The Development of a Parallel Implementation of Non-Contact Surface Measurement

By

Andrew James Sanyal

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the Degree of Doctor of Philosophy

December 1998

School of Engineering
Liverpool John Moores University

PAGE/PAGES
EXCLUDED
UNDER
INSTRUCTION
FROM
UNIVERSITY

To My Grandad,

For Understanding.

Abstract

The Development of a Parallel Implementation of Non-Contact Surface Measurement

Andrew James Sanyal **Ph.D. Thesis**

Implementing parallel-processing techniques within a practical optical metrology environment raises a number of important questions. The first key issue focuses on how optical metrology techniques are influenced and impinged upon by a parallel-processing solution. Following on from this is the question of the practical costs and benefits arising from using parallel processing. With the recent development of parallel-processing software environments the next question is how well suited are these environments to optical metrology problems.

The practical application that this work addresses is the development of fringe analysis techniques to improve patient set-up and monitoring within a radiotherapy environment. Such an application places a number of demands upon the computational system namely speed, reliability and flexibility.

This thesis traces the relevant background developments within both Fringe Analysis and Parallel Processing. Additionally the problems involved in Conformal Radiotherapy that make this research necessary are discussed.

Two interferometric phase extraction based techniques have been implemented within two parallel processing software environments. Fourier Fringe Analysis is a computationally intensive paradigm that consists of a number of algorithms while Phase Stepping Profilometry is a less computationally intensive algorithm that requires comparatively more data. Implementing both of these techniques addresses the identified goals of this research. As well as this both techniques require the implementation of an unwrapping stage, and parallel strategies to perform this raise a number of additional points.

In the final chapter of this thesis the key lessons learned in this work are summarised. As well as this the final chapter contains discussions which attempt to answer the questions of the suitability and practicality of using parallel processing to solve optical metrology problems.

Acknowledgements

The author would like to acknowledge the debt of gratitude that he owes to a number of people.

Firstly the author would like to thank his supervisory team. Professor David Burton, for his encouragement, advice and guidance, the lack of which would have prevented this work from ever being completed or started. Professor Michael Lalor, for teaching the author all the optics that he needed to know and listening when it was needed. Dr Jerry Pearson, for his short stories which have helped put the author's concerns into perspective.

The author would also like to thank his unofficial supervisory team. Francis Lilley for all his help and mentoring. Brewster LaMacchia for having the patience to understand the author's complaints and teaching him a little bit about how C40s work.

The author would also like to thank all the members of the Coherent and Electro-Optics Research group at the JMU. It is the author's belief that the team work and friendship of this group has contributed greatly to the successful completion of this work.

The author would also like to thank the European Regional Development Fund for his grant funding via the Centre for Precision Measurement and Industrial Inspection. All the research in this thesis has been performed on equipment provided via the European IVth Framework project called INFOCUS funded under the BIOMED II initiative.

Contents

Contents

Contents

List of Figures

List of Tables

Abbreviations and Symbols Used

1D	One Dimension
2D	Two Dimensions
C	Programming Language
CASE	Computer Aided Software Engineering
C40	Texas Instruments TMS320C40 DSP Processor Chip
CPU	Co-processor Unit
DDU	Discrete Data Unit
DLL	Dynamic Link Library
DMA	Direct Memory Addressing
DSP	Digital Signal Processing
EDRAM	Enhanced Dynamic RAM
FFA	Fourier Fringe Analysis
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
IDL	Interactive Data Language
I/O	Input and Output
MFLOPS	Millions of Floating Points Operations Per Second
MIMD	Multiple Instruction – Multiple Data
MIPS	Millions of Instructions Per Second
MISD	Multiple Instruction – Single Data
MLC	Multi-leave Collimator
MVI	Megavoltage Images
ms	Milliseconds
MSVC	Microsoft Visual C++
PC	Personal Computer
PSP	Phase Stepping Profilometry
PZT	Peizo Transducer
RAM	Random Access Memory
SIMD	Single Instruction – Multiple Data

SISD	Single Instruction – Single Data
TDM435	Transtech Framestore Module
TI	Texas Instruments
TIM-40	TI C40 chip
TIM411	Transtech TIM-40 Motherboard
VRAM	Video RAM

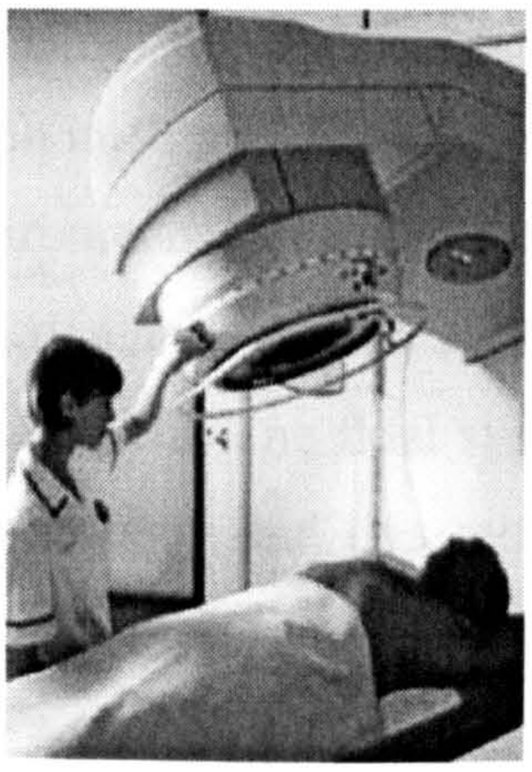
Chapter 1

Introduction

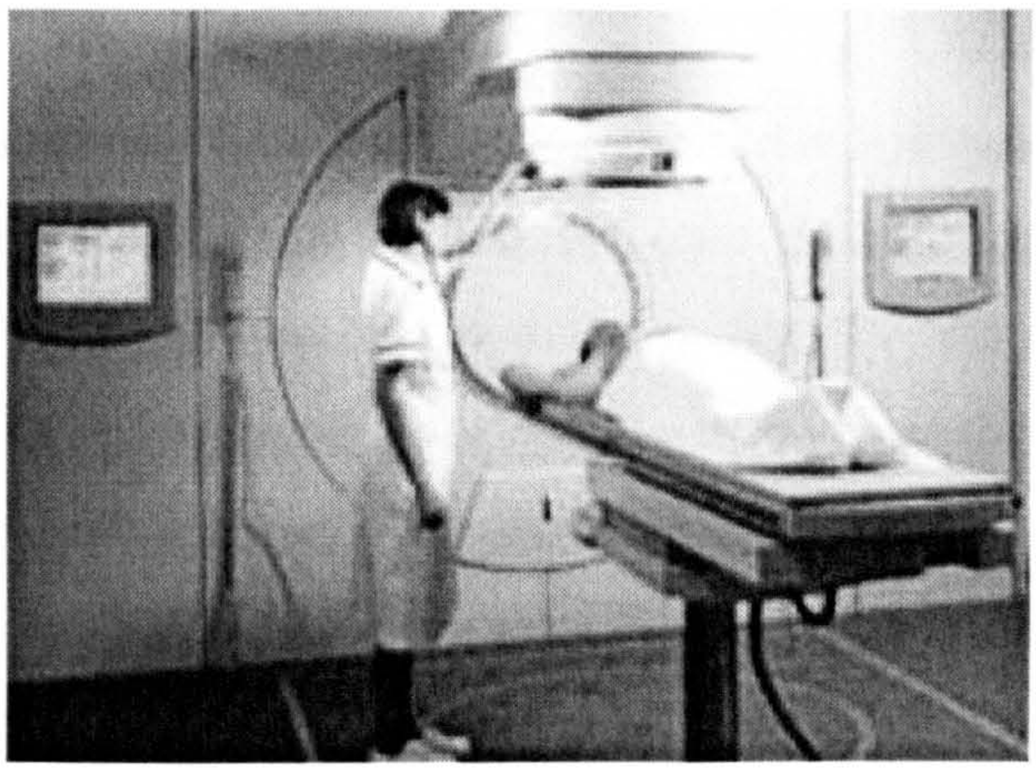
1.1Introduction

Breathing is one of the most obvious and vital signs of life. Yet this very action has proven to be a hurdle in increasing the successful treatment of cancer by Radiotherapy. Non-contact measurement techniques allow information about a surface to be obtained. With this information the exact shape and location of a surface can be calculated. A sequence of such measurements allows changes in the surface's shape and location to be tracked over a given time period. By applying these techniques in a radiotherapy environment the act of breathing should no longer be a problem in the modelling of received radiation dose in Radiotherapy.

Radiotherapy is based upon the delivery of a lethal dose of ionising radiation to a site of cancer cells. Conformal radiotherapy is the customisation of the dose of radiation received by each patient, involving both the location of the dosage and the time over which delivery occurs. Treatment is not received in one dose but in a number of fractions over a given time period. In fig.1.1(a) and fig.1.1 (b) a patient is positioned below the Linear Accelerator used to deliver the required dosage of radiation.



(a)



(b)

Fig 1.1 Conformal Radiotherapy Treatment Machine. (Pictures courtesy of EOS)

People are dynamic systems, whose external body surface shape can change through such processes as breathing, sighing or, over some time, weight loss. Monitoring the changes in the dynamic external surface after each treatment and over the entire session of treatments would greatly aid the clinician. Using Conformal Radiotherapy techniques increases the need for accurate models of where radiation has been received by a patient during a treatment session. Dosimetric models are based around tolerances of ±1mm. Therefore any surface measuring system must be capable of measuring patients position to within this level of accuracy.

The automated analysis of any surface data must be performed within paradigms that require large amounts of processing power. Where single processor architectures have been unable to match the specifications of a system, parallel processing techniques have been investigated. These techniques allow high-speed, high-performance results that to date exceed those of the "traditional" uniprocessor system.

However, parallel processing software techniques have lagged far behind those of the more conventional processor environment. Perhaps the main reason for this is that the attention of developers in this field has been focused solely upon speed. Optimising code to run as quickly as possible has been the one goal along this path. Inevitably the end users of such systems have needed to be as educated in parallel processing as the developers.

Applying parallel processing to non-contact measurement within the clinical environment of Radiotherapy changes the focus of the software developed. Speed although still a critical factor within such a domain is not the only requirement. The final system must be able to measure a number of areas of the human body and this requires different approaches for different locations. A measurement must always be achieved by the system; therefore robustness is also a goal.

The flexibility, robustness and speed of the final system must all fit into an easy to use Graphical Interface. As well as this the data must be delivered in such a way that the final user does not become swamped in a vast quantity of information.

1.2 Aims of this Research

This research has been set up within the umbrella of the INFOCUS project. INFOCUS is funded by the European Union research grant of BIOMED II. The primary goals of INFOCUS are to "develop and improve diagnostic tools, methods and standards" within the Conformal Radiotherapy field[1]. To achieve this broad ranging target there are a number of European partners involved, each with responsibility for different aspects of achieving the INFOCUS goal. These partners are:

- The Christie Hospital, Manchester, UK.
- Otto Von Guricke University, Magdeberg, Germany.
- Claude Bernard University, Lyon, France.
- SINTEF, Trondheim, Norway.
- Elekta Oncology Systems, UK.
- John Moores University, Liverpool, UK.

Within this project the development of a non-contact optical sensor for the precision measurement of the external surface position of the patient is the specific area that this research falls inside of[2]. The author's contribution to this project has been to develop a parallel processing system to perform fringe analysis.

The specific goals for this research can be summarised as:

i. To investigate the effects of a practical parallel processing network topology on "classical" linear image analysis paradigms.

To show how the practical system can alter the movement of data and paradigm structure.

ii. To design a robust, flexible and fast software system to use the data obtained from the non-contact optical sensor. This means investigating a number of different techniques to achieve near real-time measurements.

iii. To investigate the applicability of parallel processing CASE tools to system design in optical metrology.

iv. To assess whether performing parallel processing of image data is of any **true** practical value at this time.

It is perhaps this last goal that is of most long-term interest. Perhaps this goal should be re-stated as a series of questions. Firstly, is parallel processing driven by the needs of the application or by the wish to perform parallel processing? Is there an argument for using simpler processing systems, which have slightly lower processing power but benefit from other advantages? All the work within this investigation must be tested in the light of these questions.

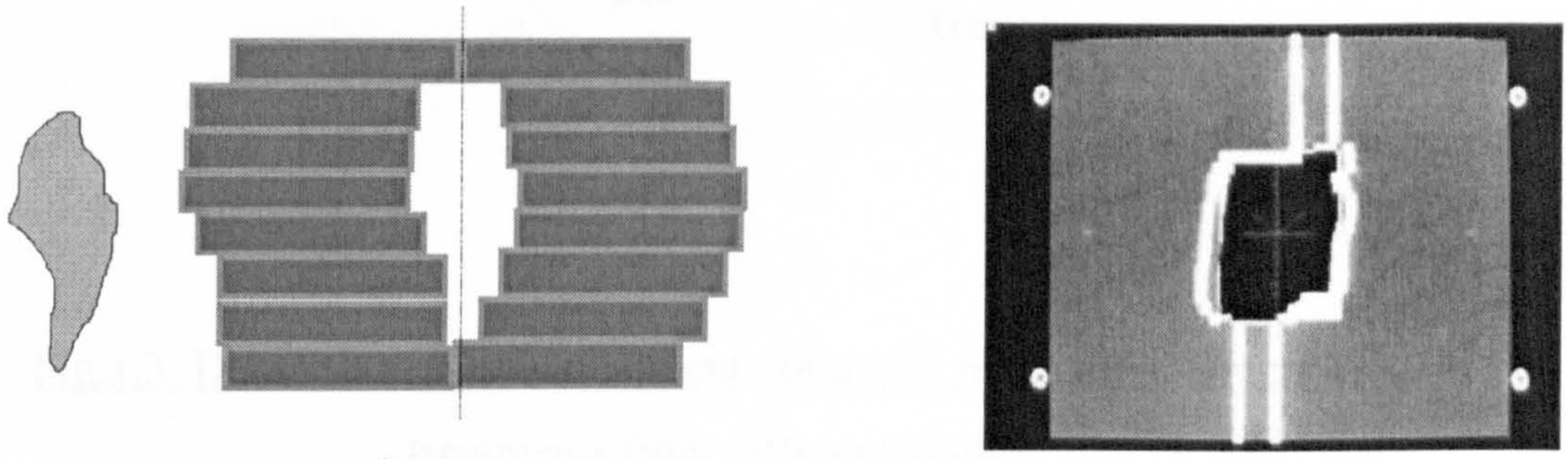
1.3 Scope of this Research

There is one main boundary placed upon the scope of this work

i. This work does not comment on the nature, cause or treatment of cancer. It is not within the scope of this research to comment on how the data obtained by this system is used by clinicians.

1.4 A Basic Introduction to Radiotherapy and its Current Limitations

As mentioned previously the main aim of radiotherapy is to deliver a fatal dose of ionising radiation to an identified cancer site. Conformal radiotherapy is a recent development of this basic idea. This idea implies that both the nature and the geometry of the therapy are customised to each patient's requirements, Fig.1.2. There is a resulting beneficial increase in the accuracy of the targeted radiation. Which in turn leads to increased cure rates and a lower incidence of morbidity in the healthy cells surrounding the cancer.



(a) (b)

Fig.1.2(a) Showing how the Treatment Head can be shaped to match the target volume. (b) Showing an actual planned treatment shape (Picture courtesy of EOS)

The clinician using Conformal Radiotherapy seeks to irradiate the site of the tumour with an individually shaped beam from 3 or 4 different angles, fig.1.3. The basic idea behind this strategy is that only the cancer cells receive a lethal dose while the healthy tissue surrounding the site receives a much lower dosage[3]. Over a number of treatments, or fractions, the healthy cells will have a far greater likelihood of recovering.

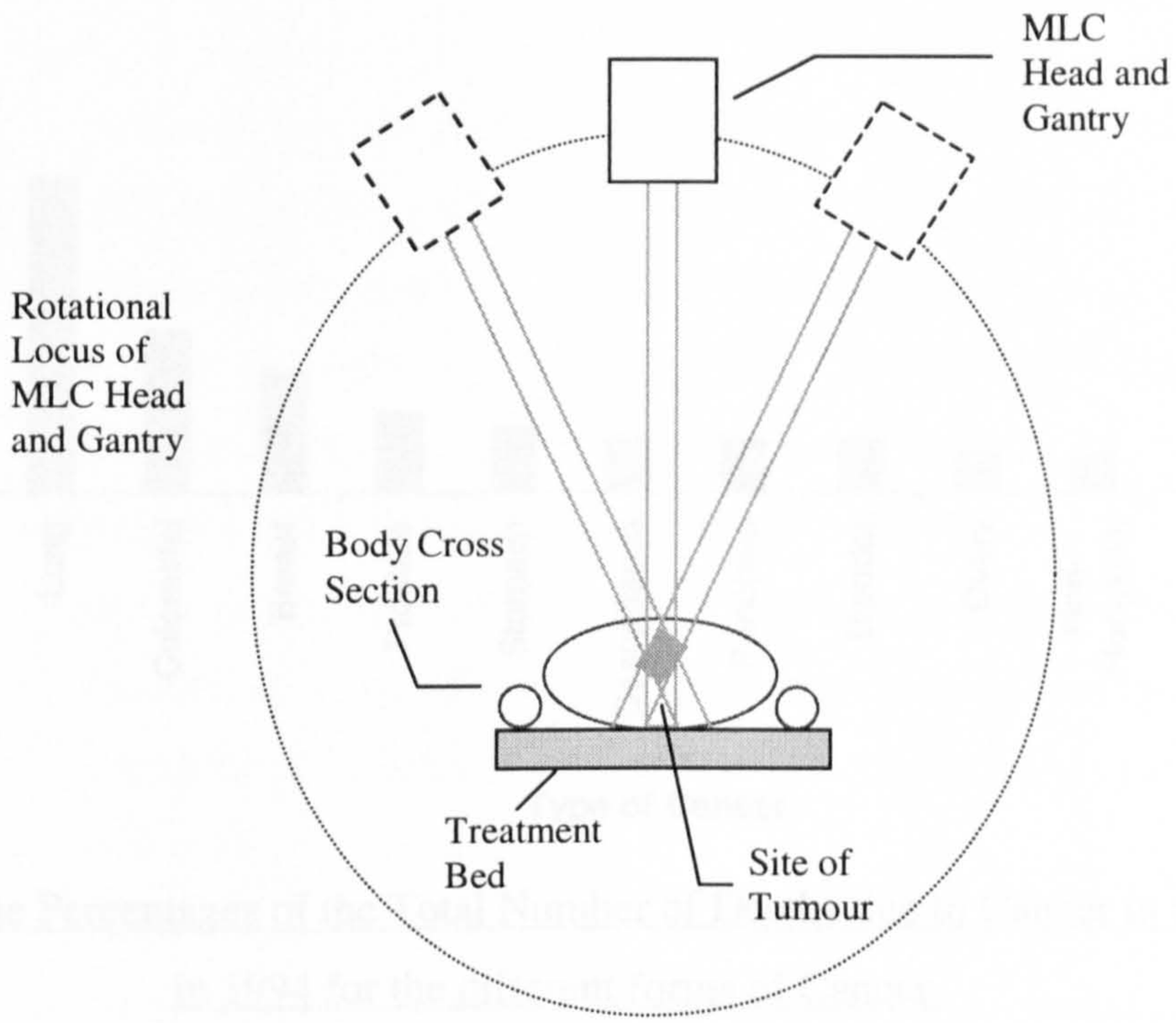


Fig.1.3. Demonstrates how treatment volume is made up of a number of treatments from different angles.

In 1994 there were over 150,000 deaths from cancer in the UK alone, fig.1.4. One method of attempting to reduce this figure is to treat people suffering certain types of cancer with Conformal radiotherapy. At the Christie Hospital Manchester conformal treatment has grown steadily in it's importance from less than 1% of annual treatments in 1991 to over 8% of all treatments performed in 1996[4]. Added to this the types of cancer that conformal therapy has been very successful in helping treat are very difficult to effectively manage with more conventional methods, fig.1.5.

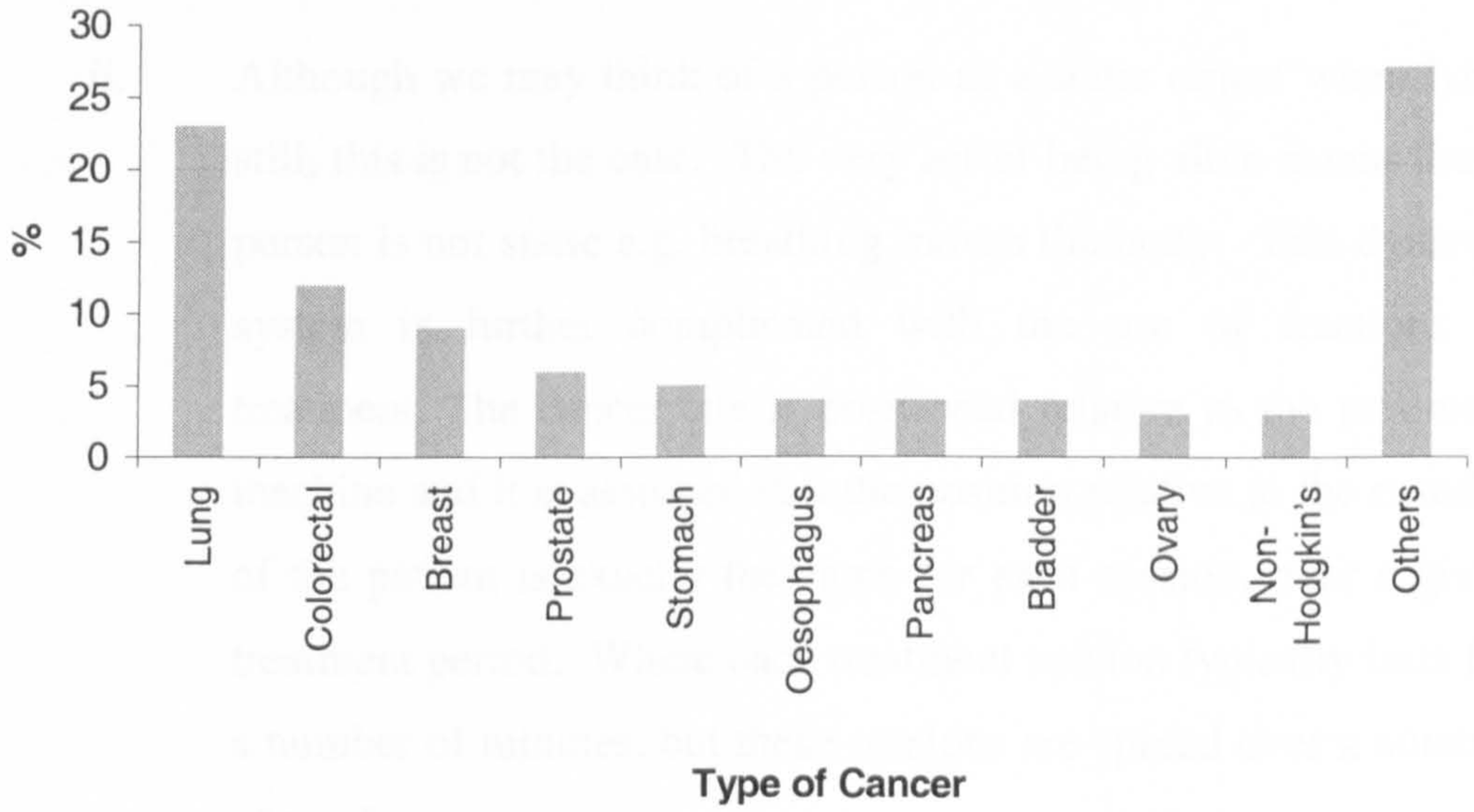


Fig.1.4 The Percentages of the Total Number of Deaths due to Cancer in the UK in 1994 for the different forms of Cancer.

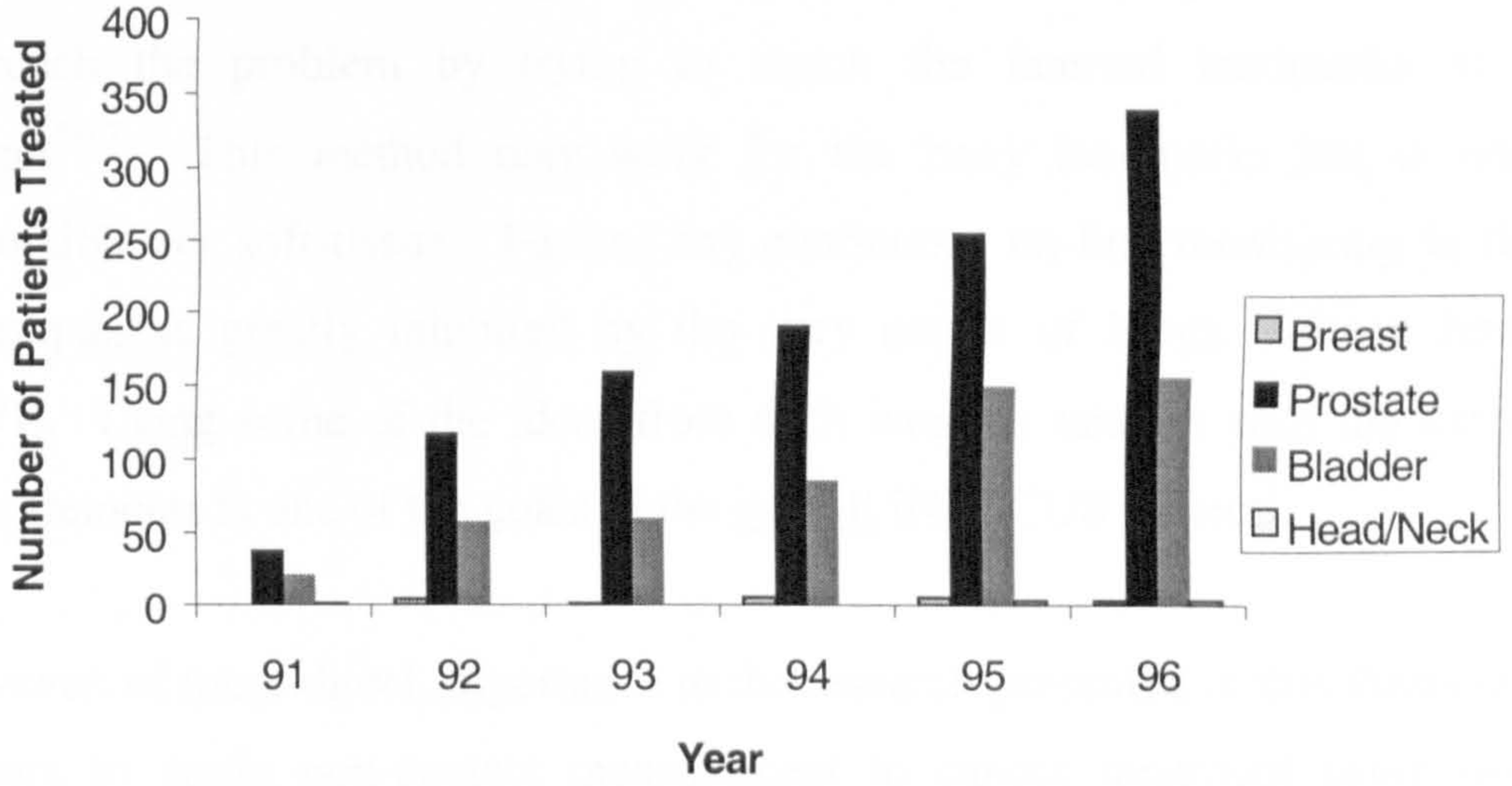


Fig.1.5 Shows the increasing use of Conformal Radiotherapy at the Christie Hospital Manchester.

However, there are significant problems with this system. In an ideal world the clinician would like to be able to irradiate **only** the cancer. This is not possible for two main reasons.

i. The cancer site may be located within a relatively mobile organ, such as the prostate. As this is often the case the clinician must allow for some error boundaries within the treatment.

ii. Although we may think of a person as a static object when lying still, this is not the case. The very act of being alive means that a person is not static e.g. breathing moves the body. This dynamic system is further complicated with the use of fractions of treatment. The cancer site is positioned relative to the treatment machine and it is assumed that the position relative to the exterior of the patient is exactly the same for each session, over a given treatment period. Where each treatment session typically lasts for a number of minutes, but these sessions are spread over a number of weeks.

It is within this second category of problem that this research is focused.

There have been several attempts to solve this problem in the past, most of which approach the problem by trying to match the internal landmarks of the patient[5,6,7]. This method may work for the bony landmarks but is nearly impossible for soft-tissue. Further any continuous on-line monitoring in these techniques is greatly inhibited by the very nature of Mega Voltage Images (MVI). Using some of the ideas from such work in tandem with the external measurements is one of the goals of the overall INFOCUS project[1].

However, of more direct importance to the research presented in this thesis is the attempt to apply non-contact measurement to cancer treatment using Moiré patterns[8]. This system uses two light sources to produce a fringe pattern on the patient's skin surface. A still camera then photographs this illuminated surface.

Wilks[9] approached the problem with an optically much simpler solution. Utilising a single line of laser light to perform single line triangulation. This work was extended by the use of a multiple line system. This system is both simple and effective. However, the final resolution of the system is very poor and only addresses the area of one type of cancer. As well as this both of these external body surface measurement systems were only implemented in planning not during treatment.

1.5 Structure of this Thesis.

This thesis consists of eight chapters.

Chapter 1:

This introductory chapter outlines the application of parallel processing to non-contact measurement. Details of the precise aims and scope of this research are also given. The chapter also contains a brief description of Radiotherapy.

Chapter 2:

This chapter acts as an introduction to the main non-contact techniques used within this research. Attention is focussed on Phase Measuring techniques, namely Fourier Fringe Analysis and to a lesser extent Phase Stepping Profilometry.

Chapter 3:

Within this chapter those aspects of parallel processing that are of specific relevance to this work are discussed. The focus of this chapter is those elements of parallel processing, which relate to image analysis and user interfaces.

Chapter 4:

This chapter briefly discusses the hardware tools used with in this research. The software tools used are also presented, with special attention on the final software environment, Pegasus. As well as this the optical system used in this research is briefly outlined.

Chapter 5:

This chapter describes in some detail the more general aspects of implementing Fringe Analysis Techniques in a Parallel Processing environment. Firstly the effects of the specialist hardware features of the Texas Instrument TMS320C40 (C40) used in this work are detailed. Following on from this the general issues of data problems are discussed. Topics include data division or partitioning and data storage.

Chapter 6:

This chapter follows on from the general issues to discuss issues which arise from the application of the phase extraction techniques within two specific parallel processing software environments. Firstly Fourier Fringe Analysis (FFA) is discussed within both the 3L and Pegasus environments. As FFA is formed from a number of algorithms these sections discuss a number of important issues. The next sections discuss the application of Phase Stepping Profilometry (PSP) in a real-time system and show how this technique is modified by the application rather than the parallel-processing environment. Finally in this chapter the issues that arise from performing parallel unwrapping algorithms are discussed.

Chapter 7:

This chapter evaluates the final system and describes the results obtained using the methods detailed. The work is assessed with regard to the aims identified in the first chapter. Some attempt at a qualitative comparison of the parallel-processing solution and an alternative Windows based system is made.

Chapter 8:

This final chapter draws conclusions from the research described in this thesis and discusses areas for further research. The discussion focuses on the question of whether or not parallel processing is of any real relevance to non-contact measurement at this time.

1.6 References

1 Information Technology in Conformal Radiotherapy, INFOCUS.
EEC Commission BIOMED II, Framework IVth application.
1995.

2. INFOCUS First Year Project Report
EEC Commission, BIOMED II, Contract N^{o}: BMH4-CT95-0567
1996.

3 Porrill J, and Ivins J.
A Semiautomatic Tool for 3D Medical Image Analysis Using Active Contour Models.
Med.Inform. Vo.19, No.1, pp.81-90, **1994.**

4 Private Communications with Dr. Chris Moore of Christie hospital Manchester.

5 Graham M.J et al
A Method to Analyse 2-Dimensional Daily radiotherapy Portal Image From An On-line Fibre-optic Imaging System.
Int.Jn.Radiation Oncology Biol. Phys. Vol. 20, pp. 613-619, **1993.**

6 Sivewright G.J and Elliott P.J.
Interactive Region and Volume Growing for Segmenting Volumes in MR and CT Images
Med. Inform. Vol. 19, No. 1, pp. 71-80, **1994.**

7 Wong J et al.
On-line Image Verification in Radiation Therapy: an Early USA Experience.
Medical Progress through Technology, Vol. 19, pp. 43-54, **1993.**

8 Chu J and Bioch P.
Application of Moiré Patterns for Obtaining Surface Contour Information on Patients receiving Radiotherapy.
SPIE Vol. 283, pp. 2-6, **1981.**

9 Wilks R.J.
An Optical System for Measuring Surface Shape for Radiotherapy Planning.
Brit. Jn. Rad. Vol. 66, pp. 351-359, **1993.**

Chapter 2

Review of Fringe Analysis

2.1 Introduction

The field of non-contact measurement has developed into a broad umbrella of methodologies in recent years[1,2]. There have been a number of applications of these techniques; see for example Chan[3] and Brown[4]. Yet the number of such experiments in real-time is a very limited set[5,6]. It is with the application of fringe analysis to near real-time that this review has been performed.

This chapter does not attempt to review all the developments within the field of non-contact measurement, but rather focuses on those aspects of the subject, which are of importance within this research.

Young was the first to demonstrate the interference of light waves to produce fringe patterns[7]. In this early experiment light from a monochromatic source is projected through two pinholes in a screen. This has the effect of forming an interference pattern, due to the pinholes acting as secondary monochromatic point sources which are mutually coherent.

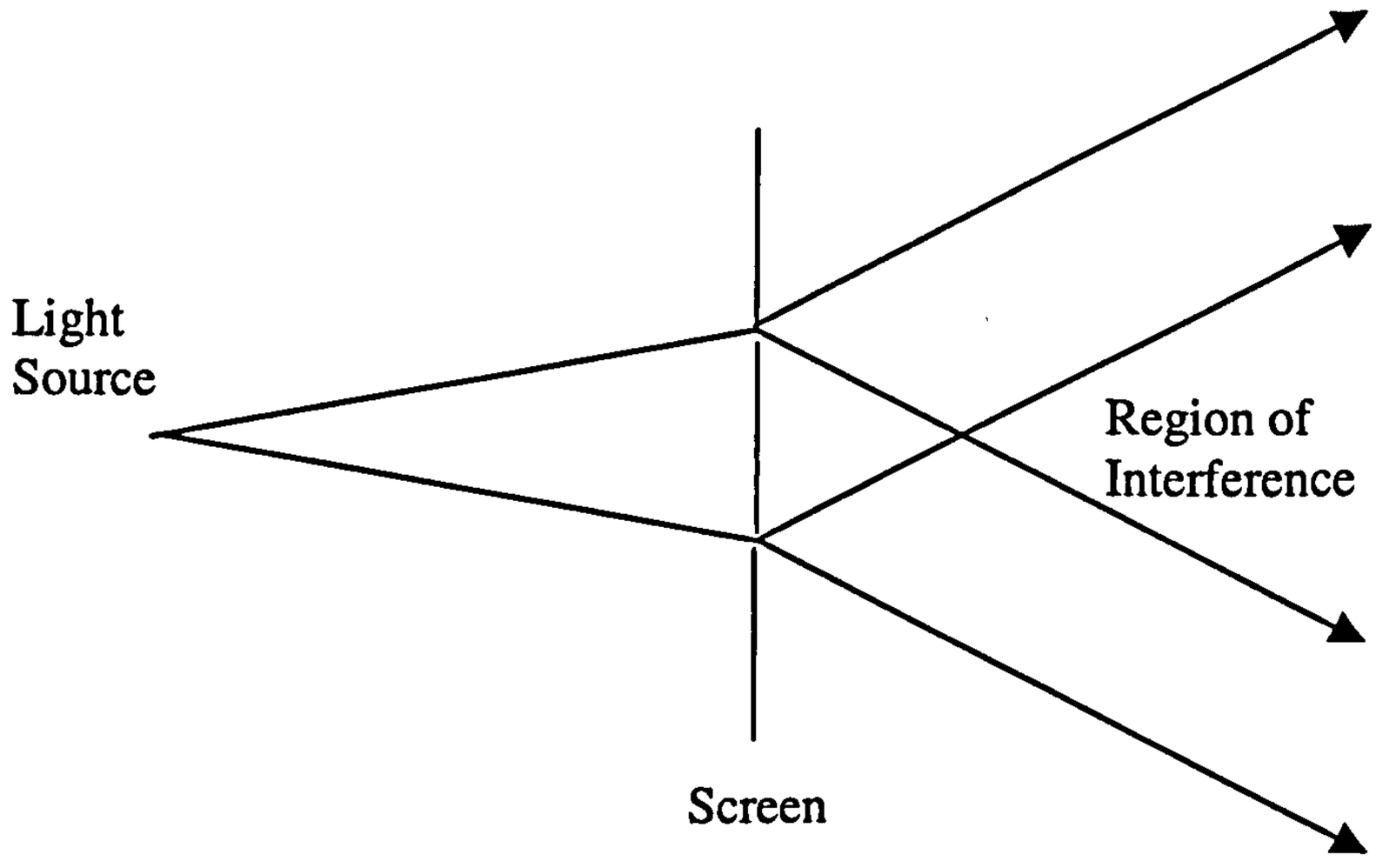


Fig.2.1 Young's Pinhole Experiment

The resulting fringes produced by the experiment in Fig.2.1 are curved in nature. By changing the set-up to use slits rather than pinholes Young produced fringes which are straight and also allowed a greater amount of light to be projected through the screen.

The modern version of this experiment uses an optical set-up based upon laser light and fibre optics[8]. Laser light is projected down a fibre optic path, at the point of the fibre coupler the laser light is split into two optical paths. The light passing along the fibre paths is mutually coherent and when projected from the fibre heads interfere in the same way as Young's slit experiment. Using optic fibres is a very elegant solution to produce Young's fringes and additionally has benefits such as allowing Peizo transducers (PZT) to manipulate the fibre heads and produce altered interference patterns. Additionally the fibre optics act as a spatial filter, which greatly increases the clarity of the fringe pattern produced when compared with the slit experiment.

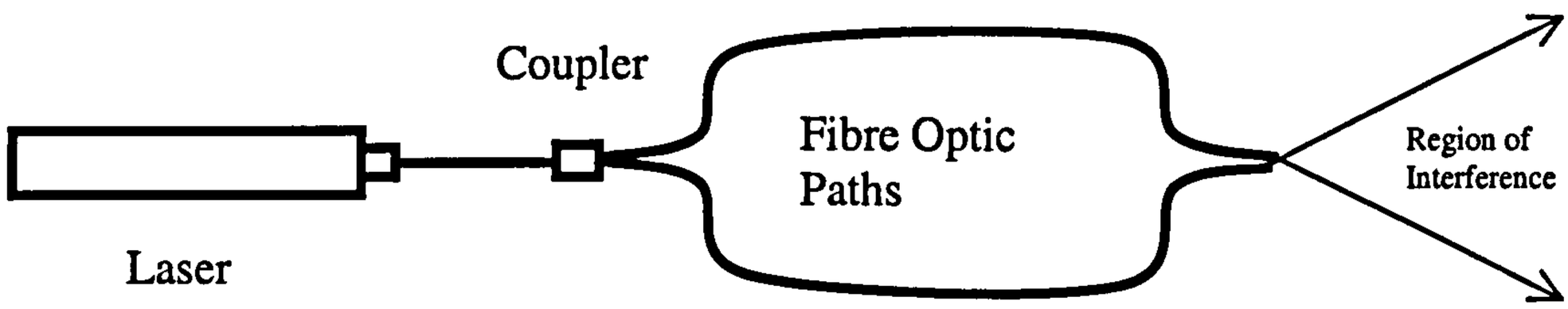


Fig.2.2 Twin Fibre Interferometer

The fringes produced by this system will be orientated at right angles to the line joining the two fibre heads. This means that by using a PZT to switching the position of the fibre heads the orientation of the fringe pattern may be altered. Viewing these fringes from some distance will make them appear straight.

The separation of adjacent bright fringes is approximately given by the equation

$$p_0 = \frac{\lambda a}{d} \tag{2.1}$$

where,

p_0 = Fringe Spacing

λ = Wavelength of laser light used

d = Distance between fibre heads

a = Distance between projection heads and observation plane

This fringe spacing may be altered by changing the distance between the fibre heads, d. Using a PZT one of the fibre heads will be shifted away from or toward the other stationary fibre head.

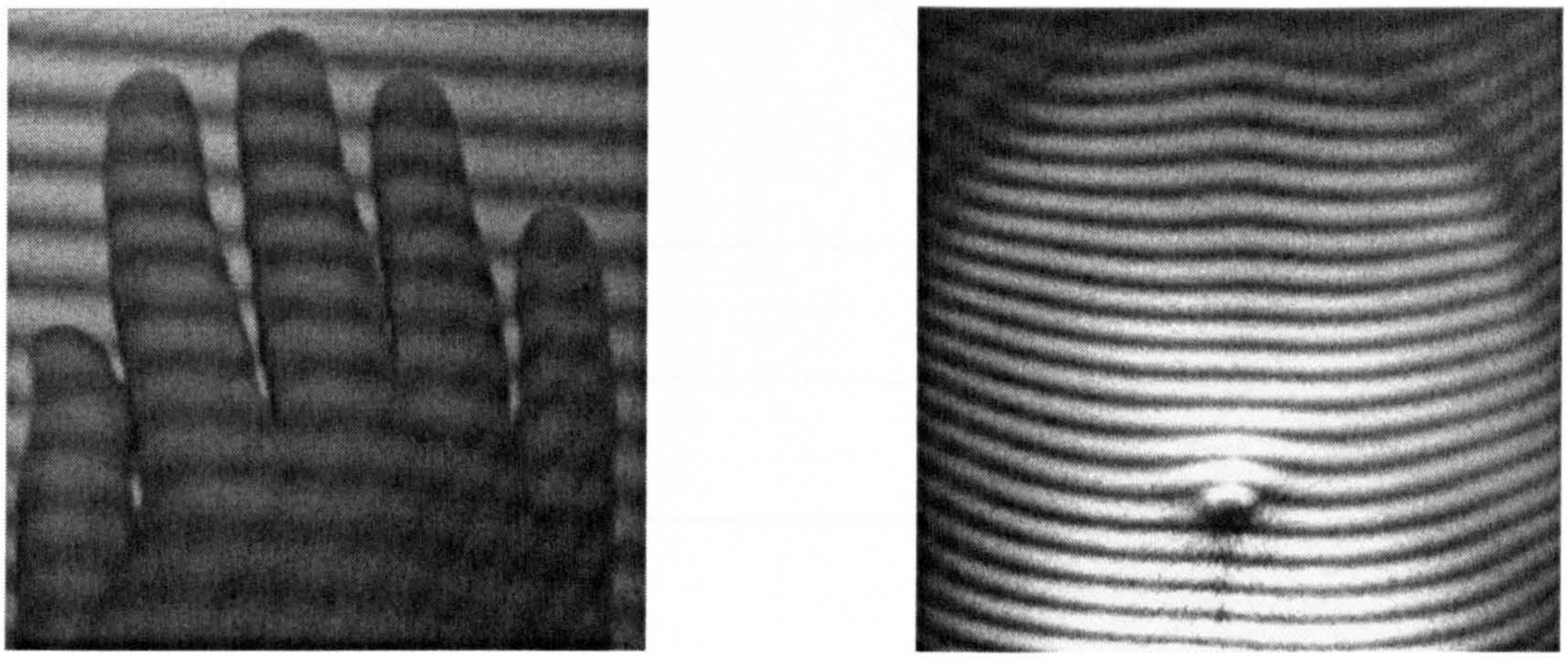


Fig.2.3(a) Fringe Pattern projected onto the surface of a hand (b) Fringe pattern projected on to the author's stomach

The two fringe analysis techniques used in this work, Fourier Fringe Analysis (FFA) and Phase-Stepping Profilometry (PSP),are phase measuring techniques.

The fringe pattern used is made up of a structured light pattern that varies between bright and dark in some regular manner such as a $\cos^2$ pattern, fig.2.3 shows examples of such patterns projected onto human skin. These patterns are projected onto a surface, and the resulting image is recorded. Because the recorded image is off axis from the projection the structure of the pattern will have been altered from the projected pattern by the surface. By the extraction of this alteration, or modulation, of the pattern surface data can be found.

2.2 Relating Phase and Height.[9]

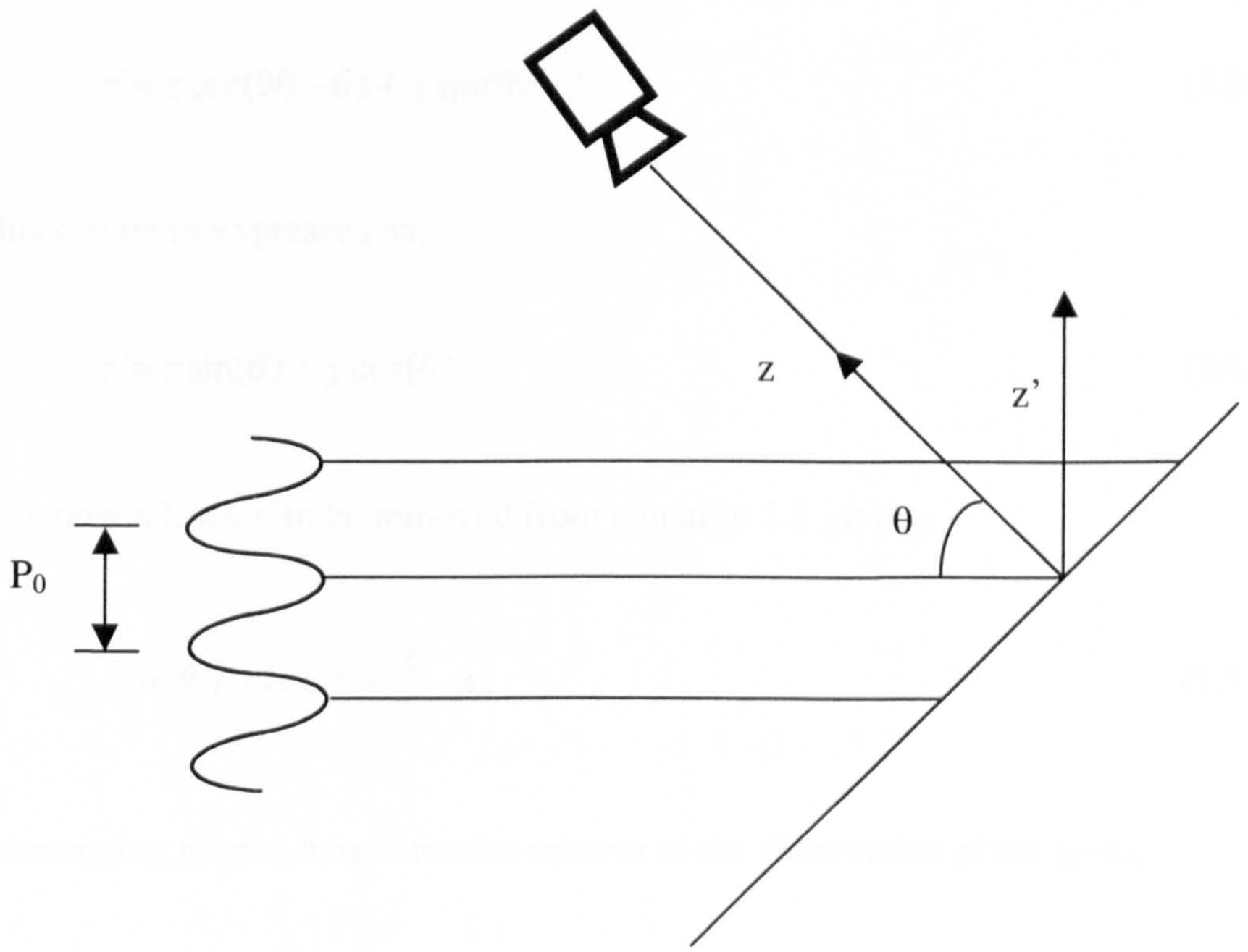


Fig.2.4 Surface illuminated by Fringe Pattern

Consider a surface illuminated with a projected fringe pattern as is shown in fig.2.4. The height within the plane of the fringe pattern is given by equation 2.2.

$$z' = \frac{\phi}{2\pi} \cdot p_0 \tag{2.2}$$

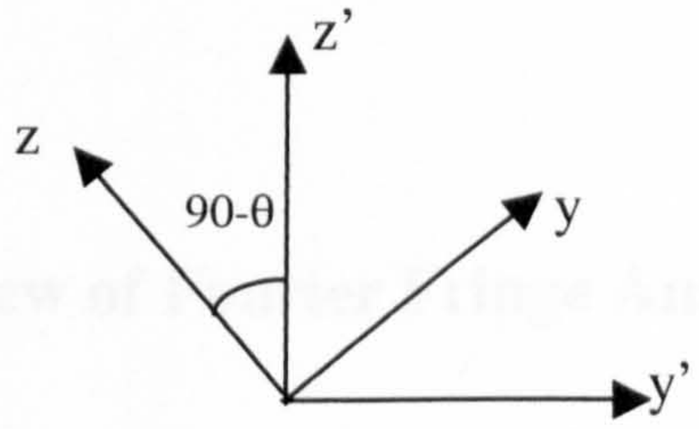


Fig.2.5 Relating projection and observation co-ordinate geometry

However, it is the height in the plane of observation that is required. By taking the co-ordinate sets shown in fig.2.5 and using the basic geometry of the system equation 2.2 can be restated as;

$$z' = z\cos(90-\theta) + y\sin(90-\theta) \tag{2.3}$$

This can be re-expressed as;

$$z' = z\sin(\theta) + y\cos(\theta) \tag{2.4}$$

This now allows z' to be removed from equation 2.2 giving;

$$z\sin\theta + y\cos\theta = \frac{\phi}{2\pi}.p_0 \tag{2.5}$$

Rearranging to give height results relative to the observation plane gives;

$$z = \frac{\phi p_0}{2\pi\sin\theta} - \frac{y}{\tan\theta} \tag{2.6}$$

2.3 Brief Review of Fourier Fringe Analysis (FFA)

In Fringe Analysis the equation of a modulated intensity of a signal at any point is often stated as[10],

$$g(x, y) = a(x, y) + b(x, y)\cos\theta \qquad (2.7)$$

The terms a(x, y) and b(x, y) represent the unwanted irradiance variations, while the term θ contains the desired modulated information. Takeda[10] proposed that we re-state this expression in the following format,

$$g(x, y) = a(x, y) + c(x, y)e^{j2\pi f_o x} + c^*(x, y)e^{-j2\pi f_o x} \qquad (2.8)$$

where,

$$c(x, y) = \frac{1}{2}b(x, y)\cos\phi + j\frac{1}{2}b(x, y)\sin\phi$$

$$c^*(x, y) = \frac{1}{2}b(x, y)\cos\phi - j\frac{1}{2}b(x, y)\sin\phi$$

Then these intensity values g(x, y) taken from a captured and digitally stored image can be transformed by the application of the Fourier Transform technique. Thus if expression (2.8) is Fourier transformed with respect to x,

$$G(f, y) = A(f, y) + C(f - f_o, y) + C^*(f - f_o, y) \qquad (2.9)$$

Where capital letters represent the Fourier Spectra and f is the frequency in the x direction. This resulting spatial domain data is made up of two harmonics or carrier signals and one DC term, fig 2.6.

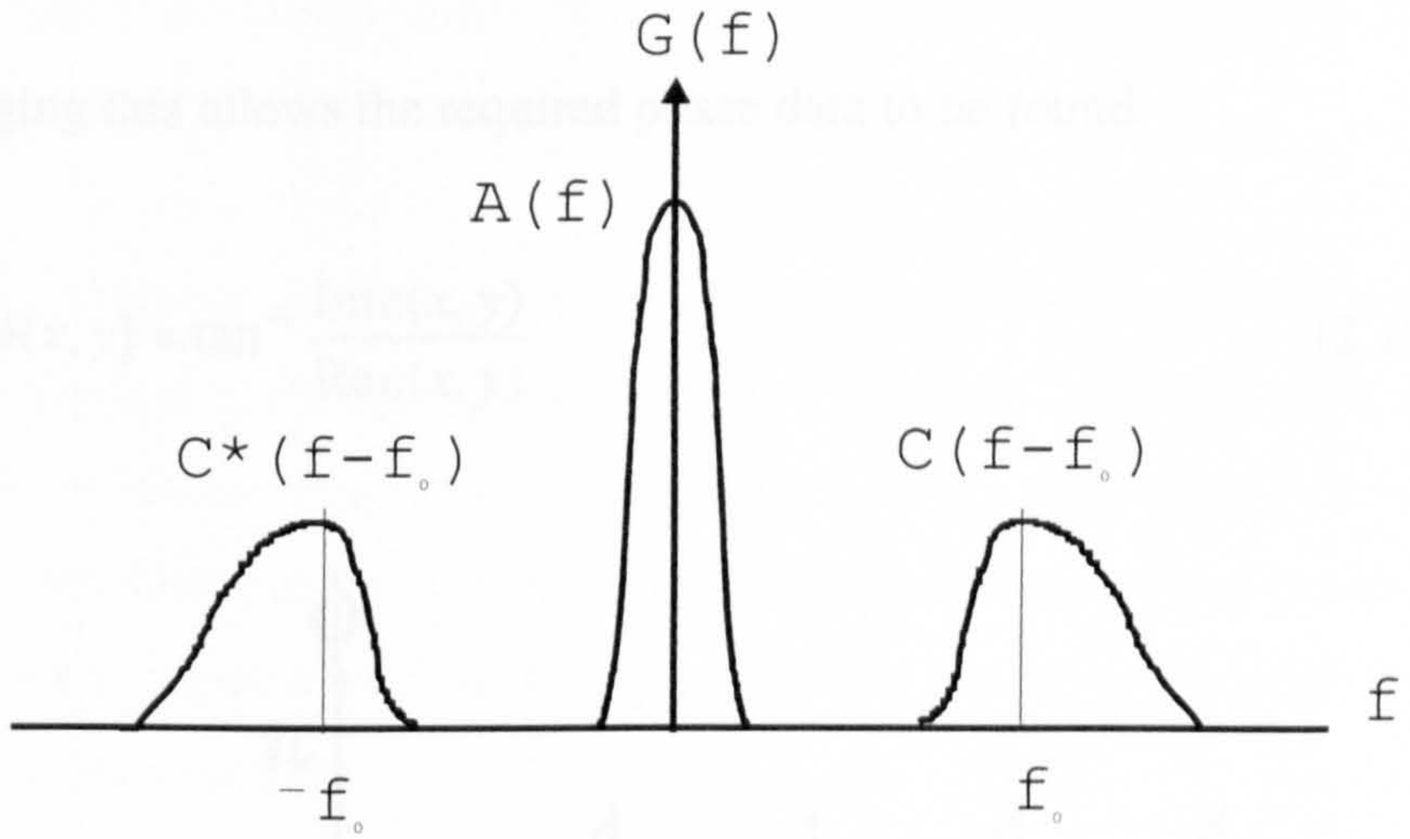


Fig.2.6 Frequency spectra, showing two carrier signals and the DC term.

The next stage is the selection of one of the carrier signals, either C ($f-f_o$, y) or C*($f-f_o$, y) which can then be translated to the origin of the spatial domain, fig. 2.7. This filters the unwanted background variation represented by A (f, y), this works because C, A and C* are physically separable. By the application of another FFT, this time acting as an inverse transform, c(x, y) is obtained. Takeda transformed this data into phase, ϕ (x, y) by the application of a complex logarithm.

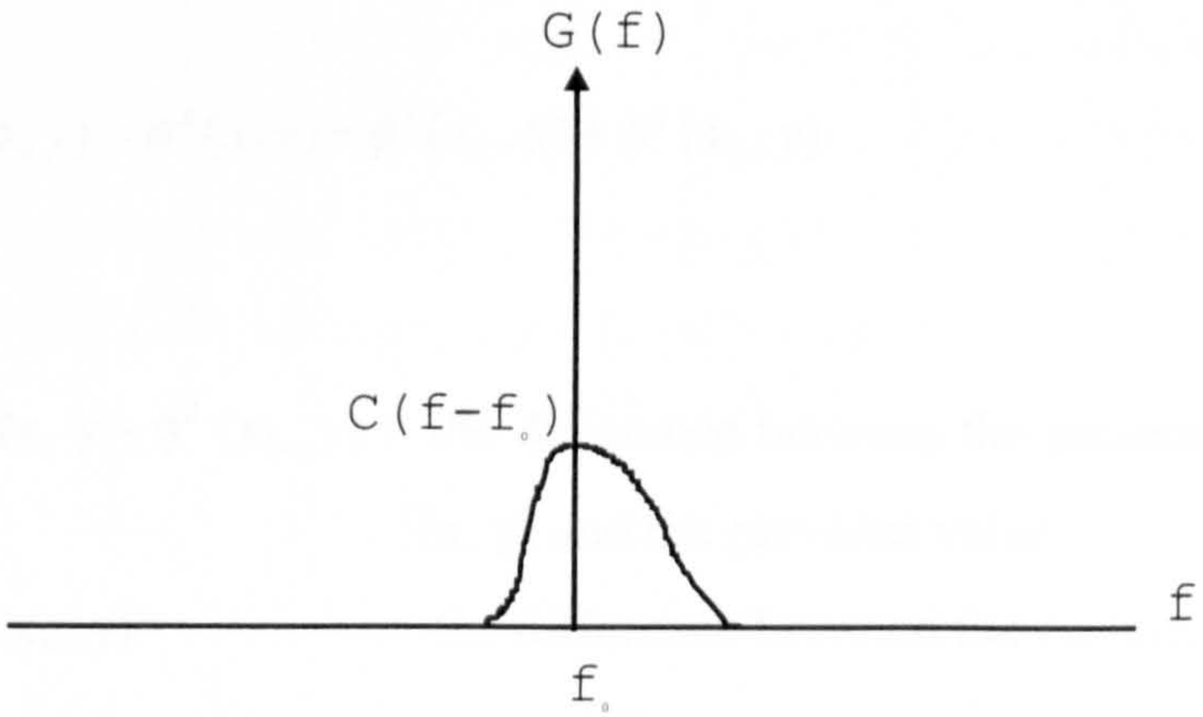


Fig.2.7 Frequency shifted carrier signal

However more frequently the data is transformed by the use of an arctan function. Having separated c(x, y) by filtering and shifting, dividing the imaginary part of c(x, y) by the real part of c(x, y) gives;

$$\frac{\mathrm{Im}[c]}{\mathrm{Re}[c]} = \frac{1/2b(x,y)\sin\phi}{1/2b(x,y)\cos\phi}$$

Rearranging this allows the required phase data to be found

$$\phi(x,y) = \tan^{-1}\frac{\mathrm{Im}\,c(x,y)}{\mathrm{Re}\,c(x,y)} \tag{2.10}$$

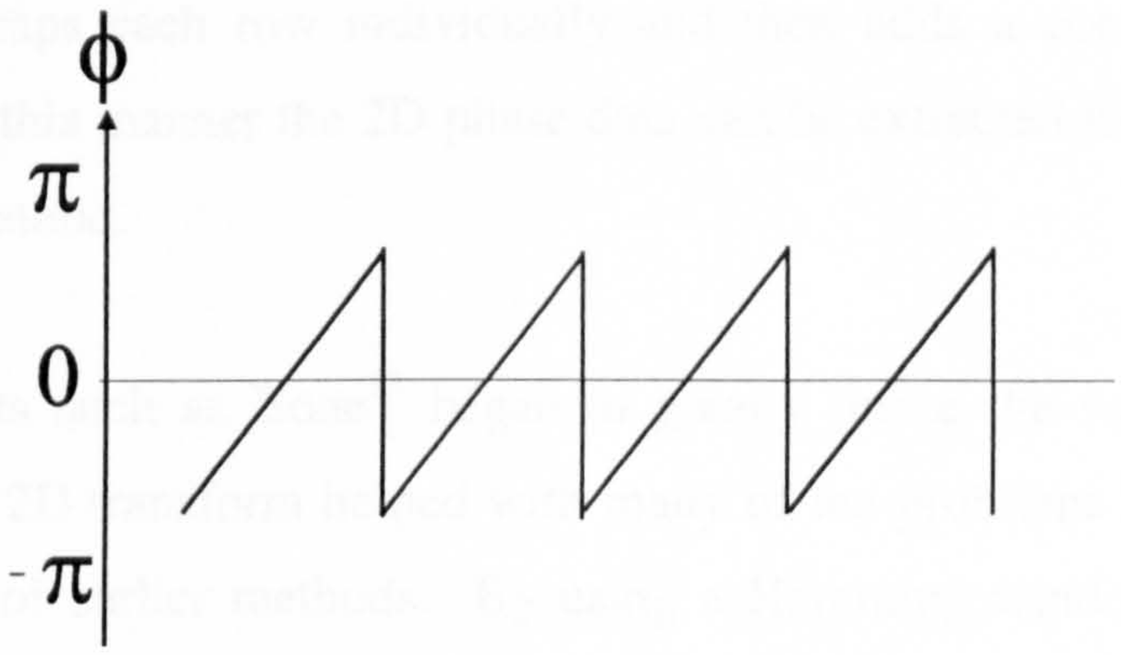


Fig2.8. Wrapped Phase

The phase data at this stage contains discontinuities as in fig.2.8, which can be corrected by the use of a phase-offset distribution or unwrapping algorithm.

$$\phi(x,y) = \phi^{x}(x,y) - \phi^{x}(x_L,y) + \phi^{y}(x_L,y) \tag{2.11}$$

where,

$\phi^{x}(x, y) - \phi^{x}(x_L, y)$ = the difference between the present value of ϕ^{x} (x, y) and the previous value

$\phi^{y}(x_L, y)$ = the difference between the current line and the previous line

This simple unwrapping function results in a surface that no longer contains discontinuities greater than 2π.

