

Different Algorithms used and their complexities for the M^{ith} octave, N=500	Basic Algorithm $O(N^2xM)$	Algorithm a 'trous' $2^{M-1}N\|g_1\|$, $\|g_1\|=100$	Recurrent Algorithm $MN\ \|h_1\|$, $\|h_1\|=50$	PWT Algorithm O(N+N+M)
M=1	$25\ 10^4$	$5\ 10^4$	$25\ 10^3$	1001
M=2	510^5	10^5	$5\ 10^4$	1002
.				
.				
M=10	$25\ 10^5$	$2^8\ 10^5$	2510^4	1010

Table 5.2 Comparison table of the running time of the different algorithms used.

5.4 Concluding Remarks :

Since the purpose of the wavelet analysis is to determine the components of the signal and their chronology as well as the distribution of energy for the modulated signals, the PWT algorithm meets its requirements. One of the interpretation of the results is the representation of the different frequencies composing the signal and the time of their apparition by different graphs and the distribution of the energies of finite energy signals in the time-frequency domains. The wavelet representation is adapted in terms of energy concentration to frequency modulated signals. It respects the marginal distribution in time and frequency. But there exist different ways to represent the data produced by the wavelet analysis and different interpretations of the results depending on what we look for. The scalogram can be used efficiently to estimate the repartition of the energy of the signal in the time-frequency plan. It is thus possible to determine at each frequency the time t which corresponds to the maximum energy.

What we can conclude is that the obtained results for the PWT computation are very similar to those obtained with other algorithms. The only difference comes from the different complexities. The 'time complexity' of the PWT shows that this algorithm presents a more reduced time. The simulation was successful since results with good accuracy better than those of the algorithm 'a trous' and as well as the recurrent algorithm were found. We have tested the validation of the PWT algorithm and its programming on different signals. We can say that the PWT algorithm has a satisfactory resolution since all frequency maxima are returned for the different signals tested. The PWT allows a good use of the wavelet transform since it is comparable to the direct calculation of the transform with a computation time very reduced. Our design presents an essential gain in speed since the systolic pipeline architecture has an advantage for real-time applications where the high speed is the main criterion.

The proposed design offers many advantages among which we can mention the following :

• A parallel implementation of the wavelet transform leads to improvement in the processing of real time signals. The architecture used is systolic and exploits pipelining and parallelism in order to obtain high speed and throughput.

• The advantage in using transputers, is that the process context switching overhead the process communication cost is very reduced, because the chip has been specially designed for parallelism.

• The simulation was made successful because of the structure of Occam where objects are simulated by processes that can run concurrently and communications are simulated by channels.

CHAPTER 6

Conclusion

6.0 Introduction :

Since the wavelet transforms applications require large amounts of data to be processed and high computing speeds, new designs were suggested to compress data and to speed up the operations.

An improved sequential algorithm using only (MxN) values of the wavelets was used compared to the (MxNxN) values needed by the basic algorithm was designed. Since the computations of the wavelets require a large amount of times for trigonometric function calculations, so it is a time saving method. Then a parallel form of the wavelet transform to give a concrete gain in performance over some other codes was presented.

Mainly, the results of the wavelet transform allows one to visualize in a very meaningful way the different frequencies composing the signal as well as the instants of their apparitions. For frequency modulated signals, they are used to determine the significant components of the signal and the time *t* which corresponds to the maximum energy.

6.1 Summary of work done:

In this work, we proposed a methodology to transform an initial specification algorithm by a series of refinements into a specification that can be implemented directly on a particular architecture such as a SIMD mesh architecture in our case. The specified algorithm PWT was formally defined using the program notation UNITY. The use of the UNITY model allowed a simple description of the parallel algorithm and a well-defined model for analyzing the performance. In order to program and simulate the systolic arrays onto the transputer, the formal program was translated to Occam language that expresses parallel computations and which was designed to be the programming language for the INMOS transputer. We have tested the validation of the PWT algorithm on different signals. We can say that the PWT has a satisfactory resolution since all frequency maxima are returned for the different signals tested.

