An embedded adaptive optics real time controller

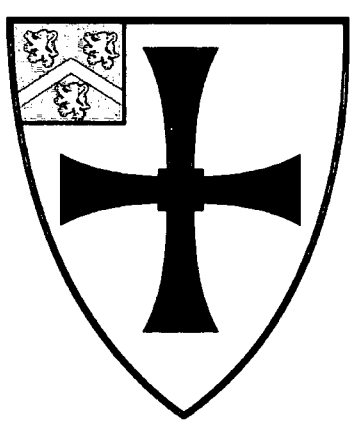
Saunter, Christopher D.

An Embedded Adaptive Optics Real Time Controller

Christopher D. Saunter

A Thesis presented for the degree of

Doctor of Philosophy



Centre for Advanced Instrumentation
Department of Physics
University of Durham
England

April 17^{th}, 2007

An Embedded Adaptive Optics Real Time Controller

Christopher D. Saunter

Submitted for the degree of Doctor of Philosophy

April 17^{th} 2007

Abstract

The design and realisation of a low cost, high speed control system for adaptive optics (AO) is presented. This control system is built around a field programmable gate array (FPGA). FPGA devices represent a fundamentally different approach to implementing control systems than conventional central processing units.

The performance of the FPGA control system is demonstrated in a specifically constructed laboratory AO experiment where closed loop AO correction is shown. An alternative application of the control system is demonstrated in the field of optical tweezing, where it is used to study the motion dynamics of particles trapped within laser foci.

Declaration

The work in this thesis is based on research carried out at the Centre for Advanced Instrumentation, the Department of Physics, University of Durham, England. Some of the data present in the results chapter has formerly been published in *FPGA technology for high speed, low cost adaptive optics Saunter C.D., Love G.D., Johns, M., Holmes, J.* from proc. 5th International Workshop on Adaptive Optics in Industry and Medicine (2005). No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all the sole work of the author with the exception of those areas listed below.

Assistance is acknowledged with PCB design - the detailed layout and fabrication of the custom DAC PCB used in the laboratory demonstrator was undertaken by the workshop services in the Physics Department at Durham. This work was carried out to the authors specification. The detailed layout and fabrication of the custom PCBs forming the enhanced prototype controller was undertaken by Sira Technologies Ltd. This work was carried out to a specification arrived at jointly by the author and Sira. The work on optical tweezing presented in chapter 7 was conducted at the Optics Group at the University of Glasgow using an experiment created and maintained by Glasgow. The results obtained and presented here are a joint result of their experimental setup and the authors control system being used as a high speed data processing camera.

Acknowledgements

Undertaking a PhD has been a fantastic experience and one I could not have started without the faith of my supervisor, Gordon Love.

During the three year period (give or take a little...) between starting and finishing my work I have had many useful discussions with members of the CfAI here in Durham, with thanks going to every member of the group, and have enjoyed the hospitality of the adaptive optics community in the UK and abroad, with special notes of thanks to Enrico, Rob and Christian at ESO in Garching, David and John-Mark (scourge of the coke machines) at Strathclyde and Sarah from London.

I am grateful to Miles Padgett from the University of Glasgow for the opportunity to work with their group, thanks to all this group for their hospitality and also to a fellow visitor there, Roberto di Leonardo, for a crash course in the behaviour of optically trapped particles and for ploughing through half a gigabyte of data.

I would like to thank the members of Sira Technology Ltd. with whom I worked for their hospitality, wonderful canteen and technical assistance. I would especially like to thank Kate for saving parts of my work during the demise of Sira.

Finally no instrumentation thesis would be complete without acknowledging the impact (!) and assistance of my contemporary travellers from Room 4, including Harold D, Tom O, Tim M, Tim B, Paul B, Jason C, Mark H and James O.

Contents

List of Figures

Chapter 1

Introduction

1.1 Synopsis

The first proposal of a method for reducing the effects of atmospheric seeing was from Horace Babcock [1] in 1953. The development of practical adaptive optics (AO) systems for such purposes required significant advances in detector technology and signals processing technology - particularly the development of the integrated circuit and central processing unit concepts necessary to provide the large real-time processing requirements.

During the last decade AO has become a de-facto technology installed on most ground based astronomical telescopes in the 4-10M diameter range and is integral to the design of future large telescopes in the 10-50M range. Recently AO has found applications to other fields such as ophthalmology, improving the resolution of retinal images[2] enabling the [3] mapping of waveband specific cones in the human eye, high power laser beam shaping to improve performance for inertial confinement fusion research [4, 5], and increasing the performance of free space optical communications links[6].

As the use of adaptive optics increases in these fields the possibility is raised of AO being used in commercial instruments and devices as well as a tool for scientific research. This thesis presents the development of a low cost, high speed control system intended for use in such commercial AO systems, offering lower cost and smaller sizes than existing AO control technologies. The control system presented here is based around field programmable gate array technology (FPGA), a relatively new class of microchip device radically different from traditional the central processing units (CPUs) and digital signals processors (DSPs) employed by existing AO control systems.

1.1.1 Chapter 2

This chapter presents a theoretical overview of adaptive optics systems intended for high bandwidth correction with an onus on exploring the devices and algorithms used to implement such a system.

1.1.2 Chapter 3

In this chapter the various options for implementing a real time control system, such as microprocessors and FPGA devices are examined, and a rationala given for the choice of FPGAs for this work. The operation and programming of FPGA devices is also examined in this chapter. Finally a review of other work on using FPGA devices for AO is presented.

1.1.3 Chapter 4

Chapter 4 presents a framework for real time image processing and control that was developed and the AO control system built on this. A broad view of the developed architecture is given, along with detailed examination of key components for AO control.

1.1.4 Chapter 5

Chapter 5 examines the design and construction of a laboratory AO system, including a turbulence source and an imaging system, for testing and developing the control system.

1.1.5 Chapter 6

Chapter 6 presents the operation and results gained from the laboratory demonstrator.

1.1.6 Chapter 7

This chapter examines further development of the control system, presenting an enhanced prototype that is a more compact system of packaged electronics providing a compact AO control system. Also examined are alternate uses of this control system including a tip/tilt beam steering control loop for free space optical communications and an open loop camera with in-built data processing and reduction, a 'smart camera'.

1.1.7 Chapter 8

Finally in chapter 8 concluding remarks are drawn.

Chapter 2

Adaptive Optics Theory

2.1 Introduction

This chapter introduces the theory underlying an adaptive optics system, paying particular attention to the algorithms used as a precursor to their implementation in chapter 4.

An adaptive optics system controls the phase of a light beam, both spatially and temporally, to modify the wavefront to be some normally flat shape. Whilst the temporal bandwidth of many sources of turbulence requires an adaptive optics system to run at high update rates the majority of the calculation involved in performing these updates are concerned with processing the spatial qualities of the wavefront. As such the spatial and temporal aspects of wavefront control are often presented and implemented separately, and are examined in sections 2.2 (Spatial control) and 2.3 (Temporal control) below. Section 2.4 draws these areas together to show how a complete system is calibrated and operated. Finally section 2.5 looks at methods used to reduce static errors.