Developing these ideas further Takeda[11] produced an automatic profilometry system. The acquired image was analysed line by line, which in later works has come to mean that this is not seen as a truly two-dimensional analysis. Added to

this a square wave fringe pattern was used and while simple to produce this leads to complications in the analysis.

Macy[12] was among the first people not only to demonstrate the accuracy obtainable with this method when compared to fringe contouring techniques, but also to begin to turn the technique into a truly 2D analysis method. The unwrapper implemented in this work unwraps each row individually and then adds a component of the column phase. In this manner the 2D phase data can be extracted from the Fourier Fringe Analysis method.

Later developments such as Bone[13] began to greatly refine the work previously outlined. Using a 2D transform helped with many of the problems inherent in the unwrapping stage of earlier methods. By using a Hamming window on the data much of the later edge effect problems were avoided. This is due to the removal of many of the lower frequencies that become centred around the zeroth, DC, component. In fact many of the problems associated with edge data are introduced by the sampling nature of the Discrete Fourier Transform.

Bone also proposed that each image should have a planar surface somewhere within the imaged object. The reasoning behind this was to aid the removal of the DC component within the frequency spectra. This proposal has the obvious drawback of requiring every measured surface to contain such a region.

Green et al[14] identified two main sources of errors within the FFA system. Firstly there are additive noise components, such as noisy fringe patterns. However Fourier is fairly robust in the face of this as long as the phase function is slowly varying. The second class of problem is caused by complicated phase functions. Conversely to the first class of problem, the FFA technique is good at reconstructing complicated phase as long as the noise is not to extreme.

The next stage of the evolution of the FFA techniques was the improvement and refinement of the filter elements to cope with the previously mentioned problems. Kreis[15] highlighted the importance of any Fourier space filtering. Not only does this remove one of the carrier frequencies; it also allows the further elimination of

distortion and noise. The choice of which of the two carrier signals to use dictates the local sign of the data, but the global sign of the surface data relies on a priori knowledge of the experiment.

Unwrapping the phase produced within this technique has proven to be one of the most problematic areas of FFA[16]. There are a number of approaches to solving this, most of which are highly sophisticated. However, this sophistication although delivering a better image than the simple line-and-column technique, comes at the cost of the time to perform the unwrap[4].

2.4 Brief Review of Phase Stepping Profilometry (PSP)[1,2]

This technique is similar to FFA in that it uses the phase of a fringe pattern to produce a 3D-height map of the surface of the object. Unlike FFA this technique uses a number of images of the object to directly state the required phase[17]. There are a number of variations of this technique but in this section a 3-Frame algorithm will be used for simplicity[18].

Equation 2.7 is the expression of the modulated intensity of a point in an inference fringe pattern. For the purposes of PSP this can be restated as:

$$I = a + b\cos(\phi + \alpha) \tag{2.12}$$

The term α is added to the expression to represent an additional phase term. This is introduced because of the technique's reliance on a number of fringe patterns. By translating the fringe pattern by a known shift α, between 0 and π for consecutive images, it is easy to arrive at a number of intensity values for each pixel in the image. This translation is performed by either shifting a sinusoidial grating with mini-translation stages or by using a Peizo Transducer (PZT) with a Twin fibre Interferometer. The PZT works by moving one of the fibre heads either fowards or backwards in relation to the other stationary fibre head.

If 3 captures are performed then there are three equations which have three unknowns, these equations can be stated as

$$I_1 = a + b\cos(\phi + \alpha_1)$$
$$I_2 = a + b\cos(\phi + \alpha_2)$$
$$I_3 = a + b\cos(\phi + \alpha_3) \qquad (2.13)$$

Where the three unknowns are a, b and ϕ. Phase is the data that is required and so the equation can be resolved in the following manner,

$$\phi = \tan^{-1}\frac{(I_2 - I_3)\cos\alpha_1 - (I_1 - I_3)\cos\alpha_2 + (I_1 - I_2)\cos\alpha_3}{(I_2 - I_3)\sin\alpha_1 - (I_1 - I_3)\sin\alpha_2 + (I_1 - I_2)\cos\alpha_3} \qquad (2.14)$$

This equation can be further resolved to give,

$$\phi = \tan^{-1}\frac{(I_3 - I_2)}{(I_1 - I_2)} \qquad (2.15)$$

where,

$\alpha_1 = \pi/4$, $\alpha_2 = 3\pi/4$, $\alpha_3 = 5\pi/4$ with phase steps of $\pi/2$ (90^0).

At this stage it can be seen that the equation to be resolved is identical in form to that of the FFA after forward and inverse transforms. This implies that in PSP a stage of unwrapping must be performed and this is indeed the case.

The phase shift error[19] caused by the need to translate the fringe patterns is one of the key areas in any phase stepping routine. Hariharan et al[20] shown that using a PZT to produce the translation induces some non-linear movement to the Phase Step. This problem may be resolved by calibration.

Further errors such as non-linear detectors or unequally spaced fringes may also hinder the PSP technique. One possible way to resolve all these errors is to increase the number of Phase steps[21]. In terms of real-time imaging this technique

has one major problem in that it requires the surface of an object to be static for a number of captured images.

2.5 Comparison of FFA and PSP [22]

It is now important to see how these techniques can help attain the goals of this research. In order to highlight important issues these techniques are compared.

	Fourier Fringe Analysis	*Phase Stepping Profilometry*
Optical requirements	Simple projection device	Projection requiring mechanical translation device
Acquisition Time	Short – 1 image only	Long: Minimum of 3 images + 2 translations
Data Size	Single digital image	Three digital images
Computation upto Arctangent	FFTs, window and filter	Partially Evaluate equ.2.14
Algorithmic intensity	Relatively High	Relatively low
Noise Resilience	Can have problems in the filtering domain	Relatively noise resilient
Area of Interest	Full-field	Full or sub field may be processed
Effects of movement	Little affected by motion of object	Object must be static or very slow moving

Table 2.1 Comparison of Fourier Fringe Analysis and Phase Stepping Profilometry.

The issues raised by this comparison will discussed in later chapters.

2.6 Summary

In this chapter the two techniques that will later be explored as examples of the application of parallel processing to fringe analysis have been reviewed. It has been shown that each of these techniques have different requirements both computationally and optically. Of interest for the later work is the computational difference. Where the key issues are the difference in data required and the complexity of the algorithms.

Simply stated Fourier Fringe Analysis is a relatively low data but highly computationally complex paradigm. While Phase Stepping is a much simpler computational problem but requires much greater amounts of data. It is important at this stage to be aware that producing the arctangent and unwrapping functions are both highly computationally expensive procedures, and required by both paradigms.

2.7 References

1 Gasvik.KJ
Optical Metrology, Chapter 11
John Wiley & Sons, 2nd Edition, **1995.**

2 Robinson and Reid
Interferogram Analysis: Digital Fringe Pattern Measurement Techniques, Chapter 4
IOP Publishing, **1993.**

3 Chan C.S, and Asundi A.
Structured light assisted CAD/CAM.
SPIE Vol. 1713 Int. Conf. on Manufacturing Automation, pp. 128 – 138, **1992.**

4 Brown G.M.
Fringe Analysis for Automotive Applications.
Optics and Lasers in Engineering, Vol. 19, pp. 203 - 220, **1993.**

5 Kent E, Wheatley T, and Nashman M.
Real-time co-operative interaction between structured-light and reflectance ranging for robot guidance.
Robotica, Vol. 3, pp. 7 – 11, **1985.**

6 Arai Y et al
High speed fringe analysis method using frequency demodulation
SPIE Vol. 2340, pp.144 – 150, **1994.**

7 Born.M & Wolf.E
Principles of Optics, Chapter 7
Pergamon Press, 4th Edition, 1970.

8 Atkinson.J.T, D.R.Burton, P.Barton & M.J.Lalor
A Fibre Optic Interferometer for Fringe Projection Contouring
2nd Int.Workshop on Automatic Processing of Fringe Patterns, pp.242-248, Akademie Verlag, **1993.**

9 Halsall G.R.
Real-time Non-contact Profilometry.
Ph.D. Thesis Liverpool Polytechnic, **1992.**

10 Takeda M, Ina H, and Kobayashi S.
Fourier-transform Method of Fringe-pattern Analysis for Computer-based Topography and Interferometry.
Jn.Opt.Soc.Am., Vol. 72, No. 1, pp. 156 – 160, **1982.**

11 Takeda M and Kazuhiro M.
Fourier Transform Profilometry for the Automatic Measurement of 3D Object Shapes
Applied Optics, Vol.22, No.24, pp. 3977-82, **1983.**

12 Macy W.M.
Two-dimensional Fringe-pattern Analysis.
Applied Optics, Vol. 22, No. 23, pp. 3898 – 3901, **1983.**

13 Bone D.J, Bachor H.A, and Sandeman R.J.
Fringe-pattern Analysis Using a 2-D Fourier transform.
Applied optics, Vol. 25, No. 10, pp. 1653 – 1660, **1986.**

14 Green R.J, Walker J.G and Robinson D.W.
Investigation of the Fourier-transform Method of Fringe Pattern Analysis.
Optics and Lasers in Engineering, Vol. 8, pp. 29 – 44, **1988.**

15 Kreis T.M. and Juptner W.P.O.
Fourier-transform Evaluation of Interference Patterns: The Role of Filtering in the Spatial Frequency Domain.
SPIE Vol. 1162 Laser Interferometry Quantitative Analysis of Interferogram, pp. 116 – 125, **1989.**

16 Stephenson P, Burton D.R, and Lalor M.J.
Data Validation Techniques in a Tiled Phase Unwrapping Algorithm.
Optical Engineering, Vol. 33, No. 11, pp. 3703 – 3708, **1994**.

17 Wyant J.C. and Creath K.
Advances in Interferometric Optical Profiling.
Int. Jn. Mach. Tools Manufact., Vol. 32, No. 1 / 2, pp. 5 – 10, **1992.**

18 Creath K.
Temporal Phase Measurement.
Interferogram Analysis, IOP Publishing, pp. 94 – 140, **1993.**

19 Creath K. and Schmit J.
Errors in Spatial Phase-stepping Techniques.
SPIE Vol. 2340 New Techniques and Analysis in Optical Measurements, pp. 170 – 176, **1994.**

20 Hariharan P, Oreb B.F, and Eiju T.
Digital Phase-shifting Interferometry: A Simple Error-compensating Phase Calculation Algorithm.
Applied Optics, Vol. 26, No. 13, pp. 2504 –2505, **1987.**

21 Hibino K, Oreb B.F, and Farrant D.I.
Phase Shifting for Nonsinusoidial Waveforms with Phase-Shift Errors.
Jn. Opt. Soc. Am., Vol. 12, No. 4, pp. 761 – 767, **1995.**

22 Pearson J.P.
Automated Visual Measurement of Body Shape in Scoliosis.
Ph.D. Thesis Liverpool John Moores University, **1996.**

Chapter 3

Brief Review of Parallel Processing for Image Analysis

3.1 Introduction

Image analysis techniques such as FFA place a high demand on the processing power of any computer system used. This is especially true when the system is constrained by the requirements of real or near-real time processing. One method of matching these specifications is to utilise parallel processing schemes.

In the field of parallel processing a number of different strategies have evolved to perform such intensive calculations. Many of the processing elements used have benefited from implementing parallel architecture, developing these chips away from "classical" Von Neumann architectures. The following sections examine those elements of parallelization that are directly relevant to this research.

3.2 Classification of parallel architectures[1]

One method of categorising parallel architectures is to split processor chips into three major groupings

- Pipelined – such architecture uses overlapping computations to achieve temporal parallelization.
- Array processors – use multiple arithmetic logic units to achieve spatial parallelization
- Multiprocessor systems – use interactive units to achieve asynchronous parallelization.

These groupings are not exclusive and a system that is pipelined may well have elements of the other two groupings. Of most interest to this work are those processors in the multiprocessor chip category.

Further divisions of computer architecture are available, perhaps the best known of which is Flynn's Classification scheme, often used to define parallel architectural systems. These may be based upon a mixture of the proceeding classes. The groupings in this scheme are

- Single Instruction Stream - Single Data Stream (SISD)
  Instructions are executed sequentially but may be overlapped. Such systems may have more than one functional unit, but one central control unit manages them all.
- Single Instruction Stream - Multiple Data Stream (SIMD)
  One control is used to instruct multiple processing units. These units will be acting on different data sets.
- Multiple Instruction Stream - Single Data Stream (MISD)
  A largely conceptual classification where shared data is manipulated by a number of processing units each controlled with it's own instruction set.
- Multiple Instruction Stream - Multiple Data Stream (MIMD)
  Most multiprocessor units and computers can be classified in this category. A number of processing elements execute different instruction sets on multiple data sets, such systems have high levels of processing element interactions.

Conventional processors are classified as SISD architectures, whereas processors chips such as the TMS320C40 can be placed in the MIMD category. Lim[2] outlines the key advantages that the C40 has over more conventional processors, these can be summarised as

- Smaller overheads when responding to interrupts
- Separate data buses for instructions and data
- Hardware support for parallel multiply and accumulate
- Fast on chip memory

However, getting the full benefits out of such hardware is based on the ability of the programmer and compiler to fully utilise these special features.

3.3 MIMD Based Topologies

There are a large number of possible theoretical topologies for MIMD machines. Most of the architectural topologies that are used in image analysis are built up from a number of simple basic shapes. The first of these is the simple pipeline structure, which may have any number of processing elements connected in a single strand fig.3.1.

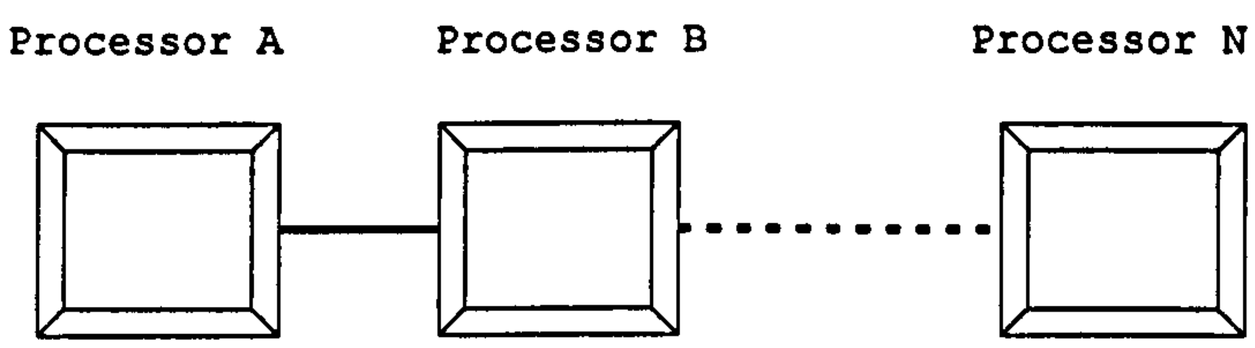


Fig.3.1 Simple Pipeline Structure

In this form of parallelization the data flows along a logical sequence of processing elements. Each element works on the same data set but uses a different instruction set. This simple to program and configure system lacks the benefits of overlapping tasks. Further, if any communication link or processor should fail there is no possible way to re-route data.

Such schemes can be used to build an entire network. One of the advantages of these systems is the simplicity of the required hardware. However, the disadvantages include the increase in the time to access memory and lowered communication bandwidth[3].

The next building block structure is based upon a simple ring, fig.3.2. This scheme allows processing elements to communicate more freely than the pipeline structure. Additionally it is a simple task to build more complicated structures from this basic configuration. However, if there is a failure in any of the communication links there

will be a great deal of re-routing needed, and this can easily overburden the C40 processor.

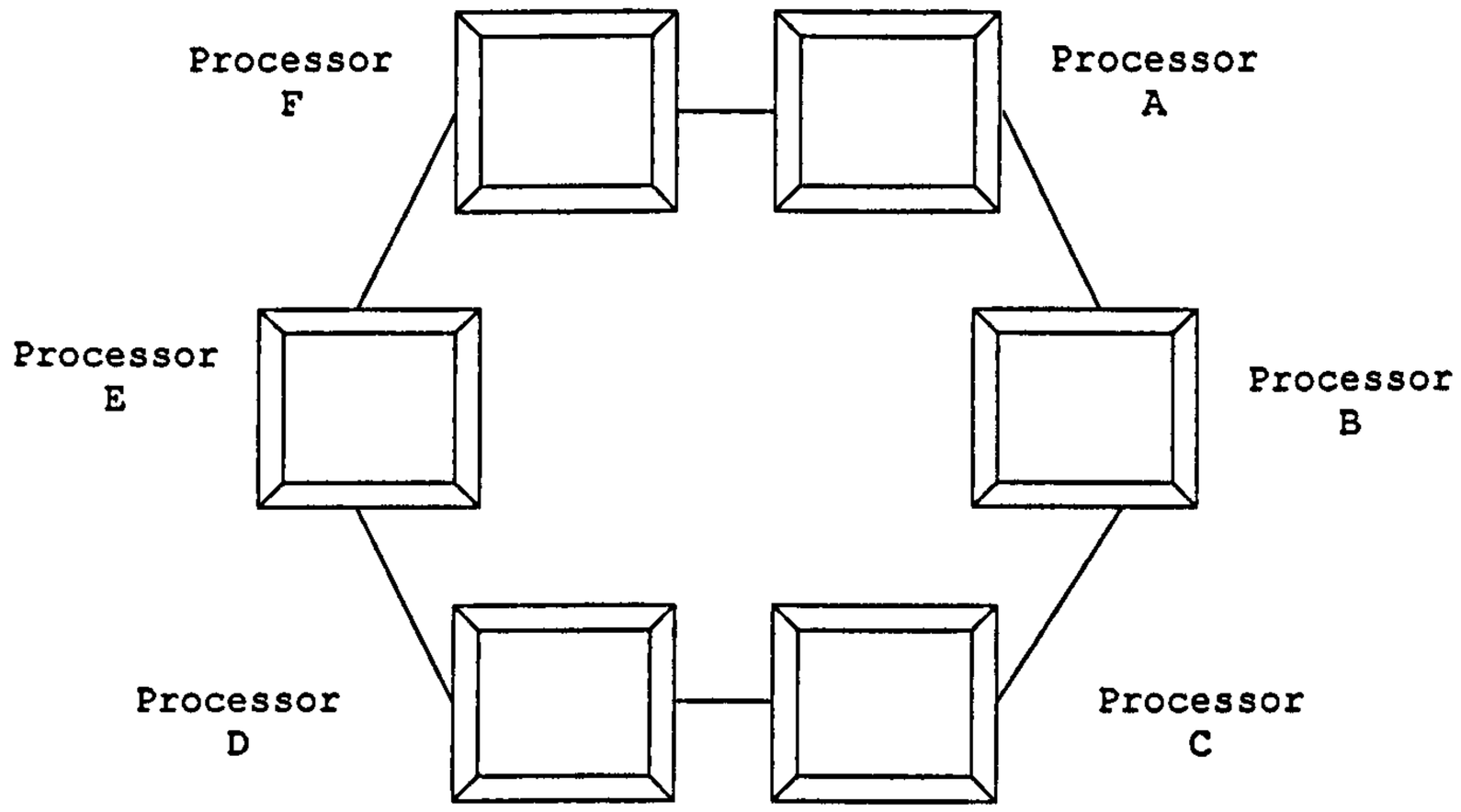


Fig.3.2 Ring Structure

In the pyramid structure one processing element acts as the parent processor and communicates data and instructions to the child processors, fig.3.3. Logically this scheme benefits from having one higher-level processor that in some way controls the flow of the process. If a processor or link should fail the data may be re-routed but if the parent processor fails all processing will be stopped.

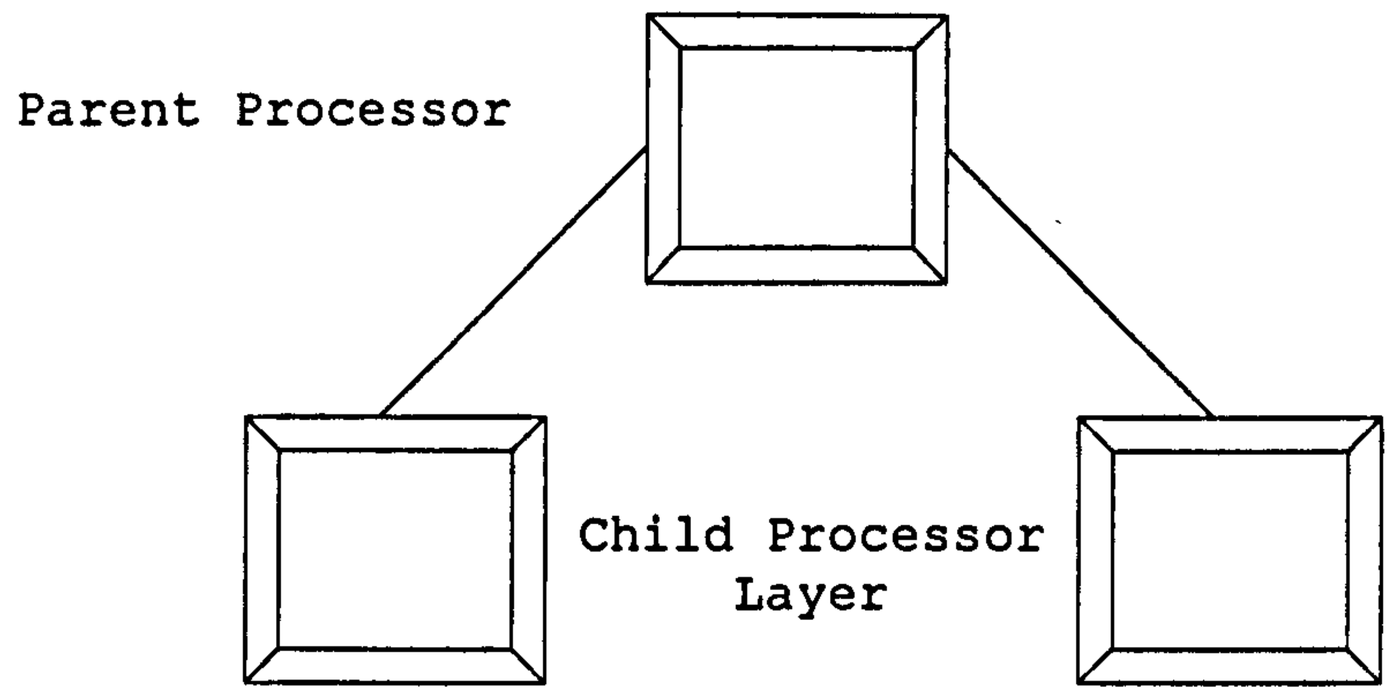


Fig.3.3. Simple Pyramid Structure

From these simple structures a great deal of more complicated and more useful structures may be achieved. A number of schemes have been used with image processing the first of which is based upon a simple mesh structure[4]. A mesh is easy to configure being based upon a ring shape, but it differs and shows enhanced performance by having fully connected processors, fig.3.4. This means that data no longer needs to flow in a pipeline manner.

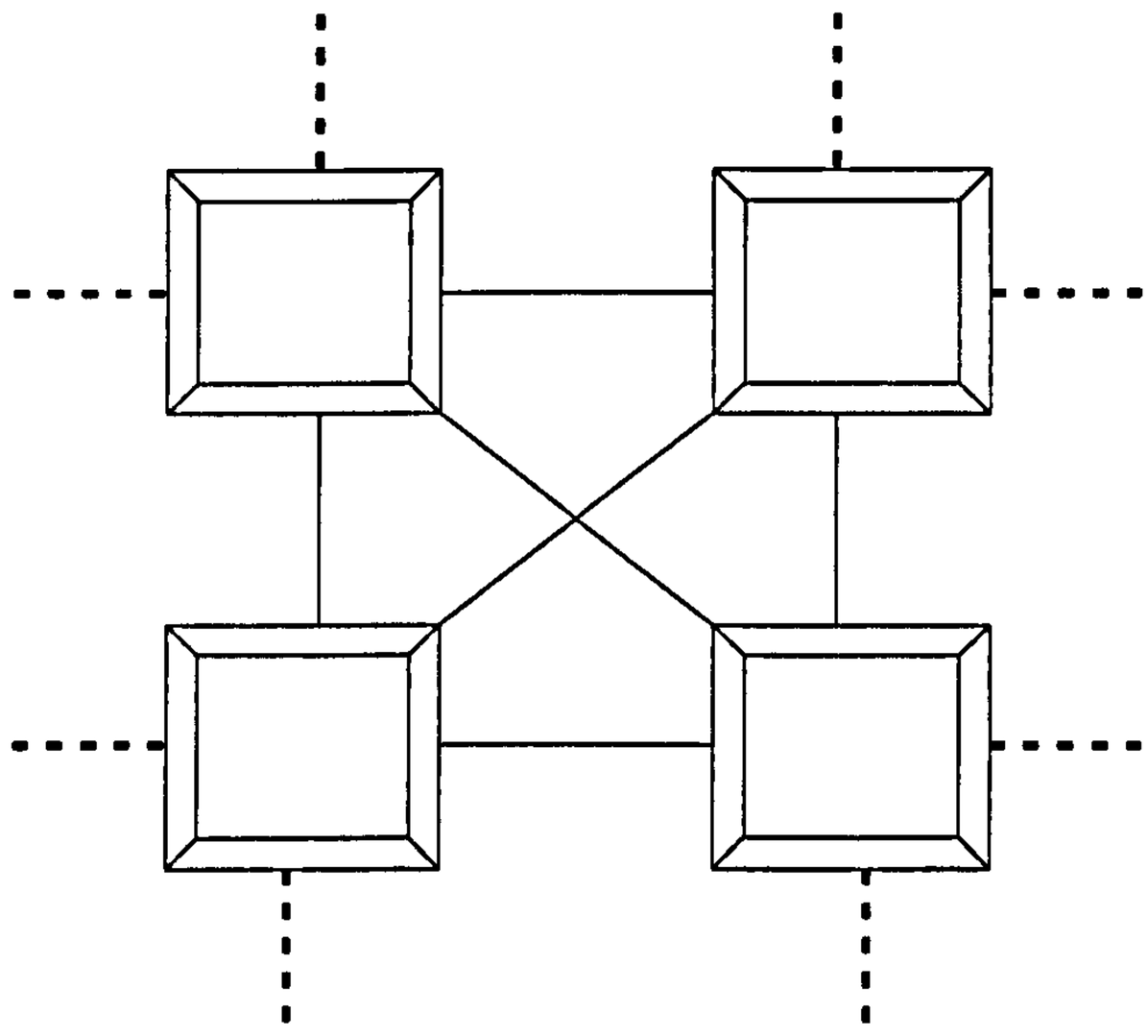


Fig.3.4 Mesh Structure

The next evolution of this mesh structure is to place one block above the mesh to form a simple pyramid structure, fig.3.5. Rutowitz[5] showed that this topology is well suited to image processing. In this scheme at the higher levels of the pyramid the complexity of the processors can be allowed to increase, this scheme fits well with a system that must use a framestore to acquire data and pass the captured image to lower level processing elements.

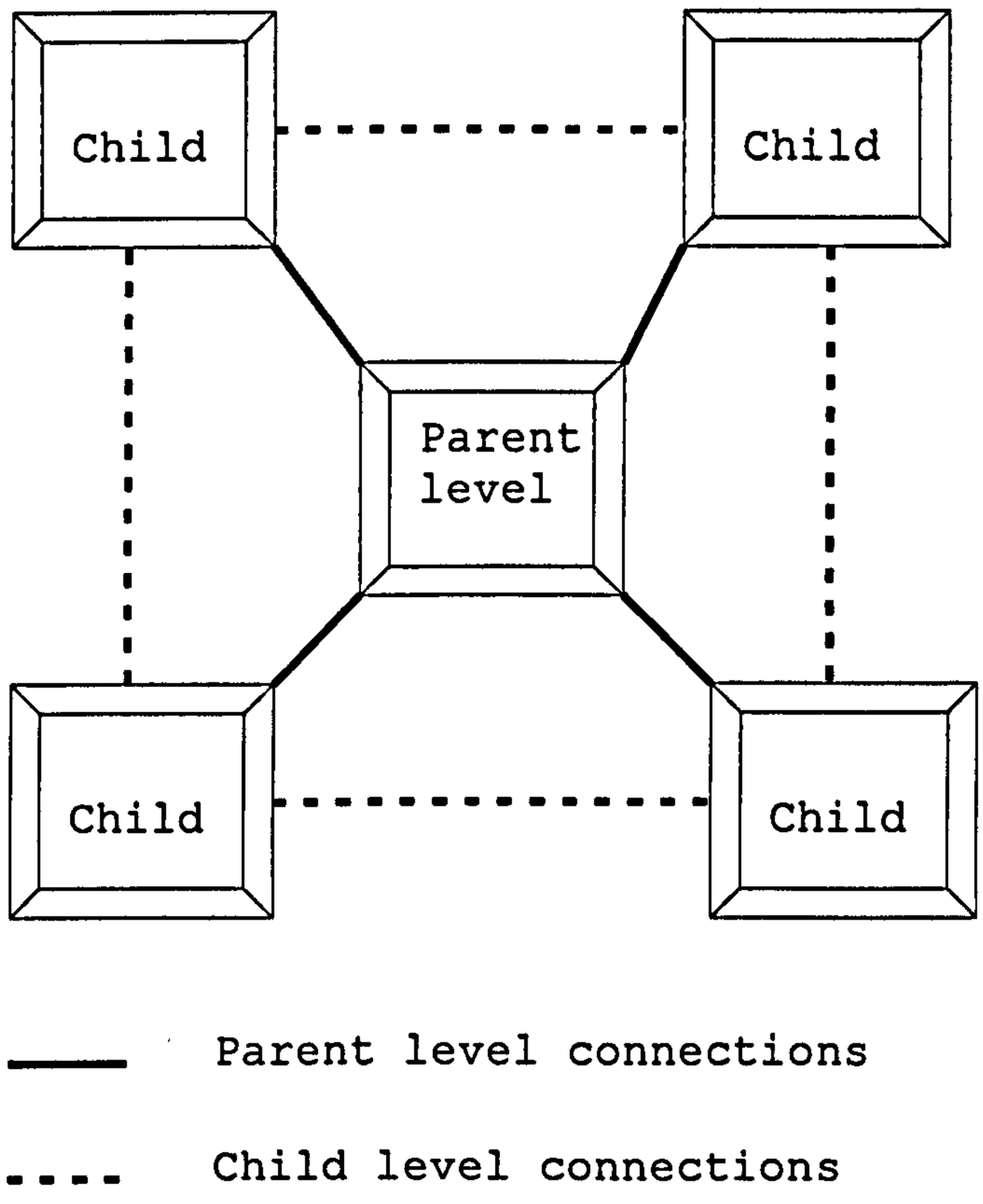


Fig.3.5 Pyramid Structure, not all connections are shown for the sake of clarity

The next development in the parallel image analysis field was the hypercube[6]. This structure can be built up from a number of lower level hypercube structures, where a hypercube is merely a ring structure of 2^N in size. By building up these structures a higher dimensional cube can be achieved. This system has the advantage that communications can never occur over a length greater than the diameter of the cube. Additional redundancy may be built into the system by using the additional communication links of the C40. In fact the basic physical structure of the C40 and it's peripherals is very well suited to this design.

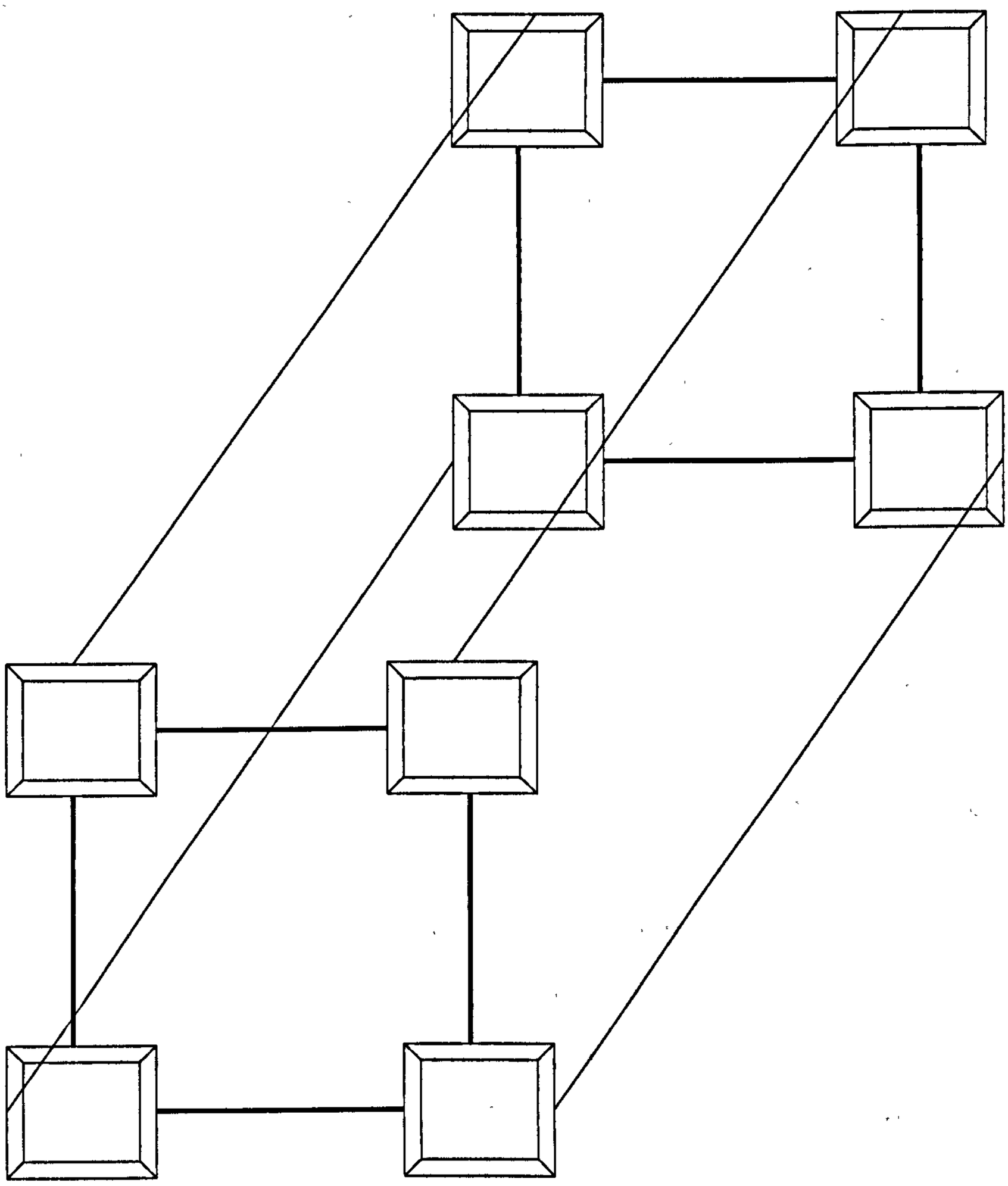


Fig.3.6. Hypercube Structure

Bhandarkar[7] recently purposed the use of a reconfigurable network based on the ring structure. In this scheme a number of processing elements are connected in a ring and rings are stacked on top of each other. One of the key problems with this scheme is that if one link fails the re-routing of data or instructions can overburden the C40. Thus, similarly to the hypercube a certain amount of redundancy must be built into the system by using additional communication ports.

3.4 DSP Parallel Image Processing

Of crucial importance in this research is the validity of using a DSP processing system when compared to a conventional processor. There is no value in comparing the MIPS or MFLOPS that any number of processors can perform. The best method is to benchmark systems by using a number of functions that can be run on a number of different machines. Tokhi[8] demonstrated that the C40 is extremely good at performing a number of DSP functions when compared to the Intel 486DX2. However, when used to perform matrix manipulation the C40 doesn't compare so favourably with the 486.

Tokhi states that the C40 only narrowly outperforms the 486 when calculating a 512 point FFT. However, other work [9,10,11] in the field points to the C40 performing these operations much quicker than the 486. Using the same benchmark function for each machine causes the large disparity in observed timings. The comparative study relies on the compiler to best optimise the code, yet work by Al-Hamdan[4] has shown that C40 code is best optimised by the programmer. Therefore, the ability of a program to run effectively is dependent not only on matching process to processor but also on matching programs to processors.

Much time has been spent on optimising the C40 code to perform DSP functions such as the FFT, but until recently the interaction between the parallel processing system and the user had not matched initial high expectations. Perhaps the reason for this is that so much time has been spent on optimising code and finding the best ways to take advantage of these systems, that little time has been spent on the software to use them.

Assembly Code has been the best and most traditional way to program the C40. This has meant that the users of such a parallel processing system have had a tendency to be people who are experts in the use and optimisation of code for specific hardware[12]. In recent years the power and ability of compilers has meant an increasing use of high-level languages such as C. Yet even these high level

languages have had a tendency to be unwieldy and full of very specialised ideas. The recent advent of HPF, C++ and Embedded Java has begun to change this [13,14].

Knowledge of the complexity of the parallel-processing problem is necessary before the applicability of software tools can be discussed. This complexity is most often labelled the granularity of a problem, where the granularity of a parallel task is the based on the range of size of the sub-tasks that can arise from the main problem[1, 10]. Granularity can be used to decide upon the number and computing power of processors required by a system. Fine-grained parallelism needs a large number of simple processors for each sub-task, while coarse-grain parallelism requires a small number of processors to perform complex computation. Between these two extremes lies medium grained parallelism

It is for this middle group that new tools for coding and controlling C40s have begun to arrive on the market. These new design tools allow the user to graphically manipulate and control data[1, 12, 15], this means that the user need no longer be solely interested in the optimisation of low-level code. By using the graphical tool the user can begin to experiment with the DSP in a much more adaptive and Object Orientated manner. Much work still needs to be done to adapt and implement such graphical tools on coarse-grained problems such as Fourier Fringe Analysis.

3.5 Summary

This chapter has described some aspects of parallel processing which are relevant to Fringe Analysis. Much work has been carried out in the classification of various forms of parallel processing, the most useful of which is the Flynn Taxonomy. From this it can be seen that processors that have both multiple instruction and data sets have been most widely exploited in this field.

In the field of image processing it can be seen that specialist DSP processors have an advantage of speed over their Von Neumann counterparts. But have the disadvantage of placing much emphasis on the optimisation of code upon the programmer. Added to this the constraints that are placed upon the programmer by conventional environments, have proven "user-unfriendly".

3.6 References

1 Hwang K. and Briggs F.A.
Computer Architecture and Parallel Processing, Chapter 2.
McGraw-Hill International Editions, **1984.**

2 Lim S.Y, Dawson D.M. and Vedagarbha P.
Advanced Motion Control of Mechtronic Systems via a High-Speed DSP and a Parallel Processing Transputer Network.
Mechtronics, Vol. 6, No. 1, pp 101 – 122, **1996.**

3 Radivojevic I. And Herath J.
DSP Architectural Features for Intelligence and Control Applications.
Proceedings of IEE International Symposium for Intelligent Control, pp. 273 – 278, **1990.**

4 Al-Hamdan S.
A Comparison of Two Parallel Computer Architectures in the Context of Interferometric Fringe Analysis.
Ph.D. Thesis Liverpool John Moores University, **1996.**

5 Rutovitz D, and Travis A.J.
Parallel Processing Architectures for Image Analysis.
Major Advances in Parallel Processing, The Technical Press, pp. 288-295, **1987.**

6 Jain Y.
Parallel Processing with the TMS320C40 Parallel Digital Signal Processing.
Parallel Processing with the TMS320C4x – Application Guide, Texas Instruments, pp.13 - 46, **1994.**

7 Bhandarkar S.M, and Arabnia H.R.
Parallel Computer Vision on a Reconfigurable Multiprocessor Network.
IEEE Transactions on Parallel and Distributed Systems, Vol.8, No.3, pp. 292 –308, **1997.**

8 Tokhi M.O, and Hossain M.A.
CISC, RISC and DSP Processors in Real-time Signal Processing and Control.
Microprocessors and Microsystems, Vol.19, Pt.5, pp.291 – 301, **1995.**

9 Peidra R.M.
Parallel 1-D FFT Implementation with TMS320C4x DSPs
Parallel Processing with the TMS320C4x – Application Guide, Texas Instruments, pp.121 - 187, **1994.**

10 Kshirsagar S.P.
High Speed Image Processing System Using Parallel DSPs.
Ph.D. Thesis Liverpool John Moores University, **1994.**

11 Harvey D.M, et al.
Digital Signal Processing Systems Architectures for Image Processing.
IEE Conf. Image Processing and Its Applications, pp.460 – 464, **1995.**

12 Decker K.M. et al.
Satisfying Application User Requirements: A Next-Generation Tool Environment for Parallel Systems.
Lecture Notes In Computer Science, Vol.919, pp.206 – 228, **1995.**

13 Foster I.
Designing and Building Parallel Programs, Chapter1
Addison-Wesley, **1995.**

14 Fox.G.C et al.
Parallel Computing Works, Chapter1.
Morgan Kaufman Publishing, **1994.**

15 Ledeczi A. et al
Modelling Paradigm for Parallel Signal Processing.
Australian Computer Journal, Vol.27, Pt. 3, pp. 92 – 102, **1995.**

Chapter 4

System Hardware, Software and Optical Equipment

4.1 Introduction

As has been previously stated this research uses a parallel processing system to achieve the desired goals of the INFOCUS project. The equipment used to achieve this has had a major impact on the system designed. There are several categories that this system can be divided into:

- Hardware
- Software
- Optical System

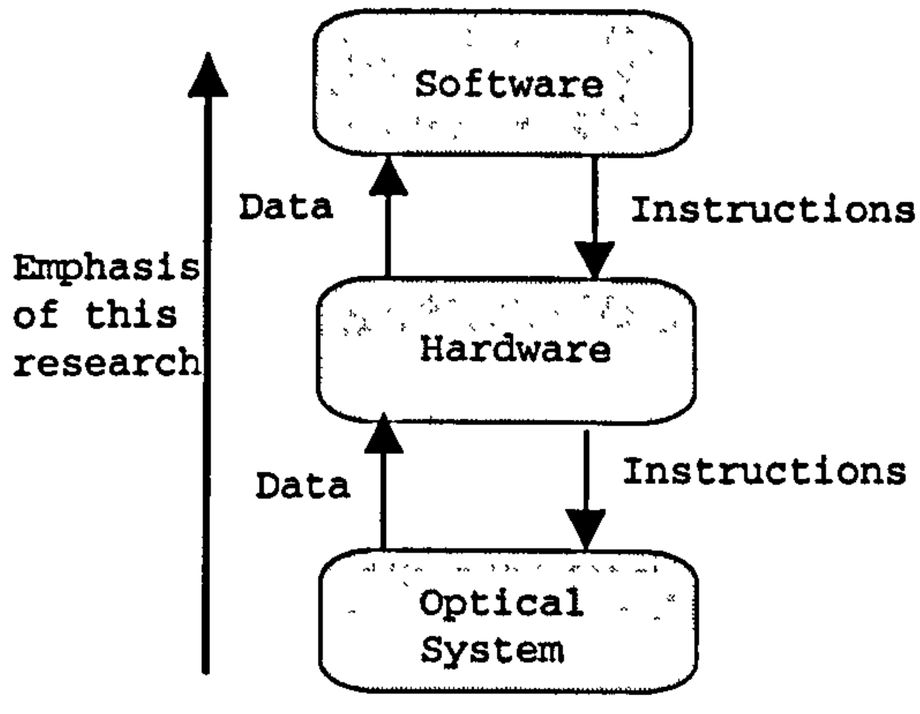


Fig.4.1 Categories involved in INFOCUS Optical Sensor project.

In this chapter those elements that have been used in the final system will be discussed.

4.2 Hardware

The computer system used in the INFOCUS Optical Sensor is comprised of:

- Host PC
- 9 TMS320C40 DSP Processors (C40s)
- 1 Framestore with on-board C40 (TDM435)

This is shown in the following diagram, fig.4.2

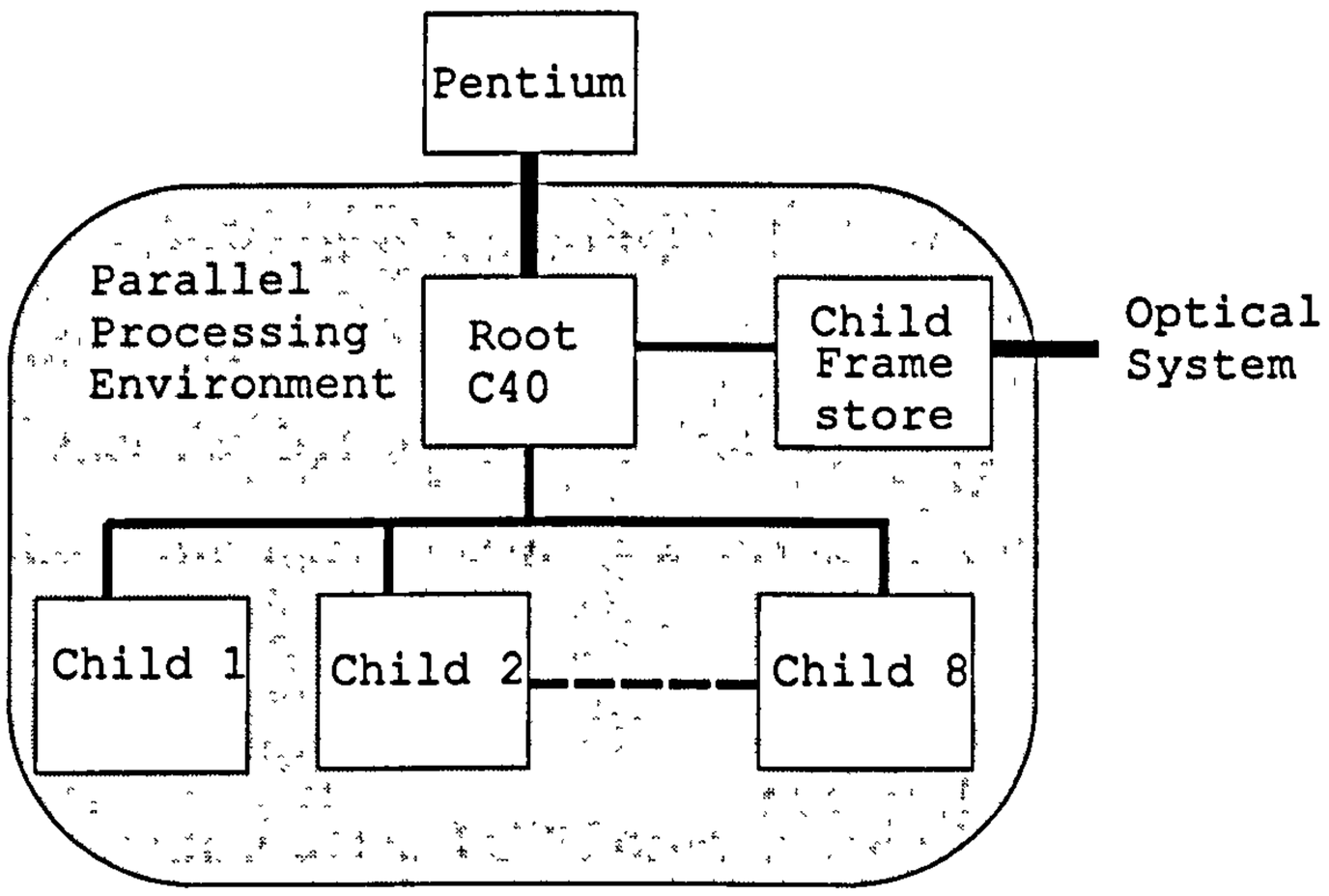


Fig.4.2 Showing the Hardware System used in this research.

The host PC is merely used to provide an operating system from which the specialist hardware can be accessed with a Graphical User Interface (GUI). As no specialised processing is performed on this system it will not be detailed in this work.

4.2.1 The C40 Processor[1,2,3]

This floating-point digital signal processor has evolved with parallel processing applications in mind. The key features that are of importance to this research are:

- **Central Processor Unit (CPU)**
  Massively parallel design of the CPU allows concurrent single-cycle multiplication and addition/subtraction in certain modes. A number of novel data addressing modes are possible such as circular and bit-reversed.
- **On-chip program cache and single-cycle RAM.**
  The 512-byte cache enhances the performance of the CPU, while the on-chip RAM can be used to lower system costs.
- **Communication Ports**
  Each C40 module has 6 communication channels for inter-processor communication. Either the CPU or the DMA can use these channels to transfer data.
- **Six Channel DMA Coprocessor.**
  This allows concurrent I/O and CPU operation, freeing the CPU for maximum sustained performance.
- **Two identical external data and address buses**
  Allowing shared memory systems and high data rate, single-cycle transfers.

4.2.2. The TDM435 Framestore [4]

Image acquisition requires specialised hardware and to achieve this a Framestore is used. The TDM435 provides the ability to display and capture on the same module. This module is based upon a 40MHz C40 DSP processor.

The TDM435 module used has 3Mbytes of Video RAM (VRAM) and 4Mbytes of Enhanced Dynamic RAM (EDRAM). The VRAM is organised into 3 banks of memory. Two of these banks may be used for either capture or display, while the

final bank is used as an overlay bank for display only. There are a number of modes of addressing the VRAM, which are available for both capture and display. Most important among these is the use of byte or word mode. These allow data to be used in either packed or unpacked fashion.

4.2.3 The TIM412 Motherboard[4]

The TDM411s (C40s) and the TDM435 are all housed on TIM-40 compatible motherboards. With 4 processors on a TIM-40 each module will be fully connected with all the other modules on that motherboard, fig.4.3. There are then three other available connections on each module, which can be used to connect to other TDM411 modules.

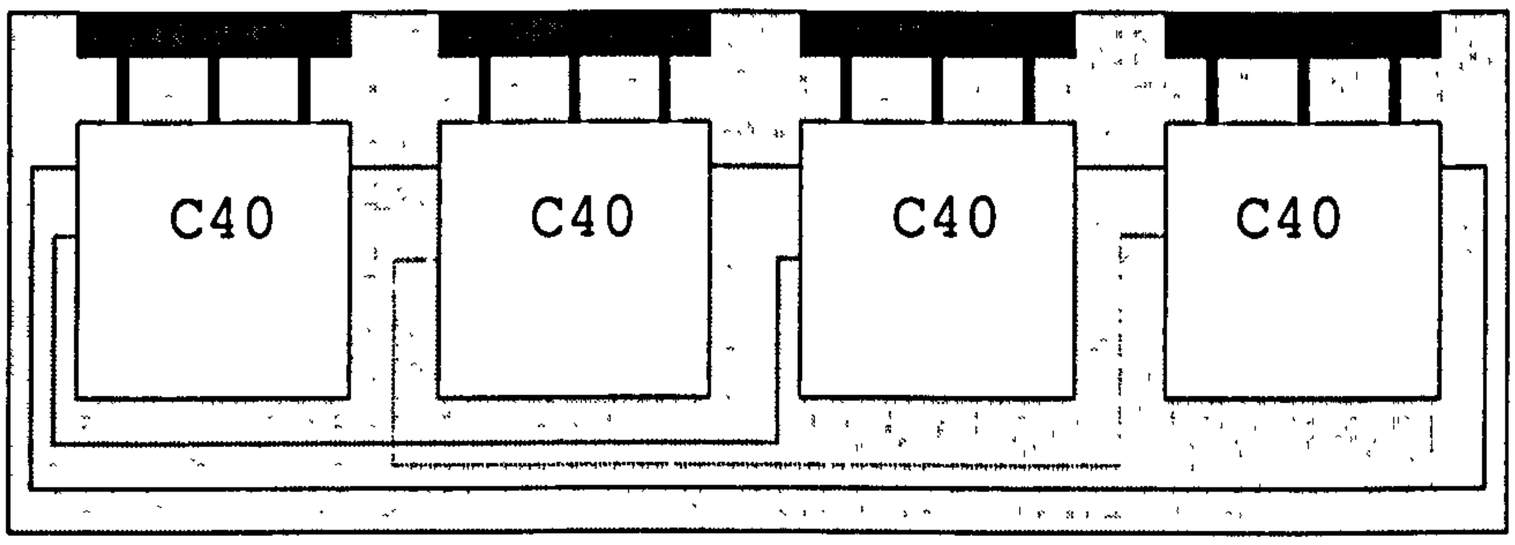


Fig.4.3 Simplified drawing of the TIM411 with 4 fully interconnected processors

When using the TDM435 on a motherboard only one other processor can be placed on the same motherboard, fig.4.4. Both of these modules will have 4 connections available to connect to external processors.

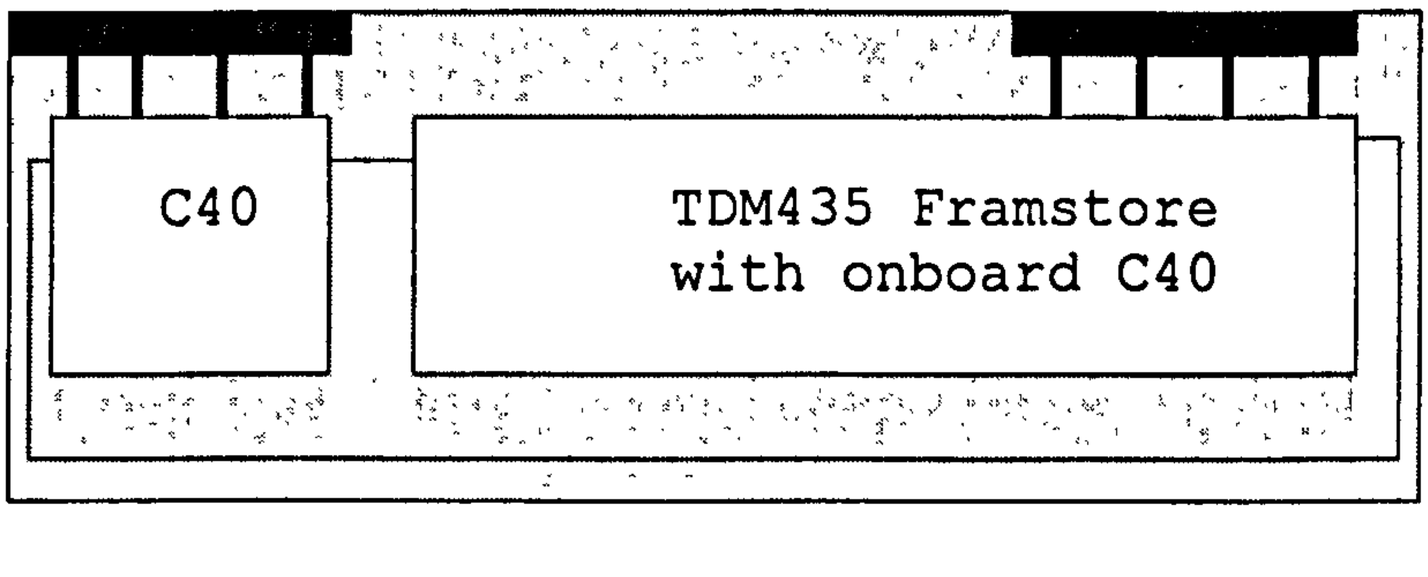


Fig.4.4 Simplified drawing of the TIM411 with Framestore and one C40

Within a parallel processing system connected to a host PC, one module must act as a root processor. This root will be the only processor able to communicate with the host PC. In other words any I/O commands, such as print, can only be performed through the Root node.

4.3 Software

This section outlines a number of different elements that make up the software used in this research, fig.4.5. Some of the elements are extremely low-level, such as assembly code, whilst others are high-level design tools, such as the Hypersignals Block Diagram Package.

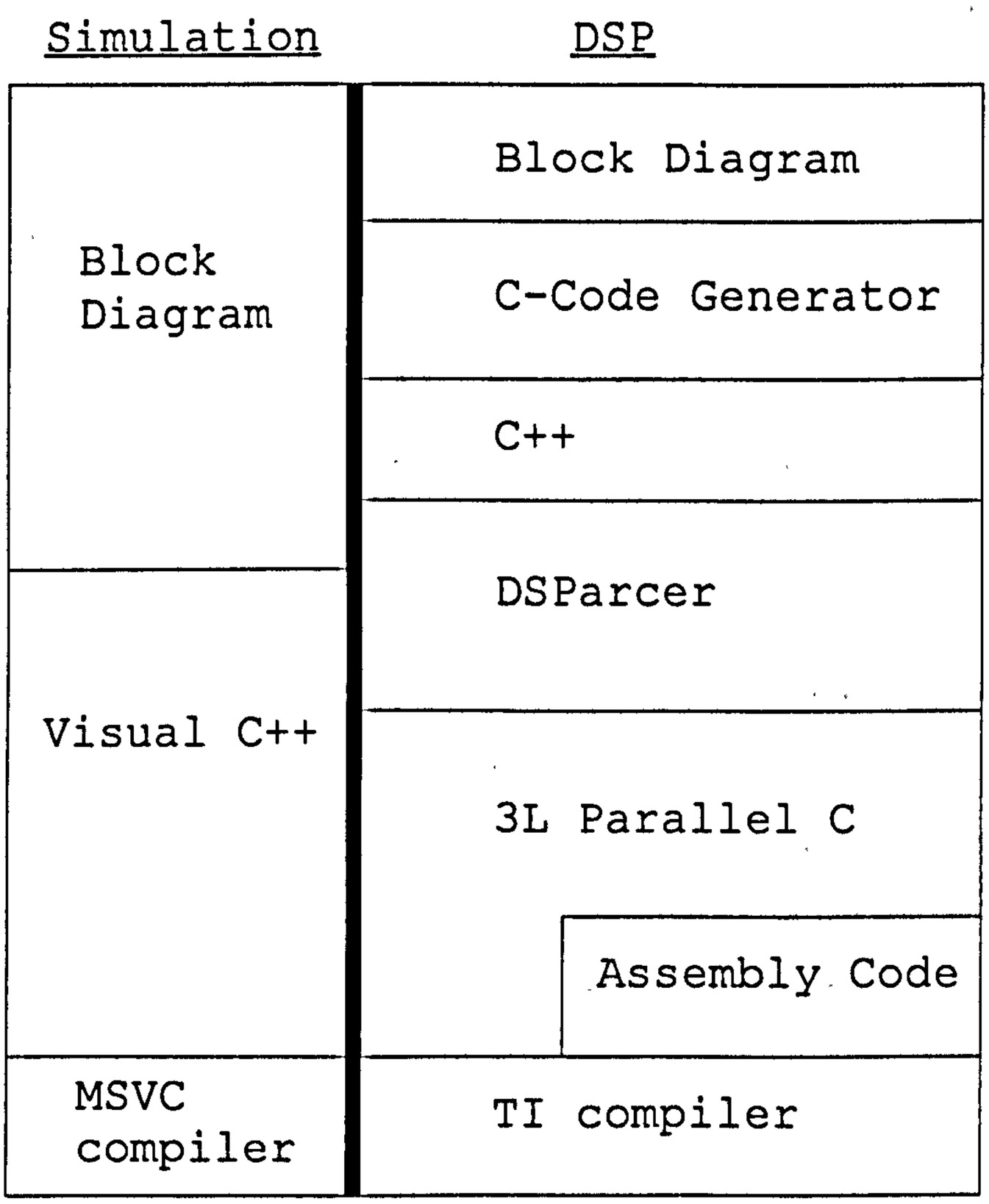


Fig.4.5 Elements that make up the programming environment of the INFOCUS Optical Sensor

4.3.1 Programming Languages Used

There are a number of programming languages available to the C40 programmer. This work used a limited number of them. Assembly code was used to create, adapt and optimise some functions, while 3L Parallel C was used to produce most of the algorithms. When working with the Pegasus environment simulation code was written in Visual C++.