6.2 Comparison with related work:

The implementation of the PWT algorithm has some advantages compared to others used up to here. It improves the time complexity of the basic algorithm by an exponential factor. While the algorithm 'a trous' leads to an extensive data redundancy, the PWT algorithm is very useful for data compression since data values of the wavelets are obtained by data recirculation over the wavelet matrix. Compared also to the algorithm "a trous " and the recurrent algorithm, the PWT results are good since they do not admit any approximations due to the truncation of the wavelets since the wavelets are taken on the same number of samplings as the signal. The PWT allows a good use of the wavelet transform since it is comparable to the direct computation of the transform with a computation time reduction.

Compared to other processing techniques, the wavelet processing of the signal as time-frequency analysis allows one to take into account the significant structures of the signal contained in every frequency band which is impossible to do with a frequency analysis only.

6.3 Original contribution and potential application of the results:

The merit of such an approach (UNITY model) is the fact that programs are viewed like mathematical objects derived straightforwardly from their specifications that can be mapped to a variety of different architectures, in contrast to the "*a posteriori*" verifications commonly found in literature. Such an architecture shows the impact of a parallel implementation on the speed in the processing of real time signals since the execution time is reduced which makes the response very rapid.

This computational advantage of low time complexity is also due to the use of transputers since overheads involved with switching the processor between processes and in process communication is very much reduced, because the chip has been specially designed for parallelism.

The wavelet analysis yields to two time-frequency representations: square modulus and phase modulus. In this representation, the elementary frequency components are separated. For illustration, we represented only the first one which is called the scalogram. The scalogram is the squared modulus of the wavelet coefficients. It characterizes the distribution of energy of the signal in the time-frequency plane, a research field that has been the major occupation of many scientists like Ville [Ville 48] and continued by many others. The energy distribution can be used to efficiently estimate the group velocity of a dispersing wave. It is thus possible to determine at each frequency band, the time *t* which corresponds to the maximum energy. Another application of the time-frequency analysis is the acoustic field. The application of the FFT does not give a correct analysis of all type of signals like illustrated by the following example taken from music. Suppose than two notes are emitted successively and are analyzed from an initial time t_i to a final time t_f : a high note, followed by a note with lower frequency. Using a Fourier analysis, the signal spectrum is very diffused and the times where each frequency starts and finishes does not appear. While the wavelet decomposition shows the starting and the finishing time of each note emitted. In echogram, the wavelet analysis is useful for localizing diseases in some organs. A signal is emitted than received, and the analysis of the perturbed area of the received signal allows to localize exactly the area of the disease, since having the velocity of the beam and the time the perturbation appears, the distance is just velocity times time. But the wavelet technique finds its best uses in seismic-reflection which is an oil searching method and for which the wavelet transform was created.

6.4 Suggestion for further improvements:

The process which is described in this thesis must be seen as one possible method for the analysis of data recorded, but it still has to be applied to many recordings of field data before its effectiveness can be really proved. Almost all examples given here are based on the use of synthetic data, which are not corrupted by noise and where obtained with a numerical method. But even if the signals tested give satisfactory results, they are still 'academic'. It will be interesting to test the method on concrete signals and to develop some real applications for it.

The background noise may affect the good precisions of the results obtained. The principal difficulty of the PWT resides in the presence of noise. When the signal is not filtered, the method does not eliminate the noise but rather amplifies it which can alter the values of the coefficients in some cases (e.g.when the noise is very important). The synthesis of the signal from these values may give aliasing signals. To detect only significant structures, the method has to be improved to reduce the incidence of the noise on the values of the wavelet coefficients.

The second improvement resides in the implementation of the PWT. Since the PWT algorithm was only simulated on the TDS system, the operation could be speed up by its implementation on a network of transputers. And since the system PWT uses simple processing elements, basically of multiply-add types, it is possible in the future to built an application specific integrated circuit (ASIC) based on the design of the PWT.