2.2 Spatial control

2.2.1 Open and closed loop control

The two principle forms that may be taken by an AO system are open loop and closed loop, as shown in figures 2.1 (a) and (b). These figures represent a simple imaging system in which a distant point source is imaged by the lens L1. In the absence of any distortions the point source produces a flat wavefront, θ_0, leading to a diffraction limited image.

Distortions to the incoming wavefront caused by perturbations, for example atmospheric turbulence, aberrate the beam, leading to a distorted wavefront θ_a and subse-

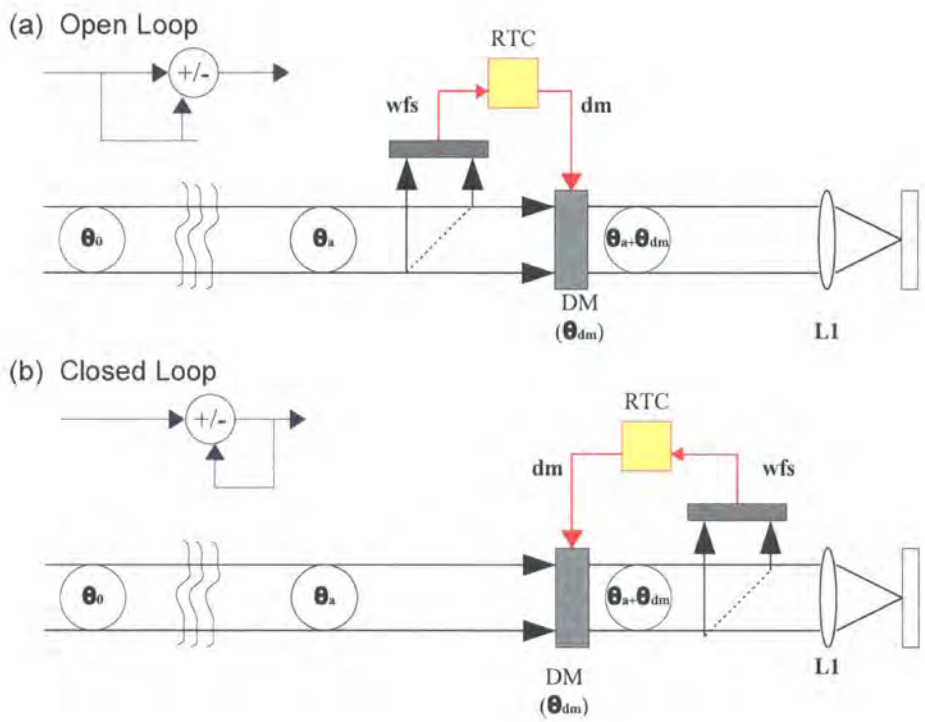


Figure 2.1: Model AO Systems - simple open loop (a) and closed loop (b) AO systems in which a flat wavefront θ_0 is distorted with a wavefront of θ_a before being corrected by a deformable mirror (DM). The DM is driven by a command vector **dm** derived from the wavefront sensor (WFS). The positioning of the WFS determines the open- or closed-loop nature of the system.

quently a distorted focal plane image. The system is examined during a time period shorter than the coherence time of the perturbations, such that they may be considered static.

In both systems an optical element with a configurable phase, the deformable mirror (DM) in the figure, responds to the control signal **dm** by adding a phase of $\theta_{dm} = DM(\mathbf{dm})$ to the beam, where the function $DM(\mathbf{x})$ represents the mirror's optical response to its electronic control vector $\mathbf{x}$. The aim of this is is to counteract the aberrations in the beam.

In the open loop case a wavefront sensor (WFS) measures the phase of the aberrated beam, quantifying it in the form of a series of measurements, collectively referred to as the wavefront sensor vector, **wfs**. This is processed by the real time controller (RTC) into the DM vector suitable for driving the DM with an equal and opposite phase thus removing the aberrations.

In the closed loop case the WFS is placed after the corrector, causing the WFS to measure a phase of $\theta_{\mathbf{w}} = \theta_{\mathbf{a}} + \theta_{\mathbf{dm}}$. This is processed by the RTC into commands for the DM that should cause the DM to reduce the measured phase to 0, i.e. $\theta_{\mathbf{dm}} = -\theta_{\mathbf{w}}$. In the close loop system the wavefront sensor is operating in a nulling mode in which the system acts to reduce the signal on the sensor to $\mathbf{0}$, reducing the dynamic range requirements on the WFS, which in practice simplifies design and allows for greater precision over this reduced range.

The majority of AO systems follow the open- or closed- loop models thus described with some form of RTC algebraically manipulating the WFS signal into DM commands by use of digital signals processing (DSP), although there have been examples of systems where the transformation is performed by a trained neural network instead such as early work with the MMT [7] or where the transform is omitted entirely and a high speed optimisation loop in employed instead [8] in a similar, but significantly faster, method to the static aberration correction outlined at the close of this chapter.

2.2.2 Wavefront Sensing

The WFS is a device that measures the wavefront or more usually some quantity related to the wavefront, presenting these measurements as an electronics signal suitable for further processing. Below a generic overview of wavefront sensing is presented followed by a brief examination of common WFS types used in AO.

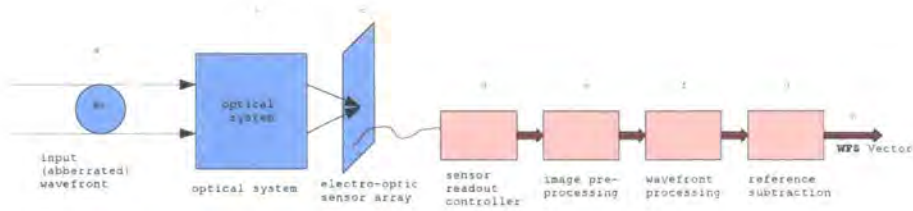


Figure 2.2: Generic wavefront sensor - most wavefront sensors can be represented by the series of optical end electronic signals processing shown in the figure

2.2.2.1 Generic Wavefront Sensor

A generic wavefront sensor is illustrated in 2.2, used to examine an incoming wavefront illustrated by (a) in the figure, with a wavefront of $\theta_{\mathbf{a}}$ and ultimately produce a vector **wfs** describing some quantities of this wavefront, (h) in the figure.

The sensor is examined in stages with a system consisting of optics (b) to convert the phase information into an intensity pattern, a sensor array (c) sampling this pattern into an electrical signal, typically analogue at this stage. The readout of this sensor array is performed by a controller (d) which emits the intensity of the sensor elements as a digital data-steam that may be regarded as an image. Image pre-processing (e) is then carried out related to the sensor array - this is independent of the type of WFS system, and is intended to compensate for non-uniformity in the sensor array and to reduce the effect of background light etc. The image is now ready for wavefront processing (f) which is specific to the type of WFS involved and extracts the measurements encoded by (a) from the image. Finally to compensate for imperfect alignment of the system, and in a closed-loop system to convert the sensor to a nulling mode a reference wavefront vector may be subtracted (g), with the system emitting the 'wavefront sensor' vector, **WFS**.