- **Assembly Code** [5]

The C40 assembly code is especially well adapted to digital signal processing. The instructions set is fundamentally suited to DSP work. Assembly code in this research has been used to optimise possible bottlenecks. These optimisations are developments from previous work[2].

- **3L Parallel C Code**[6, 7]

This variant of C contains not only the standard functions but also those special features of parallel processing, such as threads and DMA commands. This all implies that the 3L is not a language that was especially designed for parallel processing but rather one that is adapted to the problem.

The model of 3L parallel processing can be summarised in the following way. An application is built up from a number of concurrently executing tasks. Tasks are individual C programs each containing a main function and each with their own segment of memory. A linker is used to produce the image file for each task, and the collection of these files that make up an application are combined by using a configuration file.

- **Microsoft Visual C++**

Visual C++ is an Object orientated programming environment. This form of programming is based on the manipulation of data objects rather than numbers. The reason for using this environment is that Pegasus takes advantage of the Windows programming capabilities of MSVC.

- **Framestore Software** [1]

The TDM435 framestore used in this research is supplied with two libraries of functions for low-level capture and display. These libraries are linked into C programs allowing a number of functions to be called.

4.3.2 Pegasus Design Environment

Pegasus is a suite of applications that are used to address the entire DSP development cycle, including simulation. The individual elements of Pegasus are used to produce compiled and optimised code that runs on an array of C40 DSP processors. This is under the control of a parallel processing operating system.

All of the software described so far is used in varying degrees within the Pegasus environment. Perhaps the most important of these is Parallel C. The idea of tasks is used throughout those components that fall under the umbrella of Pegasus. These components are described in the following sections.

- **Hyperception Block Diagram** [8]

This graphical pseudo-flowchart is used to build up a project from blocks of code. These blocks form a library of re-usable DLLs.

- **Block Wizard**

Block wizard is used to write the Dynamic Link Library (DLL's) used in the Block Diagram system. These DLL's contain both Host and DSP code.

- **C Code-Generator (CCG)**[9]

Once a project has been designed and tested under the Block Diagram package it can be used by CCG. This takes the project and produces a single threaded piece of ANSI C code.

- **DSParcer** [10]

This is used to convert the single thread code produced by CCG into DSP code, with the correct data calls between the DSP system and the Host processor.

- **Jovian Info Server** [11]

This component of the environment is used to actually run the DSP code.

4.4 Optical Equipment [12]

For the INFOCUS Optical Sensor to meet the stated objectives the optical system used must meet the following characteristics:

- Be able to work well in the Radiotherapy environment
- Have the flexibility to produce the structured light patterns required by the various algorithms chosen

To this end a coupled fibre-optic interferometer that is capable of producing a number of structured light patterns has been used. These patterns are achieved by the interference between twin coherent beams of light, see Chapter 2.

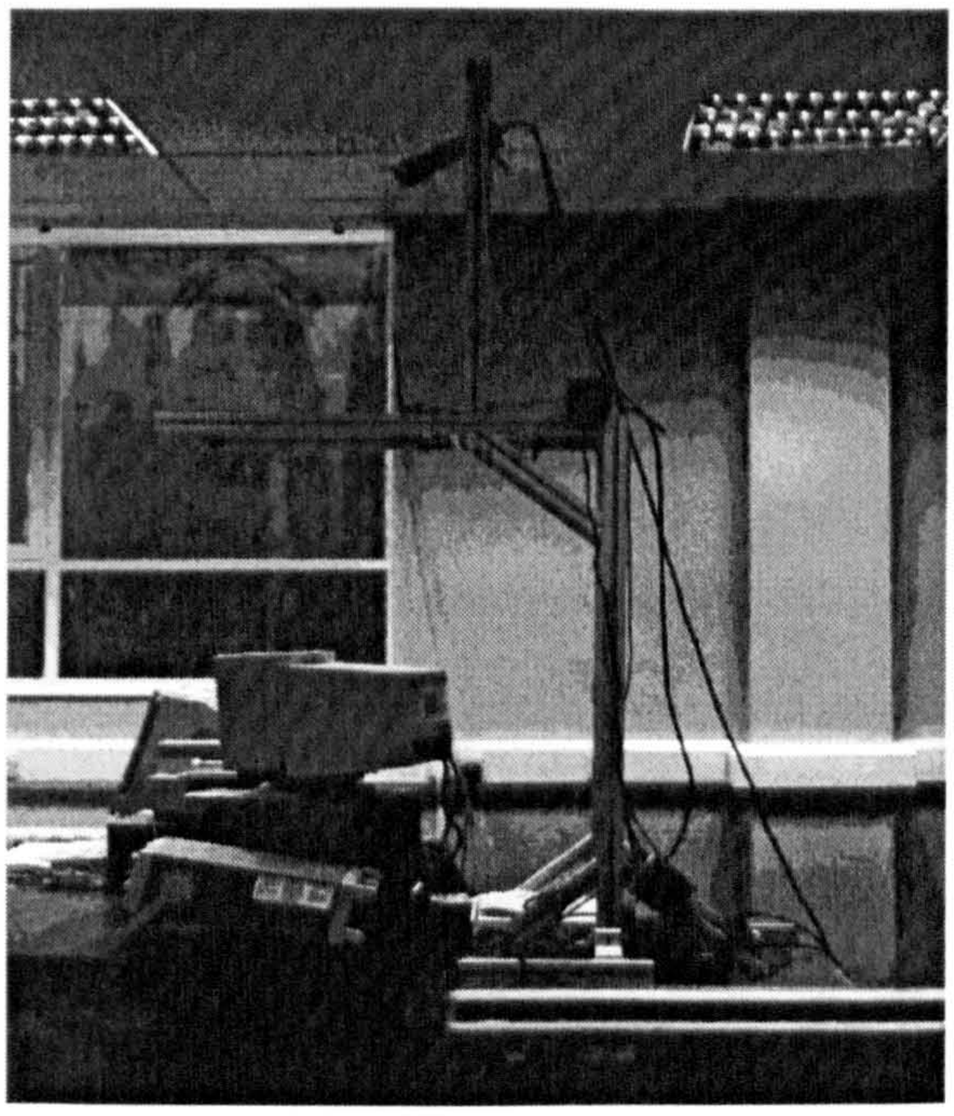


Fig.4.6 Optical Sensor Equipment in Experimental Rig

4.5 Summary

After reviewing this chapter it can be seen that there are a large number of individual parts within the INFOCUS Optical Sensor. The major practical focus of the author's research is upon the computer software to obtain data for patient monitoring. All of the elements outlined will have some impact on the final design of this system.

The major impacts will be caused by the computer hardware and the software used. The best example of this is the movement and storage of data through the system. In the next chapter the constraints that arise form the physical hardware and the software environments will be examined.

4.6 References

1 Al-Hamdan S.
A Comparison of Two Parallel Computer Architectures in the Context of Interferometric Fringe Analysis.
Ph.D. Thesis Liverpool John Moores University, **1996.**

2 Kshirsagar S.P.
High Speed Image Processing System Using Parallel DSPs.
Ph.D. Thesis Liverpool John Moores University, **1994.**

3 Texas Instruments
TMS320C4x -User's Guide, Chapter 2
1994.

4 Transtech
C40 Graphics Manual, Chapter 2.
1996.

5 Texas Instruments
TMS320 Floating-Point DSP Assembly Language Tools, Chapter 2
1994.

6 3L
C4x Parallel C , Chapter 4.
Version 2.0.2, **1995.**

7 Texas Instruments.
TMS320 Floating-Point DSP Optimizing C Compiler, Chapter 2
1994.

8 Hyperception.
Block Diagram User's Manual and Function reference, Chapter 1
Version 4.0, **1996.**

9 Hyperception.
C-Code Generator User's Guide, Chapter 2.
Version 2.0, **1996.**

10 Jovian Systems
DSParcer Reference Manual, Chapter 4.
Version 2.0, **1997**

11 Jovian Systems
JIS Reference Manual, Chapter 1.
Version 2.0, **1997**

Chapter 5

General Issues in Parallel Fringe Analysis Techniques

5.1 Introduction

What happens when a parallel solution is applied to a conventionally linear algorithm? What limitations are placed on the implementation of a parallel solution by the hardware, software and indeed the paradigm itself? Which problems need to be investigated and why?

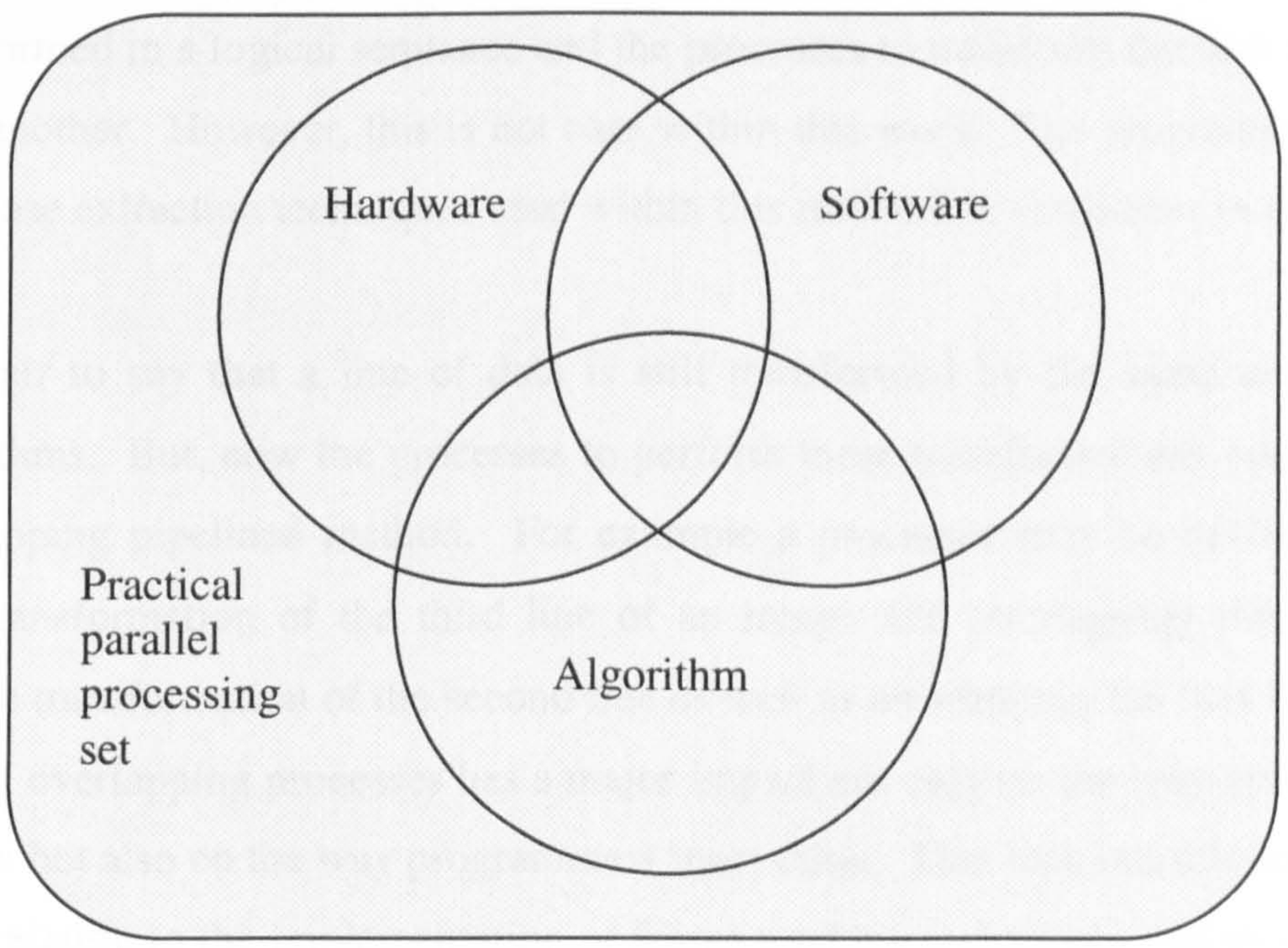


Fig.5.1 Interaction of factors which effect the final parallel system

These are just a few of the questions that any developer of a parallel system must answer. Using the fringe analysis techniques outlined previously these issues are explored. A number of examples are used to highlight specific points. These examples are taken from three systems that have been used in this work, one is based in the 3L DOS environment whilst the other two are based within the Pegasus environment.

A number of the issues that arise from implementing parallel versions of fringe analysis techniques are of a general nature, it is these topics which are discussed within this chapter. The first area covered in this chapter relates to the C40 processor used to perform the parallel processing. Following on from this, general data issues are investigated. These spring from the features of the C40 used and the software environments that this work is performed within. Both of these areas form the basis for the specific issues raised later in this thesis.

5.1.1 Non-linear Programming

Conventionally programmed phase extraction techniques consist of a number of stages that are performed in a purely sequential manner. That is to say the data is transformed in a logical sequence and the processes to transform the data occur one after another. However, this is not case within this work. The programming of all the phase extraction techniques used within this research is non-linear in nature.

It is fair to say that a line of data is still transformed by the same sequence of algorithms. But, now the processes to perform these transformations operate in an overlapping pipelined method. For example a processor may be performing the row transformation of the third line of an image and overlapping this with the inverse transformation of the second line as well as unwrapping the first line. This idea of overlapping processes has a major impact not only on the way programs are written but also on the way programmers must think. This idea impacts upon **every** issue relating to the implementation of fringe tracking techniques within a parallel-processing environment.

5.2 Reasons for Parallelization

Before dealing with the parallel algorithms, it is necessary to re-examine the reasons for performing parallel processing operations. This can be achieved by re-stating the goals of the INFOCUS Optical Sensor:

- **Flexibility**

  This system must be able to measure a number of external body features. Different regions present different problems e.g. measuring the breast area presents a different set of challenges to those posed by measuring the neck and throat region. In order to achieve flexibility a number of measurement techniques may have to be performed at the same time.

- **Reliability**

  The system must make a measurement; therefore some form of robustness is necessary. It is in fact crucial that the system deliver a measurement, even if this of very low resolution.

- **Speed**

  The system must achieve at least one measurement per second. It is again essential that a measurement be performed, if the system is to be of any practical use this measurement must be performed at least once every second.

It is fair to say that these factors all sum up to point towards parallelization as a possible solution to the challenges posed by the INFOCUS Sensor. Traditionally in parallel processing the overriding aim is speed. Although this is still of primary concern to the final system the other aims are clearly important. In effect the parallel approach is seen as a method of implementing a certain amount of redundancy. It must also be kept in mind that the system must meet these targets within a user-friendly environment.

5.3. Key Effects of the Processing Hardware

In fig.5.1 those areas that have a major effect on the final system are outlined. The first factor to be examined is that of hardware. Previously those hardware features of the C40 that are of importance within this work have been briefly described. Now the effects of these factors with specific relevance to the fringe analysis system are examined.

5.3.1 On-board Ram block

The on-board ram blocks of the C40 can be used to speed-up the processes implemented within the final system. Each C40 processor has two on-board ram blocks; these are each 4K bytes in size. This means that each of these blocks may contain up to 1024 floating point variables. By examining fig 5.2 the issues that are raised by this limitation in size can be explored.

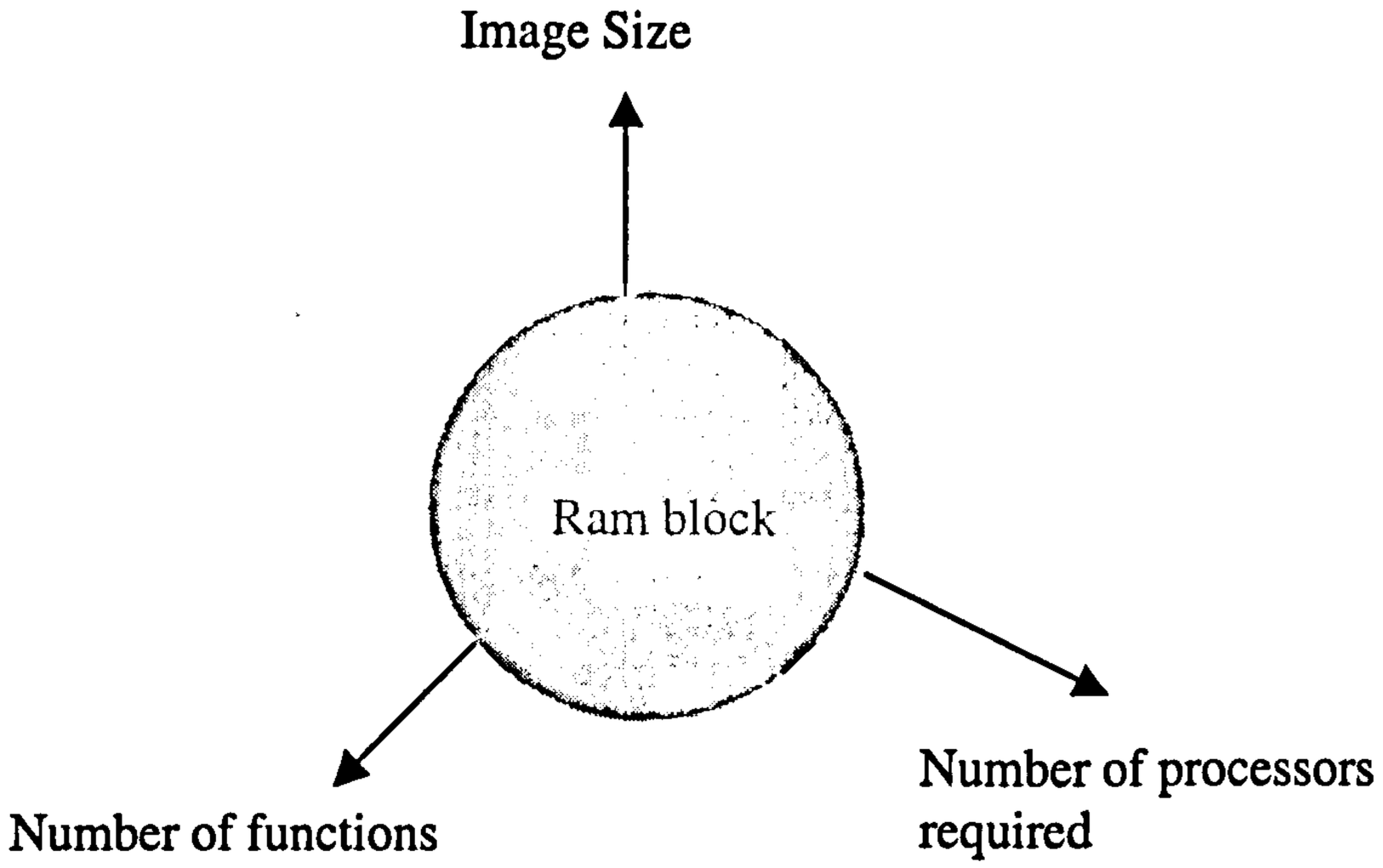


Fig.5.2 Influence of the on-board ram blocks

The limitation in size of the data that can be manipulated by the ram blocks effectively limits the size of the image. The use of complex data that is required within both of the fringe analysis techniques limits the possible image row size to

1024 "complex pairs" which would be 8Kbytes in size. It is worth noting at this stage that the complex format of passing data in the PSP algorithm is used to simplify the data handling problem. However working with this size is actually undesirable due to the nature of the physical ram blocks i.e. the separation into two blocks of ram is a physical separation. This implies that to actually perform an FFT of 1024 data points which requires the data in one contiguous form is not truly beneficial, due to this separation.

In order to avoid this problem of non-contiguous data the practical system is limited to 512 complex data pairs. This means that a whole row of a typical digital image can be fitted into one block of contiguous memory.

If the system hardware network to be used does not exist but must be commissioned, knowledge of how to effectively use these on-board ram blocks is very important for a number of reasons. Firstly a number of functions will be greatly speeded up by using this faster memory space. Secondly knowing how to effectively use this memory will dictate much of the flow of the data throughout the system.

These factors mean that the number of functions that can utilise this feature can be determined for each processor with a given image size. An example of this is that for an image of 512 by 512 in size only two processing functions requiring complex format data can be performed within the ram block space available within each of the C40s. Whilst if these images are restricted to 256 by 256 in size four such processes can use the ram blocks, it must be kept in mind that there will be some overlapping of processing of functions, which means that the functions must **not** use overlapping memory space.

This effect changes the manner in which problems are coded when compared to linear solutions. Every function that requires ram block memory within each C40 must have this allocated so that no two functions use the same or overlapping memory. If data does overlap the results of the final parallel system will be corrupted. It is therefore very important for the programmer to know exactly which functions use exactly which segments of the internal ram on each C40.

5.3.2 DMA co-processor

The DMA co-processor can be used to speed up the performance of a C40 system by moving data between the various stages of any paradigm. This DMA can operate independently of the CPU, allowing the CPU to perform calculations on one set of data whilst the DMA moves the data. Additionally the DMA may index the data in a number of methods using any indices with no overhead.

By using this facility hardware device communication and calculation can be overlapped. Within the 3L environment using this feature helped the practical performance goals of the system to be achieved. This idea of overlapping communications and calculation can be highlighted by the following example of the 3L system shown in fig.5.3.

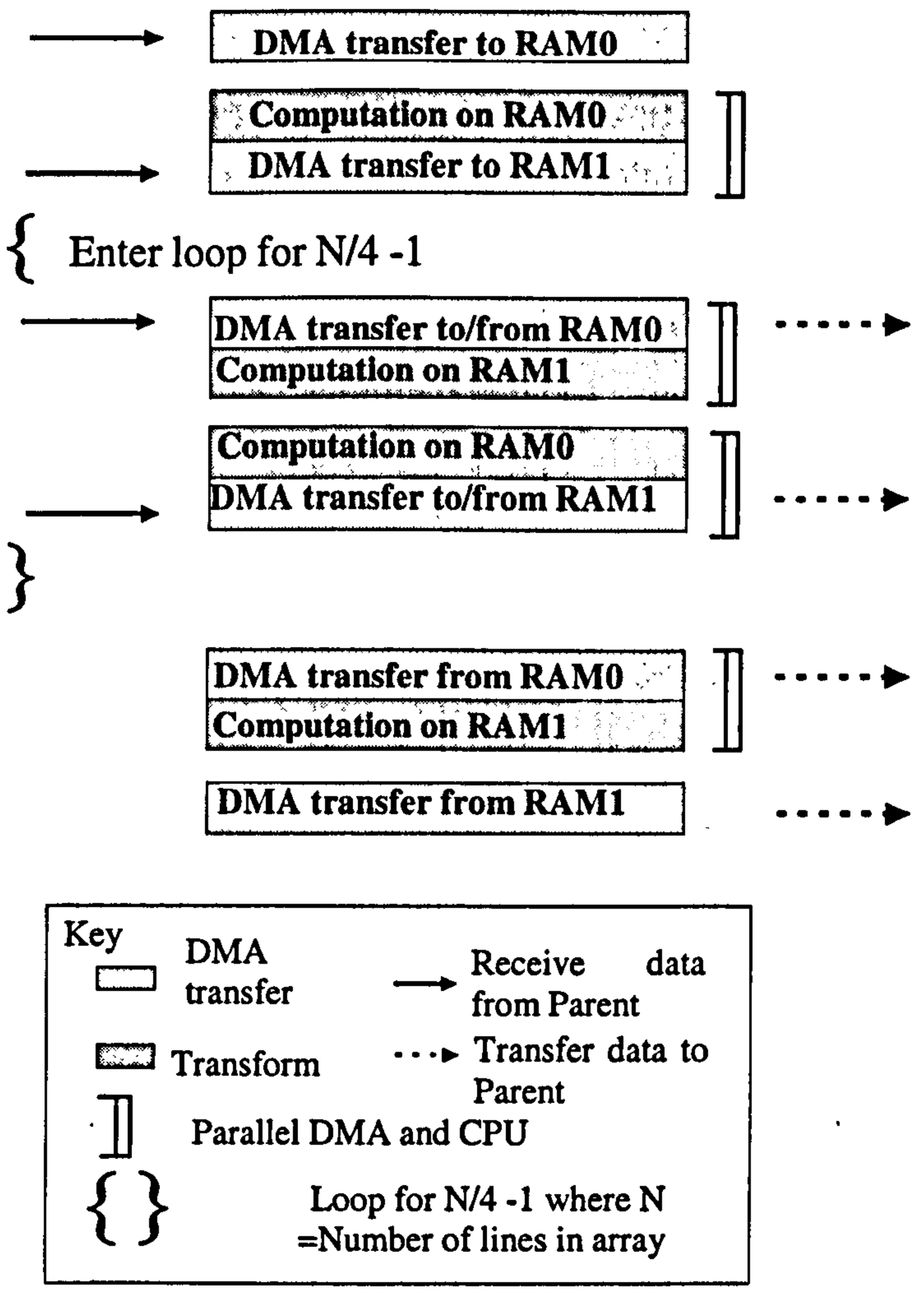


Fig.5.3 Parallel Communication and computation, showing how data flow is optimised by using the DMA coprocessor

This scheme is not implemented in the Pegasus environment due to the data partitioning scheme used; this is described later in this chapter.

5.3.3 Communication Ports (Comports)

Each C40 is designed with 6 communication ports that can be used for inter-processor communications. In a practical system, the housing of the C40s on the TIM-40 compatible motherboards will effect the final design of any software. These physical limitations are outlined in Chapter 4.2.2. The restrictions on connections greatly effects the final topology designed, which impinges upon the data partitioning and movement strategies that can be used within the hardware and software environments.

An additional feature of these comports is that they may be used as bi-directional communication links. Using the bi-directional nature of these comports means there may be some information loss due to interference of data flowing in opposite directions. This feature will decrease the size of information that can be communicated and increase the resulting communication times required.

A further feature of the communication links is that they can be used as either physical or virtual links. A physical link corresponds to the actual communication link in the hardware. Using physical links means the entire software system designed will be dictated by the available communication links. Such systems can be very fast at communicating data but rely heavily on the programmer's ability to optimise data flow and avoid any communication conflicts. This idea was used in the 3L system.

In the Pegasus environment virtual links are used. These do not rely on specific communication links but rather are connections between tasks that exist on various processors. If the overall system is defined as a block diagram scheme then each of the blocks within this scheme represents a task in the parallel system e.g. the forward FFT or the arc-tangent function. The C40 software protocol must decide on how to communicate data between tasks and avoid conflicts within this data

movement. This places the burden of connecting tasks via links upon the C40 through the software environment but it is much simpler for the programmer to use.

A further point raised with communication links is that although a C40 has 6 possible links only 4 such links may operate effectively at anyone time. This means that, although a processor can communicate with 6 other processors where the physical links exist, communicating to 4 processors is the optimum solution.

5.3.4 Topology

The previous sections have detailed hardware aspects over which the programmer has no control, although the choices of which features of the hardware to use are available to the programmer. However, the topology of the system designed is solely in the hands of the programmer. Some of the factors that influence this topological design can be seen in fig 5.4.

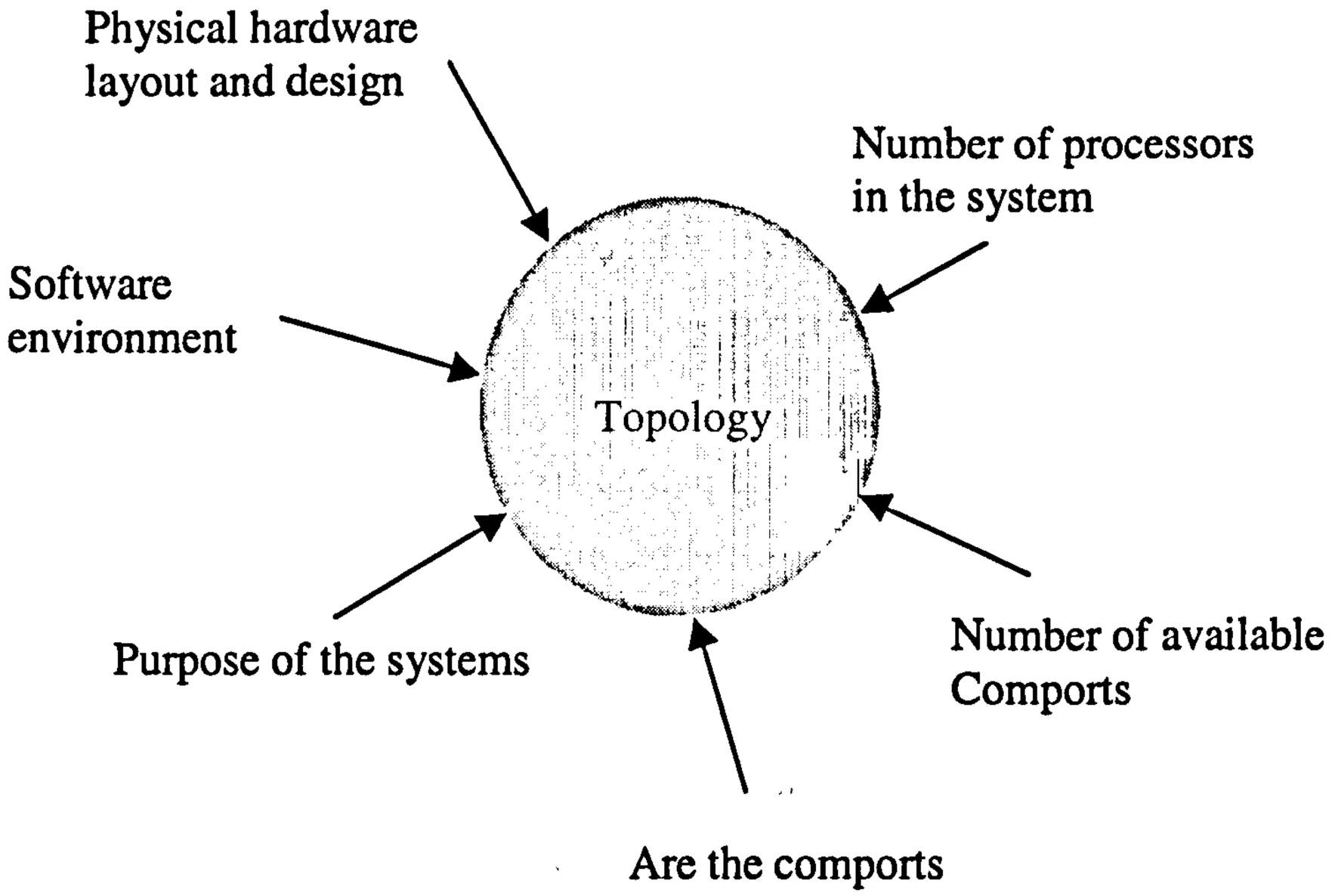


Fig.5.4 Factors influencing the practical topology designed

The final topology designed must keep the goals of the INFOCUS Sensor in mind, as stated in section 5.2. What the topology aims to allow is the concurrent

implementation of a number of techniques, which may be rapidly tested and prototyped. Keeping this goal in mind the topology designed should allow the implementation of optimised, efficient software. Further if optimum performance is to be achieved the topology must allow the system to easily re-route data so that the C40 can achieve an optimised workload.

The first key factor that affects the practical topology is the number of processors within the system. In the network for INFOCUS there are 10 processors on 3 motherboards, where one of these processors is a Framestore. This means that there are limited numbers of practical set-ups, which may be achieved.

This leads to the next factor, that of how many comports are available within the network. The physical layout of the system means that there are not enough free comports to achieve a fully connected system. It can be seen in section 4.2.2 that some of the comports are not available due to the physical layout of the housing of the C40s.

In the C40 hardware 3 of the available comports on each processor are designated as master comports while the remaining 3 ports are designated as slave ports. Those ports defined as masters will on restarting the hardware have the token required for communication while the slave processors will not have this token. In order for this token passing strategy to work master ports may only be connected to Slave ports and vice versa. This of course effects the final topology designed by impinging upon how ports may be connected.

Whether or not comports are to be used with physical or virtual links will also have some effect on the design of the topology. If purely physical connections are to be used then the system must be designed around the final tasks that are implemented. Such a system cannot be as flexible as a system designed for prototyping, and will not aid the evolution of code.

A software environment is defined in this work as the environment under which parallel code runs in this case either 3L or Pegasus. Using a system which implements virtual links means that some code which in a physical link based

system can only run on the root processor may be run on processors other than the root. This means that the software environment performs some virtual linking which is completely hidden to the programmer. In addition the software environment using any number of physical links may make virtual links between tasks performed upon different processors. In order to allow the system to optimise speed and avoid data clashes a system with more communication links than is required by a software environment using physical links only is implemented in this work.

The software design environment will have a number of effects on the topology created. Perhaps the most important of these is which of the processors is to be used as the root processor i.e. the only processor which may communicate with the host PC. In the initial experimentation, within the 3L environment, the Framestore processor was used as the root processor. The reasoning behind this choice is that it is very hard to display data on the host under this environment. Therefore the Framestore had to be able to perform both the capture and display of the image data.

However, this argument was abandoned in later work for a number of reasons:

- The Framestore has the slowest clock speed of any of the C40 modules. This in effect makes this unit a potential bottleneck, when forced to communicate between Host and child processors.
- Within the Pegasus system the display of data may be performed by the Windows environment. This releases the Framestore from needing to act as the central processor in display functions.
- Functions that require the host, such as print may be performed on any of the child processors in the Pegasus environment. This makes the storing of information on the host PC much simpler. Although it should be noted all such commands must still be routed through the root processor; the environment performs this routing function rather than the programmer.

- To improve the overall performance of the final system as little work as possible is placed on the Framestore. Effectively the Framestore in the final Pegasus system merely captures images and then pumps this data out.

The purpose of the system must also be kept in mind, section 5.2. Knowledge of how the fringe analysis techniques flow is essential to the design of the final topology.

Rather than discuss all the steps in the evolution of the final topology only the final topology used will be examined. Within this topology two distinct layers of processors are visible. Firstly there is a pyramid structure, where the root processor is acting as the parent node. This root processor is connected to all of the processors within this layer, layer 1, as well as being connected to the host processor, fig5.5. This means that only one of the 6 available comports on each C40 is left to join the root with any processors from layer 2.

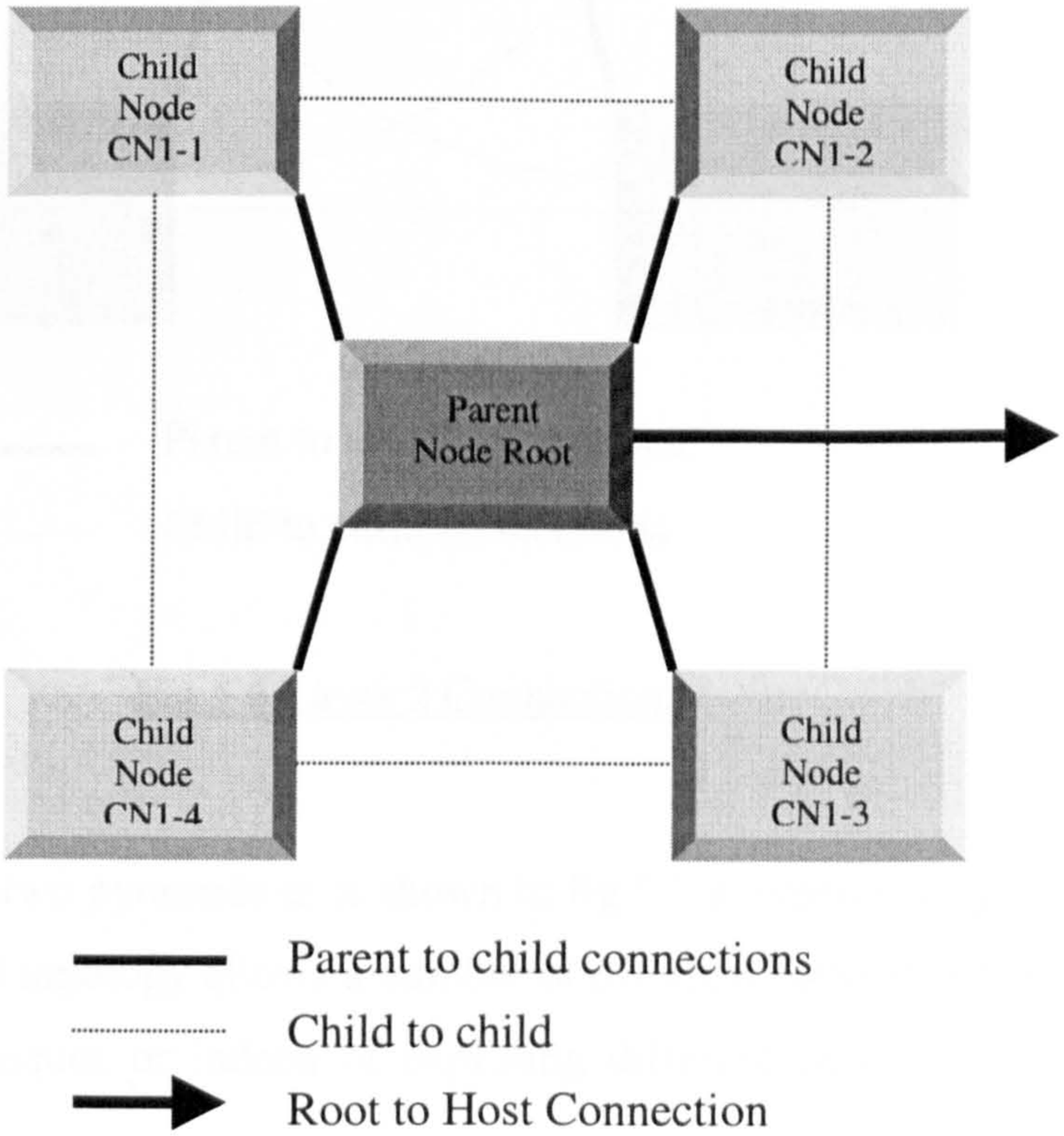


Fig.5.5 Layer 1 Connections

The products used within this research restrict the child processor connections within layer 1. However those child processors used in layer 2 **are** fully connected with each other in a mesh structure. This difference means that those processors in layer 2 have a slightly more flexible aspect and virtual routing in this layer is much less of a burden. The parent in layer 2 is the Framestore which is placed on top of the mesh of children to form a fully interconnected pyramid structure, fig.5.6.

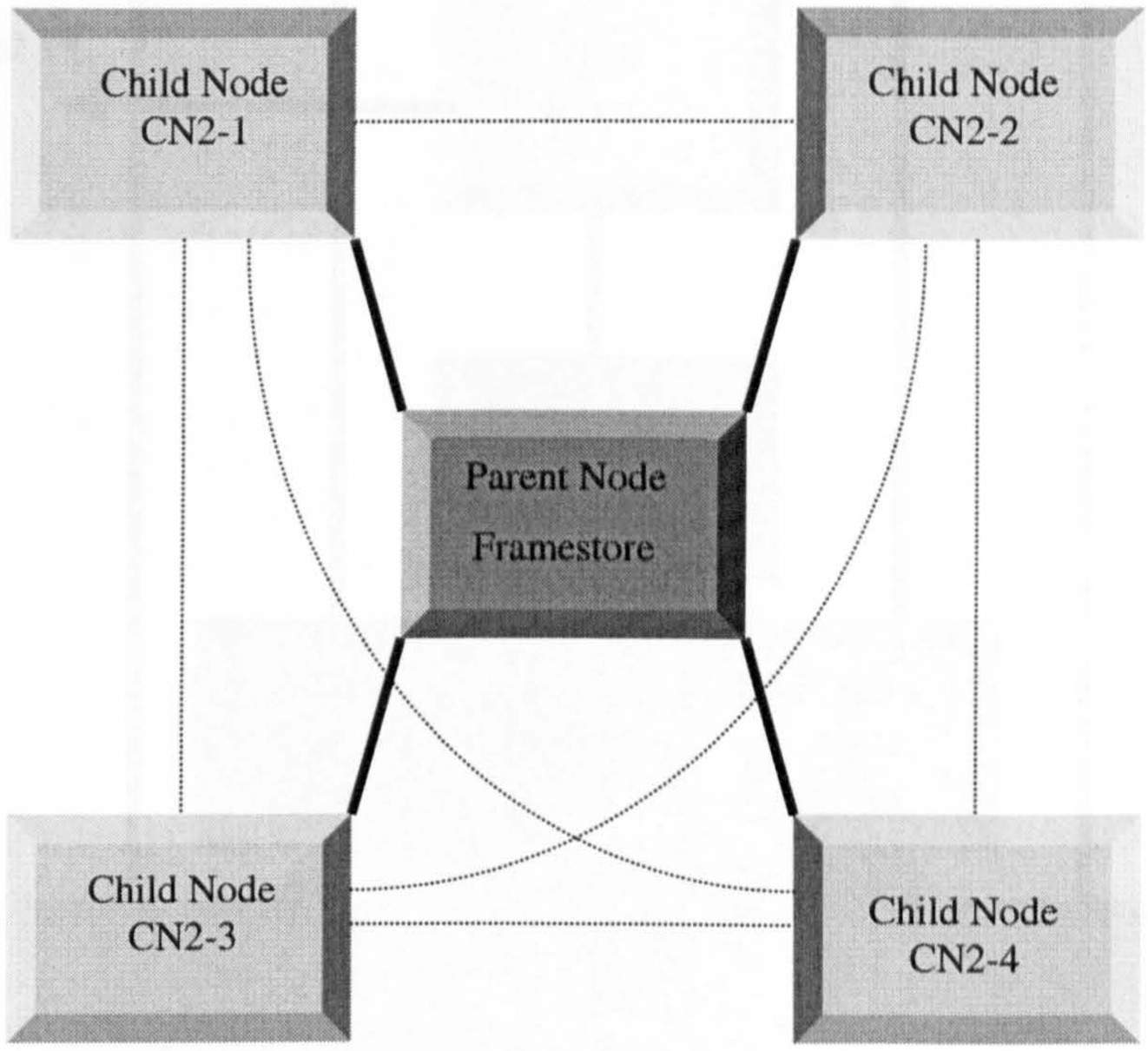


Fig.5.6 Layer 2 Connections

By connecting these two pyramids as is shown in fig 5.7 a hypercube type structure is formed. This final topology allows a number of different methods of performing the concurrent techniques or indeed of exploring different permutations of each technique.

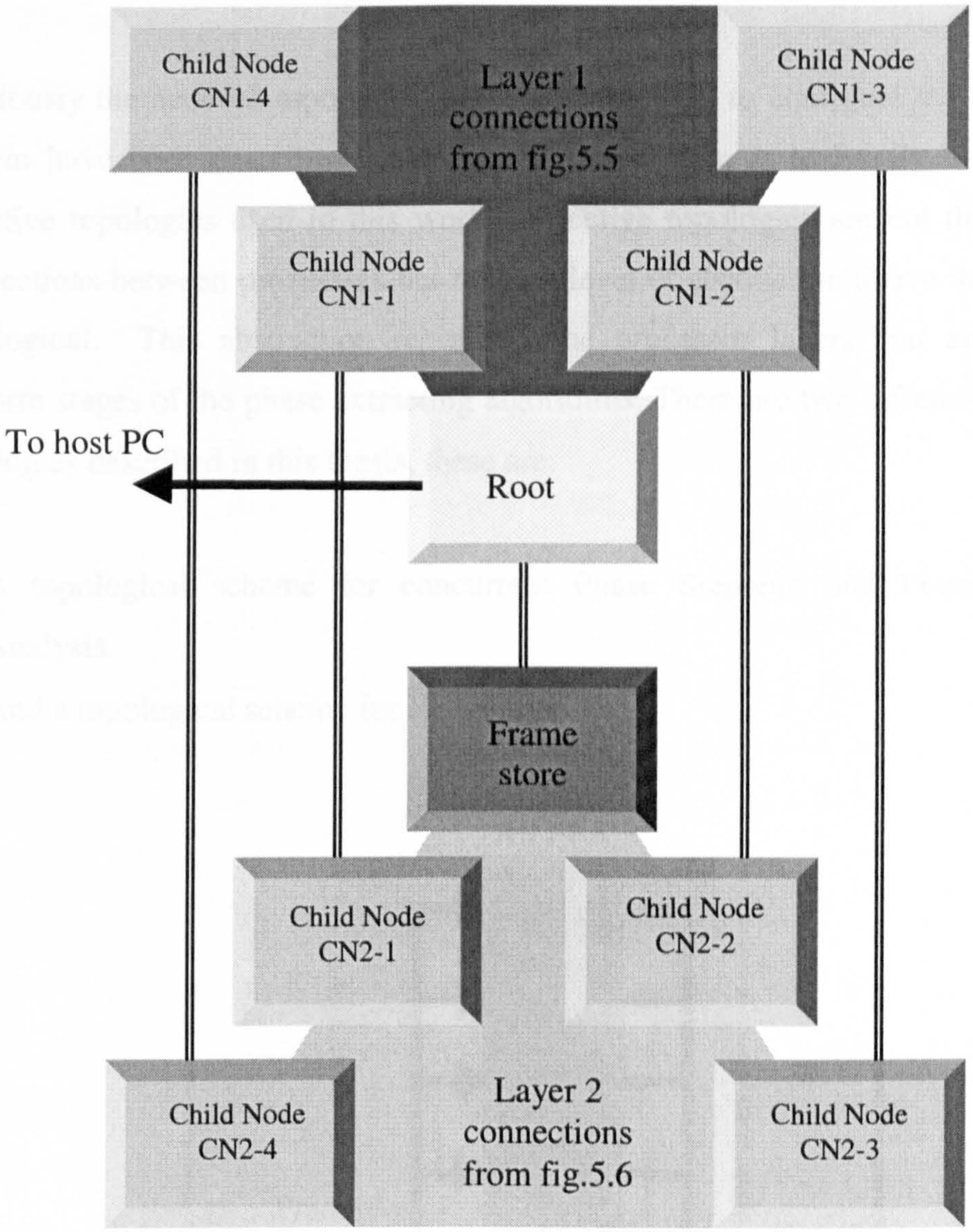


Fig.5.7 Abstract diagram of final Network topology designed

In the final topology the parent processors of the two layers are connected together, while child processors of layer 1 are connected to the corresponding child processors on layer 2. The triangles in fig.5.7 represent the layer connections of figs5.5 and 5.6. There are a number of reasons for connecting as many processors as possible within this topology. Firstly it allows for any possible comport failure, by using the virtual connection scheme. Secondly it allows a number of techniques and methodologies to be implemented.

5.3.5 Effective topologies

Previously the network topologies that have been used to configure the INFOCUS system have been described. The aim of this section is to briefly describe the effective topologies used in this work. Effective topologies are not the physical connections between processors but rather a level of abstraction above the physical topological. This abstraction represents the processor layers that are used to perform stages of the phase extracting algorithms. There are two different effective topologies described in this thesis, these are:

- A topological scheme for concurrent Phase Stepping and Fourier Fringe Analysis.
- And a topological scheme for FFA only.

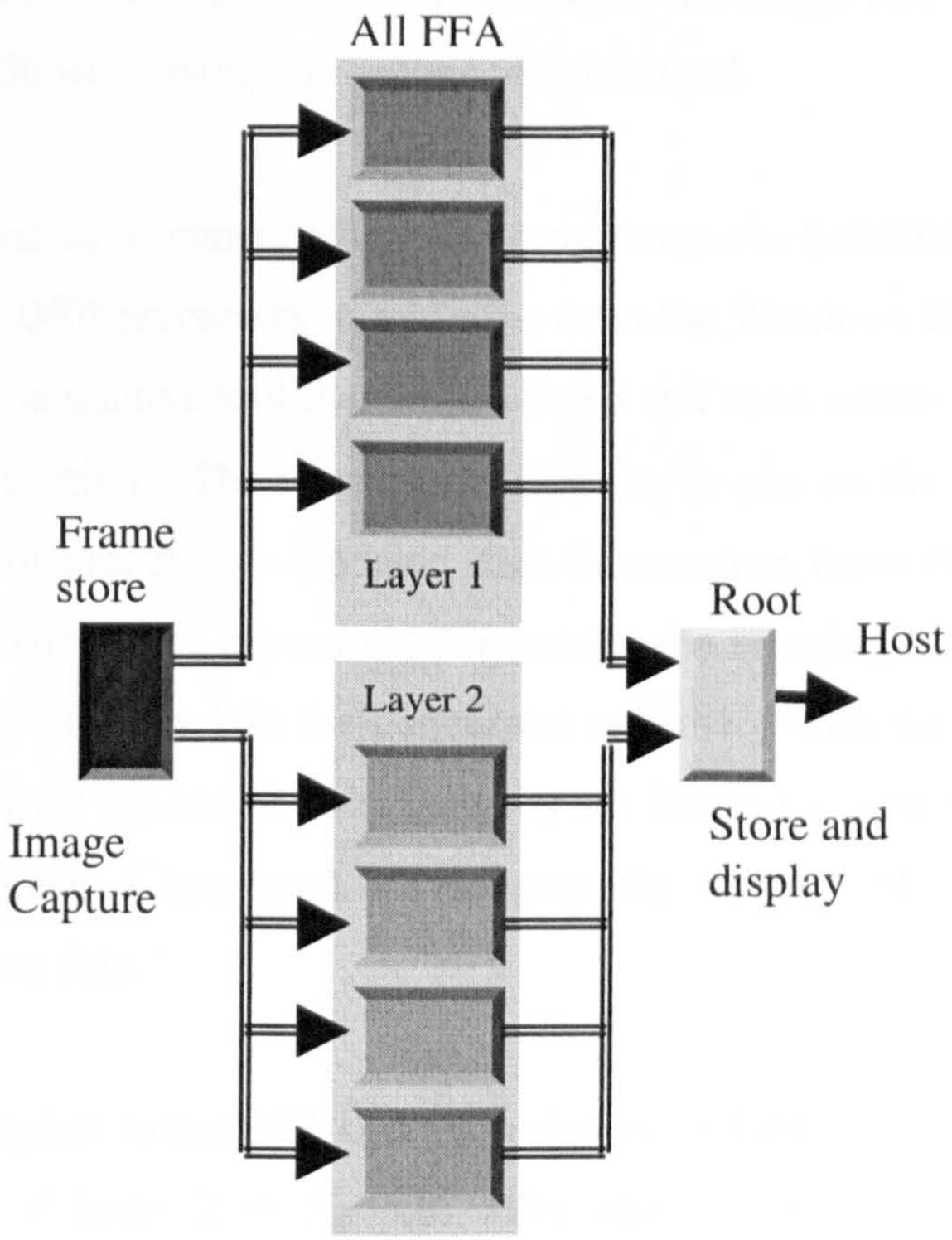


Fig.5.8 Topological scheme for both FFA and PSP under the Pegasus environment.

The first topology uses all ten processors to perform both Phase Extraction paradigms, fig5.8. The Framestore is used to capture data, and as well as this any pre-processing that is required **may** be performed here. An example of this is the removal of the background intensity value within the FFA.

All the stages of the FFA system are performed on the 4 child processors of layer 1. This allows a maximum line size of 512 data points to be used as long as the arctangent and unwrapper are not performed within the on-board ram blocks. If these are to be performed on the on-board ram blocks then the line size must be restricted to 256 data points. Deciding which of these strategies to follow has a major impact upon the system, which has been touched upon in section 5.3.1. Firstly there is the speed-up that can be achieved from using the ram blocks when compared to the local memory. This arises from the C40 code, which is optimised to run within these ram blocks. The next impact is upon the image size that must be restricted to 256 by 256 when using the second idea outlined.

Working in the Pegasus environment allows the programmer to build DLLs that can be used to control the DSP processors in real-time from the Windows Environment. One major Windows interactive tool that is developed and used within this work is a controllable filter function. The user control for this is run on the Host which interacts with the Root processor. Updated data information from the Windows DLL is sent from the Root to the processors performing the Filtering function. This is the child processors of layer 1 in the concurrent scheme. With the filter being performed upon this layer the actual links between the Root processor and the four child processors of layer 1 are used. This cuts the virtual linking task and associated time to move data.

The size restriction applies to the PSP algorithm that is performed concurrently on the child processors of layer 2 in fig 5.8. The reason for using the layer 2 processors to perform PSP is that more data must be transferred in this system than the FFA. Therefore those processors which are connected to the Framestore are used, cutting potential communication delays.

The next effective topology is based around a system that only performs Fourier Fringe Analysis, fig.5.9. This is because FFA has proven to be very good at performing surface measurement within the bounds of the INFOCUS project. Additionally FFA presents many more problems in parallel processing and therefore is of more importance to the question of whether to perform fringe analysis in a parallel environment.

In the topology of fig.5.9 those processors that are connected to the Framestore are used to perform the forward and inverse Fourier Transforms. While the layer 1 child processors are used to perform the arc-tangent and unwrapping functions. These processors do not need to be as interconnected as those processors performing the FFTs because of the neighbourhood nature of the arc-tangent and unwrapping. This means that a line size of 512 data points may be used in all the tasks that require the ram blocks, within a simple FFA system as defined in section 6.1.1.

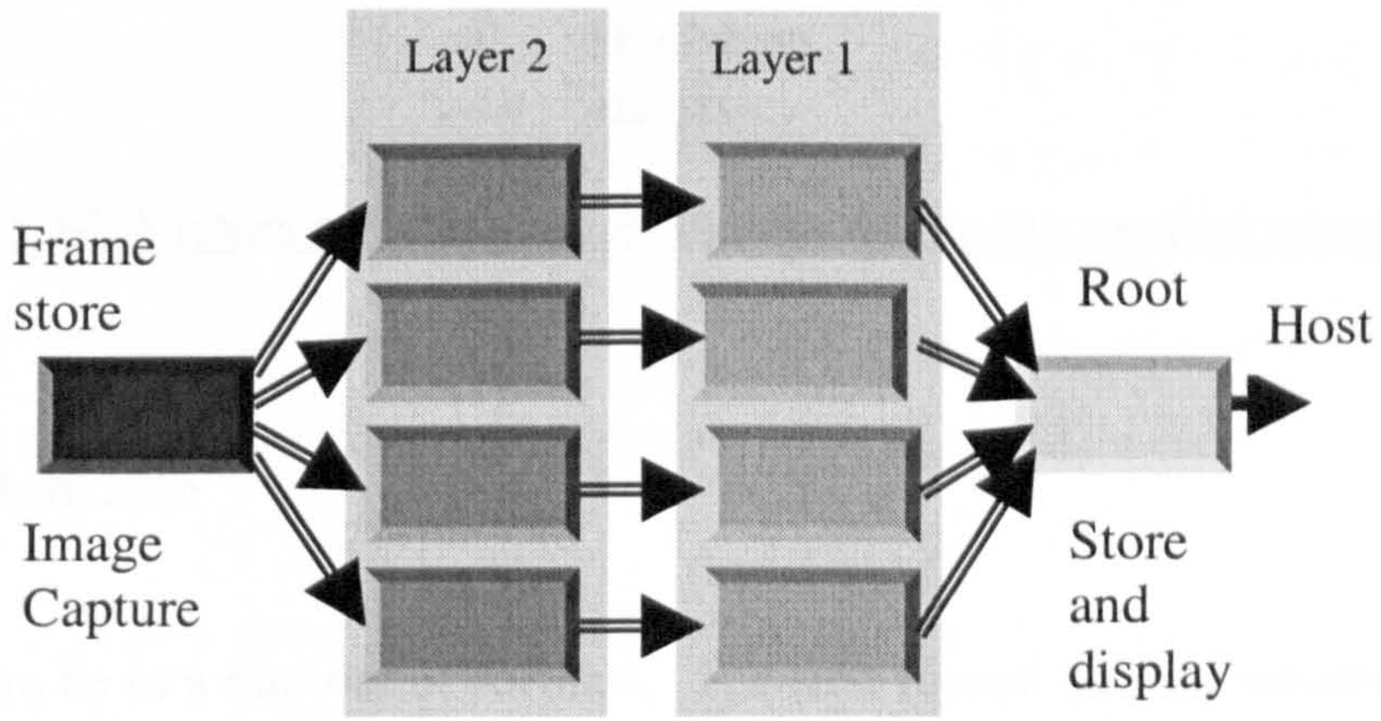


Fig.5.9 Topological scheme for FFA only under the Pegasus environment.

5.4 Data

A number of general issues are raised by the ideas used to manipulate data, fig 5.10. The interaction between these various features is somewhat complex and it is sometimes difficult to separate issues in to different categories.

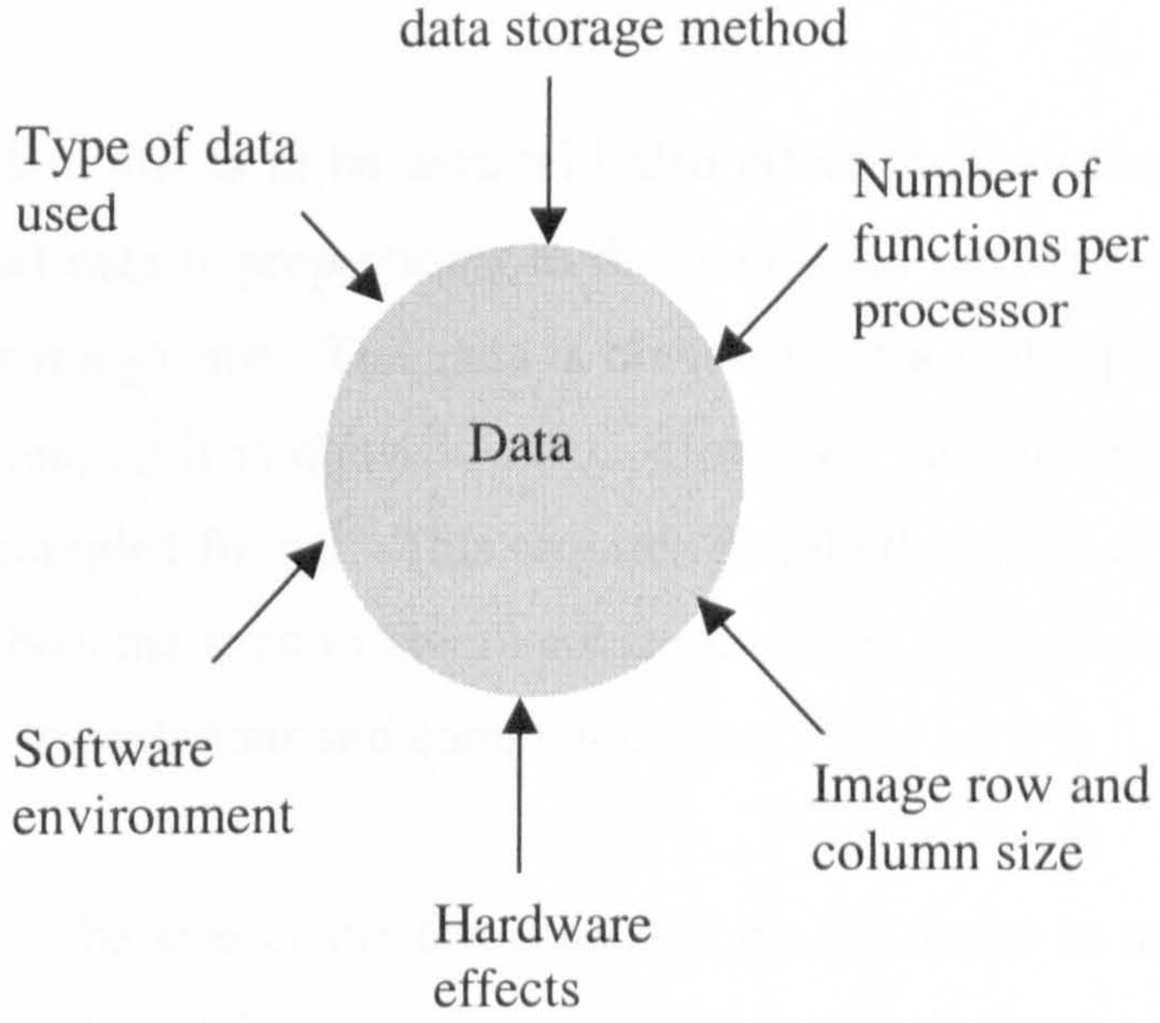


Fig.5.10. Factors affecting issues in data within the parallel processing system

5.4.1 Data Size

Data can be in a number of formats. The first format issue is whether to use packed or unpacked data values. Packed data will take up a quarter of the space and therefore time to communicate as unpacked data. However, this data must be unpacked to be of any use. This means that there is some pay off between the time to unpack data and communicate data in an unpacked format. Within the 3L environment packed data was transferred from the Framestore where it was captured. Within the Pegasus environment packed data was not used at this stage of the system. Packed data was used to send data from the C40 root processor to the host PC for display in the Windows environment.