REFERENCES

[AKL 89] S.G. AKL , *The Design And Analysis of Parallel Algorithms*, Prentice-Hall International Editions (1989).

[Askew 88] C. Askew, "Occam and the transputer -Research and applications", OUG-9. Proceedings of the 9th Occam User Group Technical Meeting. September 1988 - Southampton, UK, pp. 19-21.

[Barrat 90] M. Barrat, O. Lepetit, "Fast processing of the wavelet transform", Signal Processing, vol. 8 , nos. 1., (1990), pp. 43-49.

[Battle 87] G. Battle,"A block spin construction of ondelettes, Part 1 : Lemarie functions", Comm., Maths., Phys., vol. 110, (1987), pp. 601-615.

[Blelloch 96] G.E. Blelloch, "Programming Parallel Algorithms", Communications of the ACM, vol. 39, nos. 3, pp. 86-97, (1996).

[Bonnet 96] P. Bonnet, O. Marguin, D. Remond, "Une analyse en ondelettes par quintes", traitement du signal, vol. 8 nos. 3., (1996), pp. 117-125.

[Burns 87] A. Burns, *Programming in Occam 2,* Addison-Wesley Publishing Company, 1988.

[Burt 89] P.Burt and E. Adelson, "The Laplacien pyramid as a compact image code", IEEE Trans. Comm., 31, pp. 482-540, (1989).

[Chandy 88] K. M. Chandy, J. Misra, *Parallel program design*, Addison-Wesley Publishing Company, 1988.

[Daubechies 92] I. Daubechies, *Ten lectures on wavelets,* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[Dutilleux 89] P. Dutilleux, "An implementation of the ' algorithme a trous' to compute the wavelet transform". Wavelets, time frequency methods and phase space, Springler-Verlag, 1989, pp. 298-304.

[Hoare 85] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall International LTD, UK, (1985)

[INMOS 90] *Transputer development system*. INMOS Ltd., Prentice Hall, 1990.

[Gabel 73] R.A Gabel, R. A. Roberts, *Signals and Linear Systems,* John Wiley & Sons, Inc., (1973).

[Gabor 46] D. Gabor, *Theory of communication*, J. IEEE, 93(1946), pp. 429-457.

[Grossman 84] A. Grossmann and J. Morlet, "Decomposition of Hardy functions into square integrable wavelets of constant shape". SIAM J. Math., 15 (1984), pp. 723-736.

[Knapp 92] E. Knapp, "Derivation of concurrent programs : 2 examples". Science of computer science programming, 19 (1992) pp. 1-23.

[Kung 79] H.T. Kung, C.E Leiserson, "Systolic arrays for VLSI", Sparse Matrix Proc., DUFF, I.S., and STEWARD, G.W. (eds.), Soc. Ind. Appl. Math, 1979, pp. 256-282.

[Lemarie88] P.-G. Lemarie, "Ondelettes a localisation exponentielle", Journ. de. Math. Pures et Applications, tome 67, (1988), pp. 227-236.

[Mallat 89] S. G. Mallat "Theory for multiresolution signal decomposition: the wavelet representation", IEEE transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, July 1987, pp. 674-693.

[May 87] D. May, R. Shepard, "Communicating Process Computers", Inmos Technical Note22, Inmos Limited, Bristol, 1987.

[Meade 91] M.L. Meade and C.R.Dikon, *Signals and Systems : models and behavior,* (2nd edition), 1991.

[Meyer 87] Y. Meyer, S. Jaffard , O. Rioul, "Analyse par ondelettes", Pour la science, (septembre 1987), pp. 28-38.

[Morlet 83] J. Morlet "Sampling theory and propagation". NATO ASI, Vol. F1, Issues on acoustic signal/image Processing and Recognition, Ed. C.H. Chen. Springer-Verlog, (1983).