Most wavefront sensors can be constructed by taking this generic system and customising just the optical processing (a) and the wavefront processing (e) stages. With this in mind the AO framework developed and explored in chapter 4 seeks to be as generic and modular as possible enabling new modes of operation to be created by only changing these two components.

(b) - Optical processing

This initial stage of the system uses some physical optical system to convert the aberrated phase of the beam into an intensity pattern. Typical approaches rely on either

specialised forms of interferometry such as point diffraction interferometry, [9] or lateral shearing interferometry or dedicated AO sensing technologies such as the Shack-Hartmann, Pyramid and Curvature sensors outlined below.

(c) - Sensor array

The sensor array converts the optical intensity signals produced by (a) into electrical signals for further processing. A typical sensor array in an astronomical environment must make highly efficient use of the available light levels leading to the use of discrete avalanche photo-diode elements, as in the Subaru AO system [10] for example, or a charge coupled device (CCD).

An array of discrete sensors, such as those used by the Subaru AO system, often require fibre coupling of the sensors to the focal plane output of the optics, and a large quantity of supporting electronics making such a configuration both expensive to manufacture and physically bulky - neither trait suitable for the goal of a low cost system. CCDs as used in many astronomical AO systems represent the pinnacle of current devices with peak quantum efficiencies of 90%, frame rates of over 4000Hz and low read noise [11].

(d) - Controller

The controller provides the power requirements for the sensor array, sequences the readout of photo-electric charge from the elements and performs the amplification and subsequent analogue to digital conversion of these charges. Typically the controller will also set the frame rate, exposure and other similar parameters for the sensor array.

For an APD based system the controller is likely to be a custom development, whilst high end CCD devices tend to be interfaced to controllers from one of several prominent manufacturers such as SDSU or SciMeasure. Either approach is prohibitively expensive for a low-cost system. Previous work on low-cost adaptive optics [12][13], has focussed on the use of Dalsa CCD cameras which package a CCD device and Dalsa controller into a single physical device. A typical cost for such a camera is US$6000 which whilst significantly less than for a separate CCD and controller is still high.

(e) and (f) - Image pre-processing and Wavefront processing

These two stages occur on the digital data emitted by the controller, and are both digital signals processing operations. (e) consists of various per-pixel operations to compensate for non-linearity effects in the sensor array, and to mitigate the effects of 'hot' and 'dead' pixels in the array. (f) constitutes the bulk of the processing and is specific to the type of wavefront sensor, for example calculating the position of centroids for a

Shack-Hartmann. The output of this section is the wavefront sensor vector, **wfs**.

(g) - Reference subtraction

When observing a flat wavefront a perfect WFS used in closed-loop should produce an output of $\mathbf{wfs} = \mathbf{0}$. However due to static aberrations in the system (see later sections) and the difficulty of achieving perfect alignment in a real system this is rarely the case, and the sensor will output some constant but non-null signal of $\mathbf{wfs_0}$ when observing a flat wavefront. To convert the sensor to nulling operation $\mathbf{wfs_0}$ must be recorded before closed loop operation is commenced and then subtracted from each WFS vector during closed loop operation, the resultant vector $\mathbf{wfs'} = \mathbf{wfs} - \mathbf{wfs_0}$ is the final output of the wavefront sensor.

2.2.2.2 Geometric wavefront sensors

All geometric wavefront sensors may be modelled with the intensity transport equation (ITE), which describes how the intensity of a beam changes as it is propagated past a phase distortion.

The basic application of this equation is shown in figure 2.3(a) in which a beam of uniform intensity passes through some aberration and is then propagated some further distance x to a plane in which the intensity encodes the phase information, which may be determined by an inverse solution of the ITE. This however assumes uniform illumination of the pupil which may not be guaranteed in a real system, and is computationally expensive.

The Hartmann sensor as illustrated in figure 2.3(b) simplifies the extraction of data from images generated using the ITE. A mask with a regular grid of holes in it, referred to as the Hartman mask, is inserted before the sensor array causing only parts of the input beam to be sampled. These parts now cast a discrete array of bright spots on the sensor, the position of which may be measured to determine the gradient of the phase of the input beam as sampled by the corresponding mask holes. Variations in pupil intensity structure will cause the spots to vary in intensity, but not position, so the phase measurement is now insensitive to pupil intensity. Further the extraction of the spot positions is computationally simpler than a reverse solution of the ITE. However such a system is inherently wasteful of the input light and is prone to errors as only a fraction of the beam is physically sampled.

The Shack-Hartmann (SH) sensor [14]. shown in 2.3(c) is a modification of the Hart-

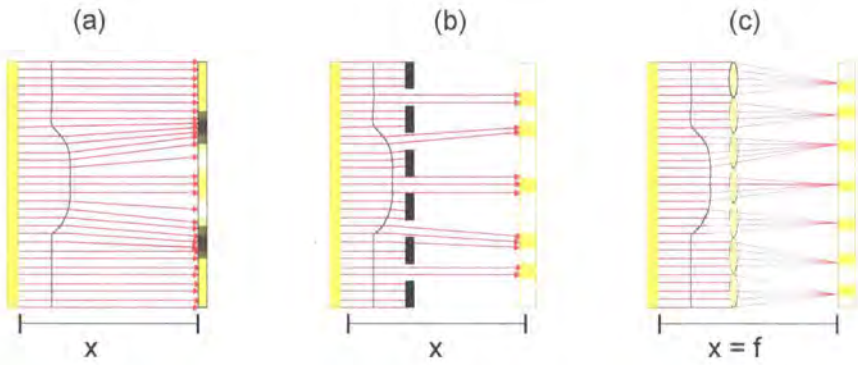


Figure 2.3: Geometric wavefront sensing - geometric propagation encodes wavefront slope in intensity in some plane (a) with a mask (b) or micro-lens array (c) employed to simplify the extraction of gradient from the intensity.

man device in which the mask is replaced with an array of lenses, often referred to as a 'micro-lens array' when fabricated on a small scale. The lenses allow the full pupil to be sampled, reducing susceptibility to aliasing and making more efficient use of available light. Further the spots produced by the device are well focussed compared to the Hartman mask, allowing their position and motion to be more accurately determined.