Within the 3L system the unpacking routine was used to allow the DMA processor to perform data movement. This allowed the DMA to bit-reverse the data, freeing the CPU from this task within the FFT function. Although the C40 may perform bit-reversed addressing with the FFT functions used in this stage, if this option is chosen the performance will be effected in a negative fashion. However, using virtual channels within the Pegasus environment places some restrictions on how the DMA co-processor can be used.

The type of data that is to be used will also affect the way the system runs. The initial captured data is proportional to the grey level intensity value of each pixel within the set image size. This data is obviously of a real type; i.e. the imaginary value is ignored, as it is equal to zero. However, data in the Fourier domain is usually of a complex format. This requires double the space of real-only data and therefore doubles the time to communicate the data. Wherever possible real only data is used in calculations and communications.

Restrictions on the size of the data imposed by the desire to use the internal ram blocks have previously been mentioned. This means that any image captured must be 2^N by 2^N in size, e.g. 256 pixels by 256 pixels. Using this size not only allows the ram blocks to be used to improve the code performance, but also allows the use of optimised FFT routines which require the data to be of 2^N in size.

A C40 processor has a number of possible storage formats for data such as floating-point, integer and fixed-point numbers. The data is initially in integer format when captured by the Framestore. However to be used by the functions written in both the 3L and Pegasus environments the data must be converted to floating-point format. In the Pegasus environment when the data is written to the host PC integer format **may** be used to save space and time.

5.4.2 Data Partitioning

For parallel processing to be performed, the initial image data must be divided in some scheme to allow for a number of processors to work upon the image. For this

reason there are a number of data partitioning schemes described in the following section.

Any image of the size 2^N by 2^N can easily be partitioned into smaller sub-arrays of data. For example it is easy to see how an image of 256 by 256 in size can be split into 4 smaller arrays; in fact there are a number of methods of performing this division.

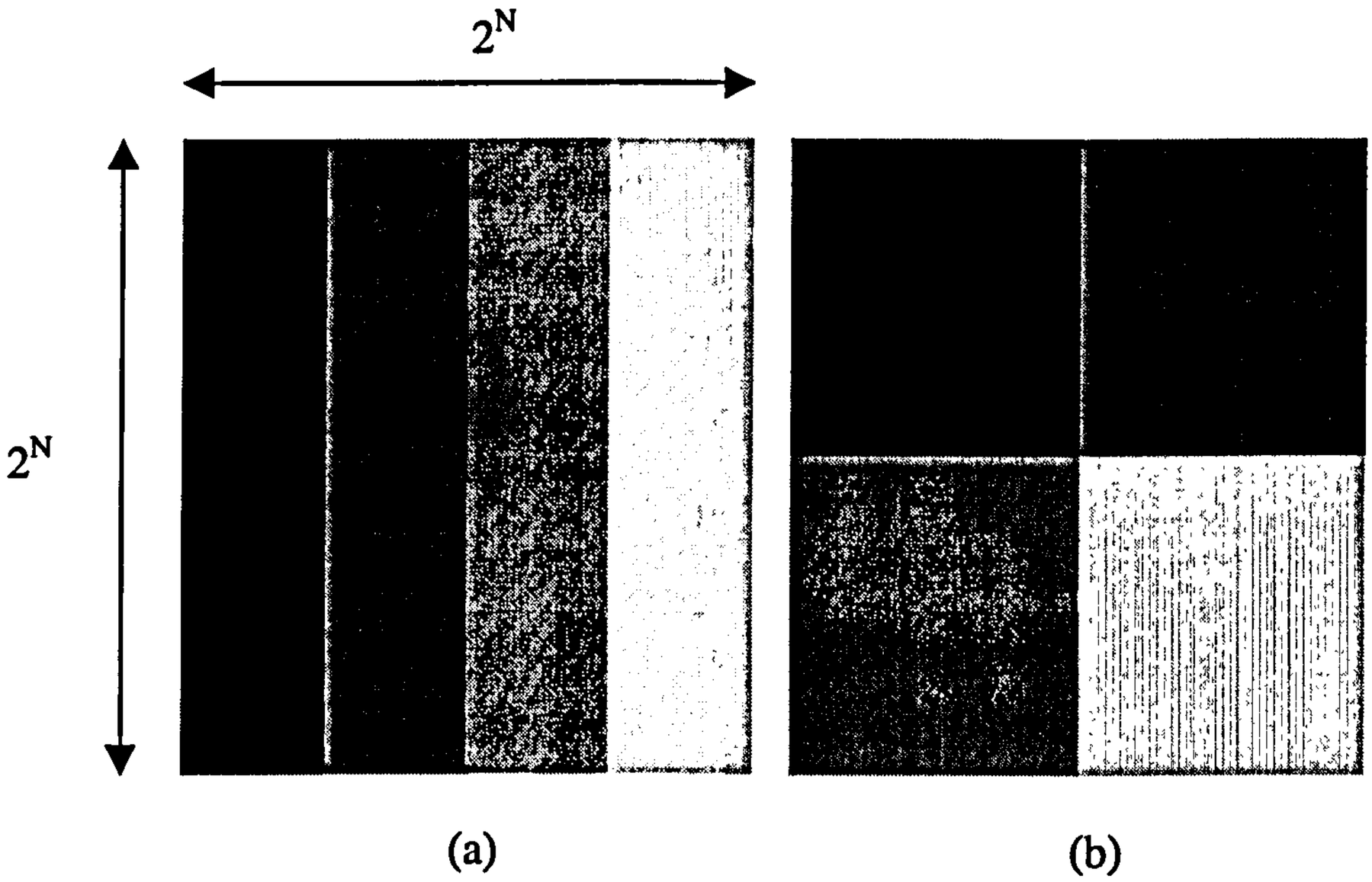


Fig.5.11 Data Partitioning using strip (a) or segment strategy (b)

In fig 5.11 two methods of dividing data are demonstrated. The first strategy is to split the image into strips of 2^N by 2^{N-2} or vice versa. Lines may be transformed in blocks as shown in fig 5.11(a) or line by line. The line by line method has the advantage of not requiring complicated pointer calculations. Using this methodology allows the optimised FFTs to be performed using the on-board ram blocks.

The second strategy consists of splitting the image into 4 segments of 2^{N-1} by 2^{N-1} in size, fig 5.11(b). This may well appear the most obvious method of splitting the image data but it cannot be used with the FFT stages of the FFA process. Segmentation can be used with the unwrapping process where it is a much neater

solution than splitting data into strips. However, this complicates the pointer arithmetic required and means that data must be transformed in segments.

Each of the software environments used, 3L or Pegasus employs a different strategy for the division of data. Under the 3L system the data is split into 4 strips for all the Fourier stages of the FFA paradigm, fig.5.12. When the data is required for unwrapping in either of the phase extraction paradigms it is split into segments. Within the 3L environment data is transferred line by line although some overlapping of communication and calculation does occur.

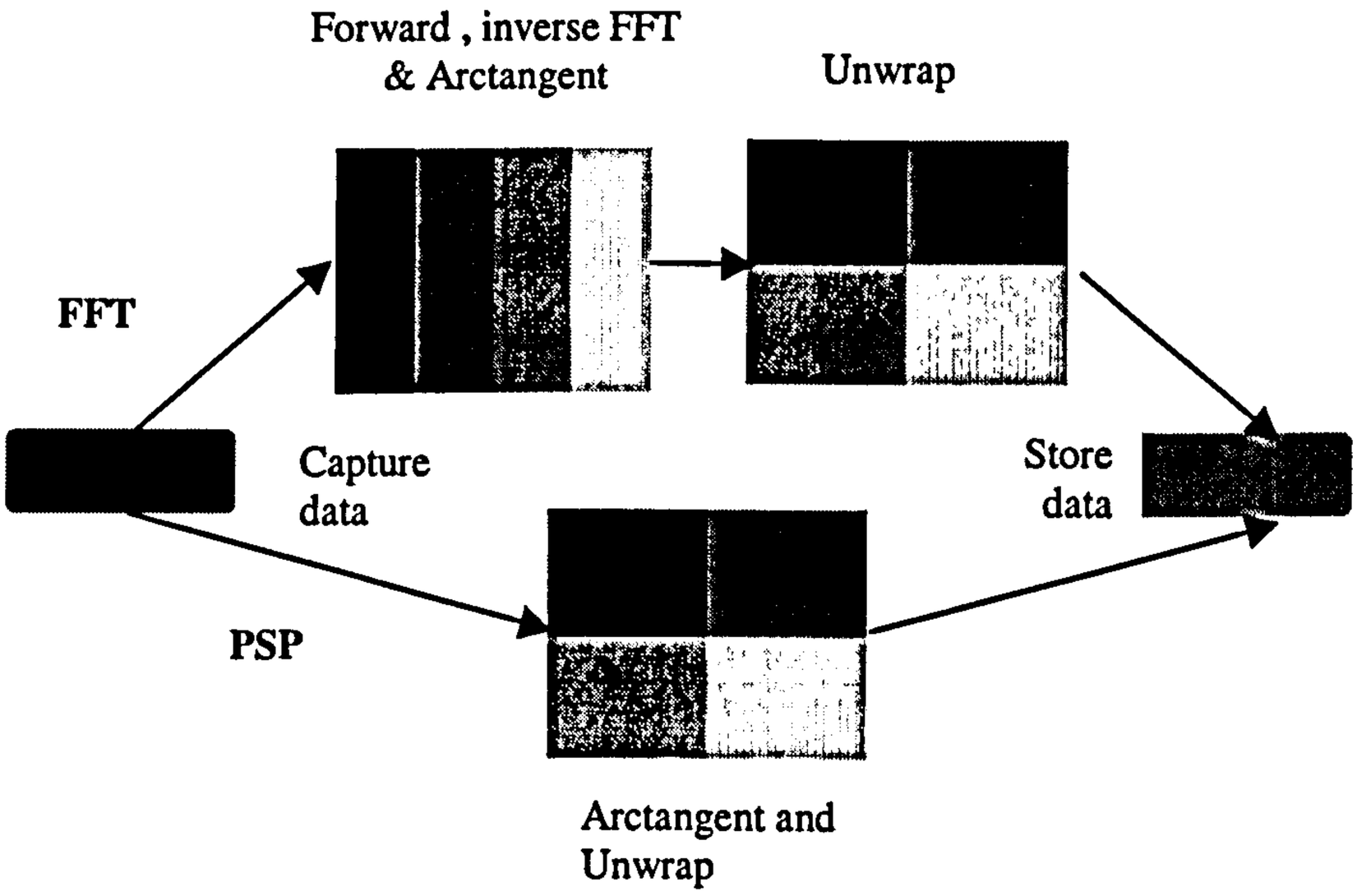


Fig 5.12. Data partitioning stages under the 3L software environment.

Within the Pegasus software environment the data is kept in strips throughout both phase extraction paradigms. However, rather than sending data a line at a time it is communicated in blocks of 8 lines at a time, fig 5.13. This means that each processor must buffer the data sent to it.

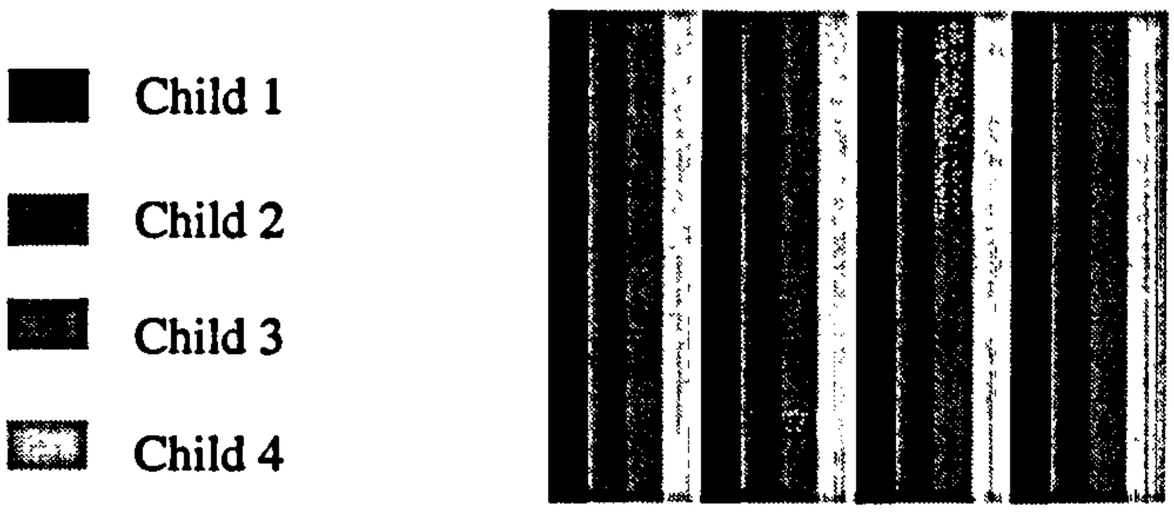


Fig.5.13 Demonstrating the 8 line strip method used in Pegasus environment

5.4.3 Data Storage Strategies

Moving the partitioned data is one of the key strategic issues within any parallel system. Deciding how to move the data depends to large extend upon the granularity of a problem. The type of problem and the sort of processor used in the solution fix this granularity. Generally fringe analysis techniques are of a coarse grained nature, that is they require a small number of processors to perform a large number of calculations.

As it is possible to move data from one C40 to 6 other processors via comports, there are a number of methods by which data can be distributed. If each processor stores some of the image and then communicates part of the transformed data in the manner of fig 5.14, that system can be said to have a distributed data strategy.

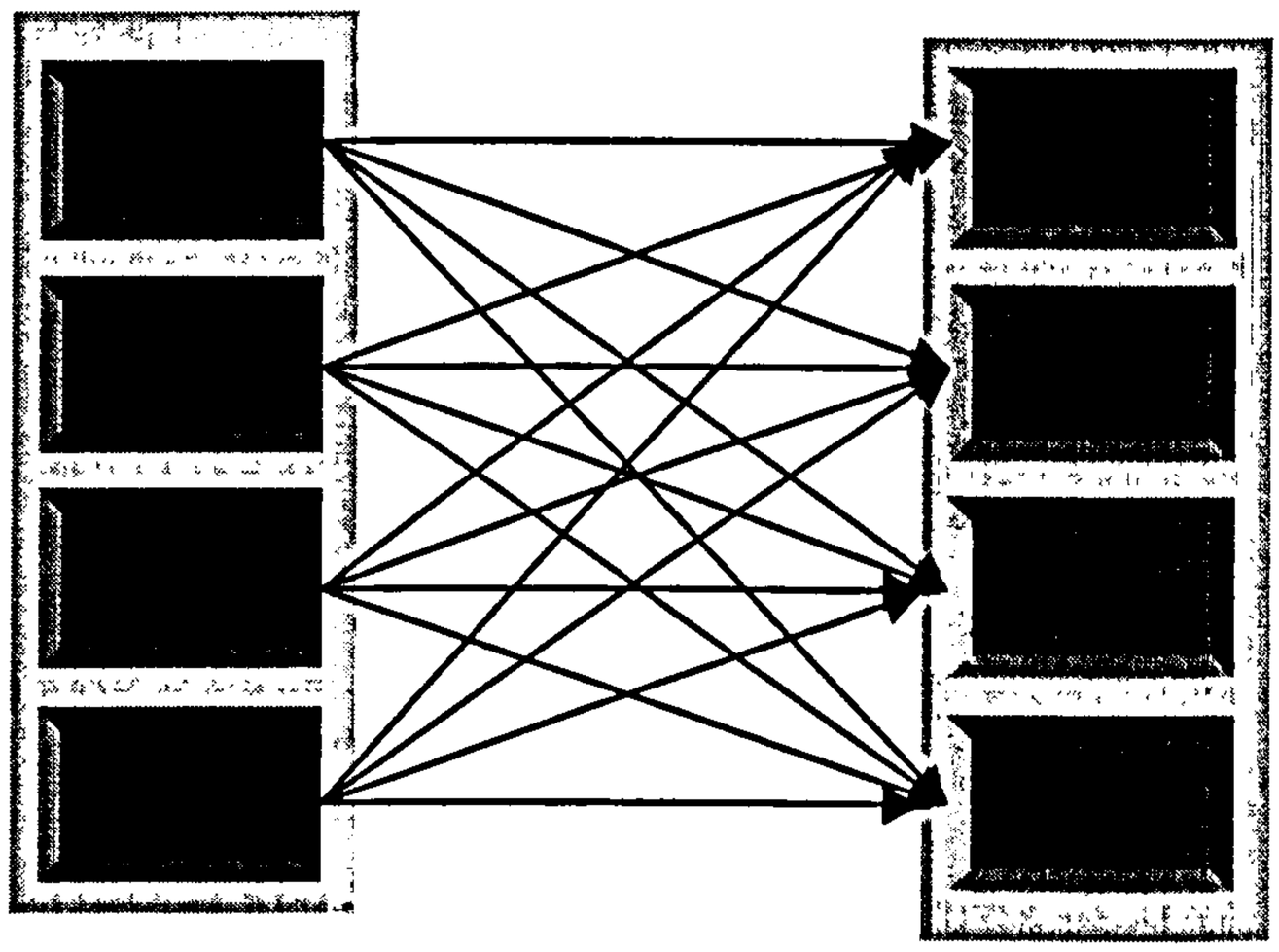


Fig.5.14 Distributed Data Scheme

A distributed strategy requires that the programmer have absolute knowledge about the data network and how data is handled. To avoid overburdening the CPU of the C40 much of the work must be performed by the DMA. This strategy can achieve very fast speeds and is a fine-grained solution.

However, there are a number of drawbacks with this scheme. To be of true benefit the system must use physical communication links and this means that any

topology used must be specially customised for this one solution. The next problem is that any code used must be designed for this one purpose and cannot be written in an object-orientated manner. Another problem may be said to lie solely with the programmer, this is the difficulty of following the conceptual solution and of knowing exactly how the data is being used within this scheme.

The next solution to this data movement problem is a more coarse-grained solution, the so-called centralised scheme fig.5.15. Under this regime the data is sent to individual processors to be transformed and then stored on one hub processor. This allows the data to be more easily translated or rotated, which may be required by a 2D FFA system. Obviously this system does not require as complicated a communication strategy as the previous solution and may not be quite as fast as the previous solution.

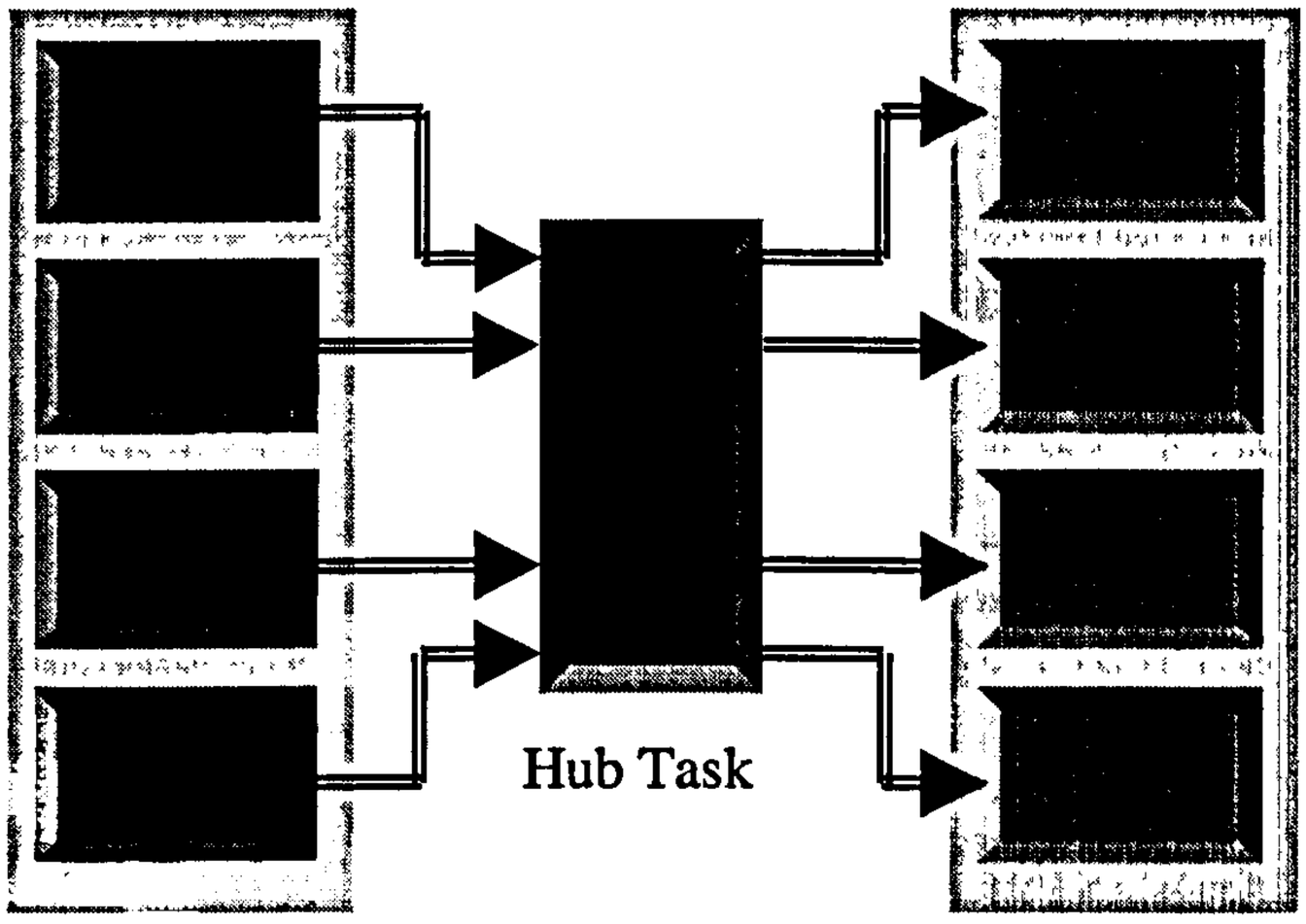


Fig.5.15 Centralised Data Scheme

However, it does have the benefit from the programmer's point of view of being conceptually easier to visualise. As less communication is being performed there is less likelihood of communication errors occurring. Further this system is much easier to adapt to the idea of buffered outputs as is used within the Pegasus environment.

The second centralised strategy has been used through much of this work.

5.5 Summary

This chapter has examined some of the more general issues raised from implementing fringe analysis techniques in a C40 based parallel processing system. The comparative results for many of the issues in this chapter are discussed in chapter 7.

- Internal Ram blocks

  The internal ram blocks of the C40 can be used to speed up an algorithms performance. But using these structures has a major impact on the data size and the way in which code must be written.

- DMA

  The DMA co-processor allows data to be moved at the same time as the CPU performs calculations. Using this feature has major impacts on the way code is written and performed.

- Comports

  Using comports is essential for parallel processing. How the features of the Comports are used has a major impact on the software performance of the system. Using comports as having physical or virtual connections dictates the systems flexibility and ability to meet the performance criteria of the system.

- Topology

  That the network topology of the C40 system is restrained by the components which make up that system. The network topology used in this work is based upon a hypercube structure. This allows the virtual connections used in the Pegasus software environment more adaptability than a system with fewer connections.

- Data Size

  The size of image that can be used is dictated by the C40 hardware that is used. Restrictions that arise when using the internal ram blocks of the C40 are compounded by the FFT code that is used in this research.

- Data Partitioning

  Often in Fringe Analysis segment division of data is the most intuitively obvious method of partitioning an image. However, in parallel processing splitting data into strips is a more frequent approach. The method of division chosen will impact on every algorithm used in this work.

- Data Storage

  There are a number of ways in which data can be stored, in this work distributed and centralised strategies have been investigated. Distributed storage requires comparatively more inter-processor communication and less storage than a centralised strategy. Additionally the coding for a distributed strategy will be less flexible than a more centralised scheme

- Effective topologies

  That a system which is connected as fully as is possible results in different topology ideas being implement for different parallel and concurrent solutions. The effective topologies used are based upon the key features of the C40 utilised in this work and all the data issues discussed.

It is worth noting that all of these factors will force a parallel-processing solution to be completely different from a single processor solution. Further a parallel C40 system will be different than any other parallel chip solution would be. And the particular hardware topology described in this work will make the solution described completely different than any other topology. This all means that this research describes a solution with unique problems and features, but with lessons of a more general nature when it comes to performing non-contact measurement in a parallel-processing environment.

These general issues will dictate many of the more specific issues for the fringe analysis techniques. And it is to these specific issues that this thesis now turns.

Chapter 6

Specific Issues in Parallel Fringe Analysis Techniques

6.1 Introduction

In the previous chapter the impacts from the hardware and some aspects of the software environments were examined. Chapter 5 focused upon the effects of the C40 hardware features upon general data issues. In this chapter these effects are examined for each stage of both fringe analysis techniques.

Whereas the focus of the previous chapter was the hardware this chapter focuses more on the software issues. Of special importance are the effects from each of the parallel development environments used in this work. To achieve this FFA and PSP are discussed in detail within both the 3L and Pegasus software environments. Although this may lead to some repetition this separation of environments is performed for the sake of clarity.

As well as discussing each of the algorithms used for FFA and PSP, parallel unwrapping strategies are discussed in some detail. This investigation highlights the ideas of data partitioning and communication.

6.1.1 Examining FFA and PSP in the Parallel Environment

The phase extraction techniques to be used within this work have previously been discussed but it is now necessary to view these paradigms in the light of a parallel environment.

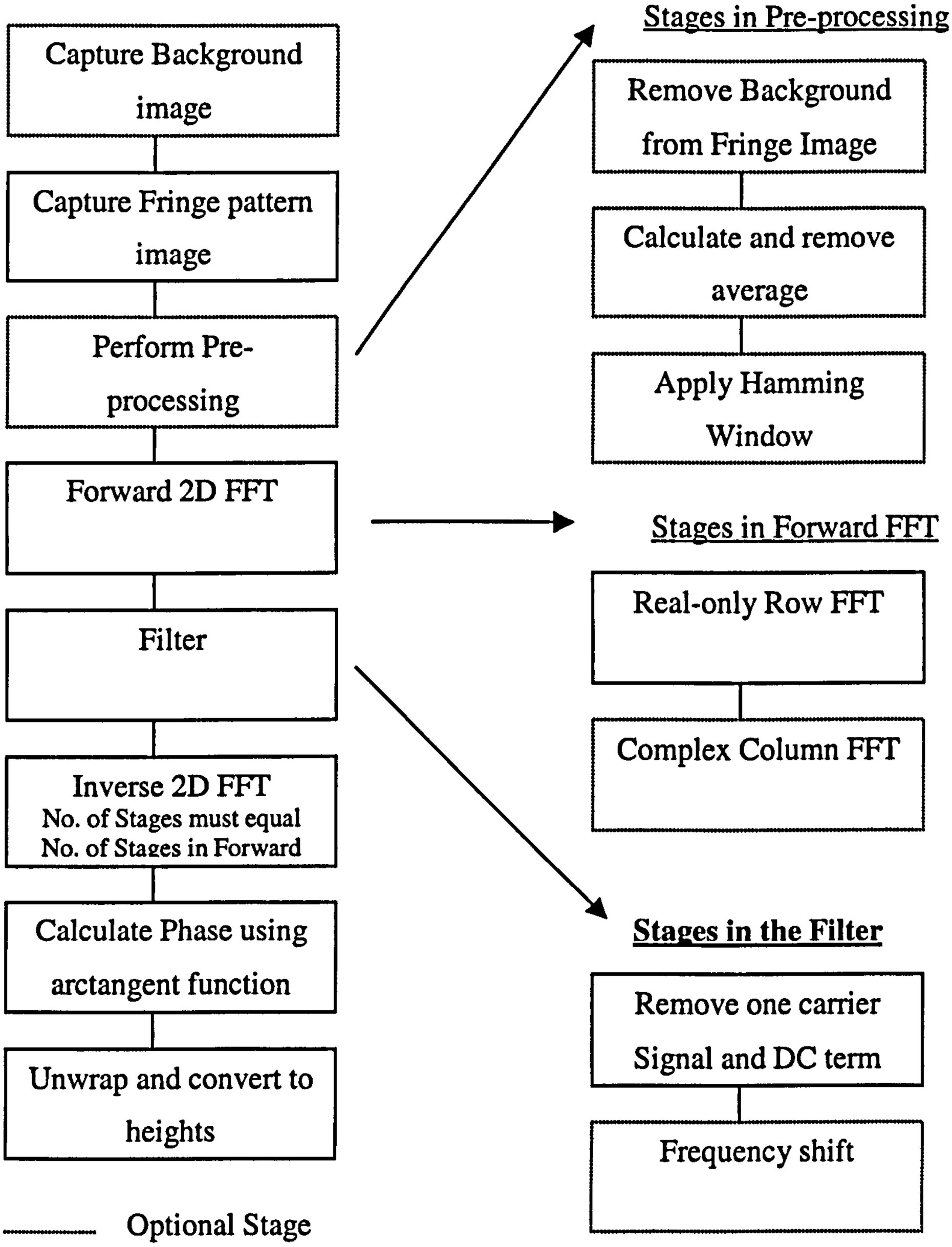


Fig.6.1 Stages in Fourier Fringe Analysis

Fig.6.1 shows the steps both essential and optional to be performed within the FFA paradigm. It can be seen that there are a number of steps in this technique, which

may or may not be performed. A system using all the required stages and the pre-processing stages of fig.6.1 is classified in this work as being scheme 1. Whilst a strategy using all the steps in fig.6.1 is classed as a scheme 2 solution.

The key difference between these two schemes is that scheme 1 is based on a 1D version of the FFA technique, while scheme 2 is a 2D version. This is the key **difference** between the two software environment implementations.

It is worth noting that although FFA is often thought of as one technique, it is in fact a composite paradigm made up from a number of smaller algorithms. This means that there is no single optimum parallel solution, the sequence of algorithms used leads to compromise.

Within the FFA paradigm there are a number of computationally intensive blocks. Much work has previously been done on the optimisation of these algorithms. This means that to a large extent the work that needs to be performed in developing a FFA system is how to communicate and manipulate data within the various stages of the paradigm.

Many of the stages within the FFA are not pixel by pixel in nature. Rather every pixel in a line, either a row or column, will have some influence over the transformed value of every other point in the line. This makes such transformations in parallel terms very coarsely grained.

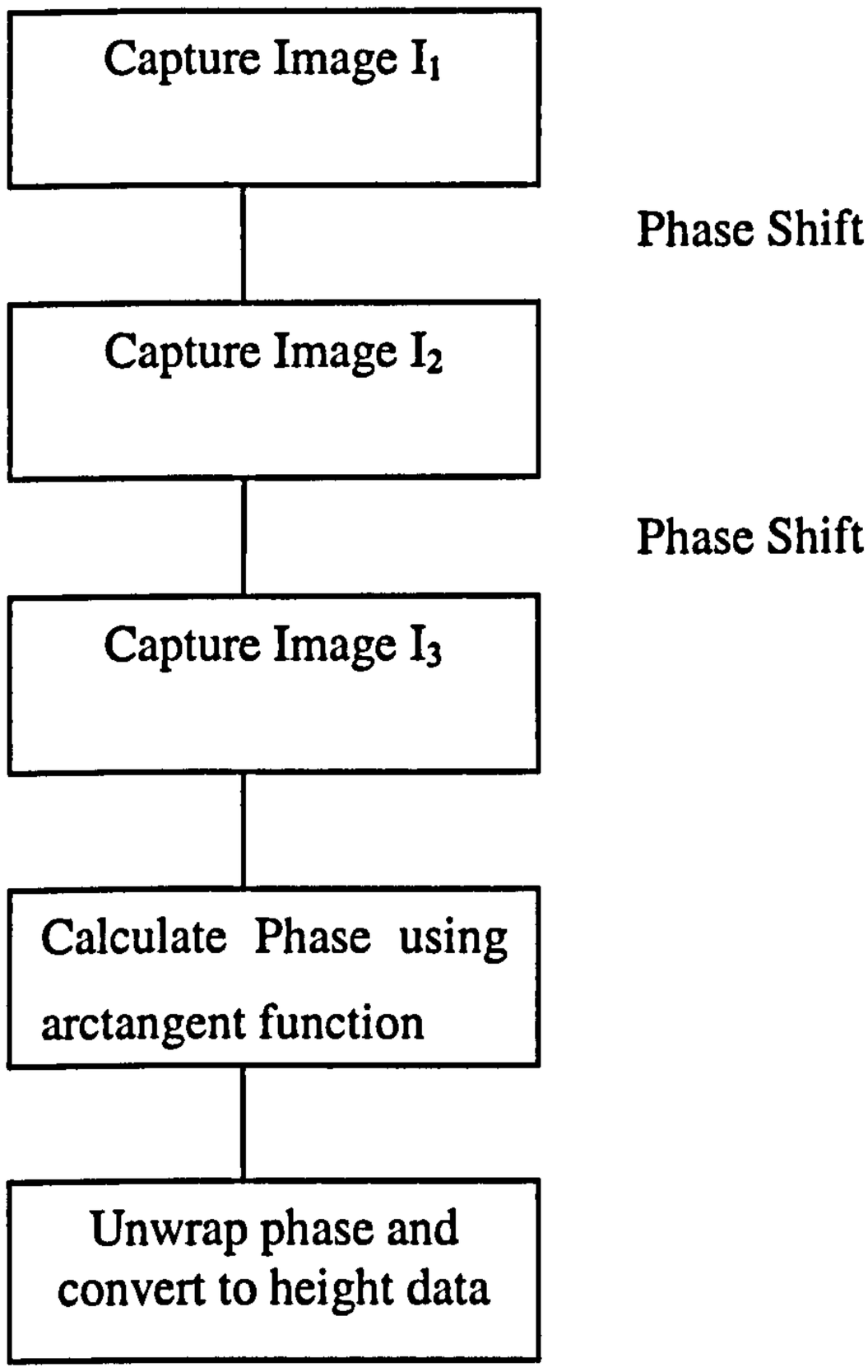


Fig.6.2 Stages in 3-Frame Phase Stepping Algorithm

Examining Fig.6.2 it can be seen that there are not as many stages involved in the algorithm of 3-Frame Phase Stepping as there are within FFA. The same is true for the Carré method of Phase stepping, which uses 1 more image and one more computational stage than the 3-Frame method. A further refinement of PSP is to use a modified "2+1" algorithm which can use the same background image as the FFA technique and requires only two phase stepping images. Thus it can be seen the PSP algorithms present simpler computational problems than the FFA paradigm.

However, a larger amount of data is required. Therefore the key issue in performing a parallel version of these algorithms is not the decomposition of computation but rather how to communicate and partition data. The capture and transfer of data are where most of the effort in PSP parallelization must be concentrated.

Both these phase extraction techniques use the same arctangent and similar unwrapper functions. Implementation of these functions raise a number of interesting aspects about the restrictions caused by the software and hardware systems used in this work. Further solving the problem of how to perform a parallel unwrapper highlights several of the important issues within data partitioning.

6.2 Implementation of a FFA Algorithm by Parts within the 3L Environment

Implementing the FFA paradigm in any software environment will change that paradigm. The aim of this section is to discuss how the 3L environment effects each of the algorithms within the FFA paradigm. The strategy identified as scheme 2 in section 6.1.1 has been used throughout the 3L work. Throughout this section all numerical examples are given for images of 256 x 256 in size. This discussion is carried out in the light of the diagram outlined for concurrent FFA and PSP calculations, fig.5.8.

6.2.1 Image Capture

Image acquisition within the fringe analysis techniques that have been used in this research forms a crucial part of the overall system. Software must be calibrated to work with the optical system. This has consequences as to the steps that can be performed in a real-time system, as will be shown.

Fourier Fringe Analysis only *requires* one image to be captured. However, it is often desirable to capture a second image i.e. a background image. This image contains the background light intensity produced by the optical set-up. This can be removed from the captured fringe image in order to present the data to the FFT functions in a more suitable data format.

Removal of the background image data from the fringe image can either be performed on the Framestore processor or passed to the child processors to perform this calculation. Implementing this algorithm changes the format in which data is communicated from the Framestore to the child processors. Not using this technique allows the data to be sent in packed format. Packed format data consists of 32-bits words, each of which contains four pixel values as packed 8-bit integers.

In the 3L system it was decided not to remove the background image so that the packed fringe image could be divided within the Framestore and sent to the 4 child

processors of layer 1. This system communicates one line at a time of data for each image. Each child processor receives 2^{N-2} lines of 2^N data points. As this is sent in packed format this means each processor receives 2^{N-2} words of data at a time. If images of 256 by 256 are captured this means each processor receives 64 lines of 64 words in size.

An additional point is that to be of real benefit the unpacking algorithm should be performed using the on-board ram blocks, this means that the precise location of the unpacking routine and the FFT must be known by the programmer. Additionally whilst the unpacking routine is performed the data is converted from the integer format it is captured in, into the floating-point format required by the C40 optimised FFT routines.

6.2.2 Pre-processing

Pre-processing is a very powerful stage in the FFA paradigm. It is made up of a number of algorithms, which are performed not only to remove noise from the data but also to present this data in a manner suited to the FFT algorithm. However, this stage is optional and has not been explored under the 3L environment. This was because of the use of packed data. Sending the data in a packed format from the Framestore to the four child processors of layer 1 made implementation of the pre-processing stages an extremely hard problem. Additionally the work carried out in the 3L environment was performed to test a number of the algorithms of the FFA paradigm purely for speed and reliability issues.

6.2.3 Forward Row FFT

The forward row FFT uses the captured image data that consists of real-only data i.e. data where the imaginary term is ignored as it is set to zero. Specialised Fourier transforms have been developed to use this data in an optimised format[1]. The conventional FFT uses complex data, this format could be achieved by padding the pre-processed data but this is a wasteful operation. Application of the real-only transform means that only 2^{N-1} complex data pairs are returned for each line of

data[2]. This exploits the fact that for real data the returned complex data will be symmetrical about the zero spatial frequency part.

All the schemes of forward FFT in this work use the on-board ram blocks of the C40 to optimise the FFT's performance[3]. The data is transformed in-place that is in the same position as it was copied to. But the data from the real-only FFT is returned in an unusual order and is Hermitean in format[4]. Therefore when moved from this stage special attention must be paid to getting the data into complex pairs, fig 6.3.

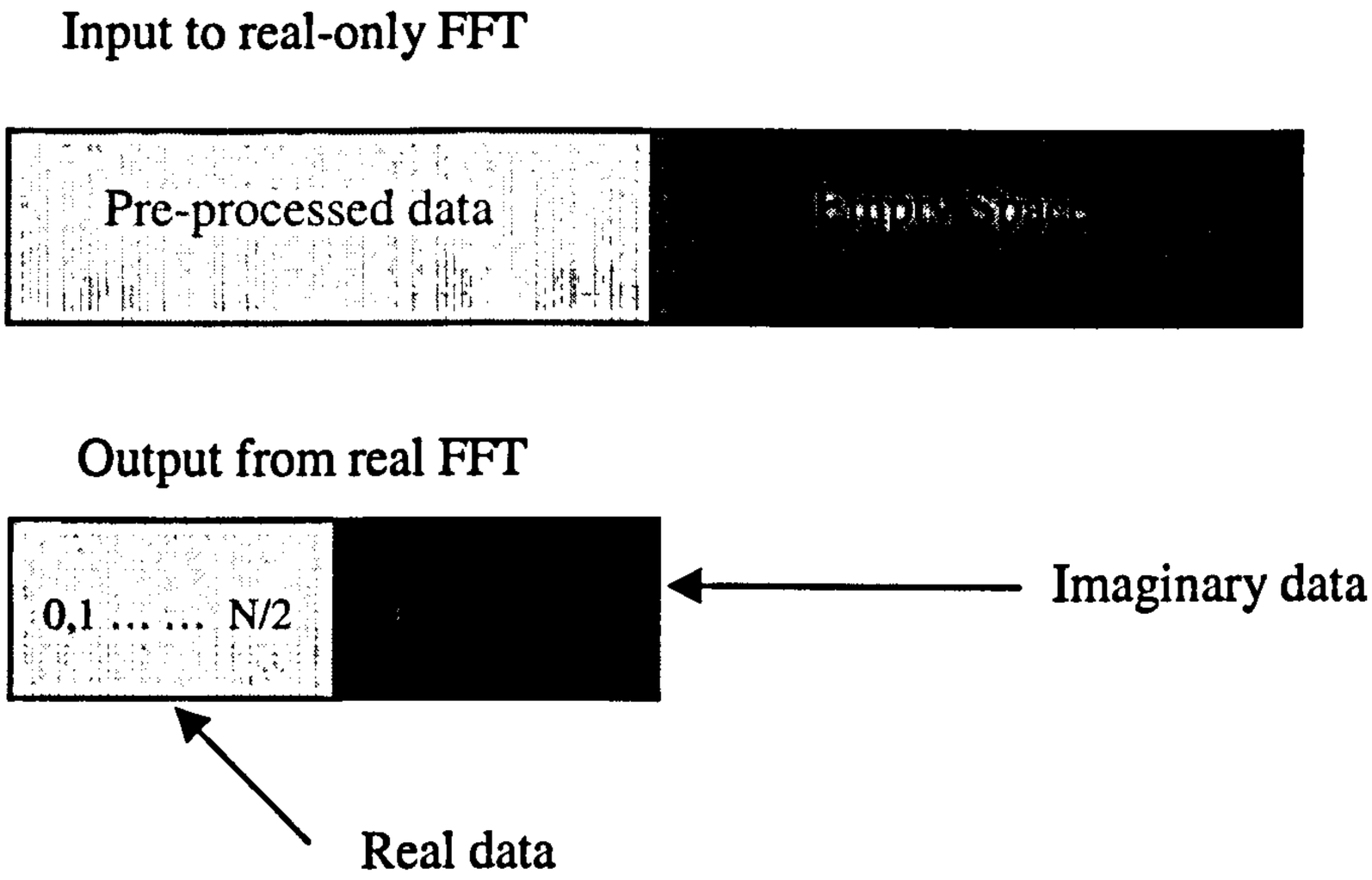


Fig. 6.3 Organisation of a line of data after the application of the real only FFT

In the 3L system before the real-only FFT can be performed the data must be unpacked. This occurs within the RAM memory of the C40 which copies data into one section of the RAM, then this data is unpacked to a different memory location and converted to floating-point format. The reason for converting to floating-point is that the code used in this work is optimised for floating point operations.

Once the data has been transformed in-place the data can be sent to the next stage of the FFA process. In the 3L system using images of 256 by 256 in size means that each child processor must communicate 64 lines of 128 complex data points to the next stage of FFA.

6.2.4 Matrix Corner Turn Block

The next **transform** stage of the scheme 2 Fourier Fringe Analysis paradigm is to perform a Forward Column transform. In a single C40 processor system this would be a simple process of merely reading the required data in a column addressing method. However, in a parallel processing system it is not as straight forward as this. Rather an additional matrix manipulation stage is required, the so-called corner turn block.

Incoming data is arranged in the same format as matrix R,

$$R = \begin{matrix} 00 & 01 & 02 & 03 \\ 10 & 11 & 12 & 13 \\ 20 & 21 & 22 & 23 \\ 30 & 31 & 32 & 33 \end{matrix}$$

The numbers used in matrix R merely represent the row and column indices of each term. The shift required is defined as

$$R\,rc = R'\,cr \qquad (6.1)$$

where,

R = received matrix value

R′ = matrix value to be sent

r = row index

c = column index

Using equation 6.1 matrix R is manipulated to become R′

$$R' = \begin{matrix} 00 & 10 & 20 & 30 \\ 01 & 11 & 21 & 31 \\ 02 & 12 & 22 & 32 \\ 03 & 13 & 23 & 33 \end{matrix}$$

Mathematically this is the transpose and is written as

$$R' = R^T$$

Now the data is organised in the correct order to be sent to the next stage of the FFA paradigm. Using a parallel strategy means that data cannot be transposed until the entire image has been transformed by the Forward Row FFT functions. In fact this corner turn operation must occur between **every** transform stage of the scheme 2 FFA paradigm.

Under the 3L environment the corner turn block is performed on the root processor, which acts as the hub for the entire corner turn blocks. Data is received from each layer 1 child processor a line at a time, using the data storage strategy from fig.5.11(a). This means that the storing of each of these lines to the corresponding location in the image is a critical operation. Once the entire transformed image has been formed, it can once more be split into strips.

The C40 processor allows any indices to be used for data storage and this means a number of methods of storing have been explored. For the stage between forward row FFTs and forward columns FFTs each row from the transformed image is stored as two lines of data in the corner turn array. The first line is made up of the real terms from 0 to 2^{N-1}-1 for the row. While, the second line contains the corresponding imaginary terms. Once these are collected the columns are sent line by line, which means each child processor receives 32 lines of 256 complex data pairs, fig.6.4.

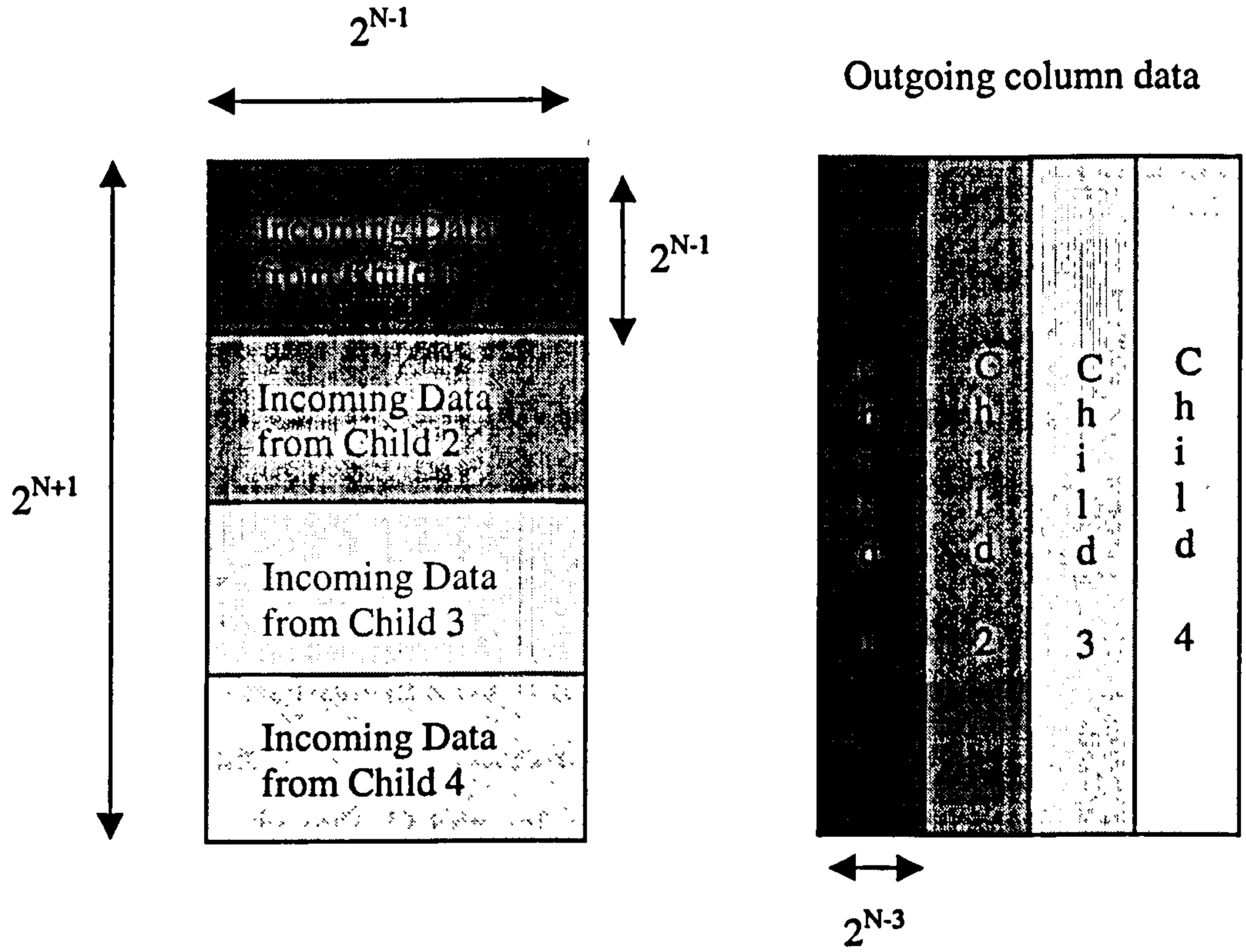


Fig.6.4 3L corner turn scheme for data from the forward row FFT

It is worth mentioning the distributed data scheme outlined in fig.5.14 at this point. If a distributed data system is used such corner turner blocks are not required, however each child processor being used to perform a parallel algorithm needs to store the entire section of the transformed image it has calculated. It is then necessary to communicate exactly the correct part of each section to the correct processor for the forward column transform.

In fact much of the early experimentation performed in this research under the 3L environment pursued this strategy. It was felt that this strategy would be significantly quicker than the more centralised data scheme discussed in fig.5.15. However, after the initial development period this method was abandoned for several reasons. Firstly, tracing exactly where data was being held and communicated to in a FFA system using the scheme 2 strategy was very difficult. An additional disadvantage is that this strategy is not easily reused in an objected orientated manner. Which means that anyone seeking to re-use the code must already be an expert in C40 systems and the particular hardware topology of the INFOCUS system.

6.2.5 Forward Column FFT.

In the 3L environment where the scheme 2 FFA strategy has been employed the next transformation stage is the Forward Column FFT. Unlike the previous real-only FFT this algorithm is much more conventional in manner. Within the 3L environment the data is transferred by the DMA co-processor from the matrix corner turn block to the Forward Column FFT in bit-reversed order. This means that the algorithm employed to perform the Fourier transform does not need to spend any time arranging the data in the correct order.

The data outputted from this stage is exactly the same size as the incoming data. For example, under the 3L system each child processor receives 32 lines of 256 complex pairs and sends 32 lines of 256 complex pairs. These lines are sent to the filter function.

6.2.6 Filtering

In the 3L environment the root processor performs the filter function. This filter function is very similar to the corner turn block, in that it requires a matrix of data to be manipulated into the correct order for the next stage of the FFA paradigm.

The transformed data from the Forward FFT algorithms represents the Fourier domain of the image and as such consists of the harmonics of the signal from the image. Where the DC term, or zeroth order harmonic, exists it must be removed. The problem presented by filtering can be defined as the isolation of the 1^{st} order harmonic of the image, this is demonstrated for a line of transformed data under the scheme 1 strategy in fig.6.5. This 1^{st} order peak contains the surface height information that has been encoded by the modulation of the carrier fringes[5].

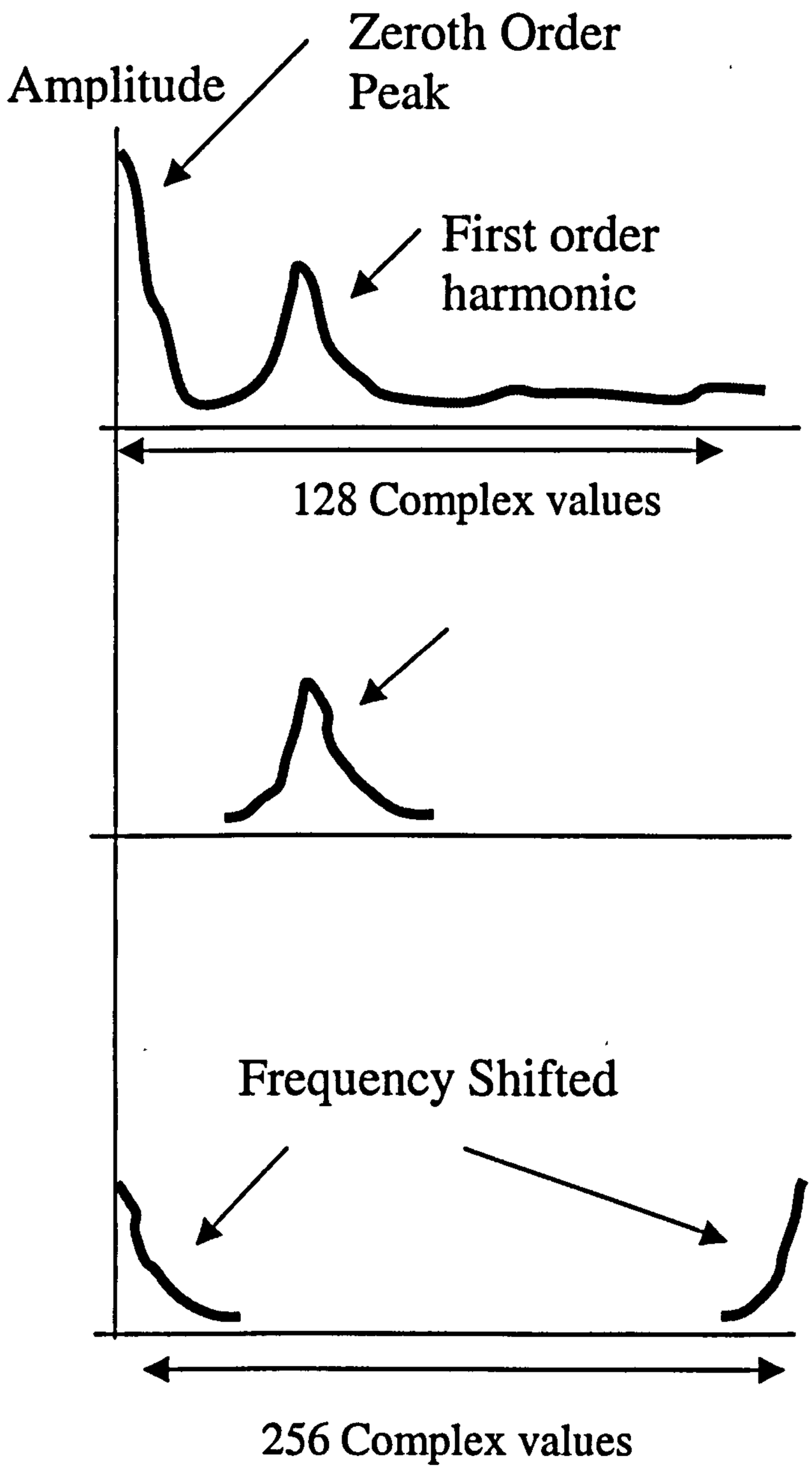


Fig.6.5 (a) Represents the signal returned from a single line of the real-only transform. (b) The 1st order harmonic has now been isolated (c) Frequency shifted 1st order harmonic

Under the scheme 2 strategy employed within the 3L environment the reality of this filtering stage is much more complex. Firstly the transformed data is not in the conventional format but rather in quadrant swapped configuration[2], fig.6.6 (b). An additional complication is that within the 3L environment not all of the data is present. In effect looking at the image in fig.6.6 (b) only half of the data is present. Rather than spend time manipulating data into the correct format it is easier to **move** the data in the correct format.

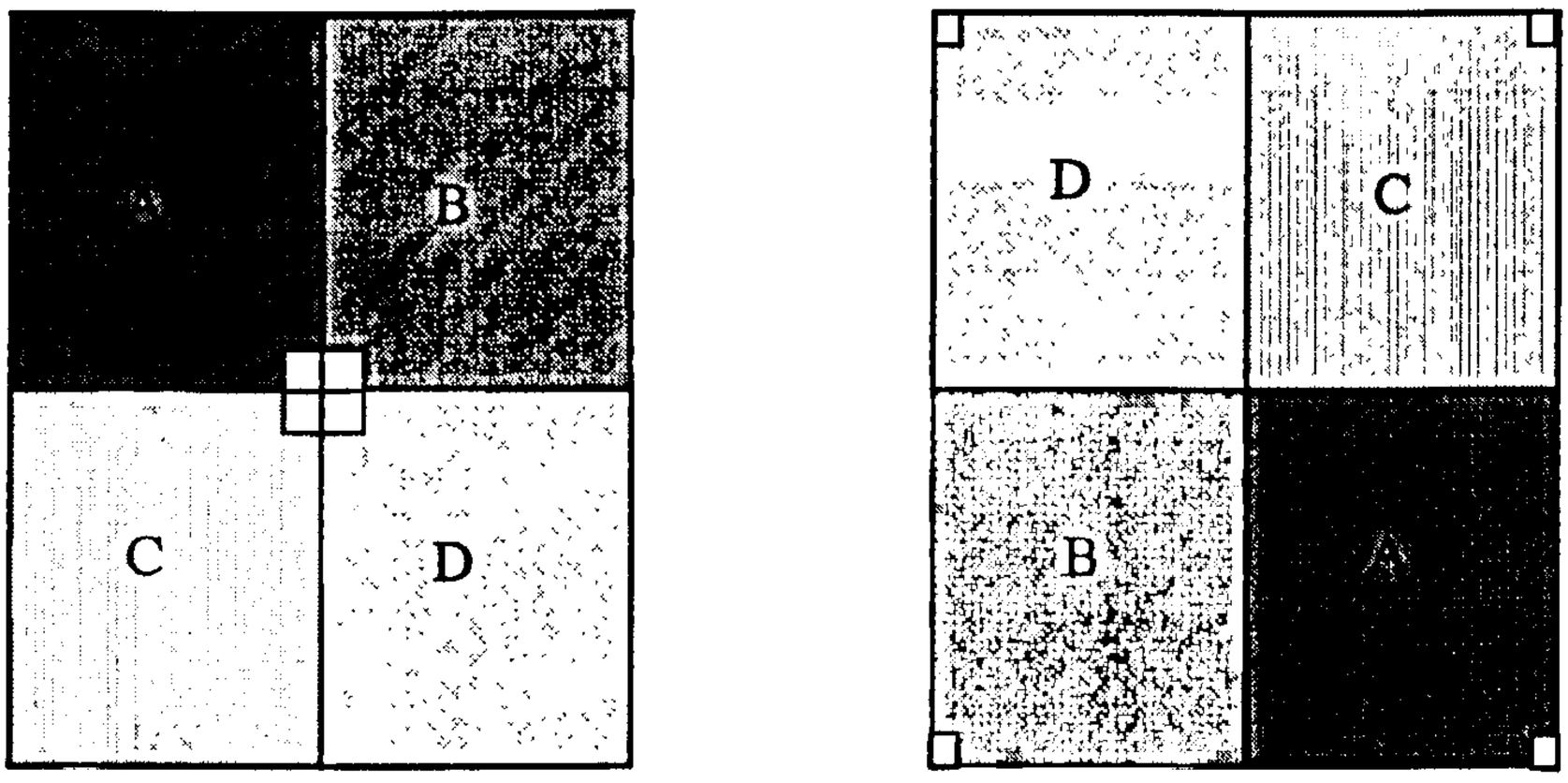


Fig.6.6 Example of quadrant swapping (a) conventional frequency space (b) in standard quadrant swapped format.