[Perrier 89] V. Perrier, C. Basdevant, "La décomposition en ondelettes périodiques, un outil pour l'analyse de champs inhomogènes, théorie et algorithmes", Rech. Aerosp., nos. 1989-3, mai-juin, pp. 53-67.

[Shahian 93] B. Shahian, M. Hassul, *Control System Design Using Matlab,* Prentice Hall, Englewoods Cliffs, New Jersey 07632, (1993).

[Thiele 92] L. Thiele, "Parallel implementation of cellular systems for numerical modeling", Journal of numerical modeling, vol. 5, (1992), pp. 203-218.

[Ville 48] J. Ville, *Théorie et applications de la notion de signal analytique*, CεT, Laboratoire de Télécommunications de la Société Alsacienne de Construction Mécanique, 2 me A. No 1, (1948)

[Wexler 89] J. Wexler, "Developing Transputer applications". OUG-11 . Proceedings of the 11th Occam User Group Technical Meeting. September 1989 - Edinbourgh, Scotland, pp. 25-26.

```
{{{ librairies
#USE strmhdr
#USE userhdr
#USE krnlhdr
#USE sphdr
#USE userio
#SE uservals
#USE streamio
#USE splib
#USE solib
#USE sklib
#USE ssinterf
#USE spinterf
}}}

{{{ declarations
INT n2,m2,n3,m3:
INT i1,j1:
INT i,j:
[300] REAL32 XVEC:
[300][6] REAL32 GWAVE:
VAL INT n IS 5:
VAL INT m IS 299:
VAL INT n1 IS 6:
VAL INT m1 IS 300:
[m1][n1] REAL32 answer:
[m1+1][n1] CHAN OF REAL32 n.s:
[n1+1][m1] CHAN OF REAL32 w.e:
}}}
```

```
{{{ procedure read signal
PROC READ.signal ( CHAN OF SP from.isv,to.isv,CHAN OF KS keyboard,
          CHAN OF SS screen,[]REAL32 sig)
 SEQ
  INT i:
  BYTE bres:
  INT kchar:
  REAL32 elmt:
  SEQ
   so.write.nl(from.isv,to.isv)
   CHAN OF KS filekeys:
   CHAN OF KS keyboard IS filekeys:
   CHAN OF SS echo:
   PAR
    so.keystream.from.file ( from.isv,to.isv,keyboard,"c:\matlab\bin\sr.dat",bres)
    ss.scrstream.sink(echo)
    SEQ
     i:=0
     kchar:=0
     elmt:=1.0(REAL32)
     WHILE ( kchar <> ft.terminated )
      SEQ
       ks.read.echo.char ( keyboard,screen,kchar )
       IF
        kchar < 0
         SKIP
        kchar = (INT '#' )
         INT hexx RETYPES elmt :
         ks.read.echo.hex.int (keyboard,echo,hexx,kchar)
        TRUE
         ks.read.echo.real32 ( keyboard,echo,elmt,kchar )
       IF
        kchar = ft.terminated
```

```
                SKIP
               TRUE
                SEQ
                 IF
                  kchar = ft.number.error
                   ss.write.char (screen,'*#07')
                  TRUE
                   SEQ
                    sig[i]:=elmt
                    i:=i+1
          ss.write.nl(echo)
  }}}

         {{{ consume rest of file
         IF
           (kchar >= 0) OR (kchar = ft.number.error)
             ks.keystream.sink(keyboard)
           TRUE
             SKIP
         ss.write.endstream(echo)
         }}}

     {{{ tabulate signal
      SEQ
       so.write.string.nl(from.isv,to.isv,"this is the signal")
       so.write.nl(from.isv,to.isv)
       SEQ j=0 FOR i
        SEQ