2.2.2.3 The Shack-Hartmann Sensor

The basic theory of a SH device is that the average wavefront tilt across one micro-lens, or 'sub-aperture', will cause a displacement of the corresponding spot in the image plane, where the displacement is proportional to the average tilt, so long as the angles are sufficiently small for the paraxial approximation to hold. For example, a reference flat wavefront $\mathbf{W_0(s,t)}$ is propagated through the microlens array as shown by the red lines in figure 2.4, forming an array of bright spots in the microlens focal plane, which is imaged by a pixelated detector array. A perturbed wavefront, shown in blue and described by $\mathbf{W'(s,t) = W_0(s,t) + \Delta W(s,t)}$ perturbs the positions of the corresponding focal plane spots with a displacement of

$$\Delta x = f.\frac{\delta \Delta W}{\delta s}; \Delta y = f.\frac{\delta \Delta W}{\delta t}. \tag{2.1}$$

Typically the sensor-specific processing for a Shack-Hartmann consists of calculating the displacement of each of the spots cast by the microlens array and outputting these values as the WFS vector. Other methods do exist such as the Fourier transform method

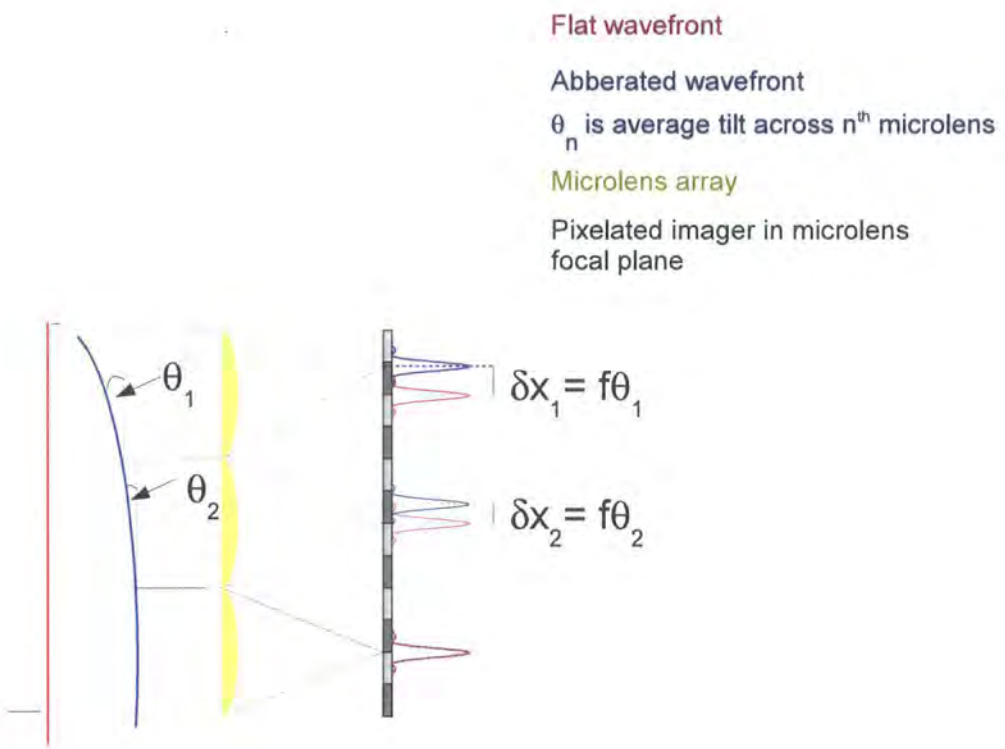


Figure 2.4: The Shack-Hartmann Sensor - an incoming flat wave, shown in red, produces a regular array of spots on the pixelated detector when propagated through the microlens array. A distorted wave, sown in blue, produces an array of displaced spots, with displacement depending on local wavefront gradient.

demonstrated by Carmon and Ribak[15], although such methods are in practice not found in high speed systems due to their computational overhead.

The position of a spot may be measured by a variety of centroiding algorithms, the most common of which is the center of mass algorithm in which the image is broken into a series of regions corresponding to individual spots, and the center of gravity of each region is calculated where mass is equivalent to the light intensity. This is done by multiplying the pixel grid values with linearly ramped weighting vectors in both dimensions and dividing the sum of weighted intensity by the sum of intensity as shown in the following equation

$$C_y = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} i.p_{i,j}}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} p_{i,j}}; C_x = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} j.p_{i,j}}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} p_{i,j}}. \tag{2.2}$$

where C_y and C_x are center of mass in 2 dimensions, $p_{i,j}$ is the intensity of the pixel at coordinates $y = i, x = j$, and n is the number of pixels across the square subaperture.

In the simplest case of a quad cell arrangement in which just 2x2 pixels of a sensor array or 4 individual sensors such as APDs are used this can be represented as

$$C_y = \frac{p_{1,0} + p_{1,1}}{p_{0,0} + p_{1,0} + p_{0,1} + p_{1,1}}; C_x = \frac{p_{0,1} + p_{1,1}}{p_{0,0} + p_{1,0} + p_{0,1} + p_{1,1}}. \tag{2.3}$$

Note that in these equations the weighting vector runs from 0 to $n-1$, where n is the number of pixels across a subaperture - often weighting vectors are used running from -n/2 to +n/2, such that a spot in the center has a centroid of 0. How the enumeration is chosen is somewhat arbitrary, although a system working with positive only numbers relaxes the requirements on the division operation, which can be of benefit for hardware implementations.

2.2.2.4 The Pyramid Sensor

The Pyramid wavefront sensor measures the first spatial derivative of optical phase across a pupil, as with the Shack-Hartmann. However they use significantly different optics detailed by Ragazzoni [16] to produce four images, with co-located pixels in each image forming a 'virtual quad-cell' that may be centroided to yield the gradient for the corresponding location in the pupil. They have been shown to have superior performance when used in closed loop systems where photon noise is an issue - i.e. low light levels, compared to a Shack-Hartmann [17]. The Pyramid sensor has primarily seen use in astronomical AO systems although Ragazzoni et. al. have demonstrated the use of Pyramid sensors to

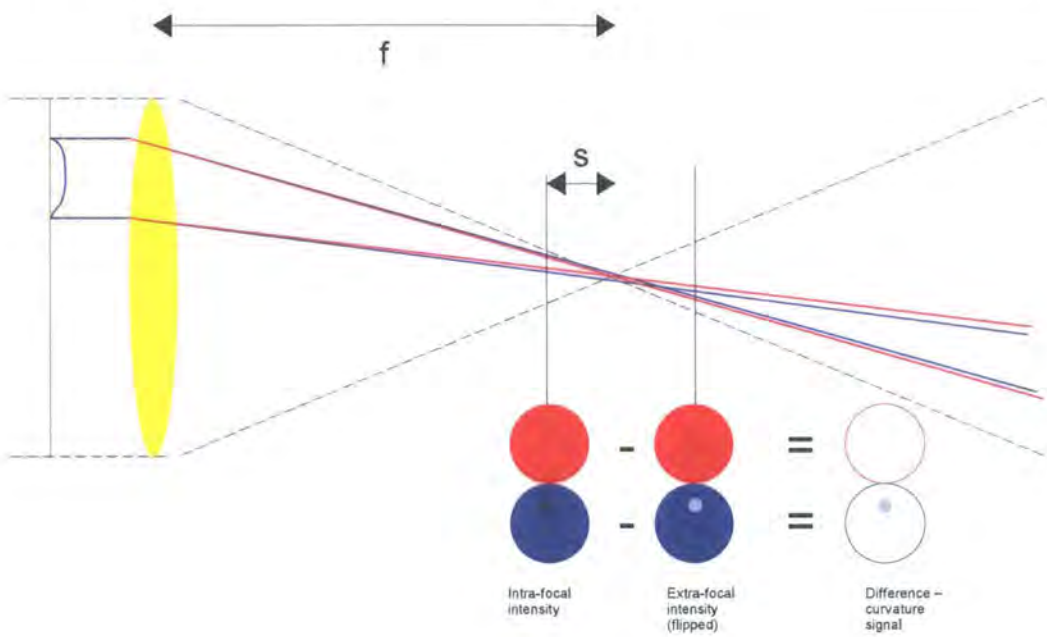


Figure 2.5: The Curvature Sensor - a flat wavefront, shown in red, is sent through a focus by a lens. If imaged in two planes equally spaced either side of the focus two images of uniform intensity are seen. When a distorted beam with some local curvature , shown in blue, is propagated the distortion causes an earlier focus for part of the beam, raising the signal in one imaged plane and lowering it in the other.

measuring aberrations in the human eye [18] and to running a closed loop AO system on the eye by Dainty et al[19].