In fig.6.6 (a) data is arranged such that the zeroth order harmonic (the D.C. term) is located at the point where all four quadrants join as indicated by the square located at this point. In fig.6.6 (b) the components of the zeroth order harmonic are located in the four-corners of the quadrant swapped image.

In the 3L environment the data after the forward transforms is in the format shown in fig.6.7 (a). Examining this diagram it can be seen that only half the data exists in this configuration when compared to the configurations of fig.6.6. It can be seen that the two quadrants that exist are arranged with the two components of the zeroth order term located in the top right and bottom left corners of the transformed data. This means that the two components, which make up the required first order harmonic are arranged in the format indicated by the two black squares in fig.6.7 (a).

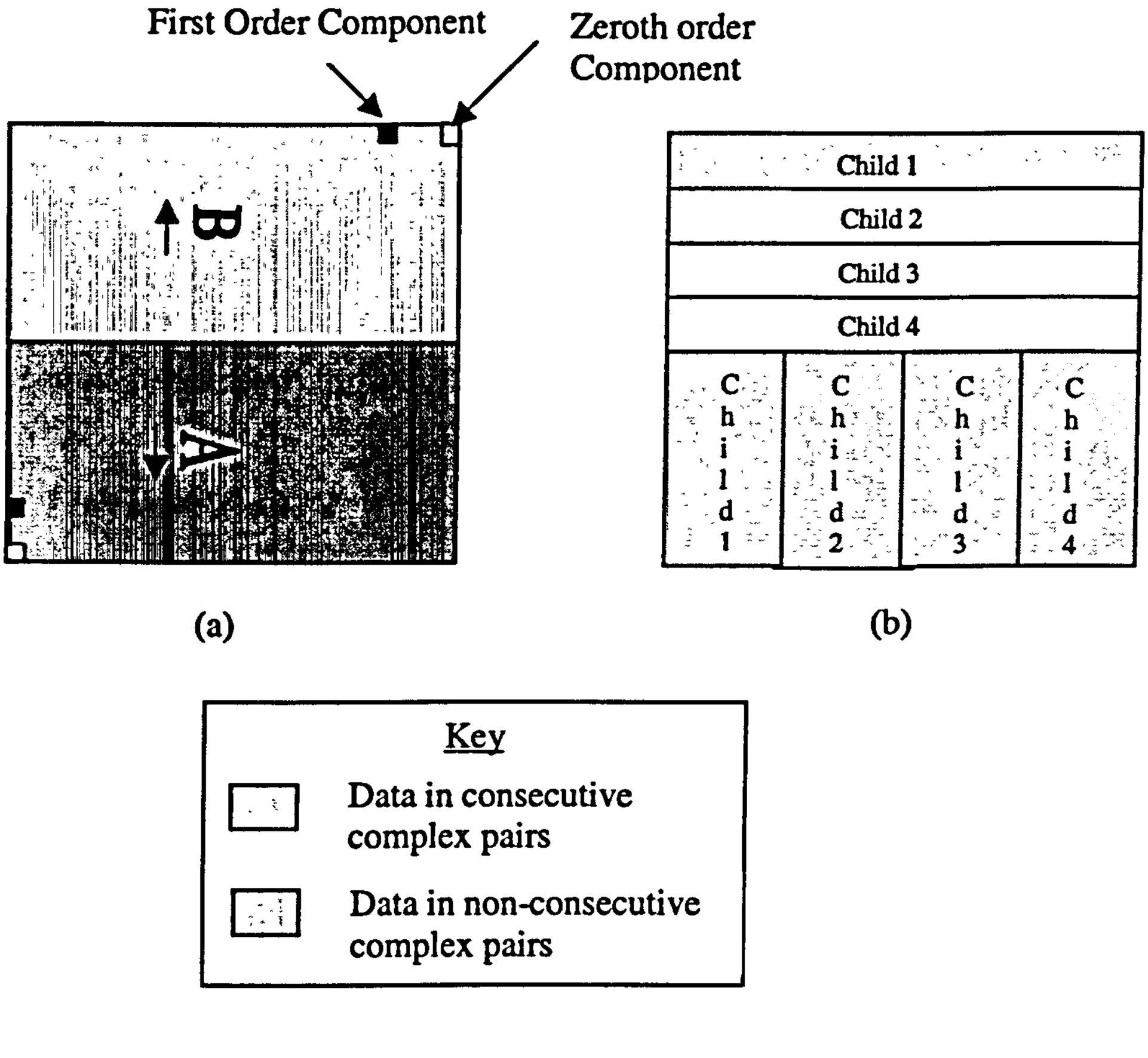


Fig.6.7 (a) Quadrant swapped data under in the 3L-environment (b) Storage of the data for inverse row FFT.

The data described in fig.6.7 (a) is stored in the root processor in format of fig.6.7 (b). In the top half of the array the data is stored in consecutive complex pairs, that is each data value is stored as complex pairs of values in a continuous line. Each of these lines consists of 2^{N-1} complex pairs, for an image of 256 by 256 in size this means that the first 128 lines of the transformed data consist of 128 complex pairs of continuous data. The second half of the data is arranged in a different format rather than a single row containing 2^{N-1} complex pairs, 2^{N-1} complex pairs are now arranged in two consecutive columns. The first of these columns contains 2^{N-1} real terms, while the second column contains 2^{N-1} imaginary terms.

At this stage the transformed data is made up of be 128 complex pairs by 256 columns. Although only a small part of the data is required e.g. 20-40 lines which

contain the first harmonic, the system nevertheless transmits 2^{N-1} lines, in effect 128 full rows of data. The reason for not cutting down this transferred number is that the exact position and location of the 1st harmonic is not fixed. Using the idea of a reduced number of communications would make the system less flexible and would limit the use of the final system.

The data from this filter stage is sent to the next stage of the FFA paradigm using the strategy described by fig.5.11 (a).

6.2.7 Inverse Row FFT

In the 3L environment exactly the same algorithm to perform complex FFTs is used as has already been used in the Forward Column FFT stage. Each child processor on layer 1 receives 2^{N-3} rows of data. Once the data has been transformed it is sent to a corner block which is identical to that used after the forward row FFT. Within this software environment this corner turn block is performed on the root processor.

6.2.8 Inverse Column FFT

Once the data has been transformed by the inverse row FFT it is necessary to get this data in the correct format for the inverse column transform. At this stage only half of the final data required exists, fig.6.8. There are a number of methods to achieve the correct final size. Firstly the second half of the array can be padded with zeros at the corner turn process and the entire column of N complex data points can be communicated. Or alternatively the data can be sent as the N/2 data points and then the second half of zero terms can be added by the child processor.

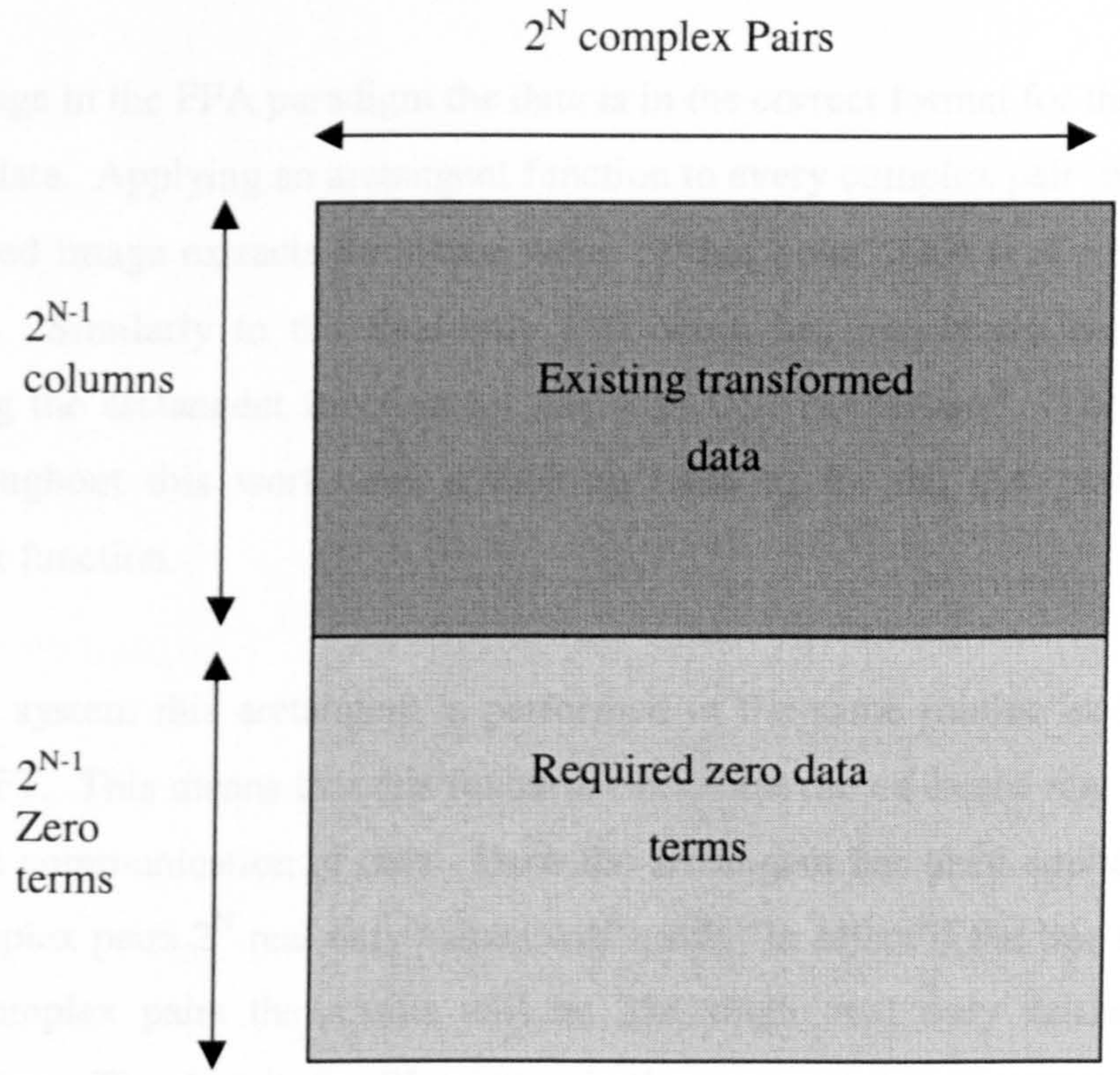


Fig.6.8 Shows why padding is required to achieve an N by N matrix for the inverse column FFT.

In the 3L environment the second method of achieving the correct size of data is used. This is because the 3L code is not as reusable as the Pegasus code i.e. each task must be coded specifically for the 3L system. The data from the inverse corner turn block is 2^{N-1} complex data pairs in length, with each child processor transforming 2^{N-2} lines. Before the inverse column transform is applied to the incoming data the memory location to be used must be padded with 2^N zero terms, effectively 2^{N-1} complex pairs of zero.

For a 256 by 256 image each child processor will perform 64 transformations of data. The incoming data is organised in 128 complex pairs, while the transformed data is 256 complex pairs. At this stage the data is ready for phase extraction.

6.2.9 Arctangent

At this stage in the FFA paradigm the data is in the correct format for the extraction of phase data. Applying an arctangent function to every complex pair in the inverse transformed image extracts the phase value of that point. This is a pixel by pixel operation. Similarly to the Real-only FFT work has previously been done on optimising the arctangent function for use with DSP processors[6]. The arctangent used throughout this work uses a look-up table to fix the data values of the arctangent function.

In the 3L system this arctangent is performed in the same routine as the inverse column FFT. This means that this function can utilise the on-board ram blocks and reduce the communication of data. Once the arctangent has been applied to a line of 2^N complex pairs 2^N real only values will result. In effect if the line is made up of 256 complex pairs the results will be 256 single real only values, half the memory size. The data in the 3L system is then communicated to another corner turn block. Each child processor will communicate 2^{N-2} lines of 2^N real only values e.g. 64 lines of 256 values.

6.3 Pegasus Environment FFA by Parts

The changes made to algorithms of the FFA paradigm under the Pegasus environment differ somewhat from those under the 3L environment. This sections aims to highlight the changes that Pegasus causes to each of the stages of FFA. Most of this discussion is concerned with the strategy of concurrent FFA and PSP outlined in fig.5.8 but where appropriate to highlight specific points the strategy of FFA only, presented in fig.5.9, is reviewed.

6.3.1 Image Capture

Within the Pegasus system the first major change when compared to the 3L environment is that the background image is removed from the fringe image. This means the communication of packed data is not used. Under the scheme 2 strategy the background image was removed from the fringe image within the Framestore. This means that the Framestore must hold data until both images have been captured. The subtraction performed by the Framestore in the Pegasus environment can then occur with very little cost. This is the case because data must be converted from integer-format into floating-point format as is required by the environment. This is an obvious infringement of the system when compared to conventional systems where integer data would be used for as long as possible.

In the strategy utilising scheme 1 a slightly different method of background removal was used. Rather than subtract the background image using the Framestore a child processor of Layer 1 is used to perform this subtraction and all the subsequent pre-processing. In the FFA-only strategy a child processor on layer 2 performs this task. There are a number of reasons for making this change the first of which is that it means there is less work for the Framestore to perform. The next point is that it simplifies the code that needs to be used by this processor, the Framestore now simply churns out image data.

An additional point is that for a system using concurrent phase extraction techniques not performing complex tasks such as background removal on the

Framestore is vital. In this concurrent system there is the need for FFA and PSP pre-processing. Performing both of these on the Framestore complicates the data problems such as data storage and data handling. A further point is that in scheme 1 FFA, when compared to scheme 2 FFA, any layer 1 child processor has enough spare capacity to perform pre-processing. While if scheme 2 is being used this is not the case, this is because of the additional stages that are required by scheme 2. These additional stages not only add time but also use up large amounts of processor memory.

In the scheme 1 strategy the child processor that performs pre-processing receives data in the format shown in figure 5.13. But rather than 8 lines at a time 32 lines of data are received at a time. At this stage in the Pegasus scheme 2 strategy the data is still held on the Framestore.

6.3.2 Pre-processing

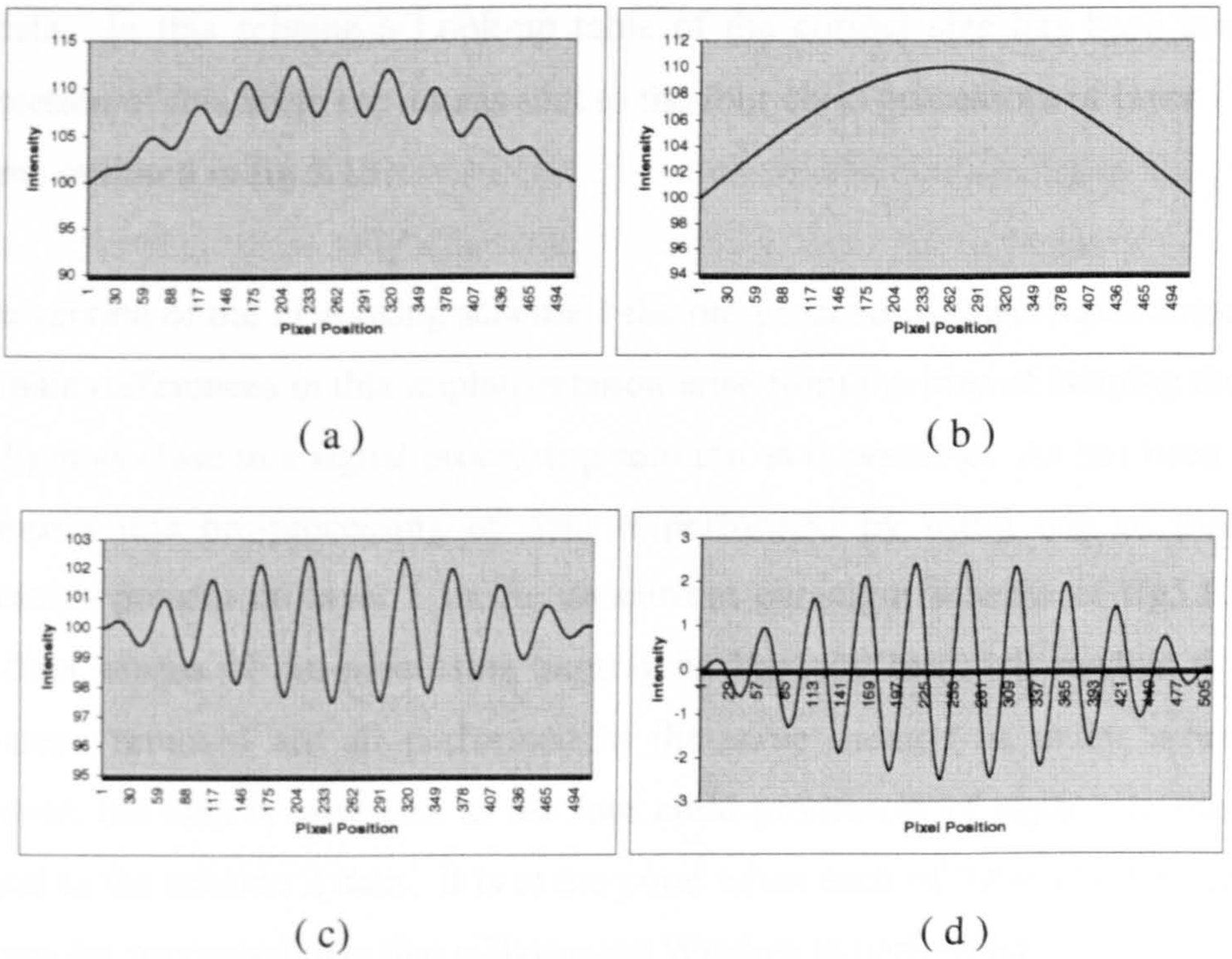


Fig.6.9 Showing various pre-processing stages. (a) Line of unprocessed fringe image (b) Line from the reference image (c) Fringe data after removal of corresponding reference data (d) Processed data once the mean or DC value has been removed

Under the 3L environment no pre-processing has been performed, however under the Pegasus environment all stages of pre-processing have been implemented. Although the various stages are optional to the overall FFA their implementation is often highly beneficial. Examining fig.6.9 shows how a row of data is transformed into a format more suitable for the application of an FFT.

Using the scheme 2 strategy all the stages of pre-processing are performed by the Framestore. The initial stage is to remove the background image as has been previously outlined. Once the background removal has been performed the next algorithm to perform is median filtering. This is performed to help reduce speckle noise in the data.

The next stage of pre-processing is to perform a mean calculation of the processed data. Once this mean has been calculated it is removed from every data point in the processed data. The final stage of this process is to apply a Hamming Window to the data. In this scheme a Look-up table of the correct size has been used. On completion of this stage the data is sent to the four child processors of layer 1 in the scheme outlined in fig.5.13.

In the version of the FFA using scheme 1 the pre-processing stage varies somewhat. The main differences in this implementation arise from the idea of keeping the FFA paradigm as close to a signal processing solution as is possible. As has been stated previously this pre-processing of data is performed by using one of the child processors present on layer 1 in the concurrent paradigm scheme of fig5.8. The first three stages of pre-processing namely background removal, median filtering and mean removal are all performed in the same method as under scheme 2. However, the data is now sent to the four child processors of layer 1 in the same method as the scheme 2 data. It is at the point when each of these child processors receives the processed data that a Hamming Window is performed.

It is worth noting that in the scheme 1 strategy data is not transferred from the child layer 1 processors until all the remaining stages of the FFA have been performed. This is because using this strategy keeps the FFA paradigm as a signal-processing

problem, which means that discrete data units (DDU) of data are transformed independently of other DDU until after the initial unwrapping stage.

6.3.3. Row Forward FFT

The data in the Pegasus version of the FFA paradigm is transformed by the application of exactly the same algorithm as under the 3L environment. In the Pegasus implementation of the scheme 2 FFA strategy each child processor on layer 1 receives 8 lines of data at a time, fig.5.13. Each of these lines is copied to a ram block of the C40 and then transformed. Once these lines have been transformed they are communicated to the next stage of the scheme 2 FFA. An additional point to raise is that whilst data is being transformed and communicated to the next stage it is also being received from the pre-processing stage. This is a continuous buffering scheme, which reduces the latency present with in the parallel FFA paradigm.

The forward row FFT algorithm in the scheme 1 FFA paradigm uses exactly the same algorithm and idea of buffered input as the Pegasus scheme 2 strategy. However, this data is not communicated to the next stage of the FFA before filtering has occurred. This is so the filtering and frequency shifting if required can be implemented in the ram blocks of the C40 used for the forward row FFT, which aids the speed of the process and the simplicity of the strategy. The filter used is discussed in section 6.3.6.

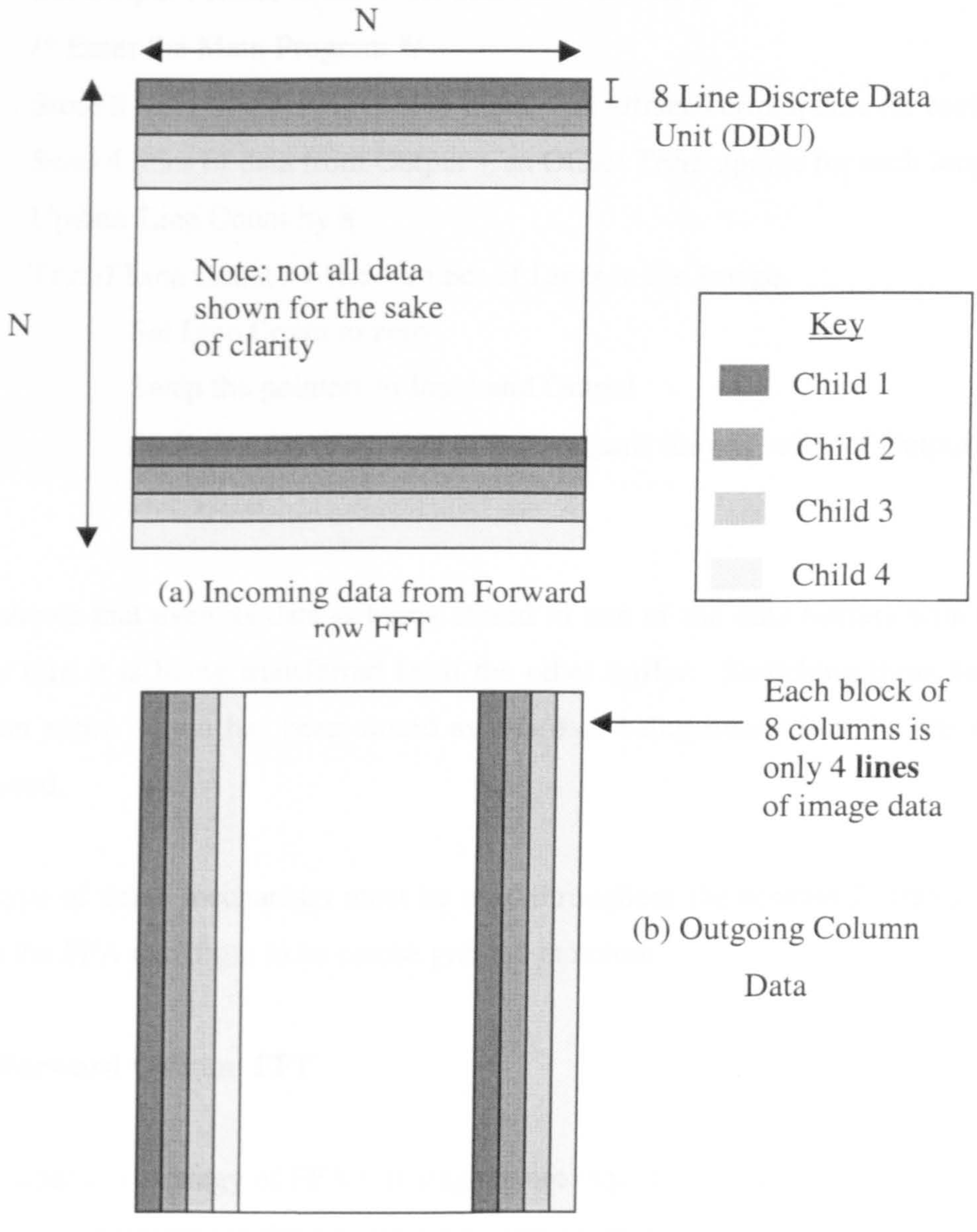
6.3.4 Matrix Corner Turn

The matrix corner turn under the Pegasus environment differs slightly from that under the 3L environment. Firstly it should be noted that this stage is only required by the scheme 2 strategy of the FFA.

In the Pegasus environment the incoming data is received by the corner turn block in DDU of 8 lines of 128 data pairs. Rather than using one of the parent processors this system uses one of the child processors to act as the corner turn block. In the concurrent PSP and FFA system this would be a child processor on layer 1. While

if the FFA only strategy of fig.5.9 is being used one of the child processors of layer 2 is used.

The incoming data is once again formed into an entire array, this time using the strategy outlined in fig.5.13. While the incoming data has been made up of DDU of 8 lines the outgoing data consists of DDU of 4 lines, where each line contains 256 data pairs. In total each child processor receives 8 such transfers.



(a) Incoming data from Forward row FFT

(b) Outgoing Column Data

Fig.6.10 Pegasus scheme of data organisation after forward row fft.

This corner turn block is a time delay caused by moving a single processor technique into a parallel-processing environment. One method of minimising the

time cost of this matrix manipulation stage is to buffer the image so that data is always being sent and received by each of the corner turn blocks in the scheme 2 FFA strategy. An example of this technique is outlined in the following pseudo code

```
/* Initialisation Code */
    Allocate Memory for two Buffers A & B of size equal to Rows by Columns
    Set Input Pointer equal to the address of Buffer A
    Set Output Pointer equal to the address of Buffer B
    /* Enter the Main Program */
    Store 8 lines of received data to Input + an Offset Term update for each loop
    Send 4 lines of data from Output + an Offset Term update for each loop
    Update Line Count by 8
    Test if Line Count >= the Number of Lines in the Image
        Set Line Count to zero
        Swap the pointers to Input and Output
        Such that the new value of Input equals the old value of Output and
        vice versa
```

This shows that even as data is being stored in one of the data buffers within the corner turn it is being transferred from the other buffer. Switching these buffers after an entire image has been stored avoids data being overwritten before it has been used.

This type of delay mechanism must be used throughout the scheme 2 strategy and forces the FFA paradigm to be coarse grained in nature.

6.3.5 Forward Column FFT

In the scheme 1 strategy of FFA this stage is not required. In the scheme 2 strategy each layer 1 processor in the concurrent system receives 2^{N-3} lines of complex data, with an image of 256 by 256 in size this means each child processor of layer 1 must perform 32 complex FFTs. The data sent to this algorithm is received and sent in DDU sizes of 4 lines at a time.

6.3.6 Filtering

Within the Pegasus system the two FFA schemes outlined in Fig.6.1 both raise different issues with regards to filtering. Under the Pegasus environment the data in the scheme 2 strategy is stored in a quadrant swapped configuration of fig.6.11.

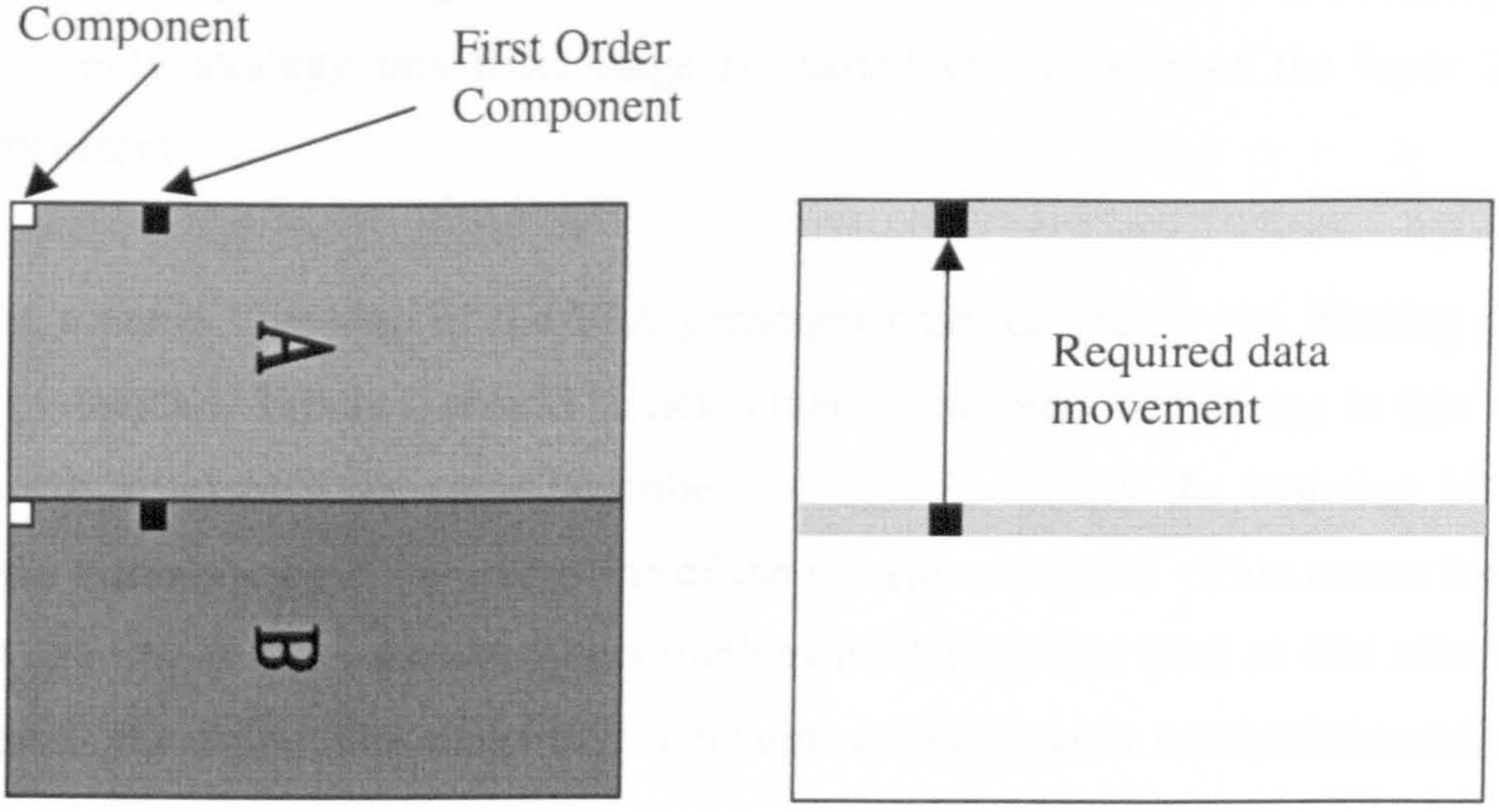


Fig.6.11 (a) Pegasus system quadrant swapped scheme (b) Showing how lines consist of components from the top and bottom half of the array

The quadrant swapped arrangement in Pegasus is such that components of the first order harmonic peak are stored at two locations the first in the top left corner of the data and the second location immediately below the left-hand side mid-point of the transformed data. The arrow in fig.6.11 (b) indicates that first part of the desired harmonic is stored in the top half of the array, while the next part of the harmonic is stored in the second half of the array.

Data for this stage is received in DDU of 4 lines at a time, where each line contains 256 complex pairs. A total of 128 lines of data are received for each image, this is because of the symmetry present in the FFT[7]. When data is sent from this filter stage it is transmitted in DDU of 4 lines. This means that after filtering only 2^{N-1}

lines of data exist, which means that at some later FFA stage the data must be padded with zero terms to achieve the required 2^N by 2^N size.

In the concurrent system the filter stage is performed by one of the child processors on layer 1, although not the child processor that performs the forward FFT corner turn block. This is because of the large amount of space that must be set aside for matrix manipulation algorithms such as the filter and corner turn blocks. In the FFA only strategy this filter stage is carried out by one of the layer 2 child processors.

The scheme 1 version of the FFA paradigm changes greatly the filtering process implemented. Firstly it should be noted that the problem of filtering in this scheme exactly resembles the ideas described by fig.6.5, namely the isolation of a first order harmonic peak for **every** line of the transformed data. This means for a 256 by 256 image there are 256 filters implemented, the filter used in this scheme is a signal-processing filter rather than a 2D dimensional matrix manipulation solution.

In the scheme 1 filter once the first order harmonic has been located this is copied to an output array which is 2^N complex pairs in length, all of which have been initialised as zero in value. The first order harmonic for each is copied to the same position in the output line as it was found at in the FFT data. An alternative to this technique is to perform a frequency shift at this point[8]. This can benefit the unwrapping process used at a later stage[9]. Both these methods of positioning the data have been explored in this work.

6.3.7 Inverse Row FFT

This transform uses exactly the same algorithm as the forward column transform. Under the scheme 1 strategy each child processor on layer 1 transforms 8 blocks of 8 lines of 256 complex pairs. Once the inverse row transform has been applied the data is ready to have phase information extracted.

Within the scheme 2 strategy exactly the same FFT algorithm is used. Each child processor on the layer performing the inverse row FFTs transforms 2^{N-3} lines. The layer of child processors used for this function is layer 1 in the concurrent strategy and layer 2 in the FFA only strategy. Once these transforms are complete the data is sent to a corner turn block. In the concurrent scheme this is the remaining child processor on layer 1 that has not been used for the FFA pre-processing block, the forward corner turn block, or the filter block.

In the FFA-only scheme this corner turn is performed on one of the layer 1 child processors, this is the first time in this scheme that the layer 1 child processors are used. This is because in this scheme this corner turn block acts as the transition point between layer 1 and layer 2 processors. The following stages of the FFA paradigm are all performed by using the layer 1 processors, therefore from this point on there are no differences between the concurrent and FFA-only schemes.

6.3.8 Inverse Column FFT

This stage is not required by the scheme 1 strategy of the FFA paradigm.

Previously in section 6.2.8 two methods of performing the required padding of data at the inverse column stage were discussed. Within the Pegasus environment the first method of organising the data for the inverse column transform is implemented, namely in the inverse corner turn block the transformed data is padded to resemble the array in fig.6.11. The reasoning behind this choice is that the Pegasus environment uses as much common and flexible code as possible. Therefore rather than use a new complex FFT just for this stage, the same algorithm as in the previous two complex transforms is used.

Each child processor receives in total 2^{N-2} lines of N in size. In the inverse corner turn block data is **not** sent in the method described by fig.5.13 as all the other turns in this scheme have been. Rather it is based on the strip segmentation strategy outlined in fig5.11 (a). In effect each layer 1 child processor will receive one contiguous block of 64 lines consisting of 256 complex points sent in DDU of 8 lines.

Once these blocks have been transform by the complex FFT the data will be the same size as the inputted data but will now be ready for phase extraction.

6.3.9 Arctangent

In the scheme 1 version of the FFA paradigm data is transformed by applying exactly the same arc-tangent algorithm as was described in section 5.6.9. Once this algorithm has completed transforming the data it is necessary to perform unwrapping.

Under the scheme 2 Pegasus system the data is transformed within exactly the same memory location as the inverse FFT is performed. The result of the transformation is identical to the 3L system. However, the data is not communicated to a corner turn block but rather is retained by the child processor, which has performed the arc-tangent so that unwrapping can be carried out by the child processor.

At this stage the data in all the FFA systems is ready to be unwrapped.

6.4 3L PSP algorithm by parts

The aim of this section is to explore those functions that make up the Phase Stepping Profilometry paradigm under the 3L environment. Largely this depends upon the equation, which has been used as the basis for the technique.

All examples use images of 256 by 256 in size.

6.4.1 Phase Stepping Equation

PSP techniques vary in the number of images that are required. Generally the more images the better the technique, however in real-time terms this increasing number must be tempered by the speed requirements of a system. Additionally PSP techniques require that a surface must be stable, this can not be true for a living human. The option of a large number of phase stepped images is not used in this work. This is because of the speed element of the goals stated for the INFOCUS sensor in section 5.2.

In the 3L environment a three-frame formulation of temporal phase stepping is used[10]. The method of obtaining the required phase for every pixel in the viewed surface is described by equation 6.2.

$$\phi = \tan^{-1}\left[\frac{I_3 - I_2}{I_1 - I_2}\right] \tag{6.2}$$

where,

ϕ = Pixel phase value

I_1, I_2, I_3 = Pixel intensity for the same pixel position in each fringe image

For this method uneven phase steps of $\pi/4$, $3\pi/4$ and $5\pi/4$ are used, rather than the traditional 3 equal steps of $2\pi/3$. The reasons for choosing this method are the relative simplicity of the equation and the reduction in the required physical phase steps.

6.4.2 PSP Image Capture and distribution

Performing Phase Stepping increases the amount of captured data that must be stored. Not only this but the code to perform capture must be altered by the need to co-ordinate the optical system and the Framestore.

Under the 3L environment a technique that requires 3 fringe images is used. The data partitioning strategy used is that of the segmentation outlined in fig.5.11 (b). This is because PSP is a pixel by pixel operation and is therefore much easier to perform in a parallel system. This means PSP is not as coarsely grained as the FFA paradigm. Further by dividing in this manner a simple unwrapper can be used.

In this 3L system each layer 2 child processor receives a section of each fringe image which is 2^{N-1} by 2^{N-1} in size. The 3 fringe images will be communicated in the format of fig.6.12.

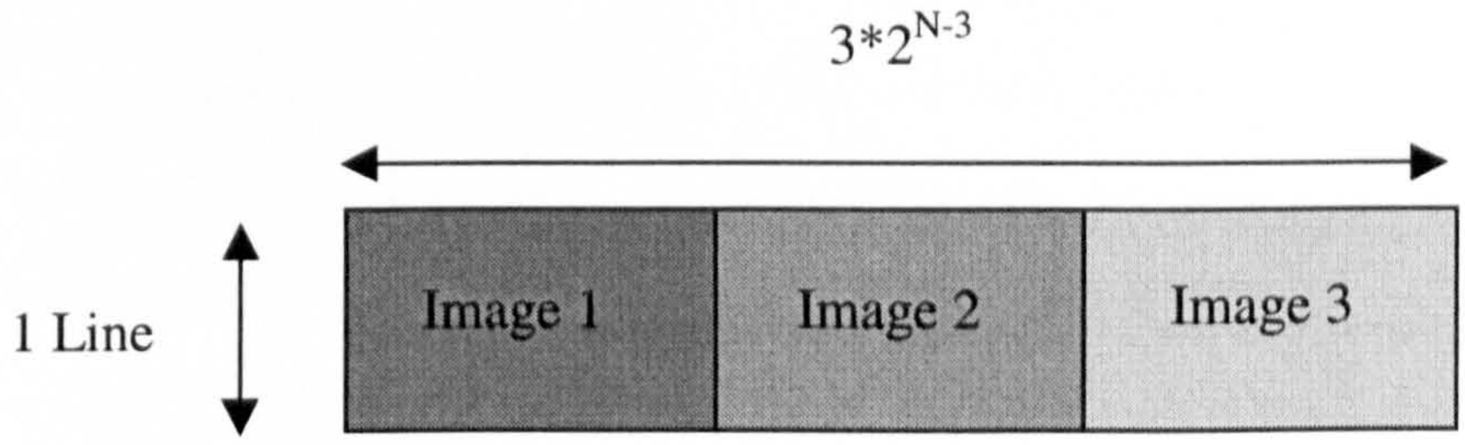


Fig.6.12 Showing how the three fringe images in the 3L PSP algorithm are divided and communicated.

In physical terms each child processor will receive 2^{N-1} communications and each of these communications will be of the size $3*2^{N-3}$ due to the packed image format. For a 256 by 256 image this means there are 128 communications to each processor. The transferred data size is 96 words in length, which when unpacked will give 3 sections of 128 real data points.

6.4.3 Phase Stepping Algorithm

The first step to performing the three-frame algorithm is to unpack the data that has been communicated by the framestore to each of the child processors in layer 2. This unpacking function is performed within the on-board ram blocks of each child processor. Once this data has been unpacked the simple subtractions required are carried out with the data being moved within the ram blocks. At this stage the data once again resembles complex pairs and is ready to be operated upon by the arctangent function.

This arctangent function transforms the data within the internal ram blocks and moves this data to another location within the internal C40 memory where the unwrapping process is performed. The transfer of data from the Framestore in the scheme of fig.5.11 (b) means that no corner turn block is required by the PSP paradigm, which will use the same unwrapper as the 3L FFA scheme.

6.5 Pegasus PSP algorithm by parts

6.5.1 Phase Stepping Equations

A number of PSP techniques have been investigated for use within the Pegasus environment. The first of these algorithms uses the equation detailed in the 3L PSP scheme, equation 6.2. However, this system has a number of inherent problems that have been discussed previously in chapter 2. To overcome these two further temporal phase stepping formulations[11] have been investigated.

The first of these two formulas is the Carré technique equation 6.3, which uses linear phase shifts.

$$\phi = \tan^{-1}\left\{\frac{\sqrt{[(I_1 - I_4) + (I_2 - I_3)][3(I_2 - I_3) - (I_1 - I_4)]}}{(I_2 + I_3) - (I_1 + I_4)}\right\} \qquad (6.3)$$

Although computationally more intensive than the three frame technique it is still relatively simple, when compared to the FFA paradigm. It benefits when compared to the 3-frame algorithm from being less prone to the inherent problems of PSP. However, this system is a more complicated problem when it comes to calibration and co-ordination between the computing and optical equipment.

The next formula that has been investigated for implementation in the Pegasus environment is the "2 + 1" technique, this can be stated as

$$\phi = \tan^{-1}\frac{I_2 - I_0}{I_1 - I_0} \qquad (6.4)$$

Between the two images I_1 and I_2 there is $\pi/2$ phase shift in the fringe pattern projected onto the surface. The intensity value of I_0 is calculated by averaging the pixel value of two images that have a 180° phase shift between them. As the "2 + 1" technique was designed with high-speed applications in mind this final value of

I_0 can *traditionally* be calculated at any time after or before the phase step system is used.

I_0 is the background intensity value for every pixel within a frame. In the FFA paradigm this background intensity value has already been used, namely when pre-processing is required and the fringe pattern is "washed-out" by oscillating one of the fibres of the interferometer by supplying the PZT with a sinusoidal signal of approximately 500 Hz to provide the background intensity image. This means that the value of I_0 does not need to be calculated again, rather the values from the background intensity image are used.

A number of points arise from using this modified "2+1" technique. Firstly a concurrent system of FFA and PSP using exactly the same images, which greatly reduces the complexity of calibration. Secondly as the "2+1" requires the background image, performing FFA pre-processing becomes a side effect of performing concurrent paradigms. Thirdly the complexity of the data storage and communication when using the modified "2+1" technique is greatly reduced. All these points make using the modified "2+1" technique in the INFOCUS system an elegant and efficient solution[15].

6.5.2 PSP Image Capture and distribution

In the version of the 3-frame step algorithm that is implemented in the Pegasus environment the data is partitioned using the strip technique of fig.5.11 (a). Each of the 3 fringe images is split into 2^{N-2} rows of size 2^N. The Framestore processor then performs the two simple subtraction routines for each pixel. This allows the data to be communicated as lines of complex pairs.

This equates to each child processor receiving 64 rows in total, in chunks of 8 lines at a time. Each of the rows is made up of 2^N complex pairs. So for a 256 x 256 image each child processor will receive 4096 words at a time.

A key issue in implementing the Carré technique within the Pegasus environment is where to perform the various calculations present in equation 6.3. Performing all

the calculations on the Framestore would cut the required communication size but would also severely hamper the Framestore. The technique that has been used in this work is to calculate the 4 intensity addition and subtraction values on the Framestore. This means the communicated rows are 2^{N+2} in size and organised as in fig.6.13.

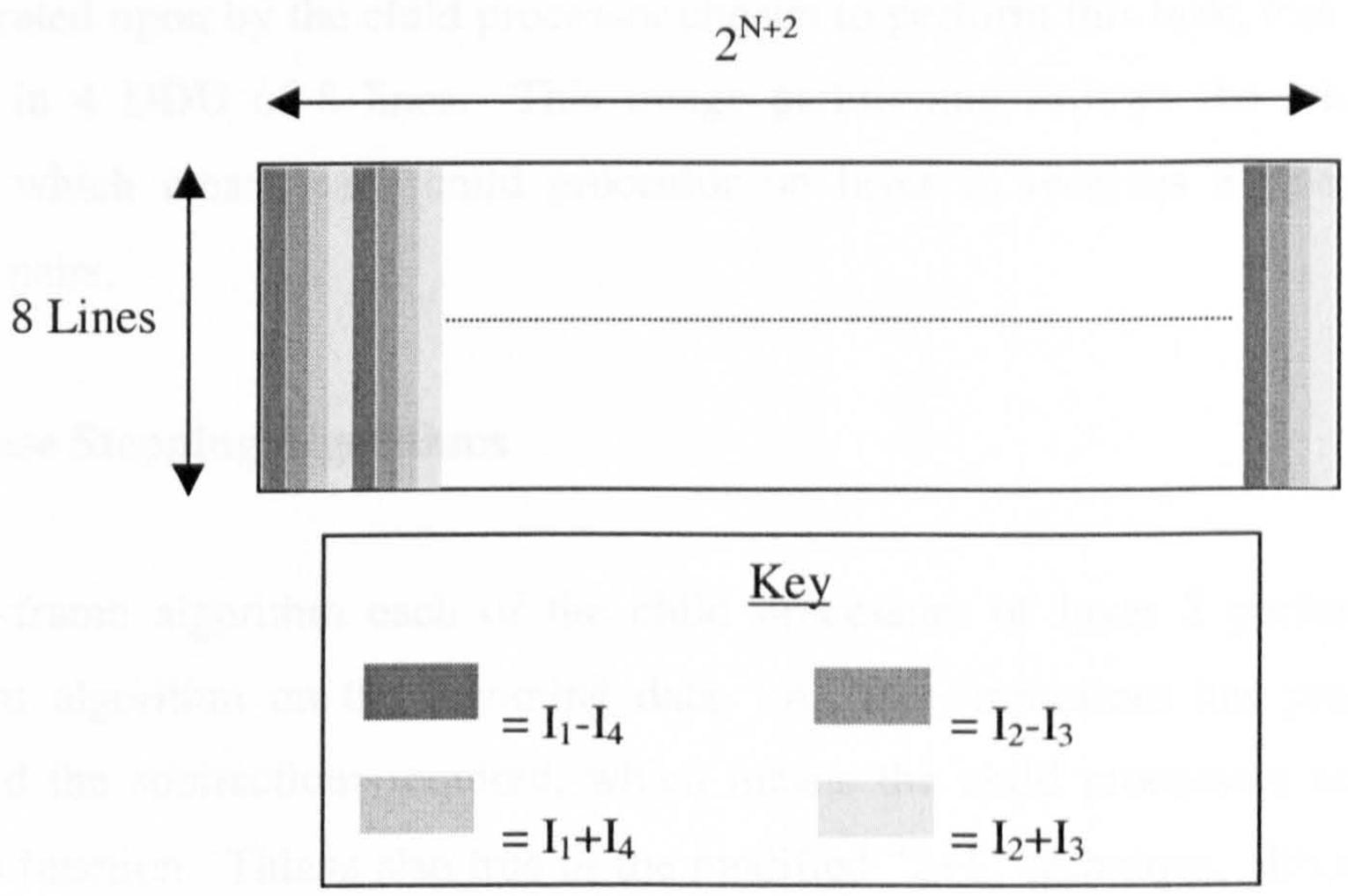


Fig.6.13 Carré data blocks

From fig.6.13 it can be seen the data partitioning strategy outlined in fig.5.11 (a) is used to divide the images to be sent to the child processors of layer 2. This means that each processor receives 8 lines of data at a time, where each line is 2^{N+2} in size. There is a need for 8 such communications to each layer 2 child processor. In terms of 256 by 256 images this means each processor must receive 8096 words of data at time.

Using the modified "2+1" technique also modifies the way data is communicated and used in the concurrent system. Rather than perform any calculations on the Framestore processor one of the child processors on layer 2 is used to perform the division of data and simple subtracts of equation 6.4. This frees the Framestore to simply capture and output image data.

The child processor that performs this PSP pre-processing firstly receives the background intensity image and stores this for later use. Then as fringe images are

captured and sent from the Framestore the child processor buffers the fringe images such that the image that was used as I_1 becomes I_2 in the next loop. This means that PSP is continuously performed.

The data is sent from the framestore in DDU of 32 lines of real only data and is then operated upon by the child processor chosen to perform this task; this outputs the data in 4 DDU of 8 lines. This image partitioning follows the scheme of fig.5.13, which means each child processor on layer 2, receives 8 lines of 2^N complex pairs.

6.5.3 Phase Stepping Algorithms

In the 3-frame algorithm each of the child processors of layer 2 performs the arctangent algorithm on the incoming data. As the Framestore has previously performed the subtractions required, which means the child processors are freed from this function. This is also true of the modified "2+1" technique, although the pre-processing has been performed by a child processor on layer 2 and not the Framestore. Once the arctangent has been performed on a chunk of data this data can be unwrapped.

Under Carré the mathematical operations which are performed by the child processors are 1 addition, 2 subtractions, 1 multiplication and a square root for each pixel. These functions are performed in such a way that the data is arranged in the required format for the arctangent function to be performed. In the C40 arctangent algorithm this is in "complex pair" format and for optimised performance this data is within the on-board ram blocks of the C40. Once this arctangent has been performed the data is ready to be unwrapped.

6.6. Unwrapper by parts

Using the arctangent function in both of the phase extraction techniques results in phase with modulo 2π discontinuities[5]. The phase at this stage will resemble fig.6.14 (a), while the aim of an unwrapping algorithm is to achieve continuous phase of 6.14 (b).

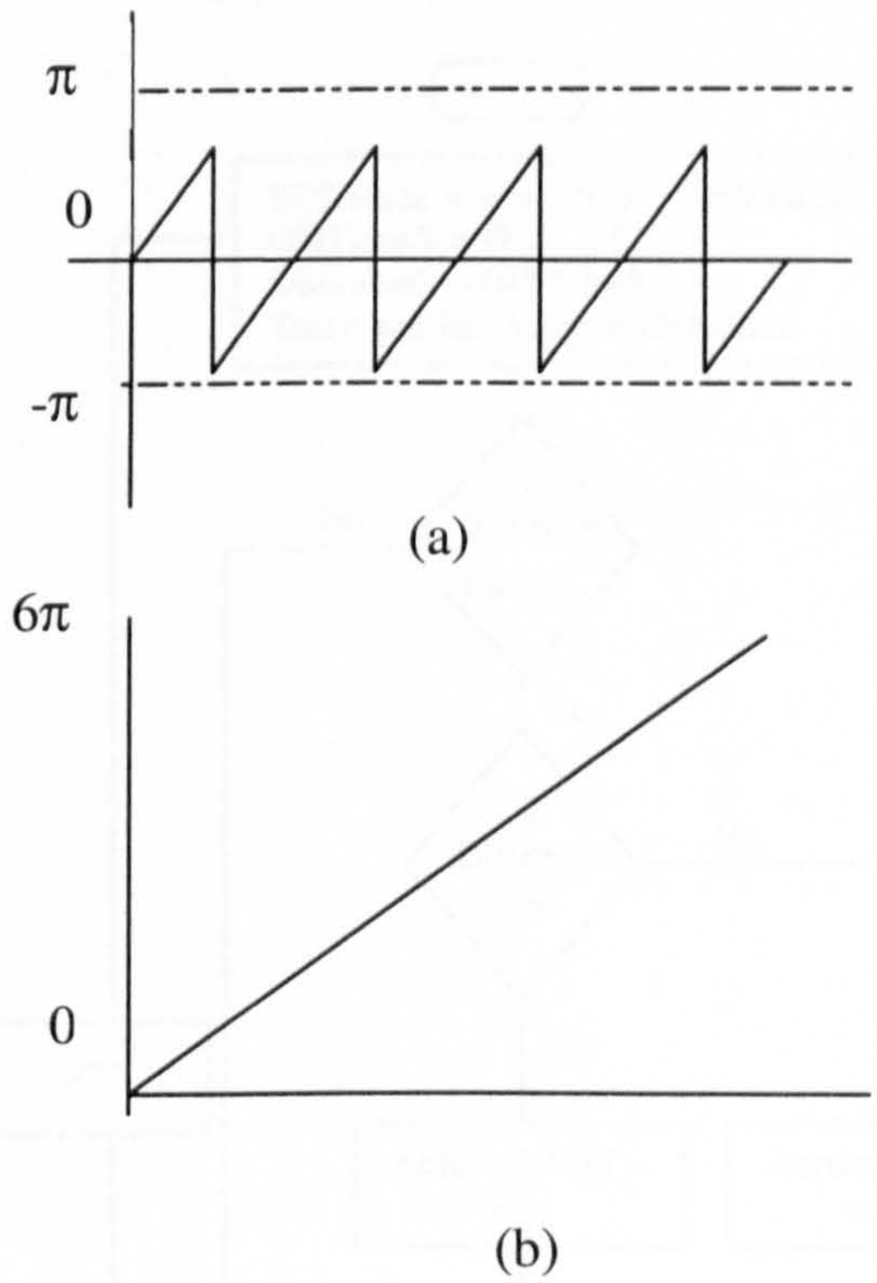


Fig.6.14 (a) Wrapped Phase Data (b) Unwrapped Phase data

There are a great number of algorithms to perform this function. The simplest form of unwrapping is to check for phase discontinuities within a line of phase data[7]. However although this works well for noise free images within a practical system, where noise often exists in the fringe images this is not the case. Therefore more sophisticated methods of removing these discontinuities have been developed[12,13,14].

For real-time phase extraction techniques the key issue in deciding which unwrapping technique to use arises from trying to find the correct balance between speed and resolution[6]. Further in parallel processing solutions the unwrapper

algorithm used is implemented on a number of processors. Therefore an algorithm which is suited to parallel implementation is essential.

6.6.1 Unwrapper algorithm

Essentially the unwrapper algorithm used in this work is of a pixel by pixel nature. However the value of a pixel is effected by its horizontal and vertical predecessor's values. In other words every pixel effects some of the nearest neighbour values.

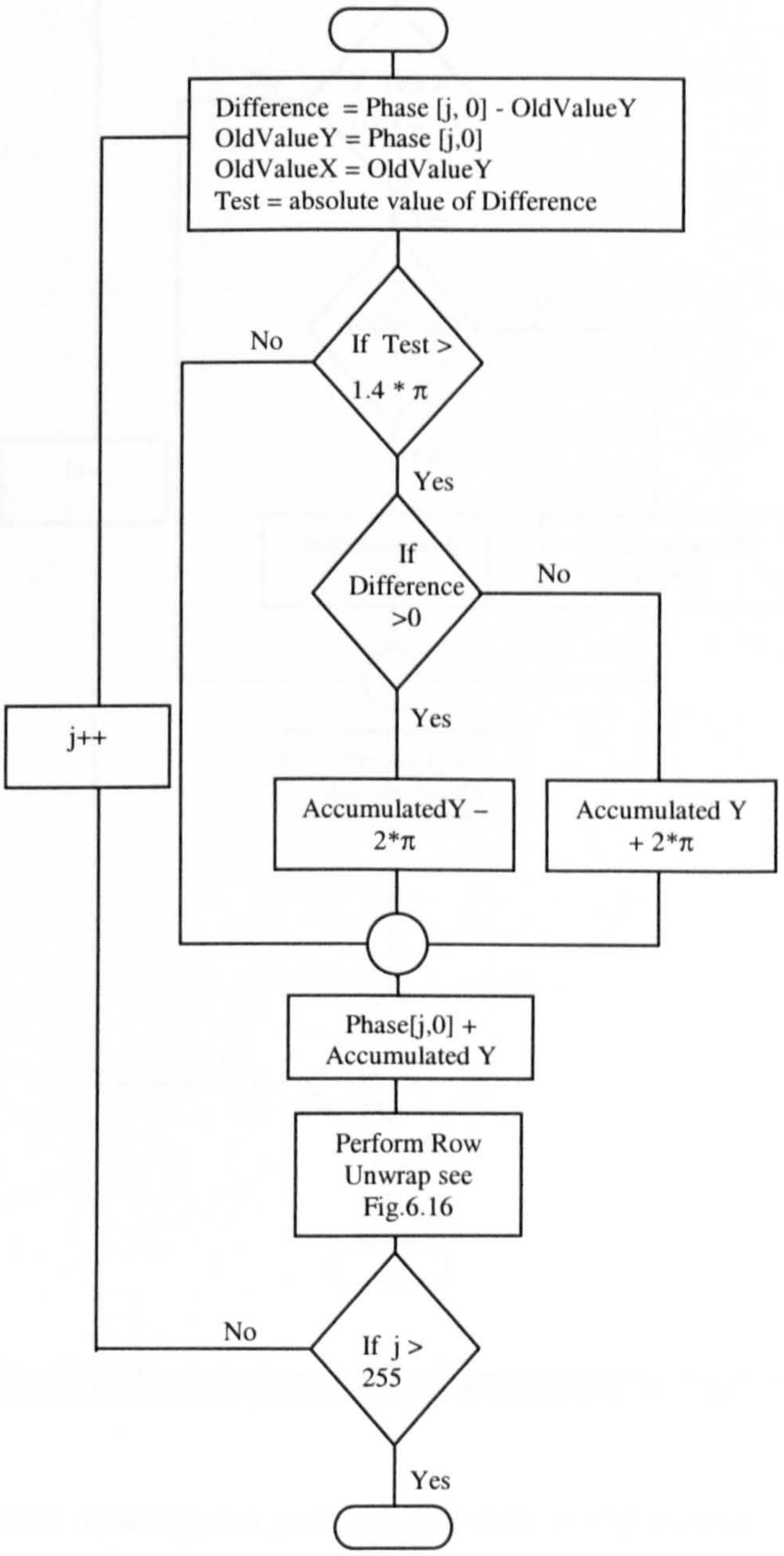


Fig.6.15 Flowchart column component of a single processor unwrapper

The flow chart in fig.6.15 shows how each column term is accounted for in the unwrapper used on a single processor system. Examining fig.6.16 shows how each of the rows is unwrapped. Using the accumulator values means that the neighbourhood feature of the data is accounted for in this technique.