         so.write.real32(from.isv,to.isv,sig[j],2,4)
         --so.write.nl(from.isv,to.isv)
     }}}
:
}}}
```

```
{{{ procedure read wave
{{{ procedure body
PROC READ.wave ( CHAN OF SP from.isv,to.isv,CHAN OF KS keyboard,
         CHAN OF SS screen,[][]REAL32 WAVE,INT n,m)
 SEQ
  BYTE bres:
  INT kchar,i3,j3:
  REAL32 elmt:
  SEQ
   so.write.nl(from.isv,to.isv)
   CHAN OF KS filekeys:
   CHAN OF KS keyboard IS filekeys:
   CHAN OF SS echo:
   PAR
    so.keystream.from.file(from.isv,to.isv,keyboard,
                "c:\matlab\bin\g1.dat",bres)
    ss.scrstream.sink(echo)
    SEQ
     i3:=0
     j3:=0
     kchar:=0
     elmt:=1.0(REAL32)
     WHILE (kchar<>ft.terminated)
      SEQ
       ks.read.echo.char(keyboard,echo,kchar)
       IF
        kchar<0
         SKIP
        kchar = (INT'#')
         INT hexx RETYPES elmt:
         ks.read.echo.hex.int(keyboard,echo,hexx,kchar)
        TRUE
         ks.read.echo.real32(keyboard,echo,elmt,kchar)
        IF
         kchar=ft.terminated
          SKIP
         TRUE
          SEQ
           IF
            kchar=ft.number.error
             ss.write.char(screen,'*#07')
            TRUE
             SEQ
              WAVE[i3][j3]:=elmt
              j3:=j3+1
              IF
               j3<=n
                SKIP
               (j3>n) AND (i3>m)
                STOP
               (j3>n) AND(i3<=m)
                SEQ
                 i3:=i3+1
                 j3:=0
      ss.write.nl(echo)
}}}

      {{{ consume rest of file
      IF
       (kchar >= 0) OR (kchar = ft.number.error)
        ks.keystream.sink(keyboard)
       TRUE
        SKIP
      ss.write.endstream(echo)
      }}}
```

```
    {{{ tabulate wave
    SEQ
     so.write.string.nl(from.isv,to.isv,"these are the wavelets")
     so.write.nl(from.isv,to.isv)
     SEQ k=0 FOR 300
      SEQ
       so.write.nl(from.isv,to.isv)
       SEQ l=0 FOR 6
        so.write.real32(from.isv,to.isv,WAVE[k][l],2,6)
       so.write.nl(from.isv,to.isv)
    }}}
:
}}}

{{{ procedure writecoefficients
PROC write.coeff(CHAN OF SP from.isv,to.isv,CHAN OF KS keyboard,
        CHAN OF SS screen, [][]REAL32 coeff,INT n,m)
  SEQ
   BOOL going:
   SEQ
    CHAN OF ANY fromprog,tofile:
    INT foldnum:
    SEQ
     PAR
      SEQ
       going:=TRUE
       i:=0
       j:=0
       WHILE going
        SEQ
         ss.write.real32(fromprog,coeff[i][j],4,4)
         j:=j+1
         IF
```

```
            j<=n
              SKIP
            j>n
              SEQ
                i:=i+1
                j:=0
                IF
                  i<=m
                    SKIP
                  i>m
                    going:=FALSE
        ss.write.endstream(fromprog)
      SEQ
        ss.scrstream.fan.out(fromprog,tofile,screen)
        ss.write.endstream(tofile)
      BYTE bres:
      SEQ
        so.scrstream.to.file(from.isv,to.isv,tofile,
                       "c:\matlab\bin\coeff.dat",bres)