2.2.2.5 The Curvature Sensor

Unlike the SH and pyramid sensors a curvature sensor[20] measures the second derivative, or curvature, of the wavefront. The basic concept of the curvature sensor is illustrated in figure 2.5 - a flat wavefront, shown in red, is brought to a focus by a lens and is examined in two planes equally spaced either side of the focus. For a flat wavefront the illumination in these two planes will be uniform. When a curvature is introduced into part of the beam, as shown with the blue lines in the figure, this curvature shifts the focus axially, causing the intensity in one of the sensor planes to increase and a corresponding decrease in the other plane. The wavefront sensor signal is generated by subtracting images from the two planes to yield a difference frame, which is then divided into a series of regions, the intensity of each forms one element of the **WFS** vector.

As a curvature sensor measures the second derivative it is insensitive to the first derivative - the gradient - of the beam. A global gradient or tilt will cause the intra- and extra-focal images to translate in the two planes rather than modifying the intensity structure within them, leading to registration errors between the beam and the sensor elements leading to false signals. This can be avoided by using a separate system to remove global tip/tilt or by monitoring the position of the pupils on the intra- and extra-focal images and compensating in software. Such compensation however requires waiting for the complete images to be acquired before the tilt can be correctly determined, introducing a significant latency with adverse effects for high speed systems. Again, as with the Shack-Hartmann sensor the requirements for a larger tilt range lead to a lowering of the speed at which the sensor must run, again differentiating the design of open- and closed-loop systems.

2.2.3 Wavefront Correction

The wavefront corrector (WFC) is a spatial light modulator (SLM) that may be a transmissive or reflective electro-optic element that modifies the phase of a light beam according to received control signals. The two main methods employed to achieve this are changing the refractive index of transmissive materials and moving the surface of a reflective material, i.e. deforming a mirror. First the operation of a generic corrector is examined below, followed by a summary of commonly used technologies, broken down into transmissive and reflective devices.

2.2.3.1 Generic wavefront corrector

The optical phase of the WFC is represented by θ_{dm} in figure 2.2. This phase is a result of control signals (typically analogue voltages) applied to the N_{dm} actuators on the device which are collectively referred to as the 'DM vector', **dm**.

For an ideal wavefront corrector the response of each control channel is linear, and is orthogonal to the response of all other channels such that the phase response of the device may be represented as a linear superposition of the individual control channel response functions, i.e.

$$\theta_{dm} = \sum_{i=0}^{N_{dm}-1} dm_i R_i \tag{2.4}$$

where r_i is the phase response of the ith actuator and dm_i are the elements of **dm**.

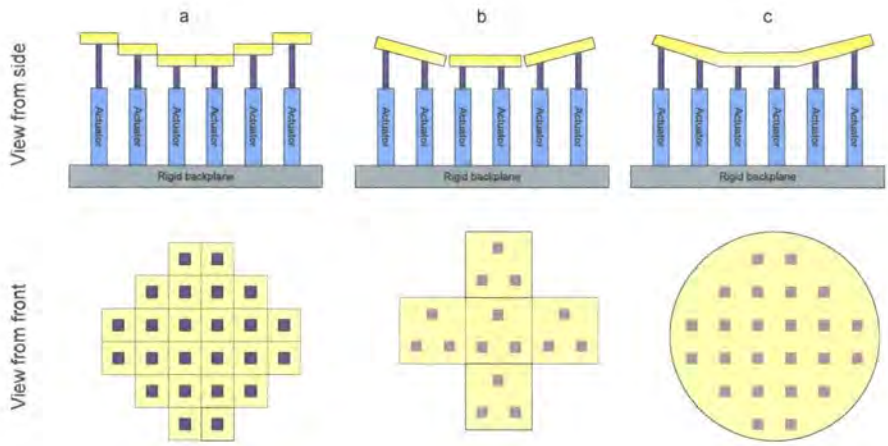


Figure 2.6: Types of push/pull actuated mirror - (a) shows a segmented piston only device that forms a stepwise approximation to a wavefront, (b) shows a device with three actuators per segment providing tip/tilt/piston control over each segment and (c) shows a continuous facesheet mirror

2.2.3.2 Reflective devices

Position actuators

The most common forms of reflective devices consist of either a continuous or segmented sheet that is coated and polished for reflection on the front face, and bonded to push/pull actuators on the rear face. The three such forms a device may take are illustrated in figure 2.6.

The simplest device in (a) consists of a series of small tessellated mirror segments, each bonded to a single push/pull actuator. The individual segments can then be moved to create a step-wise approximation to the wavefront, with discontinuities at the segment edges. In practice this is a rarely seen form of device.

More common is that shown in (b) which again consists of a series of tessellated segments, but this time bonded to three actuators to give control of the tip/tilt and piston of each segment. An example of this type of mirror is the Thermotrex DM used in the NAOMI AO system[21].

Finally (c) shows the most common form of device, a continuous facesheet DM, in which a continuous sheet is bonded to and deformed by multiple actuators.

Various technologies are employed for the actuators depending on the required actuator

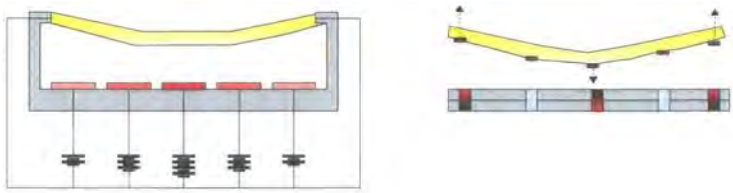


Figure 2.7: Types of force controlled mirror - in (a) an electrostatic potential is used to deform a thin membrane by means of attraction and in (b) the magnetic force is used to attract or repel mirrors bonded to a membrane

pitch, desired displacement range and temporal response etc. A typical example for a fine pitch device are the μDMs [22] from Boston Micromachines, which use parallel plate capacitors formed with MEMS technology to drive force actuators with a pitch of 60um and a stroke of 3.5um. For intermediate scales with a pitch on the order of 1cm, actuators exploiting the piezoelectric effect are a common choice and may be found on high quality astronomical AO DMs from companies such as Cilas and Xinetics. Typically the actuator pattern for such a mirror is either a square or hexagonal grid.

For larger scales such as devices intended to act as an adaptive secondary mirror in a telescope larger forces are required, leading to the use of physically bulky but effective magneto-strictive actuators, for example in the carbon fibre composite work from UCL[23].