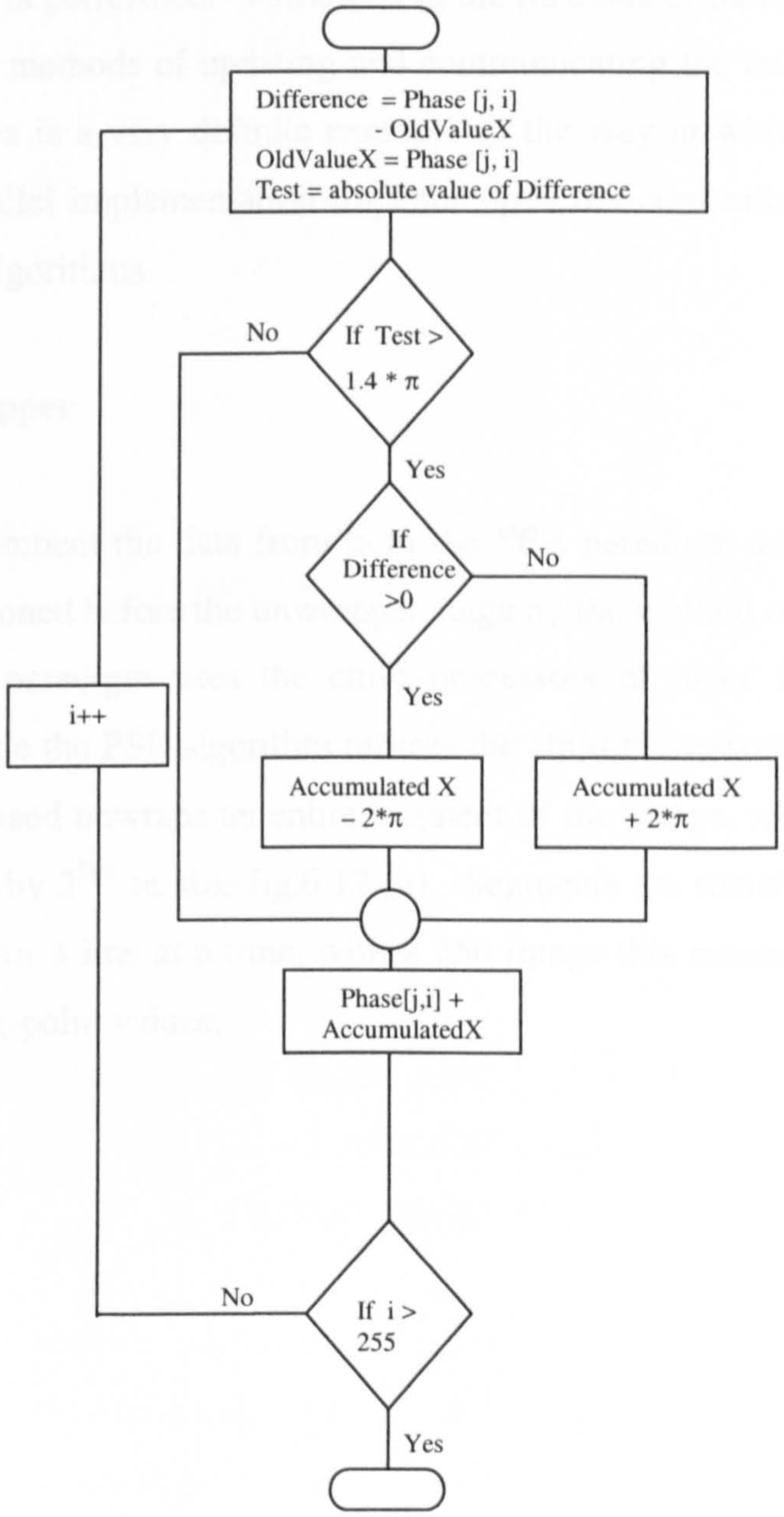


Fig.6.16 Row component of unwrapper in fig.6.15

All of the subsequent unwrappers perform the row component of unwrapping in the same method as fig.6.16. The variation in the code arises in how the column component of the unwrapper is performed. In essence this means how does the

accumulated Y value which updates the column component of the unwrapper become accounted for in each of the unwrapping methods.

The key to this problem is knowledge of how data is partitioned before the arc-tangent function is performed. Variations in the methods of partitioning data cause variations in the methods of updating and communicating the column accumulator component. This is a very definite example of the way in which a programmers selection of parallel implementation impends upon the conventional realisation of fringe analysis algorithms.

6.6.2 3L Unwrapper

In the 3L environment the data from both the FFA paradigm and PSP algorithms have been partitioned before the unwrapper stage by the method outlined in fig.5.11 (b). The FFA paradigm uses the child processors of layer 1 to perform this unwrapping, while the PSP algorithm utilises the child processors of layer 2. Each child processor used unwraps an entire segment of the image, where each of these segments is 2^{N-1} by 2^{N-1} in size fig.6.17 (a). Segments are transferred to and from the child processor a line at a time, with a 256 image this means 128 lines of 128 real-only floating-point values.

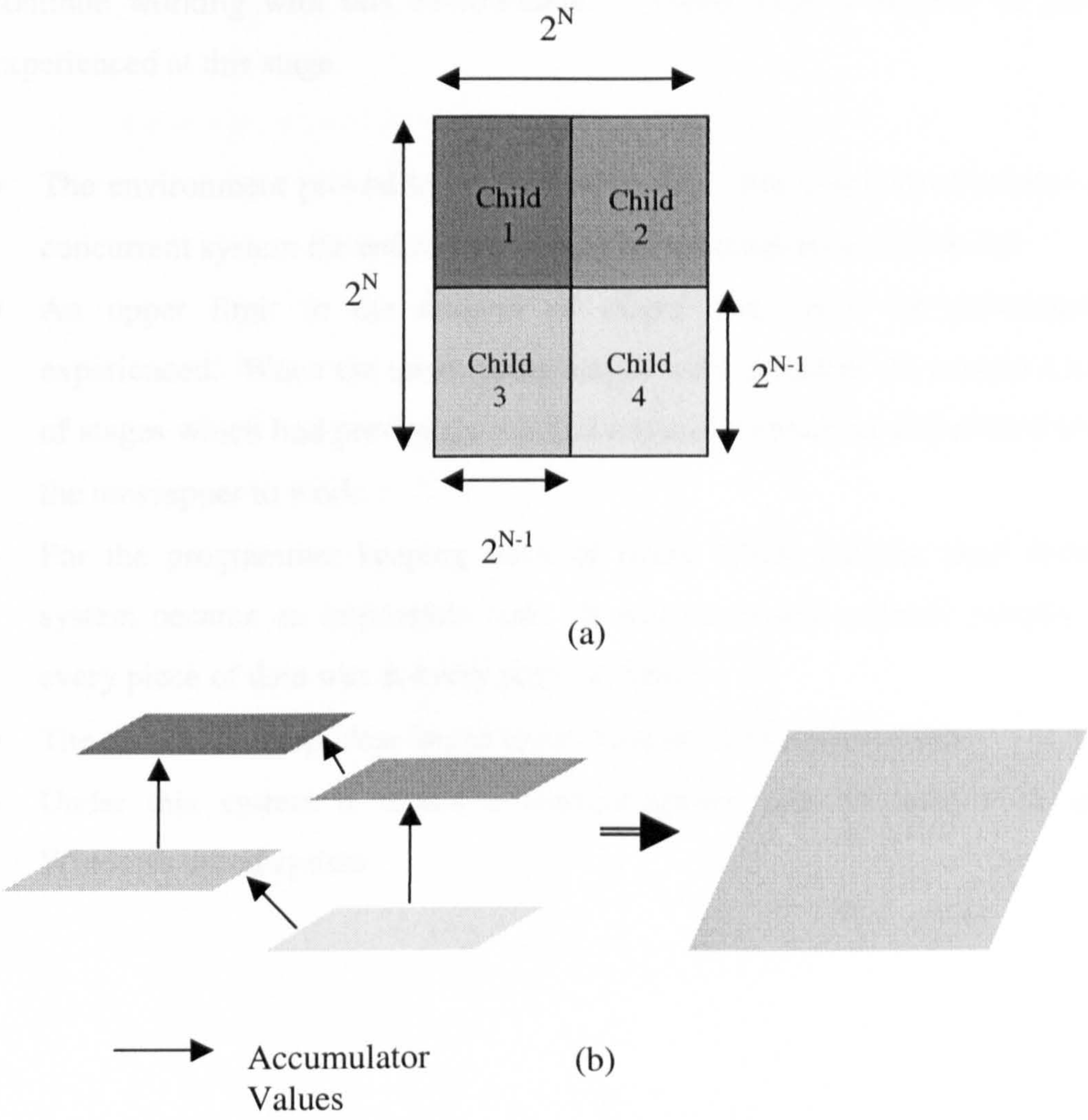


Fig.6.17 Segment based unwrapper. (a) Shows just how the phase data is unwrapped in segments. (b) The 4 unwrapped segments must be repositioned relative to each other.

Once each child processor has unwrapped a line of data this line is sent to the root processor for the FFA paradigm and to a child processor on layer 2 in the PSP algorithm. At this stage in either the FFA or PSP a further algorithm to combine the unwrapped data would be performed. This algorithm would compare the values of the pixels on the edges of the segments that must be combined to create one image. This is in effect another unwrapping stage.

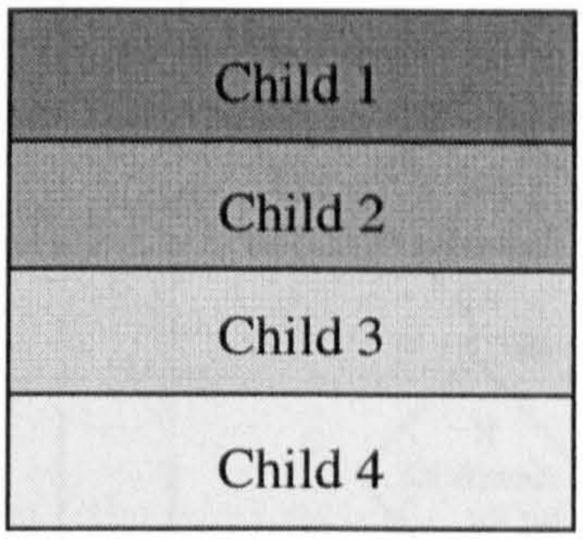
This work was all performed within the 3L environment and at this stage the problems that can arise from using the 3L environment made it impossible to

continue working with this environment. There were a number of problems experienced at this stage

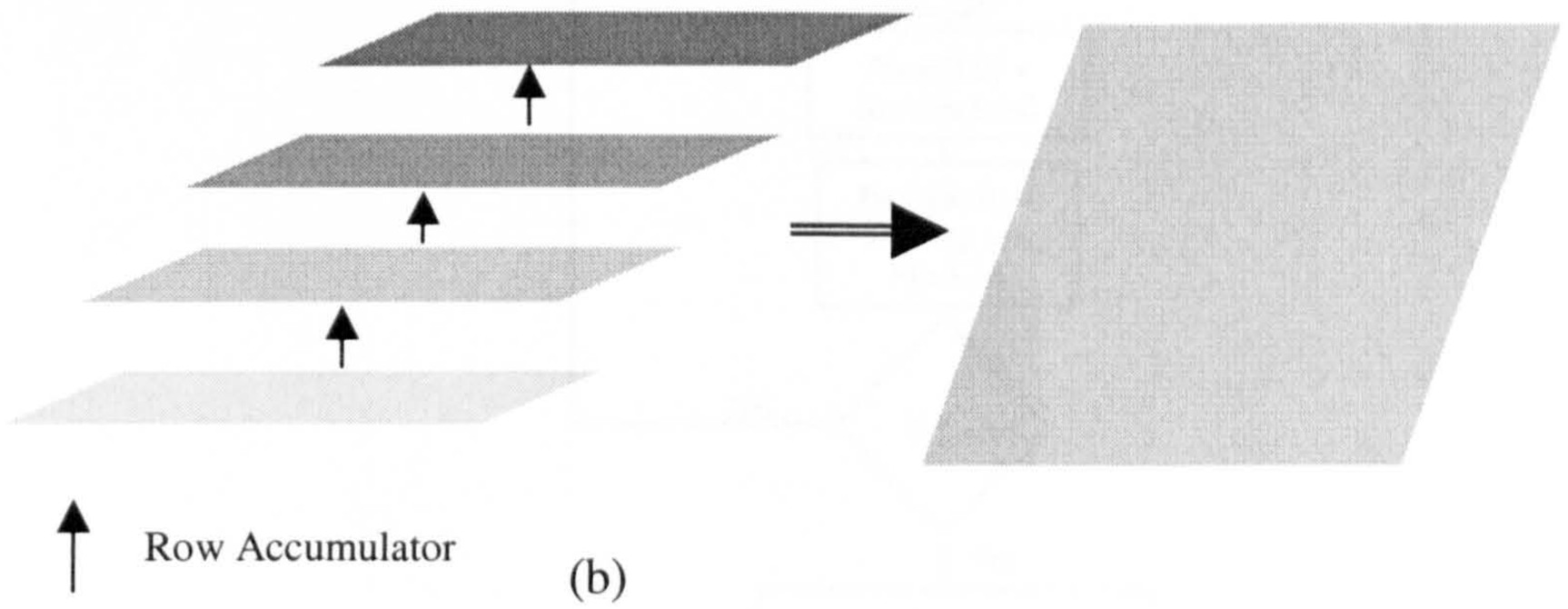
- The environment proved to be very inflexible. For every development of the concurrent system the entire code had to be re-compiled and re-tested.
- An upper limit to the number of stages that could be performed was experienced. When the unwrapping stages were added to the system a number of stages which had previously worked had to be modified and altered to allow the unwrapper to work.
- For the programmer keeping track of every DMA transfer used within the system became an impossible task. It was necessary to know exactly where every piece of data was at every point in time.
- The time to develop ideas began to increase at an exponential rate.
- Under this system it is not a straightforward task to achieve an elegant Windows based system.

6.6.3 Pegasus Strip unwrapper

In the Pegasus system which uses the scheme 2 FFA paradigm data is split into 4 strips using the strategy outlined by fig.5.11 (a). This method of division is used for both the PSP algorithm and FFA paradigm fig.6.18. Unwrapping of the FFA data occurs on the layer 1 child processors within both the FFA-only and the concurrent strategy. The PSP unwrapping will be performed on the child processors of layer 2, while the combining function will be performed on one of the layer 2 child processors for PSP and the root processor for FFA.



(a)



(b)

Fig.6.18. Strip unwrapper in the Pegasus system. (a) Showing how the phase information is split into strips in one of the Pegasus unwrappers. (b) Showing the need to recombine these strips to get a continuous surface.

For an image of 256 by 256 in size each processor receives a strip that is 64 lines in size. Each of these lines consists of 256 data points. These lines are sent in DDU of 8 lines at a time, which means that each child processor receives eight such DDU. Once these DDU have been unwrapped they are sent to the appropriate processor to be stored and then joined with the other strips from the other child processors on the same layer.

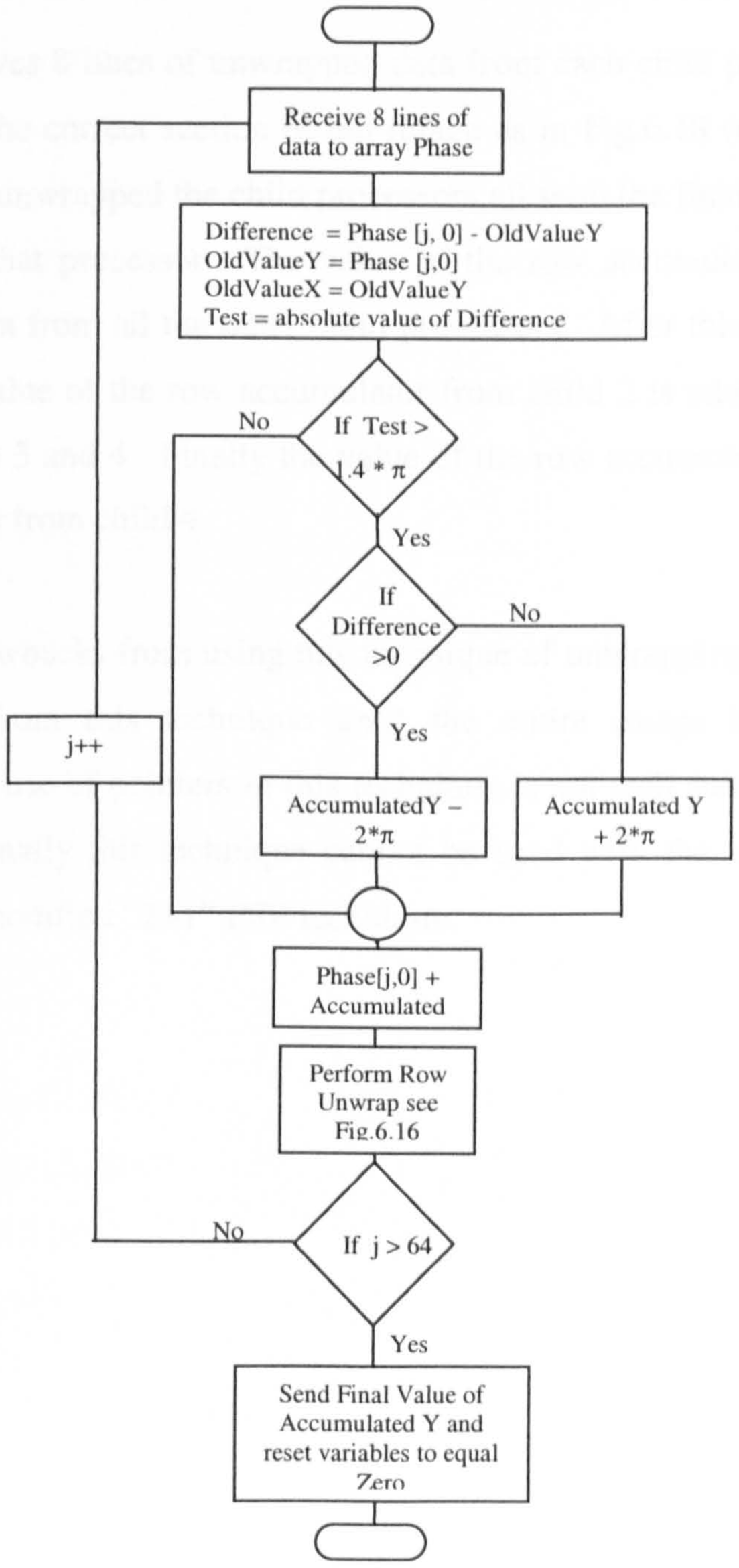


Fig.6.19 Pegasus Strip Unwrapper Column Flowchart

The main difference between fig.6.19 and that of the single processor, fig.6.15 is that the loop is performed for 2^{N-2} times rather than 2^N. This means that the variables are reset after 64 unwraps have been performed by each child processor. Additionally it can be seen that the final value of the row accumulator is sent to the combining function on the final loop of each image. This is to allow the combining task to be performed very effectively. Note the row unwrap is identical to fig.6.16.

The combining task receives 8 lines of unwrapped data from each child processor. These are then stored in the correct section of the image as in Fig.6.18 (a). Once the entire image has been unwrapped the child processors all send the final value of the row accumulator on that processor. The value of the row accumulator from child 1 is added to the data from all the other child processors. After this addition has been performed the value of the row accumulator from child 2 is added to the data from child processors 3 and 4. Finally the value of the row accumulator from child 3 is added to the data from child 4.

There are a number of drawbacks from using this technique of unwrapping. Firstly no data can be saved from this technique until the entire image has been unwrapped. Secondly the use of pointers in this technique is not well suited to the Pegasus environment. Finally this technique cannot be used with the scheme 1 FFA strategy or with the modified "2+1" PSP technique.

6.6.4 Pegasus Non-consecutive strip unwrapper

Under the scheme 1 strategy of the FFA paradigm the data to be unwrapped is in the format of fig.5.13, this means the child processors used to perform the unwrapper receive the partitioned data in the scheme of fig.6.20. In the concurrent strategy both the FFA paradigm and PSP algorithm will use this method of data division. The child processors of layer 1 perform the FFA unwrap, while the child processors of layer 2 perform the PSP unwrapping. Similarly to the previous two strategies of unwrapping a combination function is required for each phase extraction technique. For the FFA paradigm the root processor performs this combination and a child processor in layer 2 performs the PSP combination function. It is worth noting that the child processor chosen to perform combination of PSP data must not be the same child as has been used to perform PSP pre-processing.

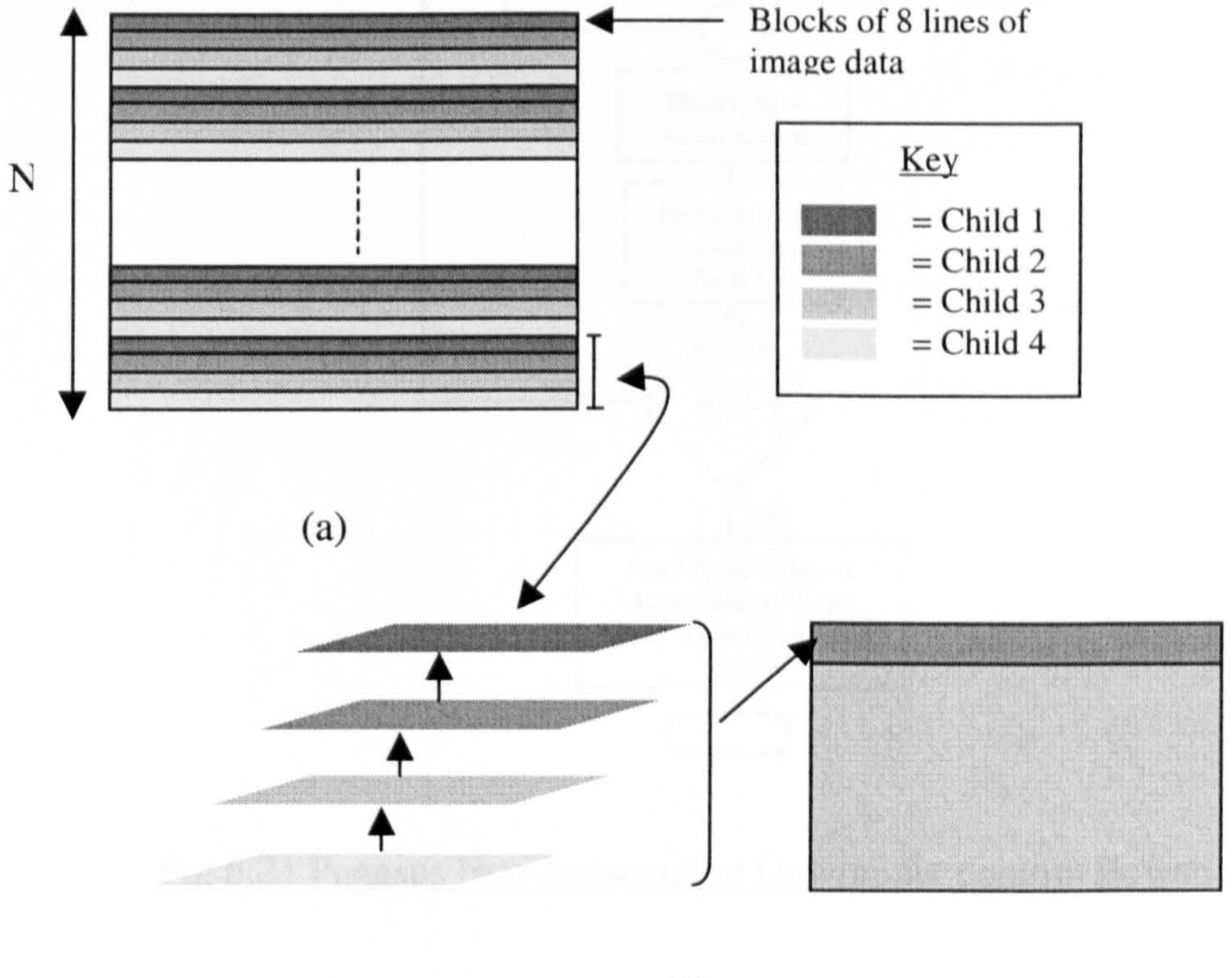


Fig.6.20 Non-consecutive strip unwrapper under Pegasus (a) Image division for unwrapping. (b) The repositioning required for 4 consecutive strips made of 8 lines of data each.

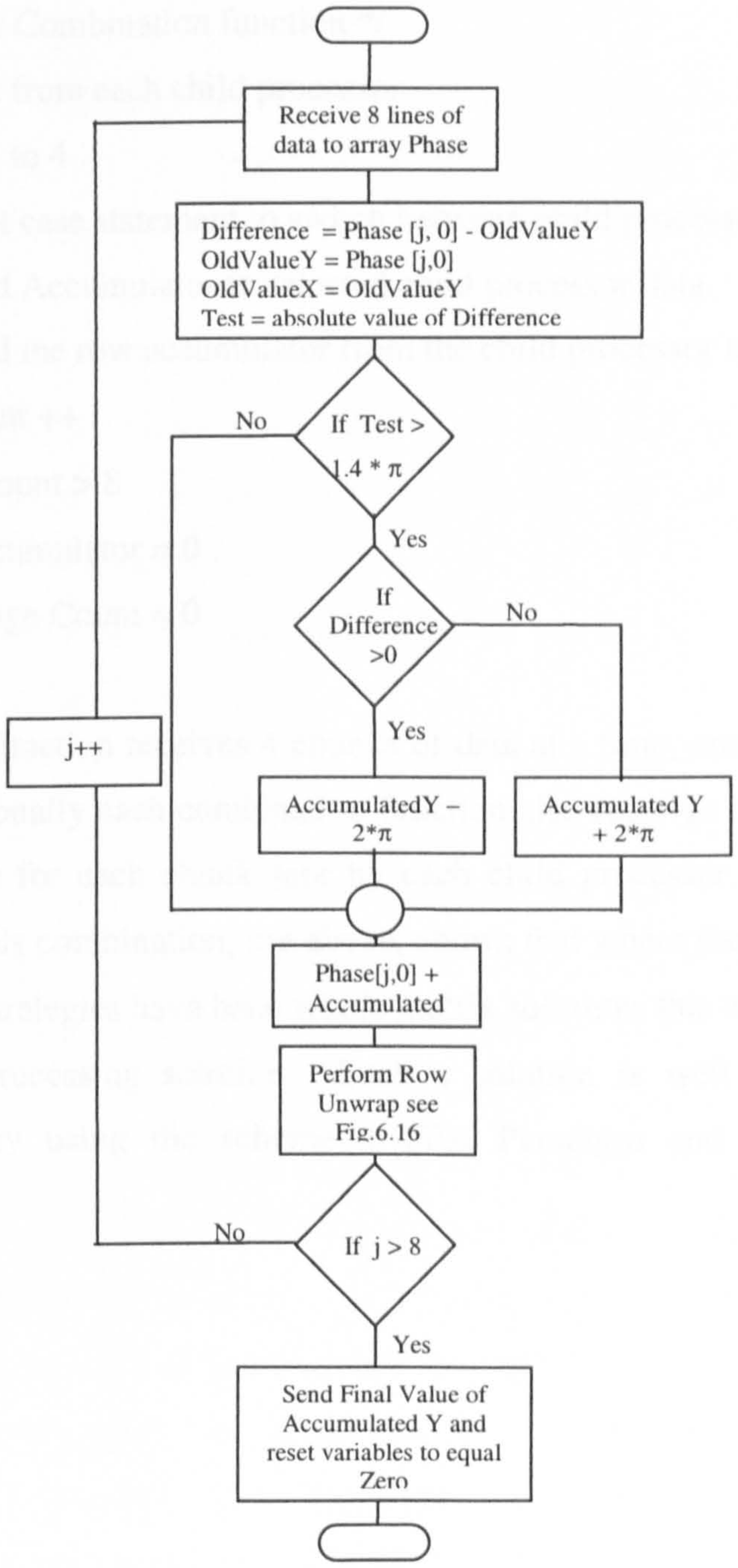


Fig.6.21 Pegasus Non-consecutive Unwrapper column flowchart

Each child processor unwraps 8 lines of data at a time, where each child processor operates upon a total of 2^{N-2} lines. This means that for a 256 square image each child processor operates upon 8 DDU of 2048 data points for each image. Examining the flowchart in fig.6.21 for this technique reveals that the algorithm used to perform this varies from fig.6.19 due to the accumulator value being sent at the end of **every** DDU of 8 lines.

```
/* Code for Combination function */
Store Input from each child processor
For loop=1 to 4
        Test case statement to switch between child processor data
        Add Accumulator to selected child processor data
        Add the row accumulator from the child processor to Accumulator
        Image Count ++
If Image Count > 8
        Accumulator = 0
        Image Count = 0
```

The combination function receives 4 chunks of data at a time, one from each child processor. Additionally each combination function also receives an accompanying accumulator value for each chunk sent by each child processor. Examining the pseudo code for this combination, see above, shows that where the combination for the previous two strategies have been whole matrix solutions this technique is much more of signal processing solution. Such a solution is well matched to the concurrent strategy using the scheme 1 FFA Paradigm and the "2+1" PSP algorithm.

6.7 Summary

In this chapter each stage of FFA and PSP have been discussed with reference to parallel implementations of these techniques within two parallel software environments. There are a number of key points:

FFA

- That Fourier Fringe Analysis is a coarse grained parallel problem to which there is no one optimum solution but rather a number of compromise solutions.
- That every stage of the FFA strategy is changed by being implemented in the parallel environment
- That 2D FFA is not suited to implementation in a parallel C40 system. Implementing this scheme forces the use of corner turn blocks to ensure that the correct data is used at each processing stage of the FFA system.
- That 1D FFA although not used in single processor systems is well suited to implementation in a parallel C40 system
- Pre-processing causes a delay of the system, as it cannot be performed until the entire image has been stored. However, the benefits from pre-processing in terms of unwrapping mean that pre-processing should be implemented.
- Filtering in a parallel C40 system is a highly complicated task. With a 1D scheme every line of the image must be filtered and frequency shifted if required. In a 2D scheme the data will be stored in a non-conventional manner, which makes finding the correct components of the 1st order harmonic extremely difficult.

PSP

- That Phase Stepping is a fine-grained problem. The main problem in PSP is how to partition data
- PSP is changed by the application rather than the parallel implementation

- Being pixel by pixel in nature PSP is a much simpler data problem than FFA
- The key problem of PSP is data manipulation. How data is stored and divided will impact upon the performance of the entire system and especially the unwrapping stage
- Performing the PSP technique concurrently with the FFA technique has lead to a modified "2 + 1" implementation.

Unwrapping

- That unwrapping techniques tend to be fine grained in texture.
- The main problems in parallel unwrapping arise from the data partitioning schemes used.
- That the unwrapping strategy chosen effects how the FFA and PSP techniques can be performed
- The communication of the accumulator values used within the unwrapping stage are a key problem
- Segment data division may be the most obvious method of partitioning data in a fringe analysis system but this is not the case in a parallel C40 implementation.
- Using strip unwrappers is the best method of performing this stage of either of the Fringe analysis techniques used in the parallel processing system.

The next chapter contains examples from the systems used in this work. Additionally in the final chapter the ideas that have been described in this chapter are discussed with reference to the goals identified for this research in section 1.2.

6.8 References

1. Peidra.R.M
   Parallel 1-D FFT Implementation with TMS320C4x DSPs
   Parallel Processing with the TMS320C4x – Application Guide, TI, **1994**.
2. Brigham.E.O
   The Fast Fourier Transform and its Applications
   Prentice-Hall International Inc, **1988**.
3. Peidra.R.M
   Parallel 2-D FFT Implementation with TMS320C4x DSPs
   Parallel Processing with the TMS320C4x – Application Guide, TI, **1994**
4. Kreis T.M. and Juptner W.P.O.
   Fourier-transform Evaluation of Interference Patterns: The Role of Filtering in the Spatial Frequency Domain.
   SPIE Vol. 1162 Laser Interferometry Quantitative Analysis of Interferogram, pp. 116 – 125, **1989.**
5. Halsall G.R.
   Real-time Non-contact Profilometry.
   Ph.D. Thesis Liverpool Polytechnic, **1992.**
6. Al-Hamdan S.
   A Comparison of Two Parallel Computer Architectures in the Context of Interferometric Fringe Analysis.
   Ph.D. Thesis Liverpool John Moores University, **1996.**
7. Burrus.C.S and Parks.T.W
   DFT/FFT and Convolution Algorithms, Theory and Implementation, Chapter 2
   John Wiley & Sons, **1985.**
8. Takeda M, Ina H, and Kobayashi S.
   Fourier-transform Method of Fringe-pattern Analysis for Computer-based Topography and Interferometry.
   Jn.Opt.Soc.Am., Vol. 72, No. 1, pp. 156 – 160, **1982.**

9. Burton.D.R et al.
The Use of Carrier Frequency Shifting for the Elimination of Phase Discontinuities in Fourier Transform Profilometry
Optics and Laser in Engineering, Vol.23, pp. 245 – 257, **1995.**

10. Gasvik.KJ
Optical Metrology Chapter 11
John Wiley & Sons, 2nd Edition, **1995.**

11. Robinson and Reid
Interferogram Analysis: Digital Fringe Pattern Measurement Techniques, Chapter 4.
IOP Publishing, **1993**.

12. Stephenson.P, Burton.D.R and Lalor.M.J
Data Validation in a Tiled Phase Unwrapping Algorithm
Optical Engineering, Vol.33, No.11, pp.3703 –3708, **1994.**

13. Herraez M et al.
Robust, Simple and Fast Algorithm for Phase Unwrapping
Applied Optics, Vol.35, No.29, pp.5847 – 5852, **1996.**

14. Huntley.J.M & Saldner.H.O
Shape Measurement of Discontinuous Objects Using a Spatial Light Modulator and Temporal Phase Unwrapping
Fringe '97, Akademie Verlag, pp.156-163, **1997.**

15 Womack.K.H
Interferometric Phase Measurement Using Spatial Synchronous Detection.
Optical Engineering, Vol.23, No. 4, pp.391-395, **1985.**

Chapter 7

Results

7.1 Introduction

The aim of this chapter is to discuss the results obtained from the parallel fringe analysis systems described in the previous chapters. Firstly some basic comparative timings are discussed. Timings are presented for each of the software environments used in this work. These timings are used to highlight the differences between the two environments, although the nature of the timing results makes any direct comparisons difficult. Thus the timing results used in this chapter are not intended to be used as absolute terms but rather represent relative performances. Each function is performed a number of times and the timing result for each function is an average of these performances.

Timings can be used as nothing more than comparative results because of the nature of the timing devices that are utilised within the C40 environment. Further the act of timing for any function will degrade the speed performance of that function by at least ten percent. This is due to the threaded nature of the timing mechanism and the print statement used to display the results obtained. Additionally the timing result of an individual function will be effected by a number of parameters such as the image being analysed and the host PCs workload. Therefore the timings in this chapter are nothing more than guides to how software environments and algorithmic functions can be compared. Measured times only are quoted in this work.

The next section of this chapter compares results, in terms of images, from the final concurrent phase extraction techniques with the results from a Windows PC Based FFA system. Comparison of these results helps to highlight the

differences between a parallel processing solution and a Windows based PC solution.

All the examples quoted within this work use an image size of 256 pixels by 256 pixels unless otherwise stated. Where 1D FFTs are discussed a complete set of 256 row only transformations has been performed. In the case of 2D FFTs both row and column FFTs have been performed. Additionally within a parallel-processing 2D FFT implementation there is an implied corner turn performed.

7.1.1 Windows PC Based FFA System

The PC based FFA system is implemented under the IDL environment. This system forms an alternative solution to the difficulties presented by the INFOCUS problem. Further this system acts as baseline for phase extraction techniques implemented within the Coherent and Electro-Optics Research Group. Therefore comparing the image results of this system with the parallel processing solution enables some qualitative discussion on the image results obtained within the Pegasus environment.

7.2 3L Results

The aim of this section is to discuss the results obtained from using the 3L environment. Many of the lower-level results form the basis for the later work within the Pegasus environment and are therefore not repeated within that environment.

7.2.1 Internal Ram Blocks

Each C40 processor has two internal ram blocks in which the speed to perform any code utilised can be optimised. In Fig.7.1 a complex 256 FFT function has been performed within one of the internal ram blocks of the C40. For comparison the same algorithm has been implemented within the local EDRAM of the C40, with an allocated contiguous block of memory.

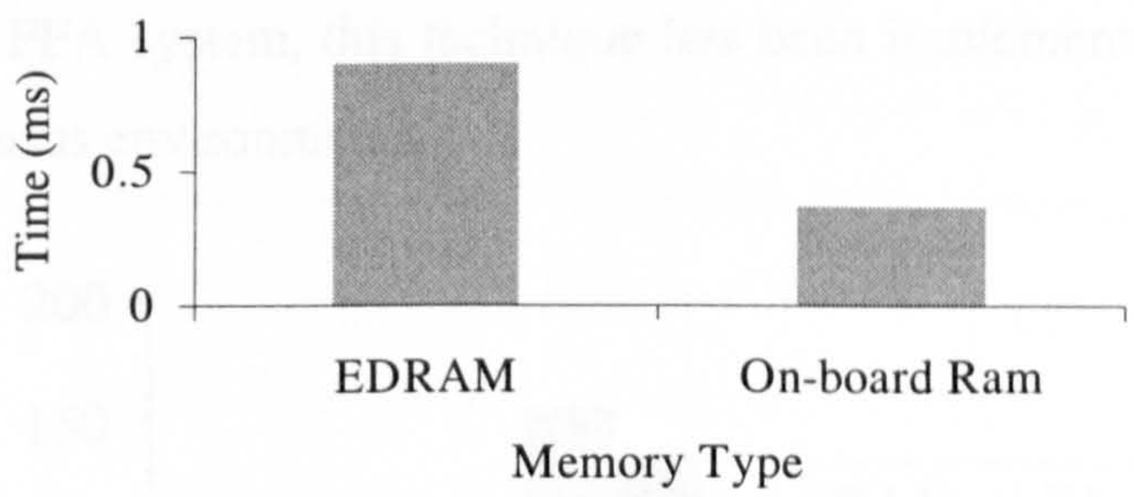


Fig.7.1 Comparing the EDRAM performance of a 256 complex FFT with the performance of the same function in the C40 internal ram memory.

These results show that for every FFT that must be performed within a FFA system there is an advantage from using the internal ram blocks. For a single line of a 256 image there is an approximate saving of 0.4ms, which does not seem that great an amount. This of course would be the case **if** only a single FFT was required. The comparative saving from using internal ram blocks for 256 lines is approximately 100ms.

This benefit arises partially from the faster access times that can be used with this memory and partial from the specialised code that has been used to perform the FFT. Where FFTs have been performed they rely on code that has been designed to utilise a great number of the key features of the C40 co-processor and this is reflected in the decrease in times when the internal memory is used. Additionally the contiguous nature of the internal memory means that an array of the required size can easily be implemented, whereas in the local memory a special form of memory allocation must be used.

If both internal ram blocks on the C40 processor are used there is a further advantage when compared to a single block of memory. This can be used advantageously to overlap communication and calculation. This idea has been used in this research within the 3L environment. Alternatively the two ram blocks can be used to overlap processes performed on a processor. An example of this is the overlapping of the real-only FFT and complex inverse FFT in the 1D FFA system, this technique has been implemented within both the 3L and Pegasus environments.

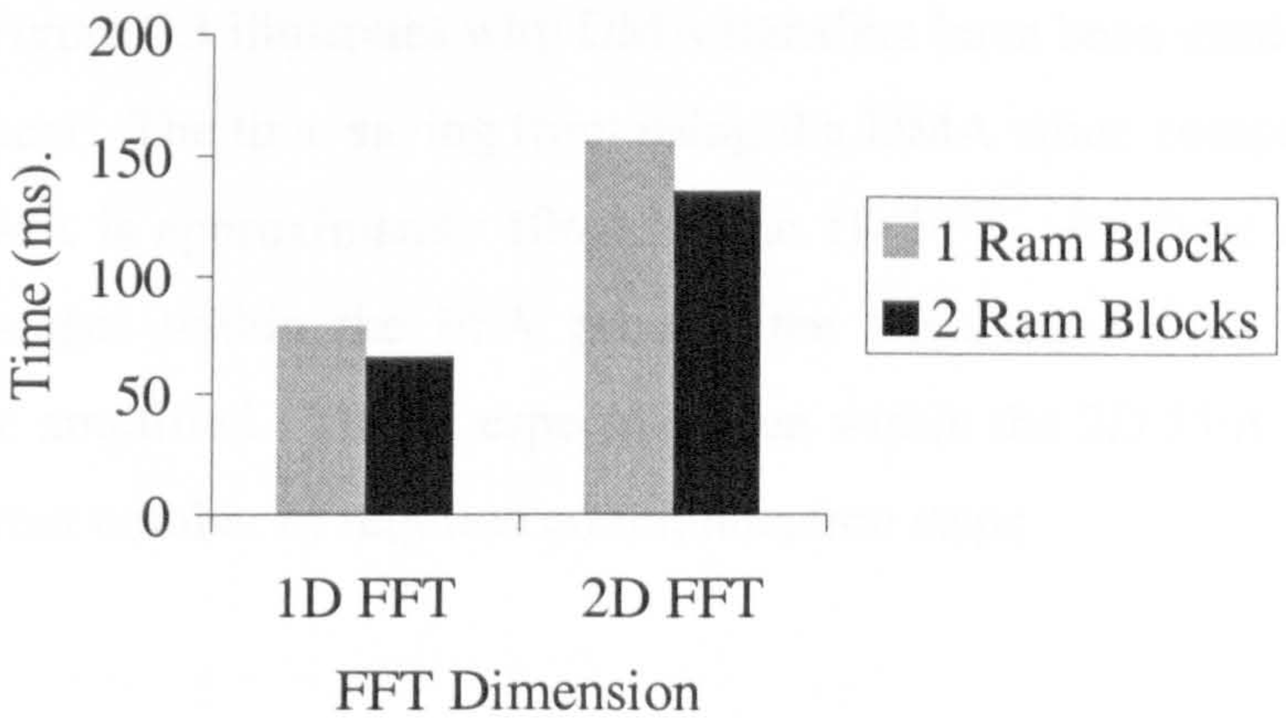


Fig.7.2 Comparing the performance of two internal RAM blocks when performing FFTs with that for one internal RAM block

As can be seen in fig.7.2 there is an effective speed up in the performance of both 1D and 2D FFTs when two internal ram blocks are used rather than one

internal ram block. This effect is amplified with the number of stages of the FFA algorithm that must be performed by the C40 processors. It should be noted that the times quoted for both 1D and 2D FFTs include the time to move the data between the internal ram blocks and the EDRAM. For this reason the 1D FFT is not simply a multiplication of the result from fig.7.1 by the number of lines in the image.

The time to perform a complete 1D FFT within 2 Ram blocks is approximately 15ms faster than using a single ram block. With a 2D FFT this time saving is increased to approximately 20 ms. As the number of stages increases so does the advantageous time saving from using both RAM blocks. This means that within FFA where there are a number of stages that can use the internal RAM blocks this saving becomes increasingly important.

7.2.2 DMA transfers

Using the DMA co-processor to move data changes the code used for an algorithm, additionally it changes how the programmer must think about that algorithm. Figure 7.3 illustrates why DMA transfers have been used within the 3L environment. The time saving from using the DMA when compared to not using the DMA is approximately 10ms for the 1D FFT. As there are a great number of stages within the FFA process the advantages from using this technique are amplified. This is especially true within the 2D FFA system, as there are a great number of required communication steps.

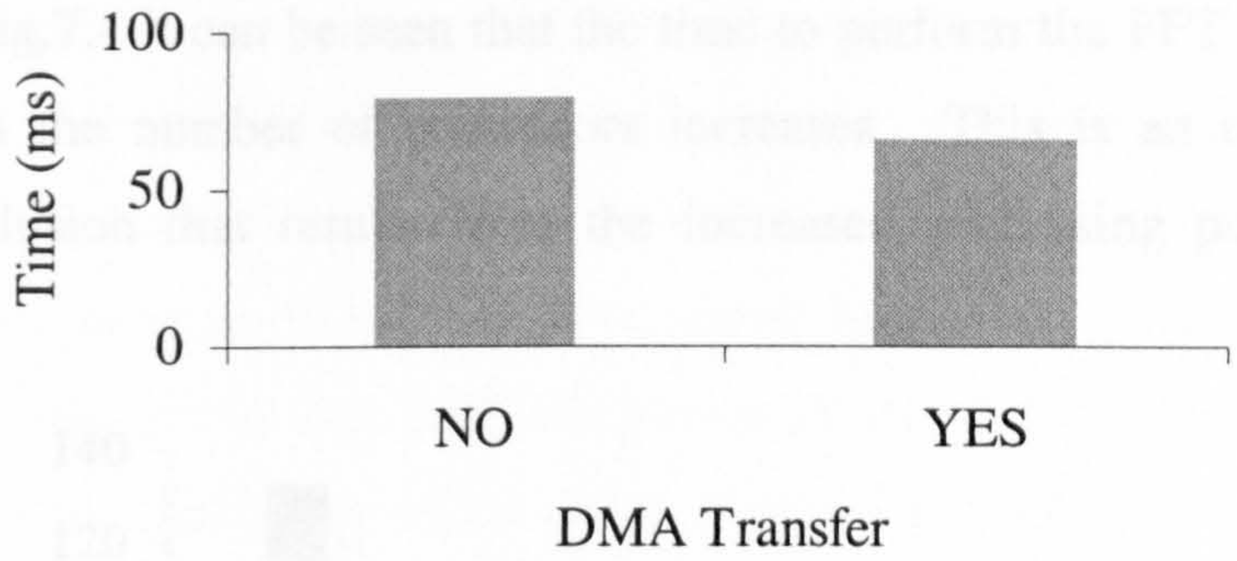


Fig.7.3 Comparing 1D FFT timings using DMA transfers with those not using DMA transfers

Purely in terms of the speed to perform a function using the DMA is indeed a powerful and effective tool. However, there are a number of problems that can arise with this technique. Firstly debugging and fault finding become more problematic. Secondly the time to code an algorithm is greatly increased. Thirdly the programmer must have absolute knowledge of data flow. These problems are effectively multiplied when virtual links are used with the parallel environment.

Therefore in the Pegasus system where any DMA transfers are performed the software environment is used to control the performance of these transfers.

7.2.3 Number of Processors to Use

The network topology of the INFOCUS sensor contains 10 C40s in a topology that has been described in section 5.3.4. The key reason behind the final topology designed was to allow a number of solutions to solving problems to be investigated. One of the results of this is that there are a number of possible permutations of how many processors may be used to perform any one stage of the FFA paradigm. For example 1 to 8 processors within this system can be used to perform the forward 1D-FFT or 2D-FFT functions.

Examining fig.7.4 it can be seen that the time to perform the FFT functions is decreased as the number of processors increases. This is an obvious and logical conclusion that results from the increased processing power of the system.

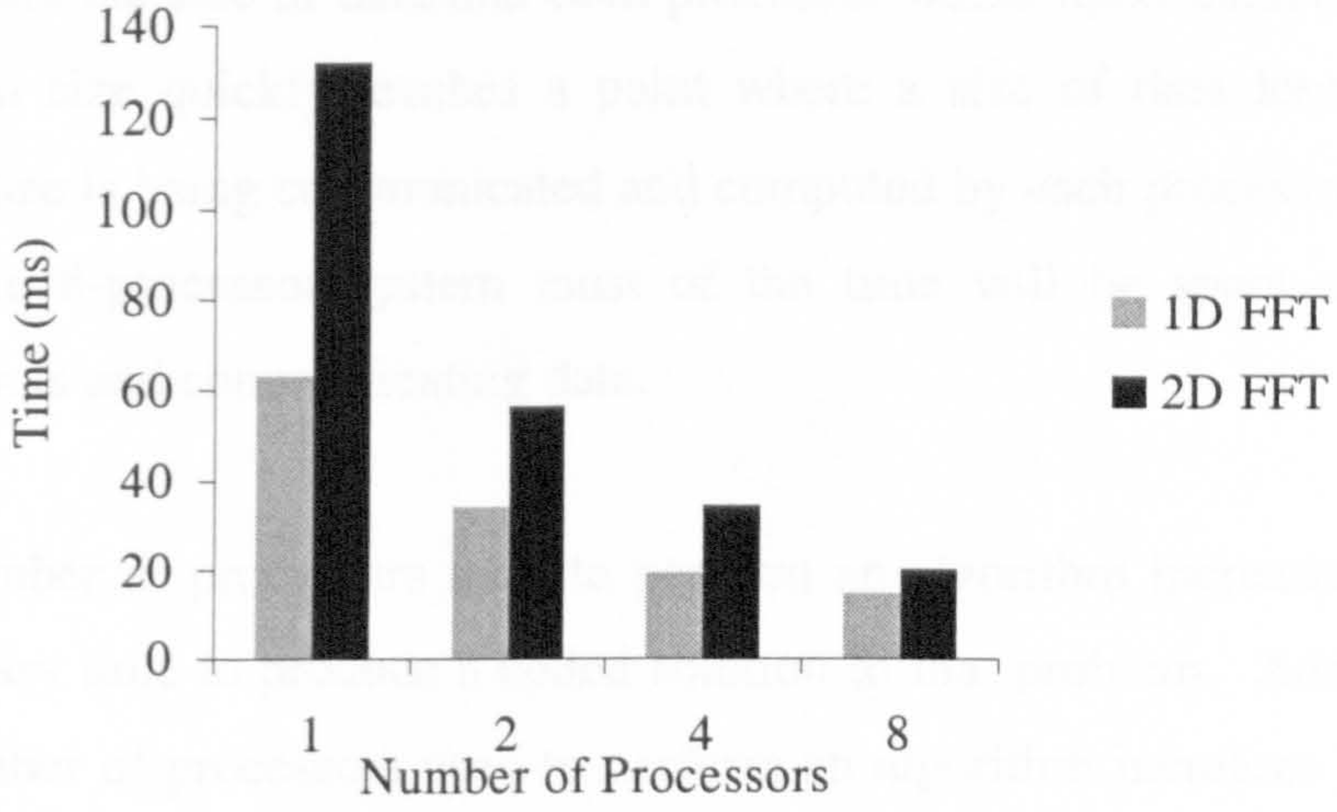


Fig.7.4 Showing decreasing performance times for both 1D and 2D FFTs as the number of processors used increases

Examining the comparative timings results from fig 7.4 it can be seen that as the number of processors used to perform a function doubles so the time to perform that function approximately halves.

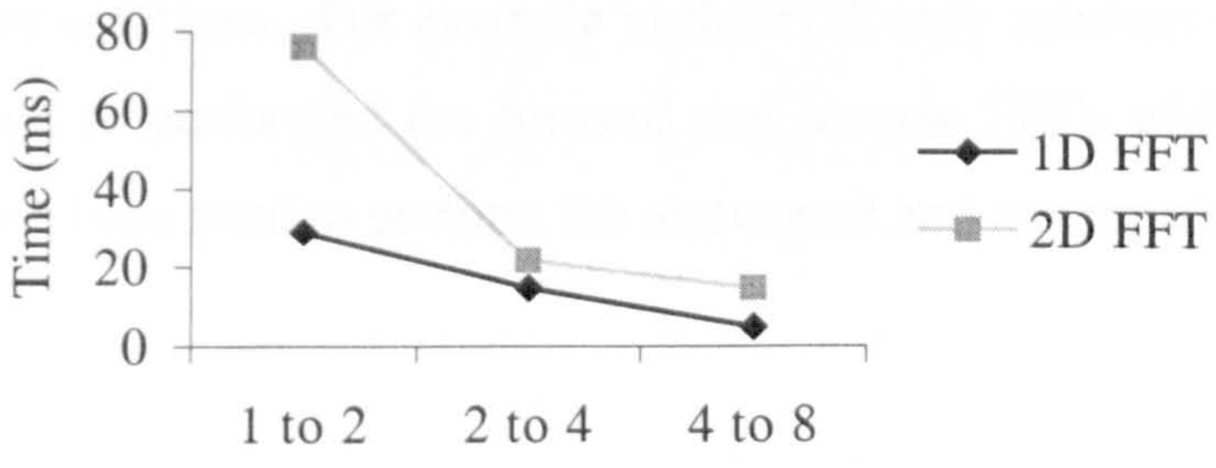


Fig.7.5 Graph Showing as the number of Processors used to perform a function increase the resulting differential lessens

Examining fig.7.5 it can be seen that the differential advantage of using more processors begins to decrease rapidly where more than 4 processors are used to perform one function. This is the case because of the number of processors that a C40 may effectively communicate with; this is limited to 4 processors at a time. Additionally as data begins to be partitioned for an increasing number of processors the size of data that each processor works upon decreases. This decrease in size quickly reaches a point where a size of data less than the optimum size is being communicated and computed by each processor. In the case of the 8-processor system most of the time will be spent arbitrating between links and communicating data.

As the number of processors used to perform an algorithm increases so does the necessary time to produce a coded solution to that problem. Additionally, as the number of processors used to perform an algorithm increases so do the number of communication links required. When greater than 4 processors are used to perform an algorithm such as the 2D FFT a large number of virtual connections are required. In the 3L environment where physical links are used the programmer must perform this threading of data through a number of processors to allow calculations to be performed.

The key effect of these factors is that where required four processor solutions to discrete stages of the phase-extractions techniques have been preferred to eight processor solutions. For example in the FFA only solution 4 processors have been used to performed the forward and inverse FFTs while another 4 processors have been used to perform the arctangent and unwrapping stages.

7.2.4 Distributed v Centralised Storage

Within chapter 5 two data storage schemes were outlined, namely centralised and distributed storage. Each of these schemes has relative advantages and disadvantages. In terms of comparative timing distributed storage is relatively faster than a centralised storage scheme.

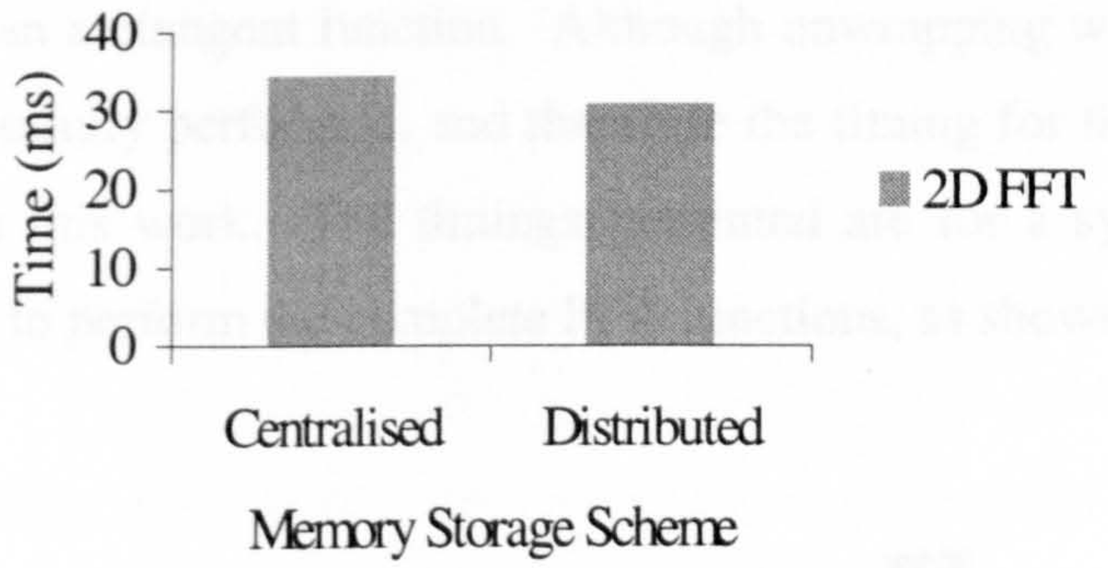


Fig.7.6 Comparing the time to perform a 2D-FFT functions using two data storage strategies.

Examining fig.7.6 it can be seen that there is a small difference between the performance of the two strategies in terms of milliseconds. Distributed storage is approximately 2 ms faster than the centralised storage scheme. Such a scheme must be performed a total of 4 times in the FFA strategy implemented in the 3L environment. This means that for a full FFA system there is 8ms advantage from using distributed storage rather than centralised storage.

Viewing this comparative time saving in the context of the INFOCUS project means that this advantage is small in practical terms. The small decrease in time for the distributed scheme when compared to the centralised scheme does not show the increased time required for programming the distributed scheme. Programming of the distributed scheme is not only more complicated but also much harder to debug. Finding out exactly where data is and how it is being

moved at each point in time is a very demanding problem. Additionally such a system can only be coded for an exact set-up of processors. This means such code is not object-orientated and therefore not easily reused.

7.2.5 FFA Stages Implemented with 3L

Within this environment a FFA system based upon the scheme 2 strategy outlined in section 6.1.1 was successfully perform up to and including the performance of an arctangent function. Although unwrapping was undertaken it was not successfully performed, and therefore the timing for this function is not presented in this work. The timings presented are for a system using 8 child processors to perform the complete FFA functions, as shown in fig.5.9.

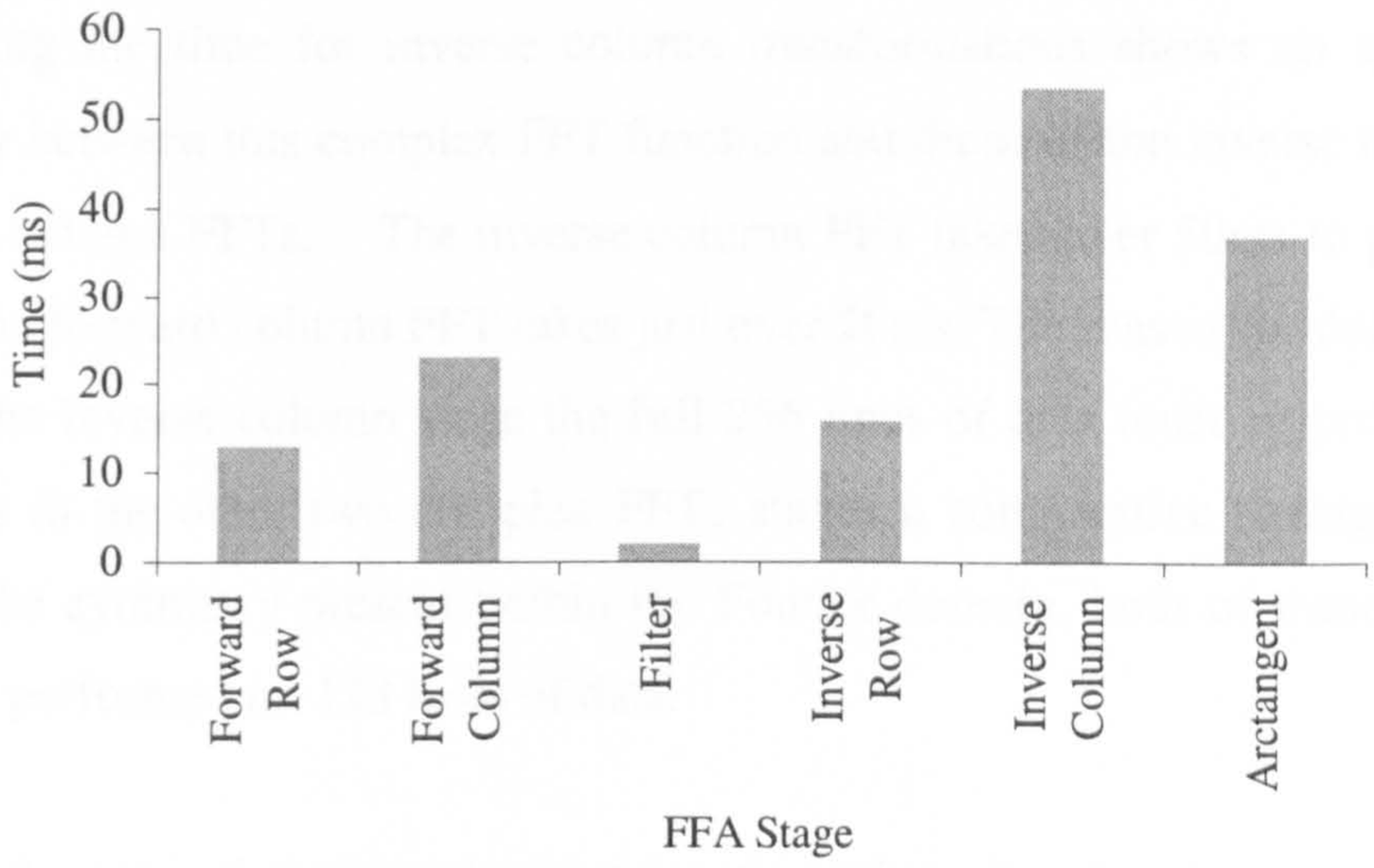


Fig.7.7 Comparing the stages of the FFA system performed under the 3L environment

By reviewing fig.7.7 a number of the stages of FFA can be compared. These timings include the time to perform the corner turn stages associated with each processing stage. Each of these corner turns takes approximately the same time as the filter stage to undertake the required data manipulation. However,

it should be noted that these timings do not include the time to perform communications.