        IF
          ((INT bres)=0)
            SKIP
          TRUE
            STOP
:
}}}

{{{ procedure multiply
PROC mult(VAL REAL32 aij, CHAN OF REAL32 n,s,w,e)
  REAL32 result,bj:
  SEQ
    PAR
      n ? result
```

```
   w ? bj
  result := result + (aij * bj)
  PAR
   s ! result
   e ! bj
:
}}}

{{{ procedure sink vertical
PROC sinkV ( CHAN OF REAL32 s,REAL32 R)
 REAL32 v :
 SEQ
  s ? v
  R:=v
:
}}}

{{{ procedure sink horizontal
PROC sinkH ( CHAN OF REAL32 e )
 REAL32 h :
 SEQ
  e ? h
:
}}}

{{{ main program
SEQ
 n2:=6
 m2:=300
 n3:=5
 m3:=299
 SEQ
  READ.signal(from.isv,to.isv,keyboard,screen ,XVEC)
```

```
so.write.nl(from.isv,to.isv)
READ.wave  (from.isv,to.isv,keyboard,screen,GWAVE,n3,m3)
SEQ
 {{{  initialize matrix
 SEQ
  PAR i=0 FOR m1
   PAR j=0 FOR n1
    answer[i][j]:=0.0(REAL32)
 }}}

 {{{  coumpute matrix coefficients
 [m1+1][n1] CHAN OF REAL32 n.s:
 [n1+1][m1] CHAN OF REAL32 w.e:
 SEQ
  SEQ k=0 FOR m1
   PAR
    PAR j=0 FOR n1
     n.s[0][j] ! 0.0(REAL32)
    PAR j=0 FOR m1
     w.e[0][j] ! XVEC[j]
    PAR l=0 FOR m1
     PAR v=0 FOR n1
      IF
       k > l
        mult(GWAVE[k-l][v],n.s[l][v],n.s[l+1][v],
                     w.e[v][l],w.e[v+1][l])
       IF
        k <= l
         mult(GWAVE[l-k][v],n.s[l][v],n.s[l+1][v],
                     w.e[v][l],w.e[v+1][l])
    SEQ j=0 FOR n2
     SEQ
      sinkV(n.s[m2][j],answer[k][j])
```

```
            ss.write.nl(screen)
          SEQ j=0 FOR m2
            sinkH(w.e[n2][j])
        write.coeff(from.isv,to.isv,keyboard,screen,answer,n4,n4)
      }}}

      {{{ tabulate answer
      SEQ
        so.write.string.nl(from.isv,to.isv,"these are the wavelets")
        so.write.nl(from.isv,to.isv)
        SEQ k=0 FOR m2
          SEQ
            so.write.nl(from.isv,to.isv)
            so.write.nl(from.isv,to.isv)
            SEQ l=0 FOR n2
              so.write.real32(from.isv,to.isv,answer[k][l],4,4)
        so.write.nl(from.isv,to.isv)
      }}}
    so.write.nl(from.isv,to.isv)
    write.coeff(from.isv,to.isv,keyboard,screen,answer,n3,m3)
    so.write.nl(from.isv,to.isv)
  }}}

   {{{ go to tds
   so.write.nl(from.isv,to.isv)
   so.write.nl(from.isv,to.isv)
   so.write.string(from.isv,to.isv,"TYPE ANY TO GO TO TDS")
   BYTE any,bres:
   so.getkey(from.isv,to.isv,any,bres)
   so.write.nl(from.isv,to.isv)
   }}}

}}}
```

ANNEXE

Liste et composition du jury en vue de la soutenance de mémoire de magister en Ingénierie des Systèmes Elèctroniques par Mme YAHIAOUI Aicha.

PRESIDENT/ - Dr. A. FARAH, Maître de Conférence à l'E.N.P.

RAPPORTEUR /- Dr BENMOHAMMED -Mahieddine.
Chargé de Cour à l'I.N.E.L.E.C.
P.H.D en Informatique.

EXAMINATEURS/ - Dr R. AKSSAS. Maître de Conférence à l'E.N.P.
- Dr K. BADARI. Maître de Conférence à l'I.N.H.
- Dr M. DJEDDI. Maître de Conférence à l'I.N.E.L.E.C.

INVITE /- Mr A. GOUGAM. Chargé de Cours à l'I.N.E.L.E.C.