Flexure actuators

An alternative form of mirror is possible in which a thin Piezo material is bonded to the rear face of a thin reflecting membrane, these devices are normally called Bimorph mirrors. By applying a voltage across an area of piezo material it may be induced to expand or compress laterally compared to the reflecting membrane, producing a bending moment much like in a bimetallic strip. Typically radial actuator patters are seen on the Piezo wafer, leading to the use of a radially patterned curvature sensor (example is Subaru) or occasionally a radially patterned Shack-Hartmann as chosen to complement the radial mirror pattern the AO system for the Vulcan laser [24] at RAL. Some work has also been conducted on building such mirrors with a regular grid of actuators[25].

Force actuators

The second class of actuated mirrors consist of those in which the deformation of a continuous facesheet is controlled by a force rather rather than actuators. The most

common forms of these are shown in figure 2.7.

Electrostatic membrane mirrors as shown in (a) are relatively inexpensive with a 37 actuator device from Flexible Optical B.V. (formerly OKO Technologies) retailing at euro 2000 and finding a place in many low-cost AO systems. In this mirror a thin reflecting membrane is clamped with radial tension by an insulating holder with the tension pulling the membrane flat. By applying various voltages to pixelated low resistance contacts under the membrane as illustrated in the figure the membrane is electro-statically attracted towards the contacts, deforming the surface. Due to the nature of the electrostatic force it is not possible to exert a push on the membrane, only a pull. Therefore it is necessary to operate the membrane from a bias position in which all contacts are at their mid-range voltage, causing a default shape of a parabola. By locally increasing or decreasing the voltage the surface may be moved in either direction. One benefit of this is that a global shift in actuator positions corresponds with a parabolic or focus term on the mirror rather than a piston, so a closed loop system with such a mirror is inherently unaffected by piston creep. A typical Flexible Optical DM is operated with a bias voltage of 100V producing a focus term of f=1200mm. A typical pitch for these devices is 1-3mm.

On a much larger scale the magnetic force may be used as illustrated in 2.7. (b). In this example a series of voice coils provide a magnetic force of controllable strength and polarity, repelling or attracting small magnets bonded to the rear surface of a thin reflecting membrane. This technology is used in various deformable secondary mirrors in large astronomical telescopes -for example on the VLT[**26**] and the LBT.

2.2.3.3 Transmissive devices

Transmissive SLMs are typically based around liquid crystal (LC) technologies in which, rather than physically moving a reflective surface as with DMs, the optical path length is altered by varying the refractive index of a transmissive layer of birefringent LC material. Devices may be addressed either electronically or optically, and usually consist of a regular pixelated structure leading to a device similar in operation to the piston only DM from figure 2.6(a). Whilst the fundamental technology is transmissive a device with more than a few control channels is often reflective in operation, consisting of a transmissive LC layer with a mirror behind it causing it to act as a two pass device, and allowing the many electronic control signals to be connected through the rear of the device.

2.2.4 Figure Reconstruction

Having both the means to measure the distortions present in the beam with the wavefront sensor, and a deformable mirror to correct these distortions, the final requirement is to connect the two such that the measurements from the sensor command the mirror surface, or figure.

As previously stated the wavefront sensor output **wfs** is not usually of the same form as the input to the deformable mirror, **dm** and so a transform is required.

If the assumptions from sections 2.2.3 and 2.2.3.1 that the WFS function and DM function are each linear and orthogonal are upheld then the reconstruction may take the form of a matrix operation:

$$\mathbf{dm} = \mathbf{C}.\mathbf{wfs} \tag{2.5}$$

where the DM vector **d** is derived by multiplying the wavefront vector(**wfs** with a transform matrix **C**, commonly referred to as the 'control matrix'

2.2.4.1 Control Matrix Generation

If the behaviour of the WFS and the DM are well characterised and their relative alignment can be accurately determined then it is possible to generate the control matrix theoretically. However these requirements are in practice hard to achieve so an experimental method outlined below is often used to generate the control matrix.

If we take the model AO system for figure 2.1(b) but open the control loop and replace the aberrated input beam θ_a with a flat reference beam θ_0 then the wavefront sensor will simply measure the phase of the DM such that

$$\mathbf{wfs} = WFS[\theta_{dm}] = WFS[DM[\mathbf{dm}]] \tag{2.6}$$

Given equation 2.4 this may be re-written as

$$\mathbf{wfs} = WFS[\theta_{\mathbf{dm}}] = \sum_{i=0}^{N_{dm}-1} d_i WFS[\mathbf{r_i}] \tag{2.7}$$

Now $WFS[\mathbf{r_i}]$ is the WFS response to a unitary actuation on the ith DM channel, and is a constant for any given system configuration. It can be referred to as $\mathbf{w_i}$ and may be theoretically calculated or experimentally measured. We now have the following for the response of the WFS to some DM command vector, **dm**:

$$\mathbf{w} = \sum_{i=0}^{N_{dm}-1} d_i \mathbf{w_i} \tag{2.8}$$

Finally, this can be rewritten as a matrix equation:

$$\mathbf{wfs} = \mathbf{M.dm} \tag{2.9}$$

where **M** is formally know as the *interaction matrix*. This matrix consists of N_{dm} rows, each of which consists of an N_{wfs} element vector - i.e. the ith row contains $\mathbf{w_i}$, the WFS unit response to the ith DM channel. This matrix is often referred to as the 'poke' matrix, due to the way DM channels are sequentially actuated, or poked, to produce this matrix.

The matrix may be visualised as

$$\mathbf{M} = \begin{pmatrix} w_{0,0} & w_{1,0} & \cdots & w_{N_{wfs}-1,0} \\ w_{0,1} & w_{1,1} & \cdots & \vdots \\ \vdots & \vdots & & \vdots \\ w_{0,N_{dm}-1} & \cdots & \cdots & w_{N_{wfs}-1,N_{dm}-1} \end{pmatrix}$$

where w[i,j] is the response of the ith channel of the wavefront sensor to a wavefront produced by actuating the jth channel of the DM by one unit.

In a system where the responses of the mirror are spatially localised (such as a continuous facesheet mirror) and the wavefront sensor measures spatially localised quantities (such as a Shack-Hartmann or Curvature sensor) the resultant matrix will be relatively sparse and may present a diagonal structure - this is dependant on the arbitrary numbering of control and sensor channels. An example of such a matrix is shown in the results chapter, figure 6.3.

Referencing equation 2.9, we can multiply both sides by the pseudo-inverse of the interaction matrix to obtain:

$$\mathbf{M^{-1}wfs} = \mathbf{MM^{-1}dm.} = \mathbf{dm} \tag{2.10}$$

This is the equation for figure reconstruction, and can be rewritten as follows, where $\mathbf{C} = \mathbf{M^{-1}}$. **C** is known as the control matrix.

$$\mathbf{dm} = \mathbf{C.wfs} \tag{2.11}$$

Although the control matrix may be derived by the generalised inverse it is more common to use the singular valued decomposition (SVD) algorithm as this process generates

a set of arbitrary modes for the mirror that however resemble common optical distortions at lower order which are used to perform the inverse. By altering the prominence of these modes, especially the higher order ones it is possible to reduce the system sensitivity to noise etc.