The forward row stage consists of 256 real-only lines being transformed. Comparing the time to perform this function with the forward column FFT the benefits from performing this real-only FFT can be seen. Although the forward column FFT is only performed for 128 lines of data the real-only FFT is still some 10 ms seconds faster. In fact comparing this real-only FFT of 256 lines with the inverse column FFT of 256 lines shows the real-only FFT is nearly 40ms faster. Further with a larger number of transformations the differential between these two FFT techniques would be more significant i.e. with a 512 image the time saved would be increased by a factor of 4.

Reviewing the time for inverse column transformations shows an apparent disparity between this complex FFT function and those of the inverse row and forward column FFTs. The inverse column FFT takes over 50ms to perform whilst the forward column FFT takes just over 20ms. The reason for this is that within the inverse column stage the **full** 256 lines of data must be processed. Whereas in the other two complex FFTs stages a comparative saving occurs due to the symmetry present within the Fourier domain, both of these stages are only performed on 128 lines of data.

It is worth noting that the arctangent function takes almost as long to perform as the inverse column FFT stage. This arctangent uses a Look-up table to speed-up this function, if this was not the case the time would be at least doubled for this stage., see fig.7.11.

7.3 Pegasus Timings

In this section the timing results for the Pegasus environment are presented using the scheme 1 FFA paradigm and the modified "2+1" PSP technique. The timer function used in this section of the research varies somewhat from the timer functions used within the 3L environment and is slightly less accurate, but as has already been stated timings are only meant for relative comparisons.

Many of the low-level ideas used in the Pegasus environment have been outlined within the previous section and therefore are not duplicated within this section. The key results that have a direct impact on the results presented now are those for the internal ram blocks, the number of processors to use and the various stages of the FFA system. However, before moving on to the FFA system some of the features of non-consecutive strip partitioning must be discussed.

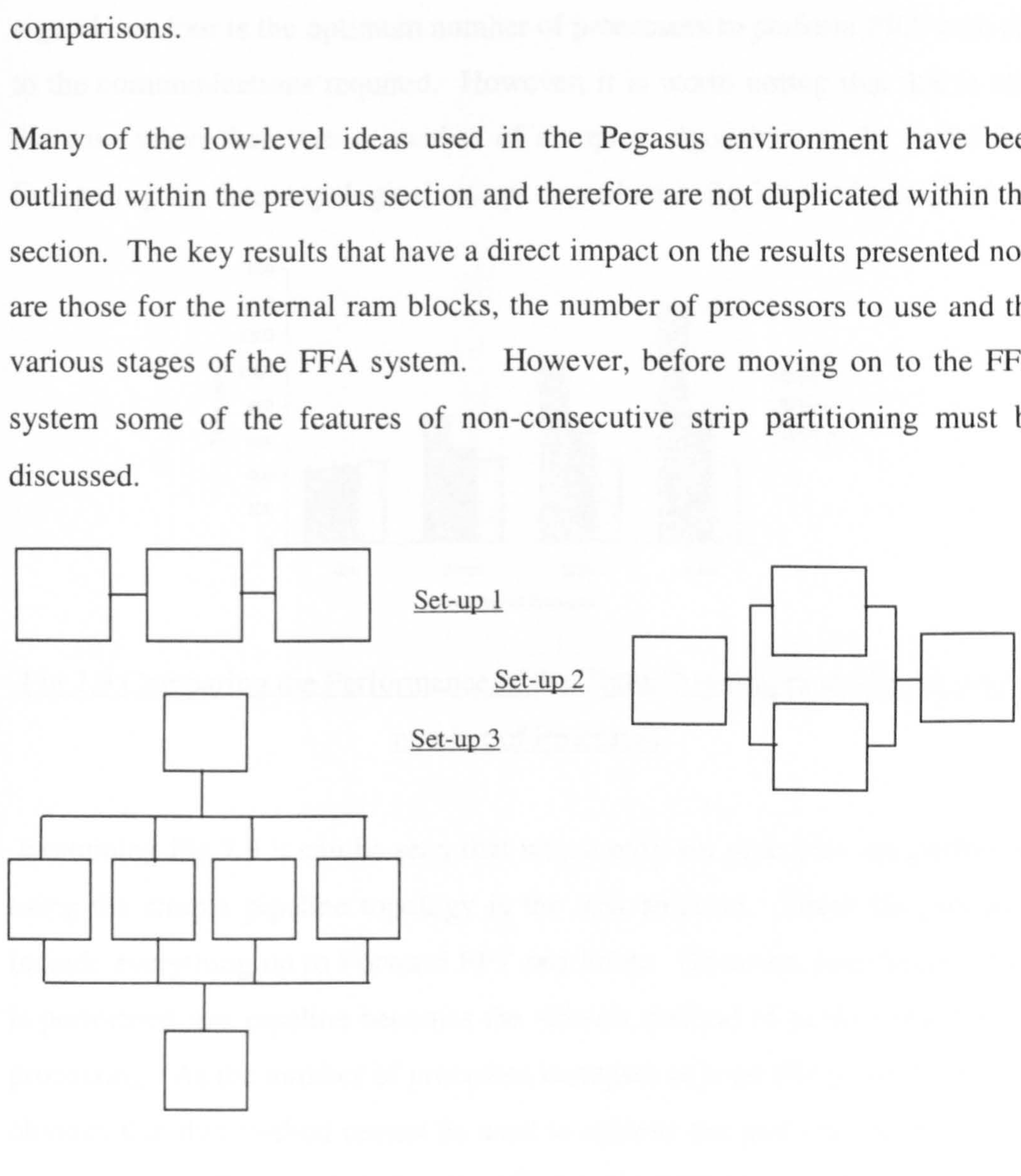


Fig.7.8 Different topologies for parallel processing

7.3.1 Arguments for Partitioning

Throughout this work it has been taken for granted that parallel processing techniques are required for the problems presented by the Fringe Analysis methods of the INFOCUS system. In the effective topology of Fig.5.8 four processors perform the stages of FFA. While in section 7.2.3 it has been argued that four is the optimum number of processors to perform FFA with due to the communications required. However, it is worth noting that this is only the case when there are a number of complex algorithms to be performed. Comparing the three topological set-ups described in fig.7.8 can highlight this.

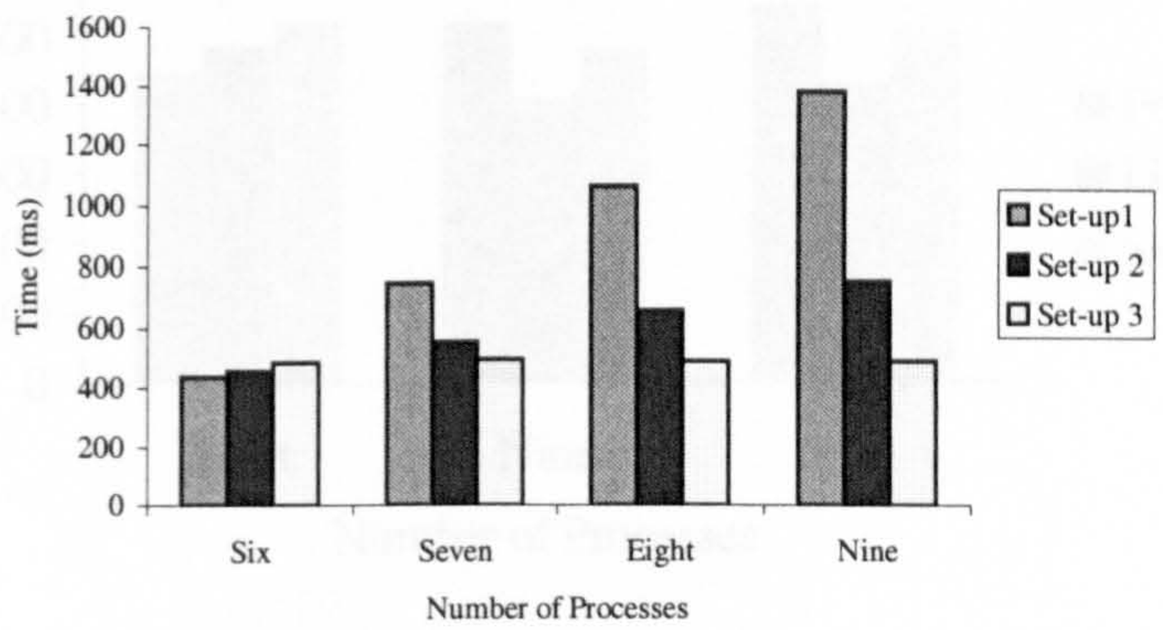


Fig.7.9 Comparing the Performance of the Three Topologies in Fig.7.8 over a number of Processes

Examining Fig.7.9 it can be seen that where only six processes are performed using the simple pipeline topology is the best solution. These six processes include everything up to Forward FFT and Filter. However, once inverse FFT is performed this pipeline becomes the slowest method of performing parallel processing. As the number of processes increases beyond this point it becomes obvious that this method cannot be used to achieve the goals of the INFOCUS system. The difference between the other topologies may not seem that great but by the time nine processes are being performed in parallel the second topology is some 250 ms slower that third topology.

7.3.2 Non-consecutive strip

Using the ideas from the previous section a FFA only system has been developed which is based upon the effective topology of Fig.7.9. In the non-consecutive strip method the number of lines which each processor receives and analyses at a time is crucial. In this research the number of lines used must be in the series 2,4,8 etc. Examining Fig.7.10 it can be seen that when the number of processes is low the number of lines analyzed should also be low. This is because the critical factor is the amount of data that can be sent to the Windows environment can be kept low.

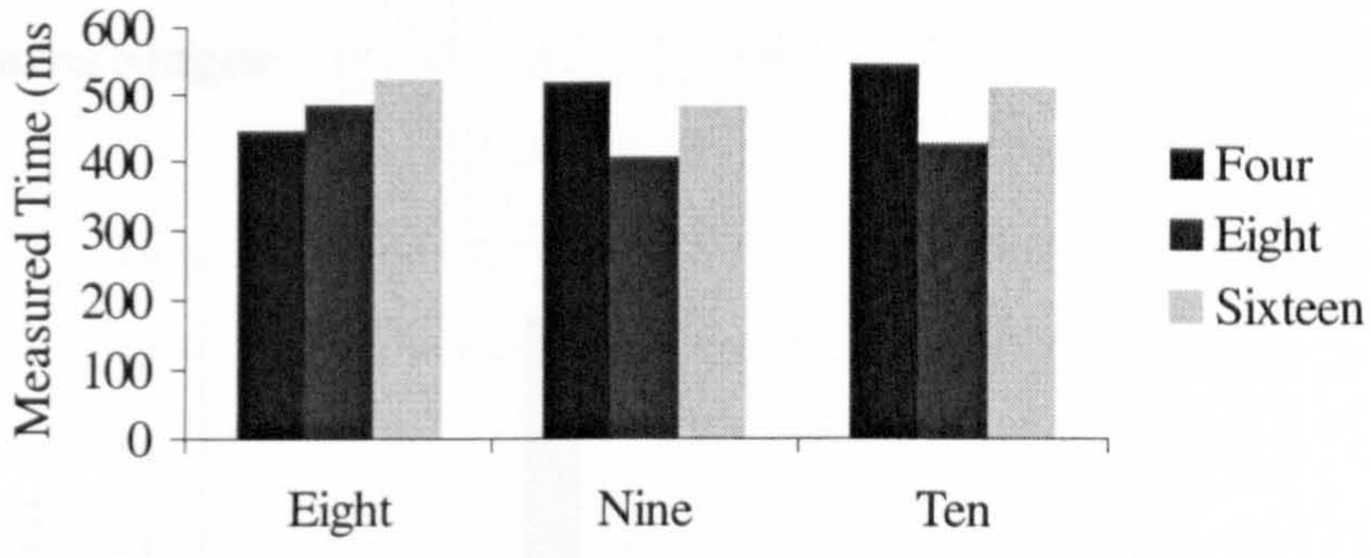


Fig.7.10 Comparing the Performance of Different Non-consecutive Strip Strategies where the Lines Per Chunk analysed are varied.

With Fig.7.10 the comparison between eight and nine processes is a little misleading in that there has been a change in the algorithms used to perform one of the processes. But comparing the three sizes of strip a rule of thumb emerges:

- Where the processes performed is less than 8 then the strip size should be kept small
- Where the amount of processes performed is very great, larger than 20 processes, using much larger strips is the preferred solution

- That between these two extremes lies the optimum solution for the INFOCUS system

Finding the optimum solution to the amount of data communicated is perhaps the most critical act of the programmer using the Pegasus system. This is a balancing act where the factors of Host interaction and DSP performance must be carefully weighed up. Successfully matching this solution to the problem is crucial in achieving the goals of the INFOCUS system. Throughout much of this work 8 lines at a time have been analyzed because this is the best match between the Host and C40 performances.

7.3.3 Pegasus Stages

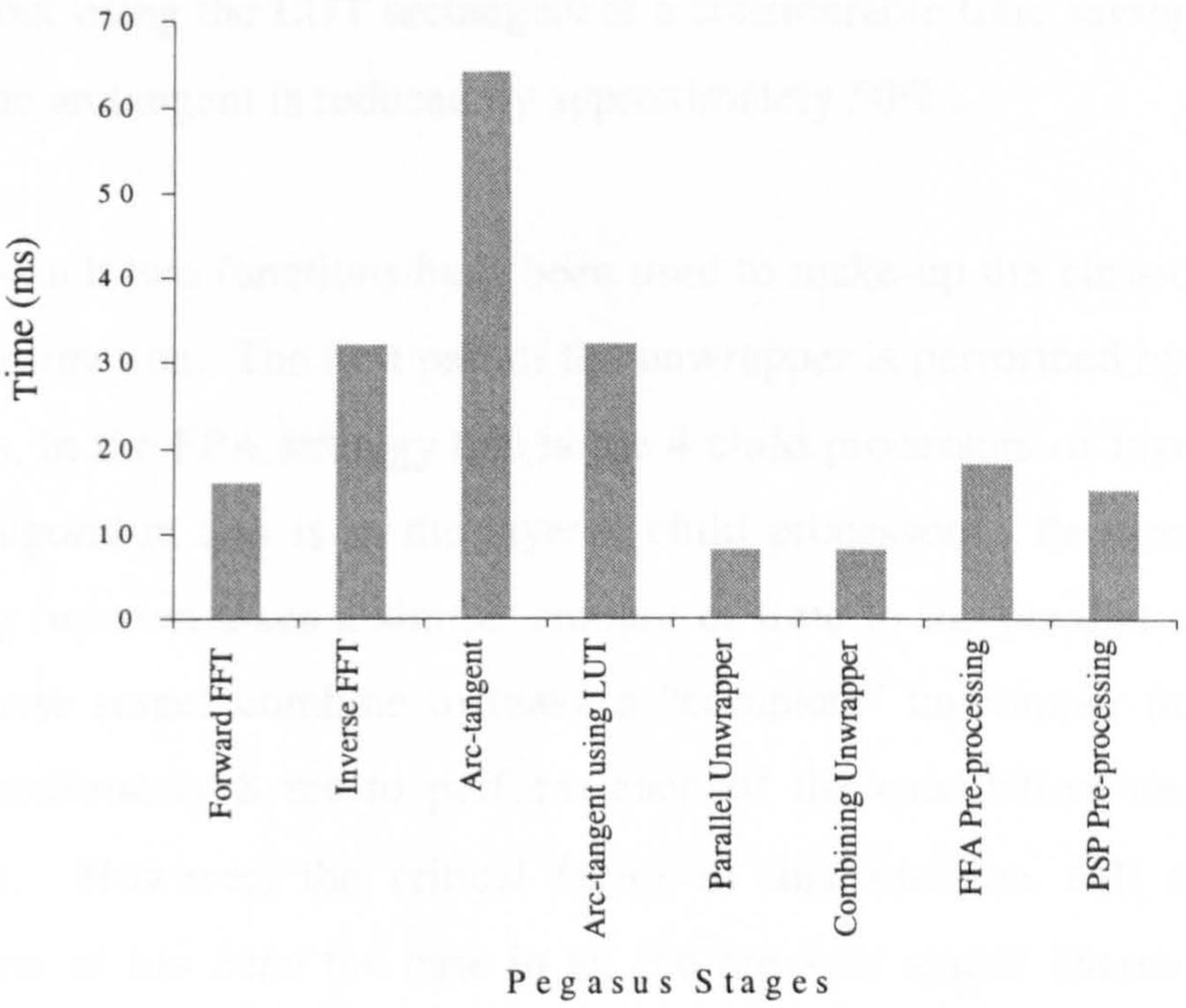


Fig.7.11 Comparing the timing results for the various stages of the Pegasus environment

The timing results quoted for each of the functions within this section, fig.7.11, are the times to perform the computation component of the algorithm but do not include the time to perform the necessary communications.

Comparing the real and complex FFTs used within Pegasus shows that there exists a similar advantage from using the real only FFT as was outlined within the 3L environment. The time to perform the real-only transform for 256 lines is just under 20 ms while the time to perform the complex FFT function for the filtered 256 lines of data is approximately 30 ms.

Examining the arc-tangent functions in more detail the benefits from using the look-up table (LUT) can easily be seen. This LUT halves the time to perform this function. Shaving milliseconds of a function is not the goal of this research, but using the LUT arctangent is a considerable time saving. Time to perform the arctangent is reduced by approximately 50% .

In this research two functions have been used to make-up the classically linear unwrapper function. The first part of the unwrapper is performed by four child processors, in the FFA strategy this is the 4 child processors of layer 1 and in the PSP algorithm this is in the layer 2 child processors. Performance of a combining function takes a similar amount of time to the parallel unwrapper, both of these stages combine to make a "complete" unwrapper function. It takes approximately 8 ms to perform each of the calculation stages of the unwrapper. However, the critical factor in this function will not be the computation as has been the case in all the previous stages examined in this section. Rather the time to store and combine the 4 unwrapped segments will be the critical factor.

Performance of both the pre-processing functions for FFA and PSP takes approximately the same time to perform as the real-only FFT. Actually

performing the FFA pre-processing stage cuts down the timings of the later filter stage and the unwrapping stage.

7.3.4 Comparing 3L and Pegasus

It should be remembered that the Pegasus software development environment is built upon the programming language and libraries of the 3L environment. This means that the environments use the same compilers and to a large extent the same programming language. This means that where Pegasus functions are identical to 3L functions in terms of the size of data analyzed and algorithm used the timing results will be more or less identical. This is proven by the timings for the real-only forward FFT block in both environments, fig.7.12.

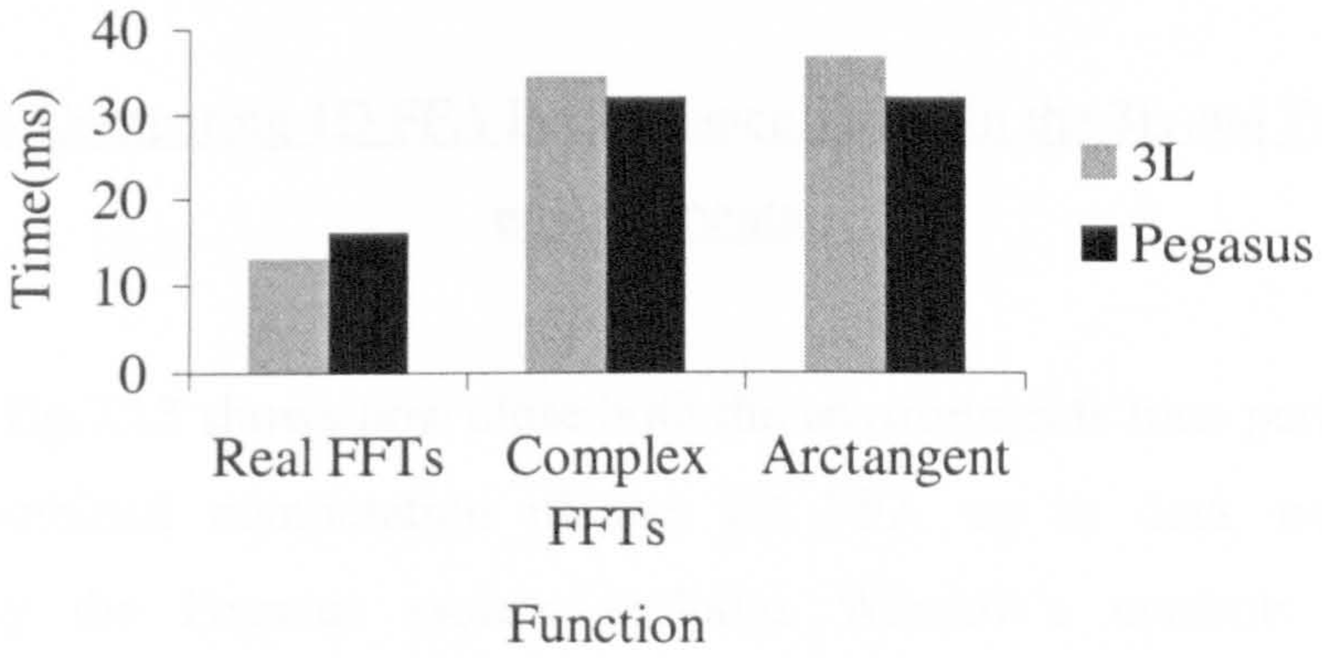


Fig.7.12 Comparison of 3L and Pegasus Functions

The real-only FFT in the 3L environment is approximately 3ms faster than the FFT performed with in the Pegasus environment. However, this difference arises for a number of reasons. Firstly, the Pegasus FFT also performs the filtering that is required, whereas in the 3L environment this is performed within another stage. Therefore this Pegasus stage does not simply consist of an FFT and DMA transfer as the 3L stage does. Secondly, the nature of Pegasus version of the FFT stage is such that there is some interaction with the

Host Windows environment required and this adversely effects the time to perform this function.

The differences between the other two stages also arise from a number of small differences between the two environments. Firstly the two environments use different timing functions. Secondly, for the arctangent function the 3L version performs a DMA transfer as well as a calculation stage.

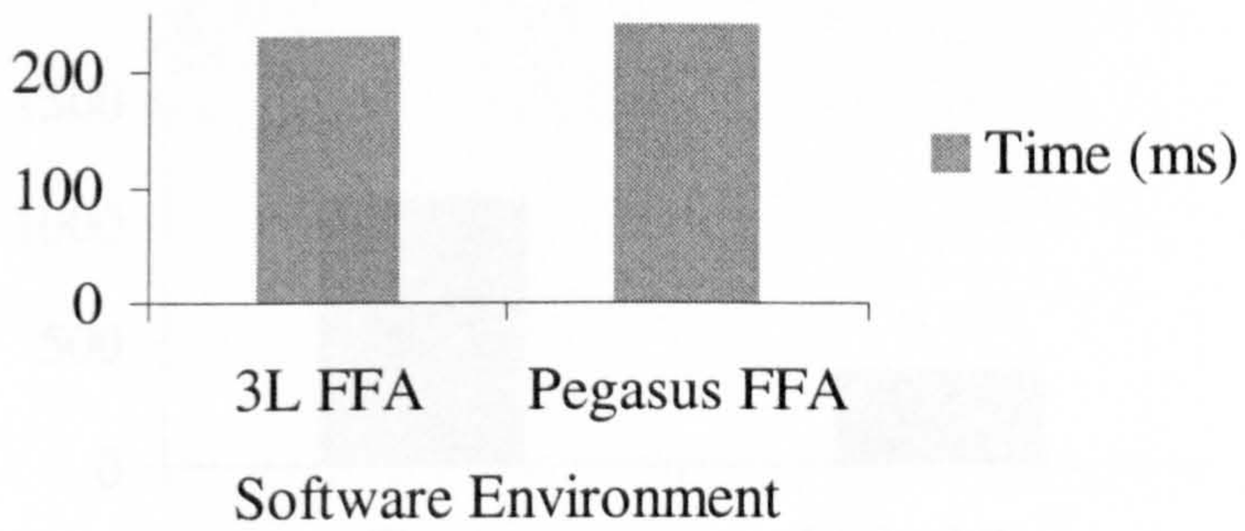


Fig.7.13 Comparing 1D FFA Performance Times in the 3L and Pegasus environments

Looking at fig.7.13 shows how close both the environments time performances for the combined computation of the 1D FFA up to data storage are. Interestingly the Pegasus system includes Window's controls and pre-processing stages while the 3L system does not. This should of course mean that the Pegasus time is much worse than the 3L performance however there are a number of major factors why this is not the case:

- Improved Compiler and Language Tools between environments
- Increased programmer experience and knowledge - the Pegasus system was written later in the research.

It can be seen from these timing results that the computational timings for 3L and Pegasus are almost identical. Additionally it can be seen that C40 timing results can **only** be used as relative comparisons.

7.3.5 Effects of the Host PC

The aim of this section is to explore some of the effects of a Windows based system upon the parallel processing solutions developed in this research. Perhaps the two main areas of this are in image display and data storage.

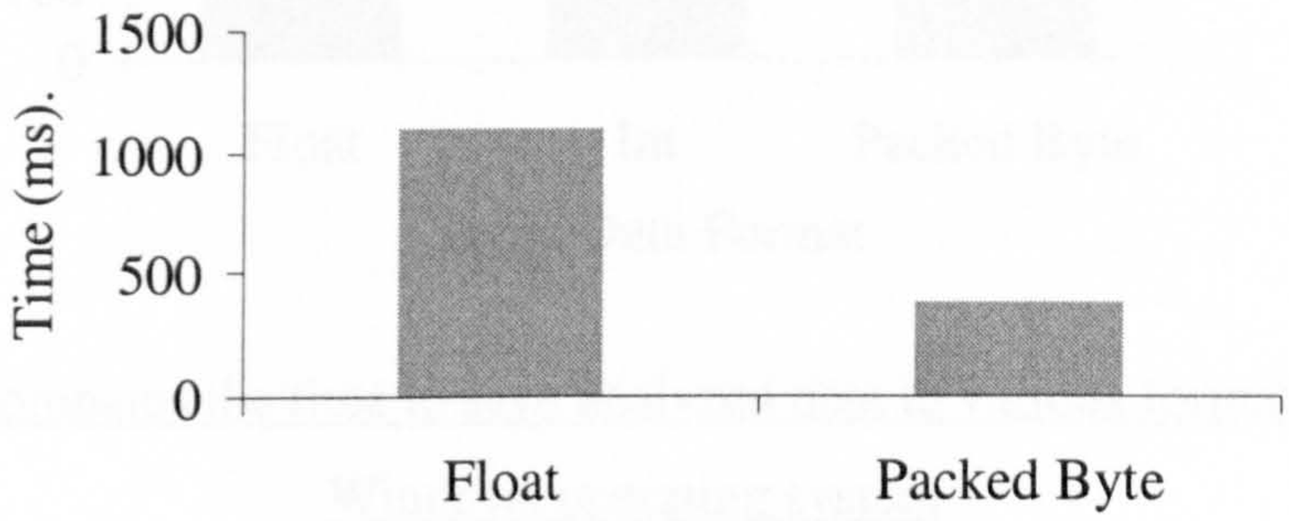


Fig.7.14. Compares the time to Perform FFA as well as the display of data in various data formats

Once data has been successfully analyzed in the C40 system it must effectively be written to the memory space of the host processor. The format that this data is displayed in by the Host PC will have a great deal of effect on the time to perform the entire FFA. This is because the FFA system will be held up by the interaction between the Host PC and the DSP root processor. As can be seen in fig.7.14 the time to perform FFA and display data in the floating-point format that has been used within the DSP system is much slower than a system using a packed format of data for the purpose of image display.

In fact comparing these times with those from fig.7.13 it can be seen that simply displaying an image in the Windows environment greatly increases the

time to perform FFA. Therefore much work has been done on using different methods of displaying data. Using 8 bit data means that some information must be lost but that the time to perform FFA although still degraded when compared to fig.7.13 is not as greatly effected.

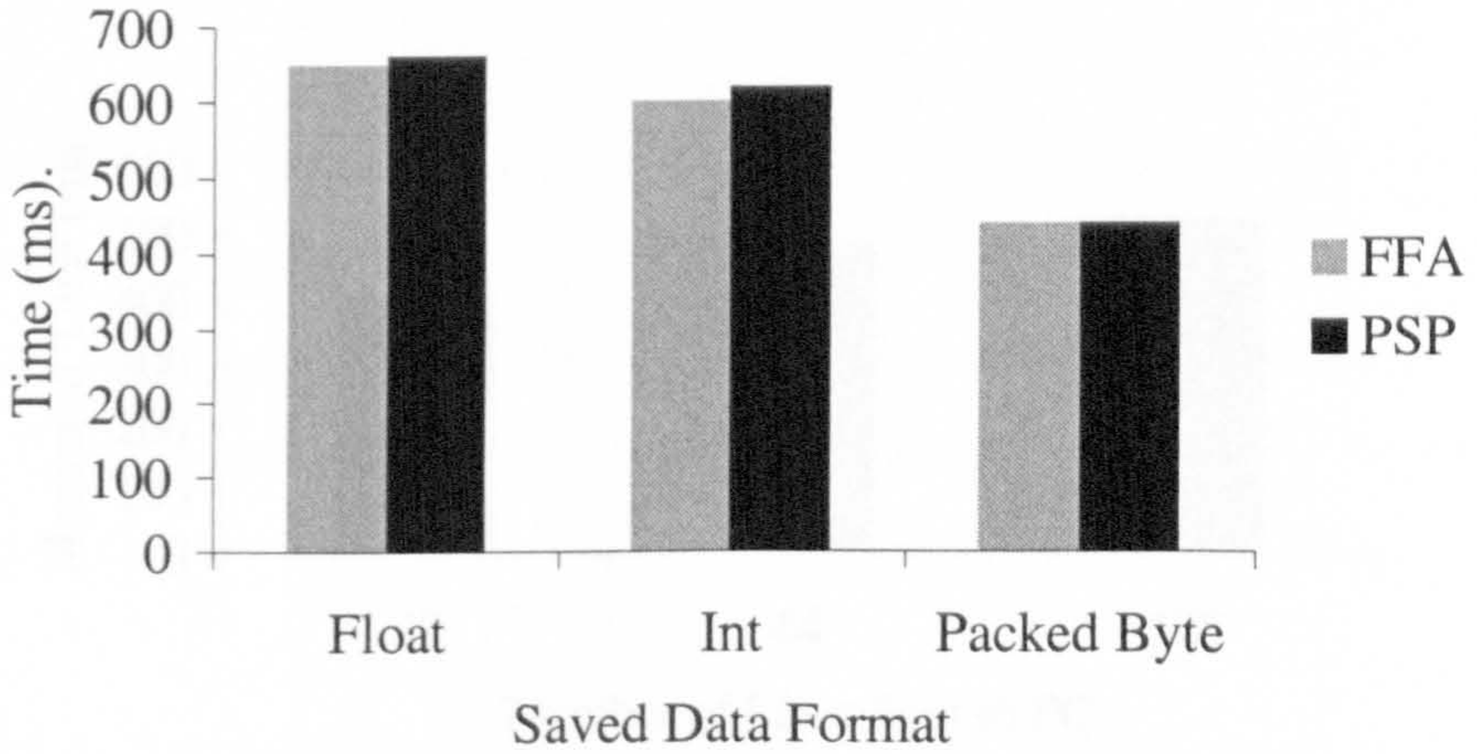


Fig.7.15 Compares the time to save analyzed data in various formats with the Windows operating system

In fig.7.15 the idea of using byte format for image display has been incorporated into both the FFA and PSP systems. Comparing the timing results it can be seen that saving both floating-point and integer format data effectively trebles the time to perform FFA when compared to fig.7.13. As well as this saving the data effectively doubles the time when compared to fig.7.14.

Storing the byte format data that has been displayed saves much of this additional time. It is worth noting that using byte format within the Windows based Pegasus system will greatly increase the importance of the user interaction because of the method of conversion that is used in this system.

Interestingly not only is the format is which data is sent to the Host PC critical but also the number of lines is also of importance, this is a development of the ideas presented in section 7.3.2. Examining 7.16 it can be seen that the effect

of increasing the numbers of lines sent is to increase the measured time of the system. It is worth noting that the actual performance for both 32 and 64 lines sent may well be comparatively much better than the 128 lines. This is because in measuring both these systems the Host PC was struggling to cope with storing data, writing messages to a window and displaying image data.

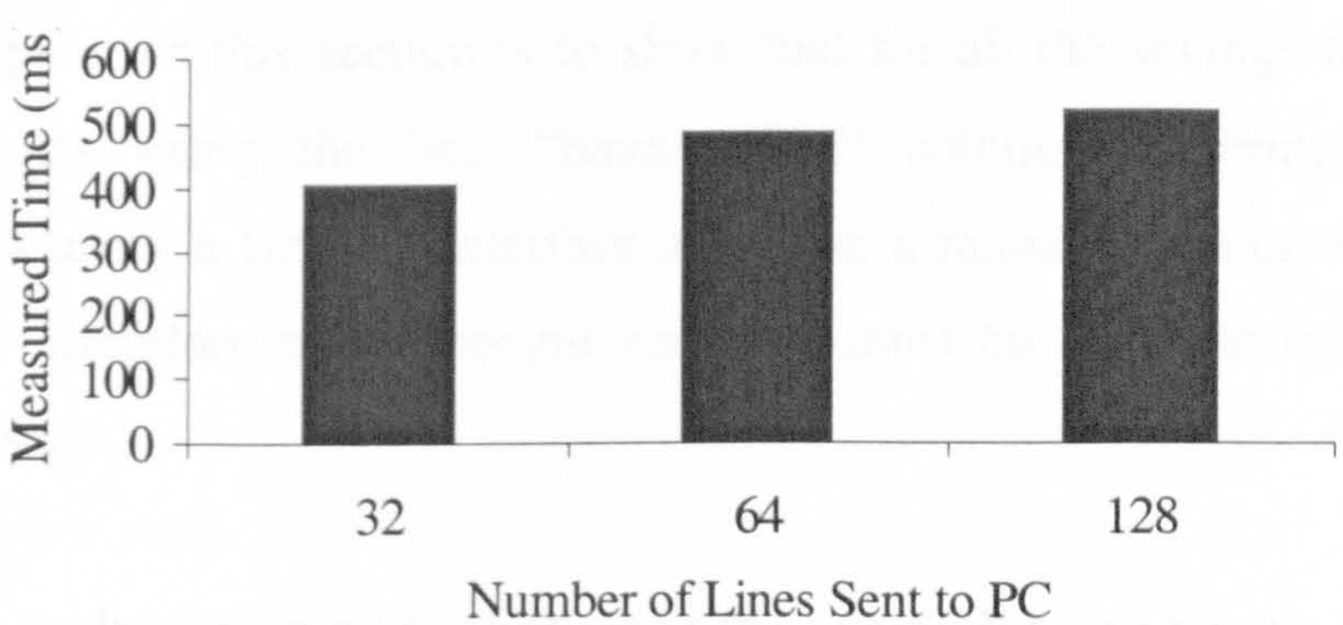


Fig.7.16 Shows that increasing the lines sent to the Host PC adversely effects the speed performance of the system

It is worth mentioning at this stage the timing results from a system using 512 lines of data. The system was designed to simply capture, display and save packed data for a 512 by 512 image. Using the data from a 256 square image as a guide a time roughly 4 times the measured performance speed would be expected, that is to say a speed of approximately 1200 ms or 1.2 seconds. This predicted time is above the 1second mark that this system must perform within. This time would be sufficient to allow all the required stages of FFA to be performed with no additional overhead. Additionally with a predicted 10 % saving for not performing the measurement the prediction for the actually speed would be very close to the 1 second mark.

However, the measured performance of this simple system was far higher than the predicted speed. The actual time performance for the system was 1600 ms, this was using the optimum arrangement of data and tasks. Even with a 10 % saving this time is well over the required 1second mark. This shows that the

performance of the Windows system is absolutely critical and extremely restrictive. The performance of the computation and DSP communication for FFA is within the 1second barrier, but the timing for storing and displaying the data is so high that 512 images cannot be used even with the simplest possible FFA system outlined in fig.6.1.

The key point of this section is to show that for all the savings that can be produced by using the best "hand-crafted" solution to Fringe analysis techniques using a Graphic Interface will have a massive impact on the final solution. Therefore much thought and time must be spent on the graphical interface stage.

This idea is demonstrated further when the results for performing concurrent 1D FFA and Modified "2+1" PSP techniques are examined, fig.7.17.

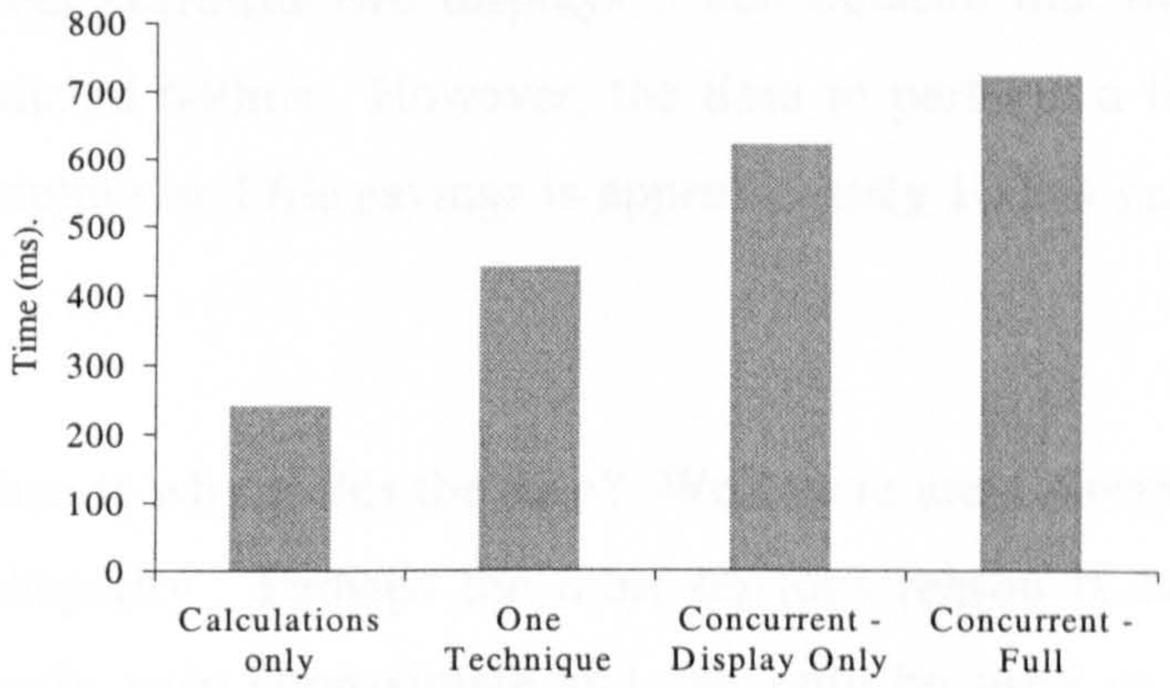


Fig.7.17 Compares the time to perform Concurrent PSP and FFA with the time to perform FFA only.

Examining fig.7.17 it can be seen that the time to perform concurrent techniques is significantly greater than the time to perform a single Fringe Analysis technique. However, the time to perform the required Fringe Analysis calculations will not be greatly altered. That is to say that it takes approximately 240 ms to perform the calculations for FFA in a single technique implementation or within a concurrent implementation. Therefore

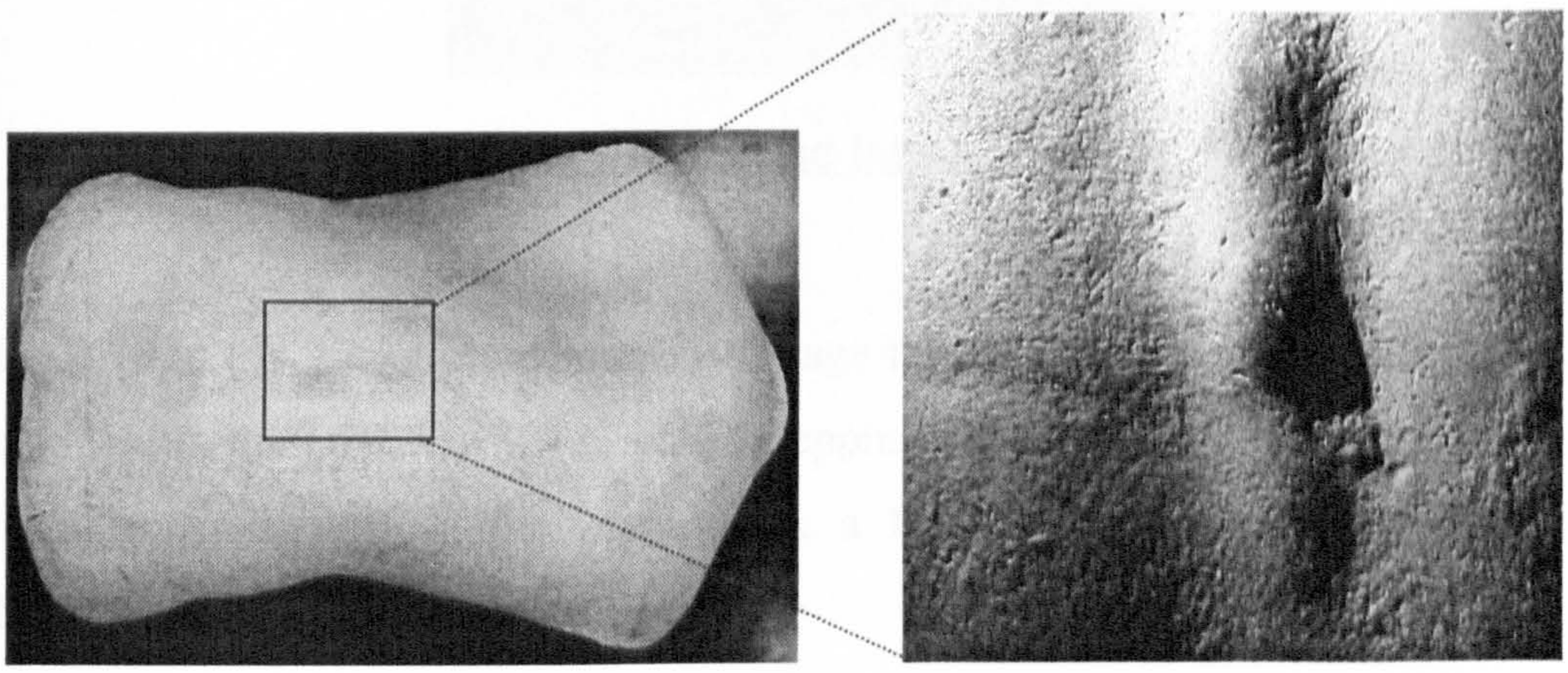
the increase in time must come from the increased amount of data that is sent to the Host processor.

From fig.7.14 the time to perform a single FFA and display the data takes approximately 380 ms, this is approximately 140 ms more than to perform only the calculations. Decreasing the speed of the system further the time to perform file saving as well as image display is approximately 60 ms more. This means that the difference between the time to perform calculations and a full FFA system is approximately 200ms. Based on this it would be expected that the time to perform a full concurrent system with two displays and data savings would be approximately 200 ms greater than the 440ms for a single system. This is because of the amount of data that is required and the added complication of having a number of communications that must interact with the Host processor. Examining the result for a concurrent system that does not save the data but performs two displays it can be seen that the time is just within the predicted 640ms. However, the time to perform a full concurrent system with displays and file savings is approximately 100ms greater than this value.

The question then is why is this the case? Well there are a number reasons for this apparent disparity. Perhaps the most obvious reason is that the timing functions are only very approximate and can only be used as a very rough guide to performance. Secondly, the increase in the number of communication threads and possible data conflicts means that the expected increase in the time to perform a concurrent system when compared to a single technique is not simply a linear relationship. The third possible reason is that the Root Processor must now perform many more tasks than in the single technique system. Based upon practical experience the difference between the FFA only timing and the full concurrent timing is well within any results that would be expected using knowledge as a guide to performance.

7.4 Images

The aim of this section is to show how the analyzed data at a number of FFA stages varies between the parallel processing solution and the Windows PC based system described in section 7.1.1. This section also shows different filtering techniques that have been investigated within the parallel processing solution.



(a) (b)

Fig.7.18 (a) Plaster Cast of back used by parallel processing system (b) Close-up view of the area analysed by the parallel-processing system.

The surface to be analyzed within the parallel processing system is shown in Fig.7.18(a), this surface is a plaster cast of a human back. Within this large area a smaller area is measured by the parallel-processing system, this area is roughly 10cm by 10cm in size. Fig.7.18(b) shows a close-up of the specific area measured by the parallel processing system in this experiment. Looking at this area closely it is possible to pick out three vertebrae located just off centre of the image. Either side of these peaks there is a hollow, within the hollow there are a number of troughs. Rising up from the hollows towards the outside of the images are muscles.

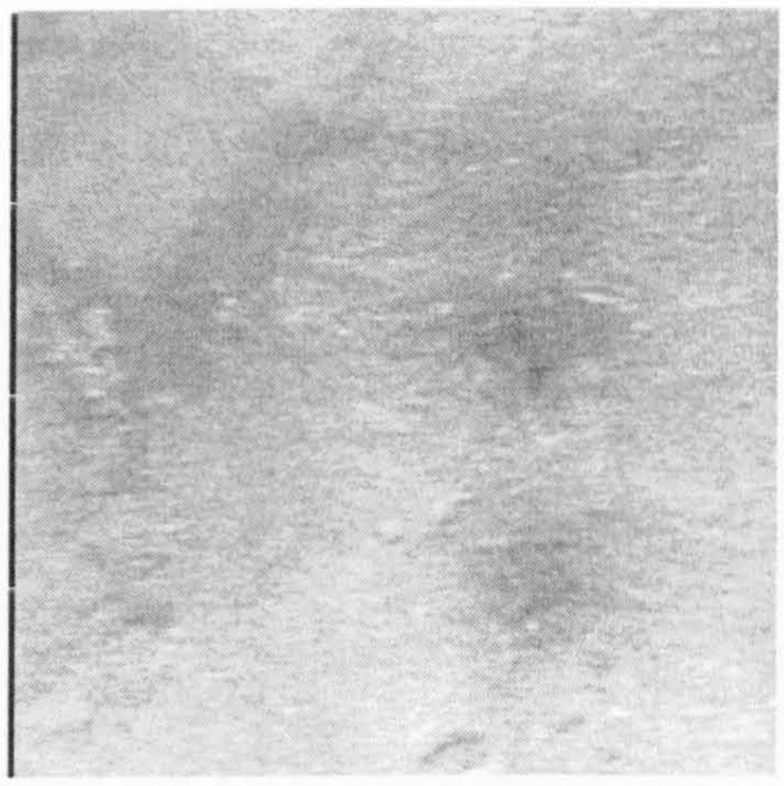


Fig.7.19 Background Intensity Image

Fig.7.19 is the background intensity image that is used in pre-processing and also within the modified "2+1" phase stepping algorithm. Once this image has been transmitted from the Framestore, a fringe pattern image is captured fig.7.20.

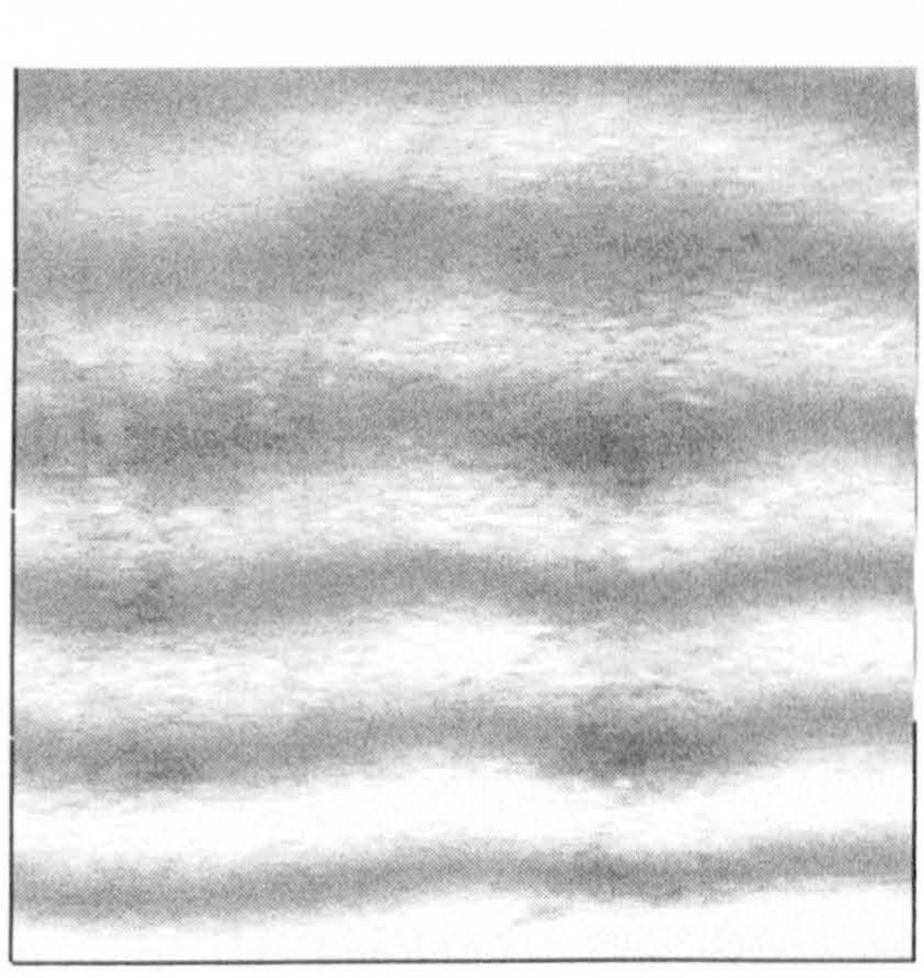


Fig.7.20 Fringe Pattern projected onto surface.

Looking a fig.7.20 the shape of the fringes shows the surface shape which can be expect to appear in the analysed data. At this stage processing can begin. In fig.7.21 the pre-processing has been performed by one of the child

processors of layer 2 in the topology defined in figure 5.9. It is worth noting at this point that the Hamming Window has not yet been performed.

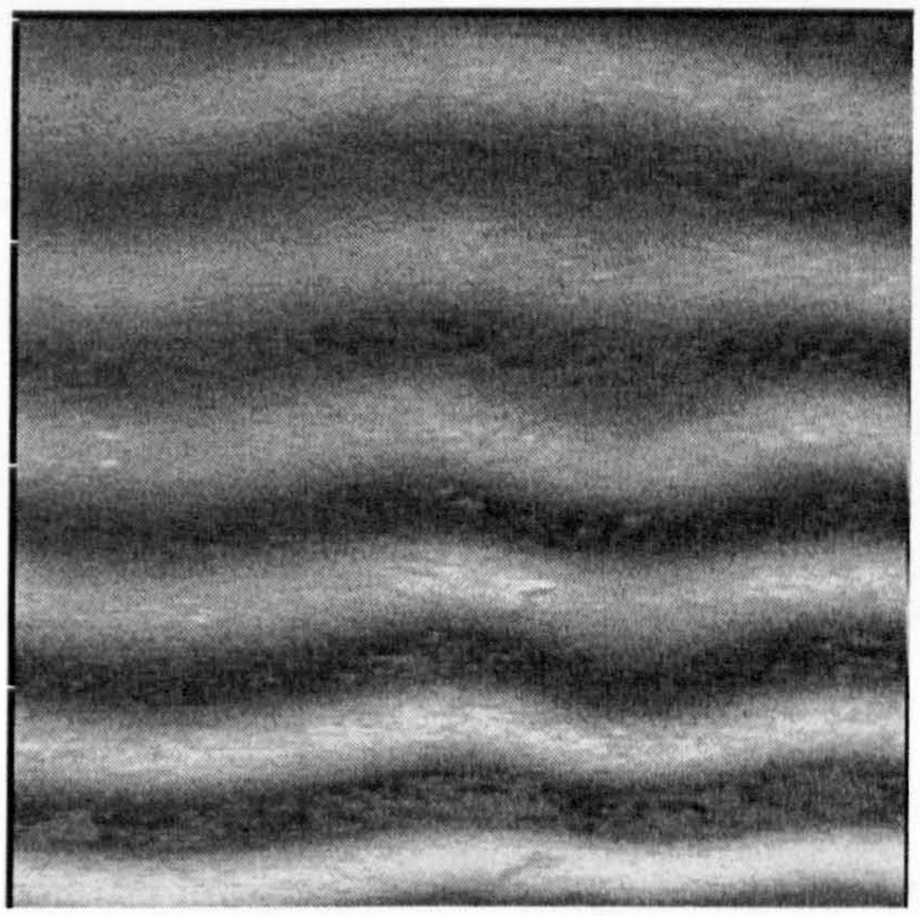
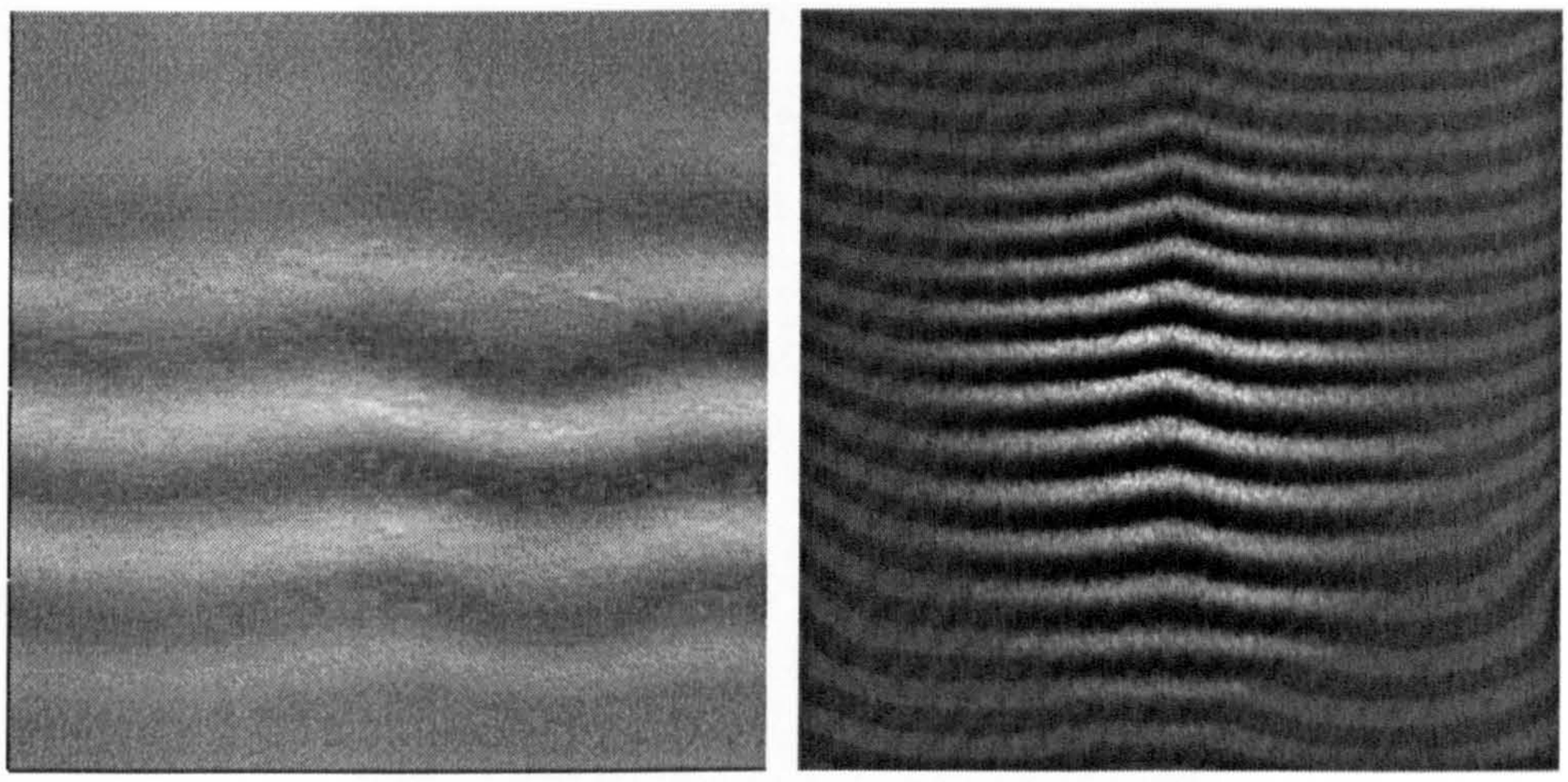


Fig.7.21 Pre-processed data where Hamming Window has not yet been performed.

Whereas the previous images have been scaled between 0 and 255 the pre-processed image is scaled between -125 and +125. Comparing this image with fig.7.20 it can be seen that the major effect of the pre-processing performed upto this point is to remove the uneven spread of illumination produced by the optical system. The fringes in Fig.7.21 are also smoother than the unprocessed fringes in Fig.7.20.

The next image Fig.7.22(a) shows the data once the Hamming window algorithm has been performed. This function is undertaken by the child processors on Layer 2 of the topology. The effect of this stage of pre-processing is to make the central data of the image have more intensity than the edges of the data.



(a) (b)

Fig.7.22(a) Fully Preprocessed Image from the parallel-processing environment (b) Pre-processed image from the IDL system

Comparing Fig7.22(a) with Fig.7.22(b) the differences between the IDL PC system and the parallel-processing version begin to emerge. Within the IDL version of the pre-processed image the effect of the Hamming Window has a more circular appearance than the parallel-processing system, this is due to the difference in one dimensional and two dimensional Hamming Windows. While IDL uses a 2D system the parallel-processing system uses a simpler 1D system. The difference is due to the way the analysis in the Fourier domain is to be performed. It is worth noting that the IDL system and the parallel-processing system are analysing similar but not identical surfaces.

Fig.7.23 shows the magnitude of the FFT once a simple rectangular filter has been applied in the parallel-processing system. This filter isolates the first order peak and sets all other data to zero in value. It can be seen each line of the image contains a first order peak which is centred around the point equal to the number of fringes within the image.

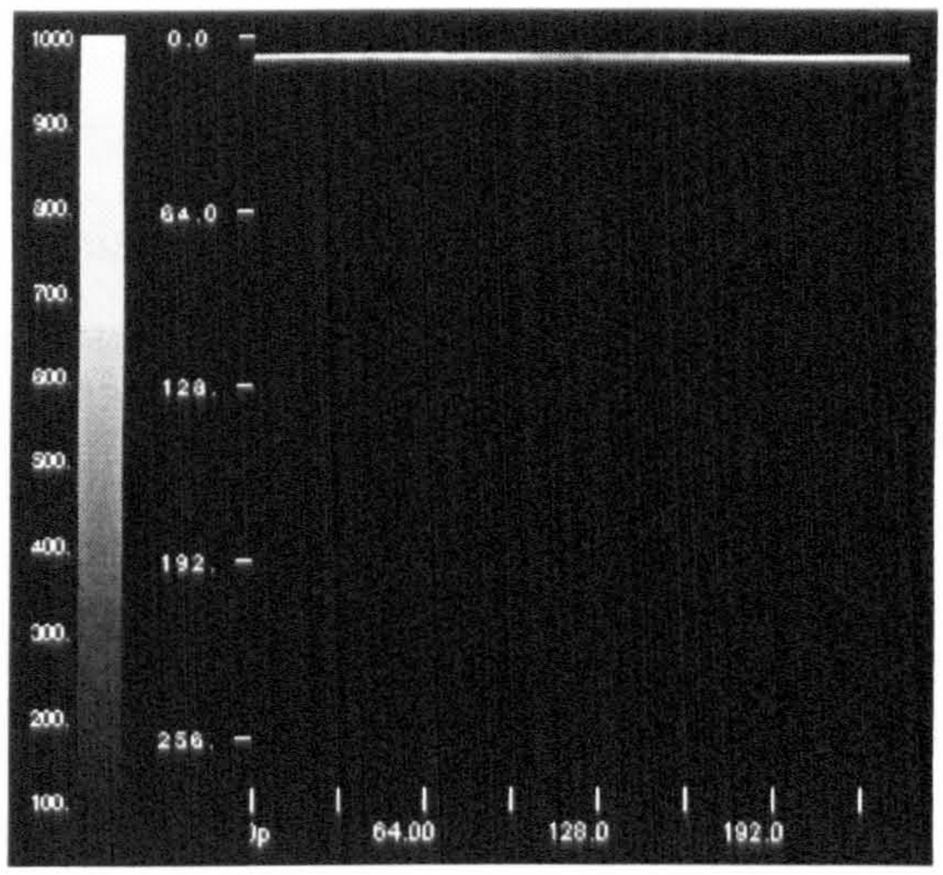
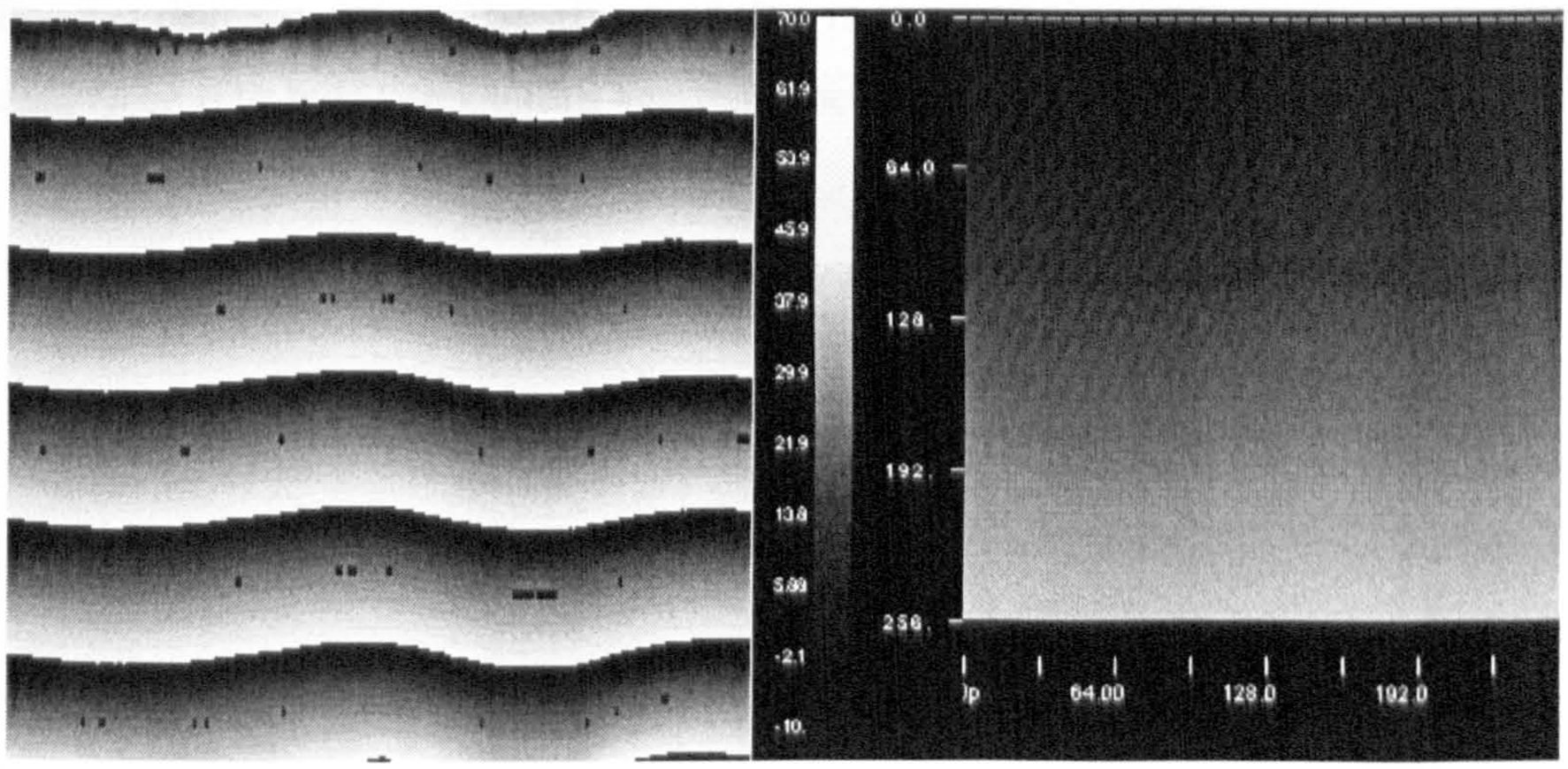


Fig.7.23 Magnitude of the application of an "Ideal rectangle" Filter in the Fourier Space.

However, using such a filter results in many difficulties in the unwrapping stage of the analysis. Examining fig.7.24(a) it can be seen that the main reason for this is the number of phase wraps that exist in the data once the inverse FFT and Arctangent have been applied. This wrapped data is scaled between $+\pi$ and $-\pi$. It is worth noting that the speckle effect present in this image is caused by an error in the scaling factors used to display the packed data.

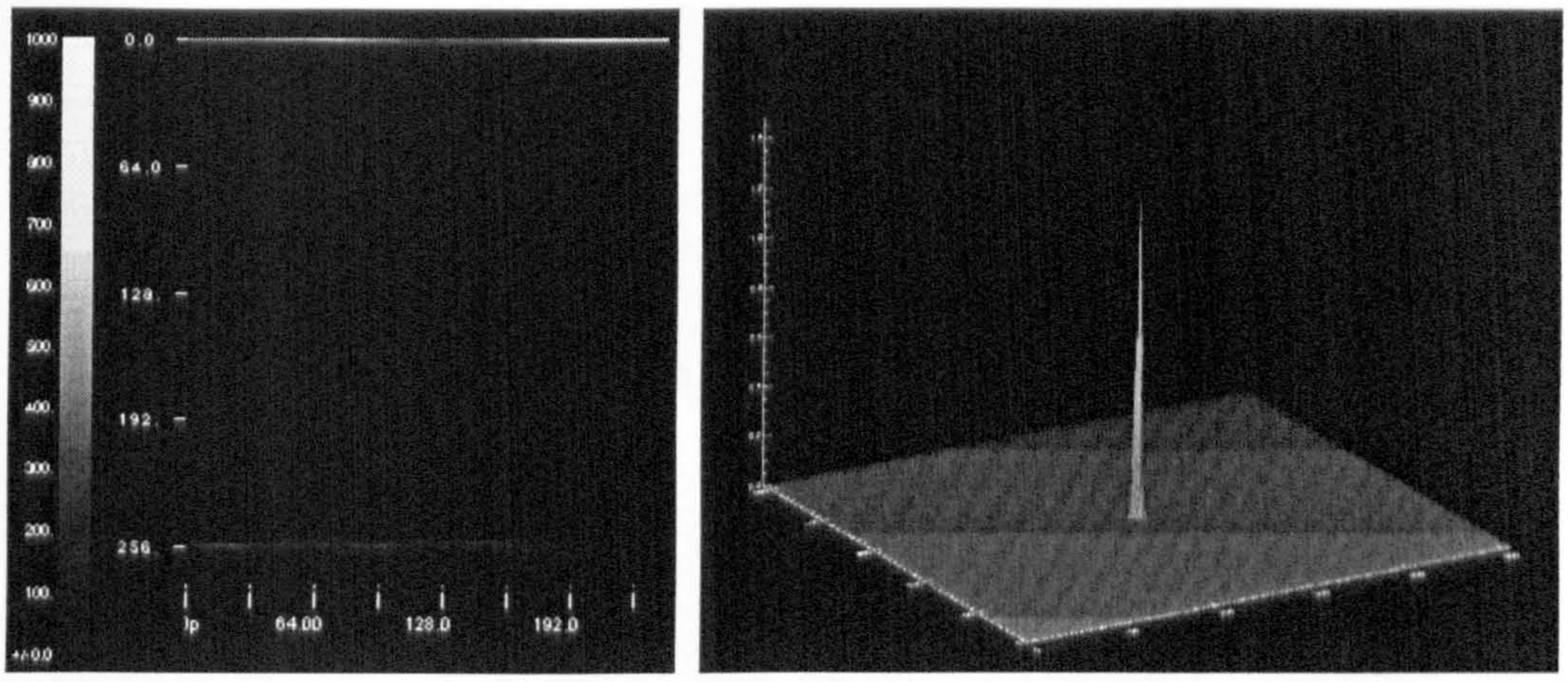


(a) (b)

Fig.7.24(a) Wrapped phase after the application of the ideal filter (b) Unwrapped Phase without Tilt removal or height multiplication.

In fig.7.24(b) the top line of the image seems out of synchronization with the rest of the image. This is not a wrap error but rather arises from the use of parallel unwrapping stages. The top line is in fact the accumulator values which will be used in the combining stage of the unwrapper. The unwrapped data in Fig.7.24(b) actually contains the unwrapped surface which has not been combined or had the tilt in the data removed or been multiplied by a height scaling factor. It is worth noting that this image is scaled between -30 and +70, which means the surface has a large amount of tilt which is produced by the analysis method, rather than from actual angular tilt in the surface. For these reasons this surface does not resemble the initial image of fig.7.19.

At this point in the research the results from another method of filtering in the IDL environment meant that a change of algorithm in the parallel-processing solution was undertaken. This changed filtering algorithm is based upon a frequency shifting method of isolating the first order peak within the Fourier domain. Looking at fig.7.25(a) it can be seen that the data is no longer located around the point equal to the number of fringes within the image. In fact the data from the origin of the first order peak has been shifted to the origin of the Fourier domain data. The effect of this is to wrap the data around origin i.e. data from 0 to half filter width and data located at 255-half filter width to 255.

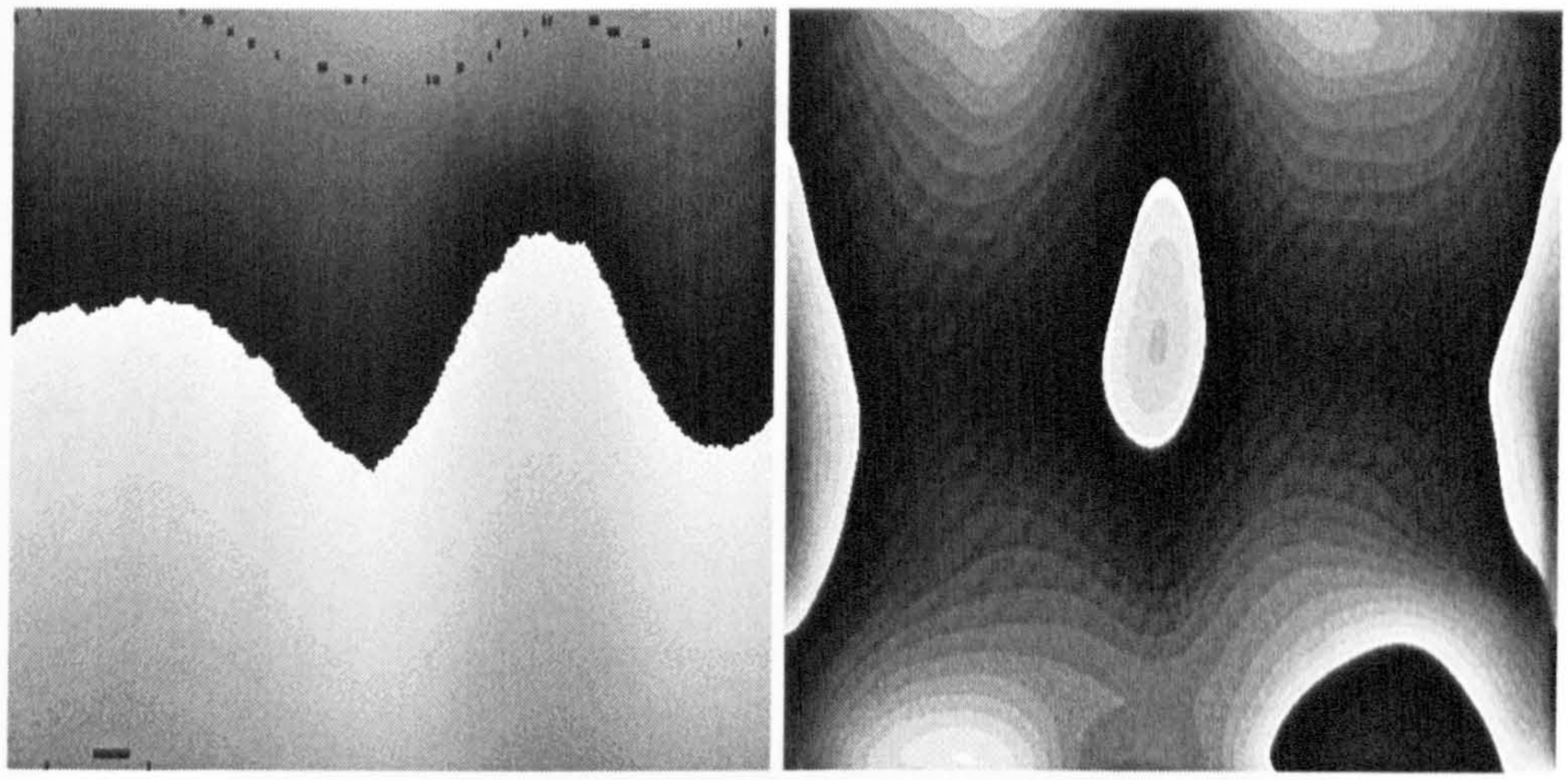


(a) (b)

Fig.7.25(a) Parallel Processing version of frequency shifted Fourier domain data (b) IDL version of Frequency shifted Fouerier domain data

Examining the two images presented in fig.7.25 shows the key difference between the parallel-processing solution and the IDL system. The IDL system uses 2D FFA analysis this means that the data in the Fourier domain is represented by a single peak once filtering been performed. Further this data exists in the non-quadrant swapped Fourier space described by fig.6.6(a). Whereas in the parallel-processing system the isolated first order peak exists for **every** line of the image. These differences exist because that the IDL solution is a matrix based solution while the parallel-processing solution is a signal processing solution.

Comparing the wrapped phase in fig.7.26(a) with that of fig.7.24(a) shows how frequency shifting simplifies the problem of unwrapping the surface data. In fig.7.24(a) 6 phase wraps are present while in fig.7.26(a) only one phase wrap is present. Additionally the surface shape visible in the original image is detectable in the wrapped phase data within the frequency shifting solution. This shows why frequency shifting filters are preferred in this research to non-frequency shifting filters.



(a) (b)

Fig.7.26(a) Wrapped phase from the Frequency shifted parallel processing solution (b) Wrapped phase from the Frequency shifted IDL solution.

It is true that the images within the IDL version of the FFA system have none of the the problems of scaling images for display and storage that the parallel system has. This explains the clarity of Fig.7.26.(b) when compared to the parallel solution. However, as both systems use the same type of filter the similarities at this stage of analysis are more obvious than the differences in the methods of analysis.

The features of the original surface become more apparent in the parallel unwrapped data from parallel-processing system, fig.7.27. Similarly to fig.7.24(b) the accumulator values used to recombine the data are contained in the top line of the image, although the frequency shifted solution used means that these accumulator values are not as apparent i.e. they are far smaller values than in the fig.7.24(b). This is to be expected from the decreased number of phase wraps present in the frequency shifted data. Utilising these accumulator values is the final stage of the unwrapping process performed on the frequency shifted filter data. This is because at this stage of the development a refinement of the filter meant that tilt removal was performed as part of the filtering process.

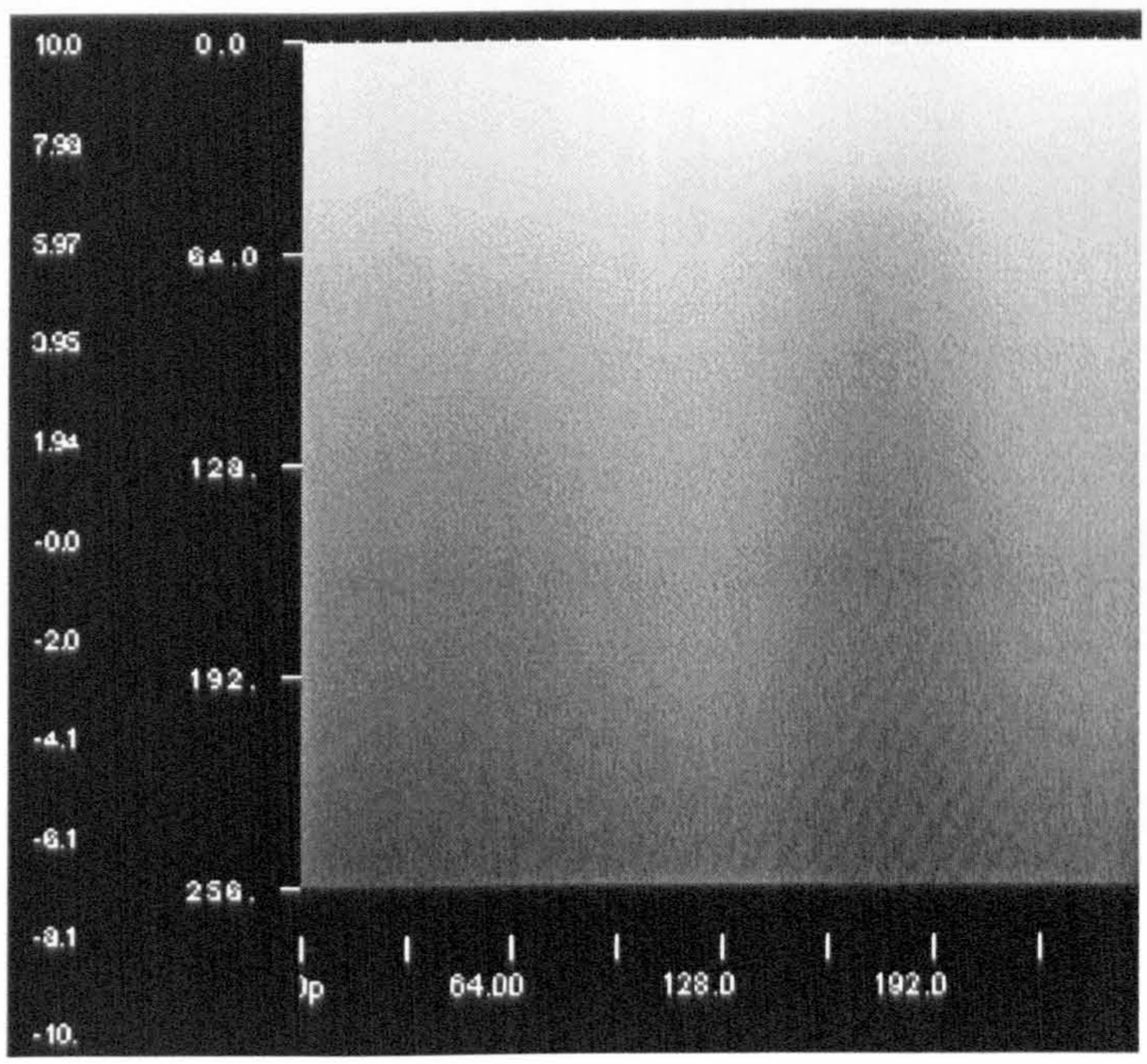
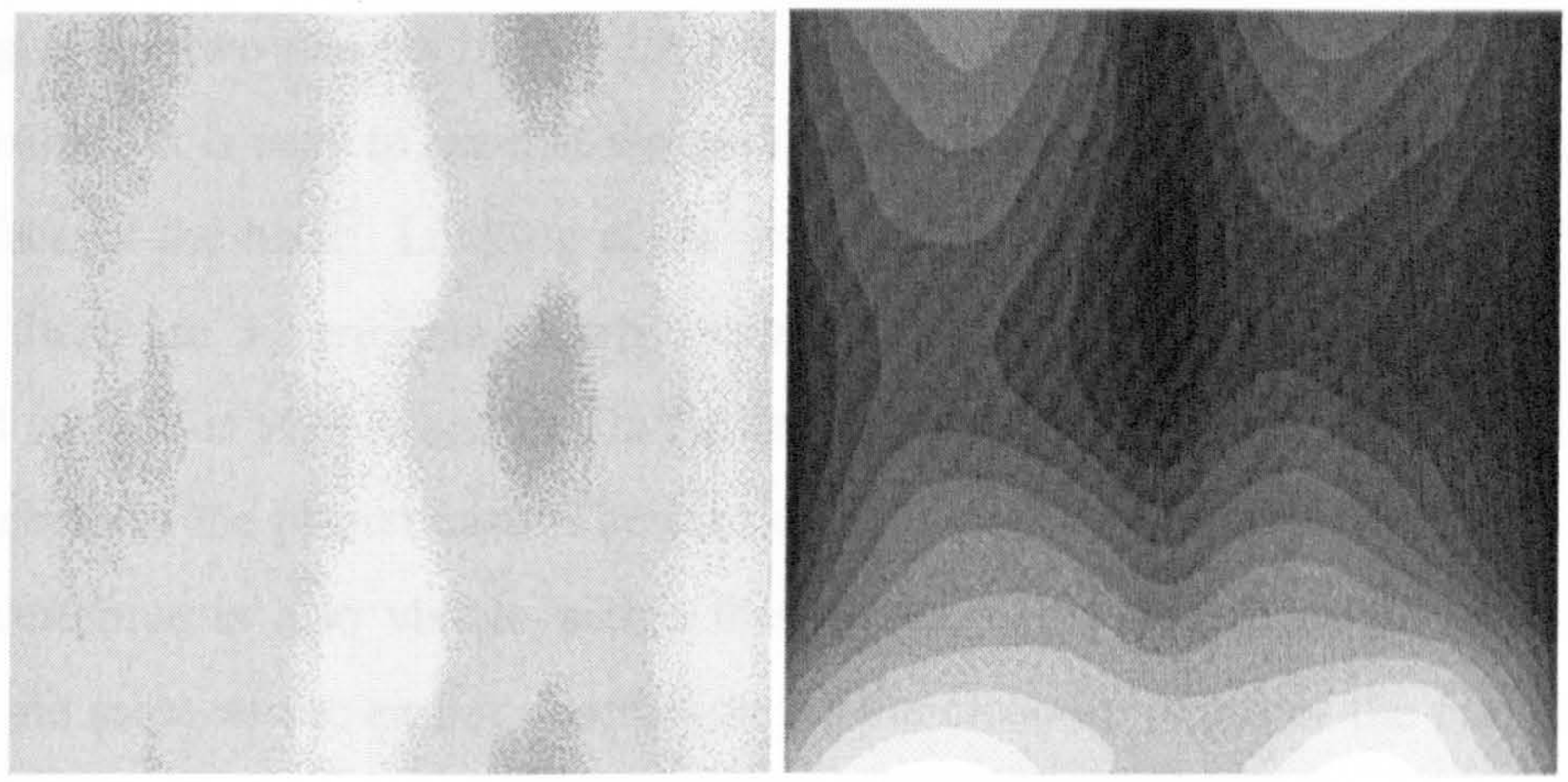


Fig.7.27 Parallel Unwrapped data

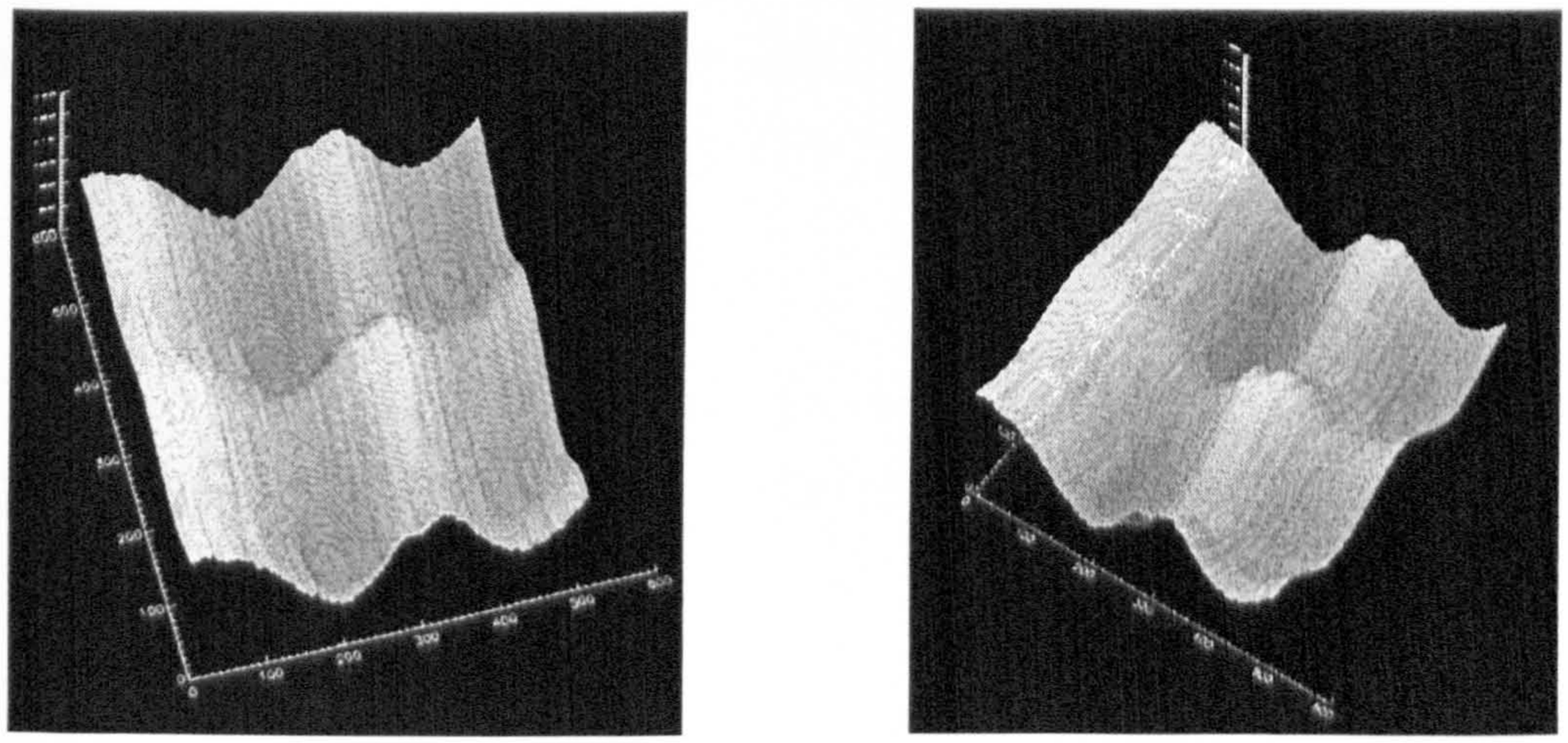
At this point in the FFA system the difference between the IDL solution and the parallel-processing system is the method of performing unwrapping. The parallel processing system uses a solution with two unwrapping stages while the IDL solution uses the commonplace version of unwrapper with only one stage. Both systems use the same algorithm to perform unwrapping. Therefore any differences in the quality of the unwrapped images is due to the parallel processing implementation of a 1D FFA system rather than the standard 2D version which the IDL system uses.



(a) (b)

Fig.7.28(a)Unwrapped surface data from the parallel processing solution
(b) Unwrapped surface data from the IDL system

In the IDL image, fig.7.28(b) the final surface is very easy to see. This surface is actually the back of phatom model used in radiotherapy. It is fair to say that this analysed surface is a very high quality result. Viewing the surface from the parallel processing system it is much harder to make out the features of the analysed surface. However, this result does appear to show the profile of the back outlined previously.



(a) (b)

Fig.7.29 Surface data from fig.7.28(a) displayed within the IDL environment.

Looking at the two images in fig7.29 this outline of the back surface is much more visible. It is easy to see that the parallel processing system has measured the surface of the back. Looking at the peaks offset to right of center in both images there are 3 vertebrae clearly visible. On the left of both images an inclined surface is visible and this is the muscle which is visible either side of the vertebrae in the plaster cast. The start of this muscle on the right-hand side of the vertebrae is also visible within this image. This desciption is almost exactly the same as the earlier description for the close-up photo of the plaster cast back in fig.7.18(b).

In comparing the final surfaces delivered by both systems it is fair to say that the IDL version demonstrates a higher quality result than the parallel-processing solution. However, the quality of the parallel processing result is a good enough result to prove that parallel processing of fringe analysis techniques is a viable solution. Where surface data is taken from the parallel processing solution and viewed within a highly quality image processing environment the validity of the results from the parallel solution is much easier to justify.

7.5 Summary

In this chapter a number of important results for the overall INFOCUS system have been presented. It has been shown that to achieve a successful Windows based Parallel-processing solution the key element is to find the balance between the C40's performance and the Host PC's performance. Achieving this means balancing the size of data sent to the Host with the analysis and communication performance of the C40s.

It is worth noting that the results from 7.3.1 effectively means that simple pipelining is not a good solution to the problems faced in this work. This is because of advantages that arise from partitioning data so that processors may concurrently perform analysis. In effect the conclusion from this section is that some form of **parallel** pipeline, as shown in figure 7.8 system 3, is the best solution. The topology of C40 network that best allows this within the specific hardware system used is based on a complex pyramid structure, refer back to section 5.3.4.

Further this chapter has shown that all the improvements that a specialist DSP programmer can bring to a system **must** be matched by the improvements made to the Host PC interaction, if a Windows based system is to be used in a real world sceneario.

Timing results are very approximate and therefore only three key results are shown here. These prove that parallel-processing can achieve the speed component of the requirements of the INFOCUS system.

Fringe Analysis System	*Measured Time (ms)*
Fourier Fringe Analysis	430
Phase Stepping Profilometry	440
Concurrent FFA and PSP	740

Table 7.1 Crucial Timing Results

Performing concurrent analysis allows the flexibilty component of the requirements to be met. The question then is how reliable is the system? The images presented in this chapter show that the system can be used for real measurements. But the system **cannot** perform analysis qualiatively as well as a Windows based matrix procesing solution.

Chapter 8

Discussion and Conclusions

8.1Introduction

The chief goal of this chapter is to analyse the work that has been carried out within this research with respect to the aims identified in Chapter 1. Firstly the specific practical issues raised by this research are discussed. These are questions of the practical restrictions that a parallel processing implementation places upon linear metrology algorithms. This highlights the impact of the hardware and software environments upon the functions of these algorithms as well as the data problems inherent in parallel versions of such techniques.

The applicability of parallel processing CASE tools to non-contact measurement problems is the next stage of discussion. The 3L and Pegasus environments are used as examples to highlight issues in this area. This discussion aims to give guidance into the future directions that parallel software development environments must follow for parallel image processing solutions to be realistic and practical in nature.

The next issue is less tangible in terms of timings and physical results, being based more upon experience. Here the true benefits and costs of parallel processing are explored. This is undertaken so that the question of whether or not parallel processing solutions are viable for optical metrology can be answered. Further questions of possible alternative solutions and future developments are examined.

Finally, this work is discussed with respect to patient monitoring and positioning in a radiotherapy environment. In this section all the themes of the discussion are formed into one coherent final picture. Allowing the question of whether this research has been successful to be answered.

8.2 Effects Upon Linear Paradigms from a Parallel Processing Solution

There are a number of different consequences that arise from moving classically linear optical metrology algorithms into a parallel-processing environment, which uses specialised hardware and software. Additionally, a number of boundaries are placed upon algorithms by the real-time requirements, in this case arising from the INFOCUS optical sensor's objectives. This section aims to discuss these issues with reference to the hardware and software of the INFOCUS system.

8.2.1 Data Issues

There are a number of general data issues that arise when performing any parallel processing application. Issues arise for a number of reasons such as:

- The desire to divide data so that parallel processing can occur.
- The requirement to store data at various stages within a paradigm.
- The coded algorithms used.
- Hardware features.
- Software environment.

Dividing the data is a very important issue within a parallel processing solution. Firstly how data is divided will impact on how algorithms can and must be performed. Secondly how data is divided will effect the ability of any system to achieve the goals of that system. Thirdly for the programmer how the data is divided is crucial for fault finding and conceptualisation.

For a square image of 2^N by 2^N in size, there are a number of methods of how data can be divided. In this case this division is performed so that 4 child processors will transform the data. The methods that have been explored within this work are:

1. Four Arrays of 2^{N-1} by 2^{N-1}
2. Four Arrays of 2^{N} by 2^{N-2}
3. 2^{N} / N Arrays of 2^{N} by N in size

Method 1 is frequently used in fringe analysis as it benefits unwrapping and neighbourhood functions. However, in the parallel-processing environment such a scheme is very difficult to implement due to pointer problems. Method 2 is very useful for performing Fourier Fringe Analysis but has similar drawbacks to the Method 1. Method 3 is much less suited to Fringe Analysis algorithms but much more suited to the parallel-processing environments used in this work.

Using Method 3 allows the FFA strategy identified as scheme 1 in section 6.1.1 to be used in this research. Further using this method is highly suited to the C40 hardware and software environments due to the size and manner in which data can be communicated.

Using the scheme 2 version of FFA necessitates the use of data storage strategies. Two such strategies have been employed within this research, namely

1. Distributed Storage where each child processor stores the part of the data it has transformed. The required segments of this data are then communicated to and received from each child processor in the same layer.
2. Centralised Storage where each child processor communicates the transformed data to one hub processor which stores and re-distributes data to the child processors in the layer used for the next transformation.

Comparing these two strategies it can be seen that distributed storage is a very finely textured solution, where a high number of communications between child processors are required. Centralised storage is much coarser grained in

comparison, requiring storage and communication with a hub processor rather *than 4 child processors.*

Both these strategies suffer in terms of simplicity and speed when compared to matrix solutions to the FFA problem. However, centralised storage has been used in this research because of the comparative programmer friendliness of this strategy. That is to say it is much easier for the programmer to trace where data is located at every point in time within the centralised strategy when compared to the task of tracking data in the distributed strategy. Further the reduced number of communications and pointers required make this strategy much easier to debug as well as maintain.

In terms of the C40 hardware the impact upon data arises from using the specialised signal processing features within the C40. The key features used in this research are:

1. Internal Ram blocks. Each C40 has two such ram blocks which can store 1024 variables. Using these in a real-time system restricts the image line size that can be transformed by an FFT to 512 points. Further if the unwrapping and arctangent functions are performed within the ram blocks this data size is further restricted to 256 values in the concurrent scheme of fig 5.8.
2. DMA co-processor. The DMA co-processor can be used to move data within the C40. It allows data to be moved and communicated in a number of ways that improve the optimised code performance.
3. Communication Links. Using the 6 available communication links on each C40 processor is the central idea behind all the parallel processing performed in this research.

8.2.2 Fourier Fringe Analysis Issues

Performing Fourier Fringe Analysis within a parallel environment effects **every** stage of the FFA paradigm. A large number of issues are raised by a parallel version of FFA. The first issue is that of whether to perform a full 2D analysis

or a more limited 1D version. Originally the FFA system was based around the 1D scheme-1 version of the technique detailed in fig.6.1, however as the applications of FFA grew so did the complexity of the system until the full 2D version became the standard approach. This is because of the increased signal separation and noise immunity of the 2D version.

However, for a real-time implementation this increased complexity comes at a high price. Taking just the forward transform in the 2D system requires a forward row and then a forward column transformation. In a single processor implementation this is a simple matrix problem. But when this is transferred to a multiple processor scheme it becomes a problem of communication and storage, this means an in-built delay to the paradigm. With this in mind the inverse and filter stages become further delays to the process. In effect between each processing stage of the linear algorithm there is the need for a stage which stores, changes orientation and communicates the data in a parallel algorithm. Effectively this is changing the problem from a signal-processing problem to a matrix manipulation problem.

An obvious implication of this is that where possible a version of the FFA close to that outlined originally should be used within a parallel-processing environment. This was the technique that was adopted at the end of this research.

Pre-processing within FFA is made up of a number of stages these are

- Background Intensity Removal. An image with no fringes within it is captured and then stored. Each subsequent fringe image has this background intensity image subtracted from the fringe image's data values.
- Median Filtering. Application of simple neighbourhood function.
- Average calculation and removal. The average for the entire image is calculated and then subtracted from every pixel value.

- Hamming Window. Application of either a 2D or 1D windowing algorithm to every pixel.

All the stages of pre-processing apart from average calculation and removal can be performed as signal processing steps. Removing the average of the image from each pixel can only be performed once the entire image has been stored. The capture function used in this work sends 32 lines of data to the pre-processing stage and until the entire image has been stored in the pre-processing stage for mean removal, no subsequent processing can be performed. This then is a matrix manipulation time delay.

Within a DSP version of a Fourier Fringe Analysis system the first FFT is most often a real-only transform. The function used to perform this leaves the data in a non-conventional "pseudo Hermitean" order, taking advantage of the symmetry present with the Fourier domain. In fact the transformed data will be half the size of the incoming data i.e. 2^N real values are transformed into 2^{N-1} complex values. This means that at some stage the data must be padded to equal the number of original data values.

If a 2D version of the paradigm is being performed the next calculation step is to perform a complex column FFT. However, before this can be performed the delaying matrix stage discussed previously must be undertaken. The column FFT that is performed is a conventional Fourier transform, although it is only performed on 2^{N-1} columns.

At this point in either the 1D system, which does not required the forward column FFT or the 2D system it is time to perform Fourier space filtering. For the 2D scheme within a parallel processing system this is a very complicated and intricate stage. Once again this stage cannot be performed until the entire data has been stored. To this point only half the data exists and that data which does exist is stored in a highly unconventional order. Therefore any filtering at this stage is a very complicated operation.

Not only must the data be filtered it must be padded and often frequency shifted. With the data organised in an unconventional arrangement the option to rearrange this data into a more conventional manner may be undertaken. However, this is an obvious matrix time delay.

Using the 1D FFA scheme avoids all these problems. While the transformed real-only data is still in a non-conventional format it is nevertheless much simpler to sort out as the necessary rearrangements are kept as a 1D signal-processing problem. This means that filtering, padding and frequency shifting can quickly be undertaken.

The next step in the FFA process is to perform the inverse Fourier transforms. With the 1D system this is only one stage, and once this is finished from each line or block of lines conversion to phase data can be performed. In the 2D system not only are these row transforms performed but column and matrix manipulation stages must also be performed. Once these are complete the data is once more ready for phase extraction.

The data is now passed to an arctangent function that is identical in most respects to a classical arctangent algorithm. Using a look-up table to perform conversions is the only manner in which this parallel version varies from the linear version. The same algorithm is used for PSP as well as FFA.

8.2.3 Phase Stepping Profilometry

Phase Stepping techniques are often seen as being highly susceptible to a number of errors. One counter of these errors is to increase the number of fringe images captured to perform the PSP algorithm. This results in increased noise suppression. However, in a real-time version of PSP the number of steps is greatly restricted by the demand for speed. Additionally the measurement of a dynamic system means that performing a great number of steps is not possible due to changes in the surface shape between steps.

Therefore the PSP algorithm used is restricted by the application rather than by the parallel-processing environment. Unlike the FFT algorithms phase stepping algorithms are pixel by pixel in nature, this means that they are much simpler data manipulation problems. However, the complexity in phase stepping generally arises from the requirement to move the fringe pattern between each image captured.

In this work three basic PSP algorithms have been investigated. The first a simple 3 Frame algorithm is fairly straightforward to implement. Once three fringe images have been captured the PSP algorithm is performed by the Framestore processor. The result of performing the required algorithm is that the PSP data can be stored as one "complex" format image. This means that the Framestore can reduce the data size that must be communicated. This allows the data before phase extraction to be communicated as a complex image to the child processors.

The next PSP algorithm investigated was the Carré technique. This method is much less effected by noise than any other practical PSP technique. However as four images are required and the algorithm is of increased complexity making this technique is somewhat more difficult to perform.

The final PSP technique explored was the “2+1” technique. One of the advantages of this technique is that it was designed with high-speed applications in mind. This technique requires only 2 images to be captured for every cycle, although initial calibration images are required and must be stored on the processor. These are used to calculate the background light intensity. Additionally the information required to extract phase can once again be communicated in the correct format from the capturing processor very simply.

8.2.4 Unwrappers

The development of unwrapping solutions has followed a very similar path to the progress of Fourier Fringe Analysis, moving away from a signal-processing problem to become a 2D image-processing problem. Once again this results in

an in-built time delay when using a parallel processing system. Time must be spent collecting and then redistributing data in the correct order. As well as this delay many of the most effective unwrapping solutions are performed using some version of neighbourhood manipulation. Although this is indeed possible on the C40 and could be said to be merely a pointer arithmetic problem this doesn't tell the whole story for a real-time solution.

Such refinements of the unwrapping process do lead to a greater ability to unwrap complex images. However, this must be measured with respect to the increased time and memory consumed by these complex unwrappers. Once again these highly developed ideas have been discarded in favour of techniques closer to the original signal-processing solutions.

If a row or column only unwrapping algorithm is performed this can be kept as very simple 1D problem. However, although neat this system does not unwrap real images well enough to be used. Therefore an unwrapper that uses both row and column information is performed in this system.

As a data partitioning scheme must be used in a parallel system i.e. there must be some parallel processing, there is a need to communicate some data to enable unwrapped sections to be joined together. This means that the algorithm of a simple unwrapper becomes more complicated in the parallel system. However, this is not a great increase in complexity and does allow the unwrapper to still be performed as a signal-processing technique.

It is worth noting at this point that in a signal processing based solution the performance of the unwrapping strategy will dictate how every stage in the fringe analysis techniques are performed. That is to say that unlike traditional fringe analysis solutions where unwrapping is the final stage which can be performed independent of all other stages in a parallel processing system unwrapping becomes the critical stage. In effect a parallel solution is designed from the last algorithm back up to the first algorithm, reverse design.

8.2.5 Concurrent paradigms

The implications from concurrently performing Fourier Fringe Analysis and Phase Stepping Profilometry re-enforce many of the points that have already been discussed. Firstly any concurrent system not only requires time and space for each of these paradigms to be performed but also for each of the phase extraction techniques to perform an arctangent and unwrapping stage. This means that the restrictions on the size of data that can be used are re-enforced.

Another consequence of using concurrent systems is to make the placement of tasks of heightened importance. For example either the Framestore processor or child processors can perform the pre-processing functions required. In FFA, where pre-processing is dependent upon the entire image, under a concurrent scheme it is better to perform the required manipulations upon one of the child processors. This child processor now becomes a hub processor used to store raw image data and distribute the pre-processed data. Within PSP the algorithm, which in the concurrent system will be of the "2+1" method, is once again performed on a hub processor. The reason for performing all of this pre-processing on processors other than the Framestore is that it avoids the Framestore becoming a potential bottleneck within the concurrent system.

Another interesting development of using concurrent systems has been the modification of the method of performing the "2+1" technique. The original version of this technique requires two images with 180° phase shifts to be captured and one image to be subtracted from the other image. This results in the background intensity value. Within the FFA paradigm an image containing the background intensity is also used, however in the INFOCUS Sensor this is achieved by washing out the fringe pattern. The obvious implication of this is a modified method of performing the "2+1" technique using the background intensity value used in FFA. This reduces the optical and computational problem of the concurrent system.

Both Fourier Fringe Analysis and Phase Stepping Profilometry have been used in this work to measure the external body surface of people. It is the author's opinion that within the hardware and software environments used in this within this research that using FFA is the better of these two techniques for such problems. This is partially because of the need to move fringe patterns in PSP very rapidly and partially due to the increase data that must be used within PSP. It is the author's belief that FFA is much more beneficial in measuring this type of dynamic surface whether the computer system is parallel or simply a single processor solution.

8.2.6 Conclusions on the Effects on Linear Algorithms

There are a number of conclusions that must be drawn from the preceding discussion. Firstly, that the C40 processor is best suited to 1D signal processing solutions rather than multidimensional signal processing problems such as image processing. Secondly **any** image processing or optical metrology algorithm implemented in a parallel environment will be greatly effected by the hardware and software of that environment.

The third and most important conclusion is that anyone seeking to perform parallel processing for optical metrology **must** learn how to think and work within a parallel-processing environment.

8.3 Discussion on Parallel Processing Software Environments

Within this section the aim is to discuss the pros and cons of parallel processing environments. Using the models of 3L and Pegasus as examples of parallel software environments, broad issues of the applicability of parallel processing software to perform optical metrology techniques are examined. Firstly each of these tools is discussed in some detail, effectively providing an overview of the subject. This view is then expanded to look at less tangible issues.

8.3.1 3L

Parallel C has for an extensive period been one of the primary language tools used in programming DSP processors. The environment most often runs within a simple DOS based system, although with the recent Windows Server package 3L systems can now be run from the Windows Environment. This interface to the Windows environment is a very basic way of accessing the DSP code from a PC Host system.

As a basic language tool Parallel C is robust enough to allow for fairly complex systems to be built. However, there are a number of drawbacks with this system, which have been experienced in this work. The first of these relates to problems that arose as a concurrent system was built in the 3L environment. It became apparent that as the system evolved to become a fully concurrent implementation the environment limited some of the decisions made by the programmer.

The next problem relates to the user-friendliness of the 3L compiler. This very powerful tool is not a user-friendly system, which can make debugging, and error checking virtually impossible. One of the worst problems experienced with this system within this work was the requirement to rebuild every stage of any parallel system as a full system was built upon. This required a great deal of time and effort.

It is fair to say that this 3L product is very much a traditional programming environment adapted to parallelism. This of course has benefits such as the ability to easily take full advantage of the C40's specialist hardware features but these are largely outweighed by the disadvantages when compared to systems based more strongly in the Windows environment.

8.3.2 Pegasus

While the 3L system is a very basic environment Pegasus is a very much more intricate system. Pegasus works on a number of levels. Firstly there is the simulation code which is used to model the DSP on the Host system. This allows ideas and code to be tested as well as shared between the systems. One of the key benefits of this is that data models can be tested in a very interactive way. However where code is not common to both systems, code for the Host PC can become a distraction from the DSP problem.

The next steps in the Pegasus system are to convert common code to DSP threaded code. Using two systems, the second of which is a DSP compiler based around the Parallel C compiler, to achieve this. Obviously this means that this system is susceptible to some of the same compiler problems as the 3L system.

Finally there is the DSP server which acts as the interface between the host and DSPs in the same way as the 3L server. One of the advantages that Pegasus holds over 3L is that it allows real-time control tools to be built, which are Windows based but can control the DSP system. This allows a user-friendly front end to the system to be built.

A further advantage of the Pegasus environment is that it allows users to build large complicated systems from re-usable DLL modules. This means that code which has been used to solve one problem may simply be plugged straight into another problem, such as the arc-tangent and unwrapper blocks which were used initially in the FFA system but simply plugged into the PSP paradigm much later on. An obvious step from here is that very quickly problems can be modelled, that is as long as the code already exists.

However, Pegasus is a relatively new signal-processing system and is as yet not truly an image-processing solution. Yet the flexibility of the system is such that a user can begin to change the system to solve such problems.

8.3.3 CASE Tools for Image Analysis

As has been demonstrated so far in this section there are two approaches to providing a software environment in which parallel image processing can occur. Both of these approaches have their own advantages and drawbacks. It is now time to see if these characteristics make using such systems to solve practical image processing problems worthwhile.

One of the crucial arguments for using CASE tools is that they allow object-oriented solutions to problems. For example with a more traditional programming environment to re-use a coded solution to a problem in another system, would mean re-writing the code or rather cutting and pasting code. However, in an object orientated CASE tool exactly the same block of code may be reused, with no need to recompile the task as the same DLL is used repeatedly.

A further argument in favour of CASE tools is that they allow solutions that are Windows based and user-friendly. For example a filter block may use one set of values for parameters in the simulation and a different set in the DSP environment. In fact these values may be changed in the run-time version of the DSP system, meaning that recompiling for changes in variable values is not required. This allows a much greater ability to quickly test ideas than under a less flexible system where new values would mean a new compilation.

Another advantage of using a CASE tool is that if a solution to a problem does not require new blocks to be built, the designer of a system does not need as much chip specific knowledge as is required by traditional solutions. Extending this further means that methodologies and ideas, which use existing code, can quickly be modelled and undertaken. Yet another advantage of this is that different DSP chips make no great difference to the non-expert user. Traditionally skills learnt on one DSP chip are not easily transferred to other DSP chips; some learning period is required for each new hardware solution.

When compared to more conventional tools newer environments can be seen to shift the balance between programmer and compiler to the advantage of the programmer. That is the compiler is becoming more effective, efficient and powerful. This means that lower level programming may no longer be a crucial requirement within a DSP system.

For all these very good points there are a number of drawbacks with using the CASE tools as they currently stand. The first, and most significant of these for image analysis, is that CASE tools presently available are aimed squarely at the signal-processing market. What this means is that solutions are not simply the plug and play ideas outlined previously. If no code to solve the problem exists the user must write the code. This is not a simple task and requires a number of skills, such as some ability to program in the Windows environment and also to program for the DSP processor used.

This leads to the next problem namely the difficulty of interacting with the Windows environment, not only for the programmer but also for the processors. In image analysis where large amounts of data are required the speed and efficiency of the host PC processor becomes a bottleneck.

Another slight drawback to CASE tools arises partly from this desire to perform Windows interactions and partly from the environment performing all information routing to the Host, there will be a decrease in the achievable speed of the final system when compared to a more basic traditional type of system. Partly this lowering of achievable speeds is due to the increased functions and partly due to the environment performing virtual tasking. However, although this maybe considered a drawback for some applications, in general the increased ease of use more than outweighs the slight speed trade-off.

8.3.4 Conclusions on CASE Tools

Characteristic	Conventional	CASE Tool
Speed	Very high speed due to simplicity of the system	Slightly lower speed due to complex nature
Modularity	Not modular	Highly modular allowing easy re-use of code
Error Checking	Very hard to error check until either compilation fails or tasks hang	Allows the use of simulation to check ideas and data movement although DSP code suffers same problems as conventional
Windows Interaction	Complex due to needing to write hooks from DSP code to Host	Still complex but much of the basic work is done by the system
User Expertise	Expert at using DSP chip, Expert at a large number of Programming languages, Some knowledge of problem	Understanding of key features of DSP chip, Expert programmer if new code is required otherwise basic programming skills, Expert in the problem to be solved.

Table 8.1. Highlighting some of the differences between Conventional DSP environments and CASE tool environments.

So are parallel CASE tool solutions worth pursuing? The simple answer to this is yes. As can be seen in Table 8.1 there are a number of problems with CASE tool environments. And although these environments are constantly improving they do not yet allow image analysis to be merely a plug and play solution.

The potential for a system, which does achieve this, is limited only by the imagination. The easier the tools become for image processing the greater will become the demand for increasing the level of abstraction that the user sees. That is to say as CASE tools become more widely recognised and used so the level of user expertise in DSPs and coding will decrease; in short bringing parallel processing into the mainstream. An additional point is that newer DSP processors such as the SHARC processor are designed for multidimensional signal processing. This means these systems are more suited to image analysis problems. A point to raise at this stage is that a CASE tool such as Pegasus

allows code from one processor chip to be adapted for any other DSP processor, or in the near future for any Pentium based chip.

All of this means that if CASE tools are to be used more frequently in image-analysis solutions to matrix problems must be the main thrust of the CASE tools development. And the only way for image analysis experts to influence this direction is to be in at the sharp end of development, sculpting the shape of future systems now.

Of course, the counter argument to this is that the present CASE tools are still not at a stage where a user can come to a computer and build up a full image analysis system from simple building blocks, programming in a building block style. As this means that time must be spent learning the new system many users will stick to the more conventional software but this behaviour simply ignores the future potential and could limit the future of high speed image processing.

8.4 Discussion of Parallel Processing for Image Analysis

Parallel processing has largely been within the realm of specialists in computer hardware and programming. Such users and their applications are driven by the desire to perform parallel processing. However, this work and the problems it examines were not initiated with this focus, performing parallel processing has been a means to end rather than an end in itself. The reasoning behind this choice of solution highlights some of the goals for the use of parallel techniques in an image analysis system. These reasons were:

- The need for real-time measurements
- The need for flexible measurement systems
- The need for adaptable measurement systems
- The need for concurrent measurement systems

It is now necessary to discern whether applying such parallel techniques to image analysis, with specific reference to fringe analysis and non-contact measurement, is of any **true** practical value. To achieve this the true cost and benefits of practical parallel implementations must be examined.

8.4.1. Arguments against Parallel Processing

The first argument against parallel processing is that of the time it takes to learn about the crucial elements within such environments. The processing hardware used in DSP chips often varies a great deal from the more abundant PC processors. Learning how to utilise the specialised features of such systems to their fullest potential takes some considerable time. Further as different chips have different architectures most of the experience of using one processing chip is largely not transferable to another chip architecture.

The next argument against these types of solutions is based upon that of the software environments that must be used. As has been previously discussed CASE tools have improved the software scenario but time is still required to learn how to use all the available tools effectively. Additionally skills learned under one environment may suffer from the non-transference of skills problem.

Arising from these two arguments is a third more subtle but perhaps more complex argument. This is that learning to think of problems in a parallel environment is a time consuming skill that must be achieved. In fact learning how to partition and store data in a parallel way can be one of the more complicated tasks facing the programmer. Grasping these fundamental concepts is crucial to the outcome of the system.

Any technique that is used in image analysis will make data handling the most important problem in a parallel solution. Experience about how to manipulate data correctly can only be learnt within a parallel setting. It could be argued that commissioning and attempting to solve problems with a parallel solution demands that this knowledge already exists.

Parallel systems are inherently complex in nature. Fault-finding and debugging in such environments are specialised skills in themselves. With the many different aspects present within a parallel environment finding the exact cause of any problem can be very time consuming.

The next problem that follows logically from this is that of the maintainability and reliability of the system. Any parallel system will contain a large number of different elements, how these interact and communicate means that there is some amount of uncertainty within the system. Therefore, some specialist skills are required to maintain and check such dynamic structures.

Being able to quickly implement and design parallel solutions to problems is still not an option in image analysis. If signal-processing solutions can be used then rapid development is possible. This of course means that time to test ideas and coding solutions is a cost to the system. This is one aspect in which parallel solutions still lag far behind more conventional processor systems.

8.4.2 Arguments for Parallel Processing

The first argument for parallel processing is that it is presently still in a developmental stage for practical purposes. Being involved in parallel processing now is fraught with "teething problems", but now is the time to develop ideas which foster parallel processing for image analysis. In image analysis the ability to work on a whole matrix of data is vital, not being able to use such solutions effectively forces the user to learn new concepts but also to discard certain techniques. Working in this field now enables the thrust and context of parallel processing to be pushed towards solutions of matrix problems. One of the first stages in this direction is the use of DSP processors that are more suited to image analysis than C40s.

Applications that require real-time solutions place a high demand on processing power, this is an obvious reason for choosing a parallel solution. Division of a process into a number of smaller packets that may be performed by a number of

processors greatly increases the obtainable speed to perform that function. Using a parallel solution allows this divide and conquer strategy to be utilised.

A parallel environment provides the capability to perform not only parallel tasks but also parallel paradigms. Such concurrent strategies mean that parallel solutions can be used in a much more flexible and robust manner than uniprocessor systems. Such interaction of algorithms means that innovative concepts and solutions to problems can evolve. This leads down new paths of discovery.

Although PC processors are rapidly developing and the use of such chips to perform real-time image processing is an emerging technology, the most up to date DSP chips are still someway ahead of these more conventional architectures. This means that there is still some technological advantage to using specialised hardware. Added to this as real-time signal processing environments are adapted and improved to perform image-analysis the gap in speed of development is decreasing. This gap presently exists in the PCs favour and although this will continue to be the case DSP environments are narrowing the difference.

Additionally PCs are moving closer to parallel solutions. This means that to a large extent lessons that are learnt now in the parallel field will within a few years become mainstream ideas and concepts. It also means that the crossover influence from one field to another will become more evident. This is especially the case with tools such as Pegasus which are moving into the PC based market.

8.4.3 Conclusions on Image Analysis with Parallel Processing

The problem now is to see how the arguments for and against parallel processing balance with regards to the scales imposed by the requirement to perform image processing.

It is impossible to give a strict rule of exactly where and when parallel processing should be used. However, if the problem is complicated enough and the requirements are exacting enough then it would seem to be a powerful tool to employ. For optical metrology using parallel processing would seem to be a good way forward, where the algorithms are computationally demanding. But, anyone seeking to follow this path must be aware of the problems and pitfalls that lie ahead.

Taking three scenarios of research project some idea of whether or not to use parallel processing in fringe analysis can be better stated.

1. If no prior knowledge of applying parallel processing exists then it is the author's view that parallel processing **should not** be used due to the steep learning curve associated with using parallel processing.

2. If one of the more senior members of the project team has proven knowledge of implementing parallel processing then parallelism is a good option but only if the specifications to be met are rigorous enough.

3. If the systems engineer developing a solution has a good in-depth knowledge of implementing parallel processing techniques then the advantages in terms of the obtainable speed and the ability to build redundancy into the system, in the author's opinion makes parallelism a very strong option.

8.5 Future Developments for the INFOCUS Sensor

The first development to the INFOCUS system should be an improved framegrabber. The system used in this research relies on the low level programming ability of the user. This means that variations of the capture routine require detailed knowledge not only of the hardware but also of the

software structures used in the Transtech system. Improving the capture system would speed-up the time to obtain and move data with in the system.

Any research that relies heavily on computer processing power will suffer from the effects of time, as the state of the art becomes common place. A processors technological lead will become outmoded and surpassed by new or improved technology. An obvious example in this research is the improved SHARC technology currently on the market[1,2]. Additional speed-ups and improvements such as increased image sizes may be obtained by using this technology which is designed for multidimensional signal processing.

Recent developments in the field of Field Programmable Gate Arrays[3,4] may make this another possible hardware solution. The flexibility that a FPGA system offers may well make this one of the best solutions for systems which use a number of algorithms such as FFA.

Emerging PC technology[5] may also show considerable advantages over the present INFOCUS hardware. Systems will develop which allow parallel processing within operating systems designed with parallel processing in mind. Electing to use such systems benefits from the great number of packages available for fast implementation of matrix manipulation crucial for image processing developments.

Improving the algorithms used with this work would also benefit the INFOCUS system. Perhaps the most important of these developments is within the field of unwrappers. The simple unwrapper used in this work has demonstrated some of the ideas of how unwrappers can be used in a parallel DSP system. Taking these ideas forward and improving the unwrapping quality must be one of the next stages of evolution for the INFOCUS system.

Using concurrent systems allows data from different paradigms to be compared and contrasted. Investigating any benefits, which arise from this ability, is another field of possible research. Concurrent algorithms may well benefit new forms of unwrappers.

8.6 Conclusions

This work has proven that non-contact measurement techniques are well suited to resolving the problems of patient positioning and monitoring in a radiotherapy environment. Further this work has proven that performing concurrent measurement systems is a practical idea within this environment. Additionally this work has shown that the benefits and costs of using the present software environments of parallel processing are very finely balanced. But in order for these to shift further in favour of parallel image processing solutions the disadvantages must be accepted now so that future benefits can be achieved.

With the improving performance of PC based hardware the benefits of C40 based parallel processing for non-contact measurement become harder to justify and define. If these benefits are thought of merely in terms of speed then the disadvantages of the parallel processing will outweigh the advantages. However if criteria other than just speed are used this balance is shifted in favour of parallel processing. Perhaps this is the most important lessons that can be learnt amount practical parallel processing.

It will always be the case that a developed system will be outperformed by some new or improve chip. But those things that last are the lessons and skills that have been learnt in performing the parallel solution. In the final analysis parallel processing is the way forward but not with the C40 processor.

It is the author's opinion that other processing chips such as the Pentium 400, SHARC, FPGA or some combination of these is the way forward for parallel processing. If these tools can truly begin to match the expectations and desires of practical image processing then maybe the lessons learnt from this research can be used to shape the future of the CASE tools used to achieve the goals of fast, flexible and reliable optical metrology systems.

8.7 References

1. Quintek SHARC Pack
   www.alex.com
2. APEX SHARC
   www.spectrumsignal.com
4. Reconfigurable Computing Coming of Age
   Mokhoff..N
   www.icspat.com
5. Chaffold.G
   Designing Digital Signal Processing Systems: A New Approach
   Techonline, Vol.2, No.3, www.techonline.com, 1998.
6. Alpha Data
   www.alphadata.co.uk