2.2.5 Wavefront Reconstruction

Using the theory outlined above, measurements from a derivative sensor such as a SH or Curvature are transformed directly into mirror drive commands, without the actual wavefront observed by the WFS ever being calculated. It may be desired to calculate the wavefront for either diagnostic purposes or use in a system monitoring GUI etc. This may be performed by running a second figure reconstruction for a virtual DM, the response functions of which are some set of orthogonal modes such as Zernike modes. The interaction and control matrices for such a virtual mirror may be calculated theoretically, and used to transform the wfs vector into modes, from which a phase map can be created.

2.2.6 Modal Control

Modal control refers to a modification of the figure reconstruction in which the WFS vector is initially transformed into some set of intermediate modes (as with the wavefront reconstruction) as which may then be filtered etc. before being transformed into deformable mirror commands. This process may be useful to suppress high order modes that are poorly sensed by the WFS due to noise levels or aliasing etc., or to separate low order modes such as tilt and defocus for offloading to a separate low order mirror.

A further use is for coupling highly disparate sensors and correctors, for example when using a Shack-Hartmann sensor with a pixelated, piston only phase addressable element such as an optically addressed LC, the standard reconstructor approach breaks down, as the native modes of the corrector - piston - are unsensed by the WFS, meaning that it is not possible to directly generate an interaction matrix. For such a system an extra layer must be inserted such that a modified DM vector DM' addresses Zernike modes (for example) which are approximated into a piston only phase map for display on the LC. It is this modified vector that is used to generate the interaction matrix.

2.3 Temporal control

To fully correct an aberration, regardless of the temporal bandwidth of the aberration, an open loop system must operate with a gain of unity which is often not feasible due to non-linear and hysteresis effects in current DMs and non-linearity in wavefront sensors when required to operate over a high dynamic range. As such closed loop systems are used in the majority of cases, a gain of less than unity should be used, such that the system converges on the correct solution over several iterations allowing the closed loop to measure and adapt to these effects, whilst also reducing the dynamic range over which the WFS will typically operate improving uniformity.

Consequent to a gain of less than unity a closed loop system will require a faster sensing and DM update compared to an open loop system for any given temporal bandwidth aberrations, with the shorter times being required compensate for iteration, thus reducing the time available to integrate light and perform computation. Despite this most AO systems in practical use are closed loop systems.

In a closed loop system the WFS measures the deviation of the current mirror figure from the desired turbulence, so the dm commands derived from equation 2.11 refer to the change in mirror figure required to move from the current figure to one compensating the turbulence. Therefore the control system must include the current mirror figure in determination of the new one. This may be achieved by use of an integrator or PID control, e.g. with an integrator the mirror figure at time $n+1$ where n is the iteration number is given by

$$dm_{n+1} = \mathbf{dm}_n + \mathbf{wfs_n}.\mathbf{C}. \tag{2.12}$$

2.4 System operation

Having examined the theory behind the essential components of an AO system the procedure for calibrating the system and closing the loop is presented below.

2.4.1 System calibration

Before closed-loop operation can commence, the reference wavefront θ_0 must be measured and the control matrix must be generated. To measure the reference wavefront the deformable mirror is placed in its default flat, midrange position and a calibration

source replaces the usual light source into the system, providing a flat aberration free wavefront. The corresponding **WFS** vector is recorded, typically averaged over several hundred measurements spanning several seconds to eliminate low-order turbulence caused by air currents around the optical bench with this average forming θ_0.

The interaction matrix is then generated by sequentially moving each mirror actuator from mid-range towards an extremity then back to mid-range with the corresponding **WFS** vectors being recorded at the extremity, again with averaging over several seconds. After subtracting the reference θ_0 from these readings the sequence of vectors forms the interaction matrix which is then conditioned and pseudo-inverted to give the control matrix.

The control matrix and reference wavefront are uploaded to the RTC, which once also informed of the desired gain is able to perform the various calculations summarised below and therefore close the loop.

2.4.2 Closed loop operation

Figure 2.8 draws together the various operations described earlier to show a complete breakdown in the stages of a closed loop AO system. These are (1) the measurement of some image encoding the properties of the wavefront, (2) the extraction of these properties from the image by signals processing, (3) the conversion of this sensor into a nulling mode by reference subtraction, (4) multiplication of this signal by the control matrix to transform the signal into mirror commands, (5) gain and integration for closed loop operation and (6) application of this signal to the DM.

2.5 Static correction

Various sources of aberrations may exist in an AO system - perhaps imperfections in the quality of the optics themselves or in their alignment, or perhaps in the default surface of the deformable mirror due to such causes as the positional tolerance of the actuators , the mirror finish, or the differing responses of the various drive electronics.

These aberrations add their own distortions to the light propagated through the system and must be carefully managed to ensure that they do not impact the required performance. Whilst the simple model systems described earlier in this chapter only contain a few optical elements, a system designed for and constrained by a practical use often con-

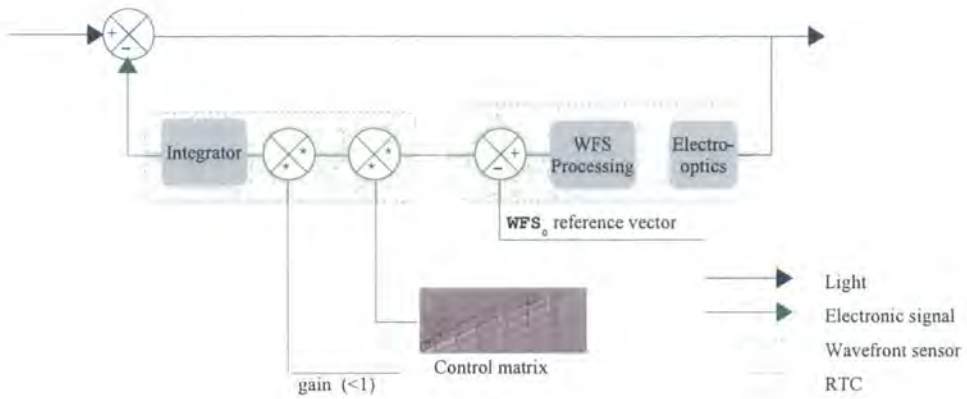


Figure 2.8: Detailed breakdown of operations in an AO system

tains many more elements, for example the retinal imaging system described by Williams et. al.[27] contains 10 elements in the imaging path, each of which as the potential to impact on the system performance. Given the presence of a DM in the AO system it is possible to correct for these aberrations with a DM figure.

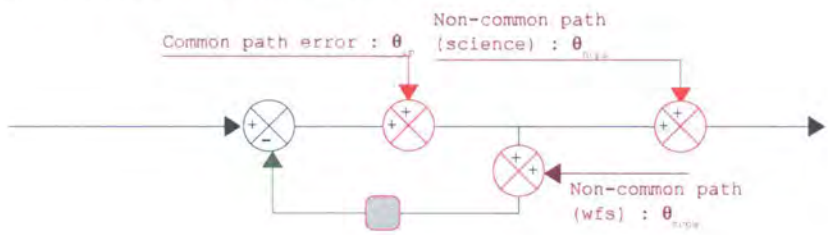
Static aberrations may be classified as one of two types according to their location in an AO system, as illustrated in figure 2.9(b) with 'common path' aberrations occurring before the separation of light for the wavefront sensor, and therefore being observed by both the wavefront sensing and science arms of the system, and 'non common path' errors which exist after this separation and affect only one path. Non-common path errors in the wavefront sensor arm are typically not an issue as these will be calibrated out in the reference wavefront sensor vector (section 2.4) and are not considered further, whereas errors in the science arm - which may be in optics external to the AO system - present more of a concern.

During the system calibration procedure outlined in section 2.4 the AO system is given a flat input wavefront reference, $\theta_0 = \mathbf{0}$. In the presence of common path aberrations the wavefront sensor measures a phase of $\theta_0 + \theta_{cp} + \theta_{ncpw}$ and once the loop is closed will try and shape any incoming wavefront to this reference, so even if an input aberration is perfectly corrected, the common and non-common errors are still present after the DM.

These static aberrations may be mitigated by running the closed loop AO system with a default mirror figure that opposes the aberrations as shown in figure 2.9(c), achieved by adding the static corrective signal to the mirror drive signal.

(a) Closed Loop AO system

(b) Closed Loop AO system with static aberrations



(c) Closed Loop AO system with static aberration and correction

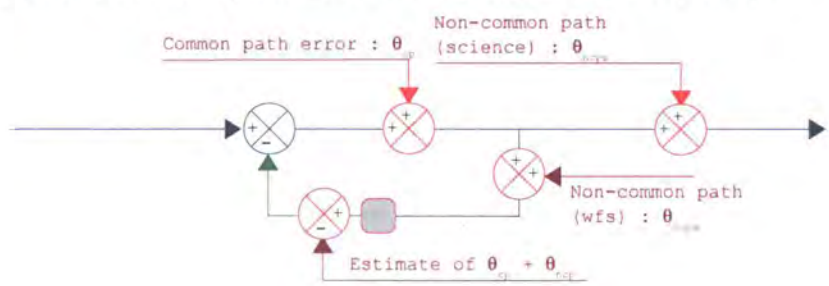


Figure 2.9: Static aberrations in an AO system

If the primary source of static aberration is the poor 'default' surface of the DM correction may be achieved by examining the mirror with a phase shifting interferometer whilst using an iterative feedback algorithm such as that developed by Sivaramakrishnan and Openheimer [28] to flatten this figure. This method is suitable for the rapid optimisation of a system of many hundreds of channels with [28] demonstrating the flattening of a 241 channel device in 12 iterations. However this method requires the use of a phase shifting interferometer which typically costs many tens of thousands of pounds.

A more accommodating approach is to examine the final output of the system and determine some metric representing the quality (for example focal spot size; power in a box etc.) and to use an iterative method to maximise this by varying the optimising figure. The Simplex algorithm has been shown to be highly efficient at this task [29] and is, for example, used in the NAOMI AO system with an emphasis on non-common path correction[30]. A plethora of other optimisation methods are available such as genetic algorithms and stochastic gradient descent. In applications where the temporal bandwidth of the turbulence is very low an optimisation method may suffice as the only required control loop, replacing the WFS and full-blown RTC.

Chapter 3

FPGA/AO Theory

3.1 Introduction

As seen in chapter 2, a large quantity of calculations must be performed as a key component of an adaptive optics system. The forerunner of the modern AO system, fast steering tip/tilt correction systems, were first operated in the 1950s with simple analogue control systems. The first higher order AO system based around a WFS, figure reconstructor and DM was the ARPA real-time atmospheric compensator (RTAC), described in Hardy [**31**]. The requirements of this system exceeded digital technology which was still in its infancy, and thus an analogue control system was used with a 2D resistor network performing the matrix vector multiplication from the reconstructor. Such an analogue network has inherent speed benefits in that it is completely parallel. However such a system lacks the simple configurability and flexibility of digital systems, which have been rapidly adopted with a progression through systems using discrete multiply-accumulate (MAC) microchips through to central processing units (CPUs) and CPUs with specialist features for digital signals processing (CPU/DSPs). Whilst bringing flexibility and configurability compared to analogue systems, the poor ability of inherently serial systems to scale is leading to an examination of the use of field programmable gate array (FPGA) devices for future ELT AO systems and for use in second generation AO systems for current telescopes in the 8m-10m range. An FPGA is a two dimensional grid of parallel, configurable digital logic that combines the parallel execution of an analogue system with the flexibility of a programmable digital system. As seen in chapter 2, a large quantity of calculations must be performed as a key component of an adaptive optics system. The forerunner of the modern AO system, fast steering tip/tilt correction systems, were first operated

in the 1950s with simple analogue control systems. The first higher order AO system based around a WFS, figure reconstructor and DM was the ARPA real-time atmospheric compensator (RTAC), described in Hardy [31]. The requirements of this system exceeded digital technology which was still in its infancy, and thus an analogue control system was used with a 2D resistor network performing the matrix vector multiplication from the reconstructor. Such an analogue network has inherent speed benefits in that it is completely parallel. However such a system lacks the simple configurability and flexibility of digital systems, which have been rapidly adopted with a progression through systems using discrete multiply-accumulate (MAC) microchips through to central processing units (CPUs) and CPUs with specialist features for digital signals processing (CPU/DSPs). Whilst bringing flexibility and configurability compared to analogue systems, the poor ability of inherently serial systems to scale is leading to an examination of the use of field programmable gate array (FPGA) devices for future ELT AO systems and for use in second generation AO systems for current telescopes in the 8m-10m range. An FPGA is a two dimensional grid of parallel, configurable digital logic that combines the parallel execution of an analogue system with the flexibility of a programmable digital system.

Work to date on low cost AO has focused on the use of commodity personal computer (PC) hardware, with the PC's CPU performing the calculations, for example in systems by the NOAO[12] and TNG[32] observatories, and on the use of reasonably inexpensive DSP hardware assisted by a common PC, for example in the LOCADO system[13].

3.2 Architecture of Modern Devices

In this section we focus on the architecture of FPGA devices from Xilinx, although broadly speaking devices from other manufacturers such as Altera are comparable. The foundation of all modern FPGA devices from Xilinx was laid with the XC4000 series FPGA devices from 1999[33]. This device consisted off a 2 dimensional array of configurable logic blocks or CLBs as illustrated in figure 3.1. Each CLB consists of two independent 4-input look up tables that may be configured to produce any arbitrary function of between 1 and 4 inputs. These LUTs are asynchronous with their output stabilising some short time after the inputs stabilise. To convert the LUT into a synchronous element the output may be registered by a flip-flop (FF).

The 2D array of these CLBs is immersed in a configurable routing fabric such that outputs of one CLB may be connected to the inputs of another to cascade these simple 4-

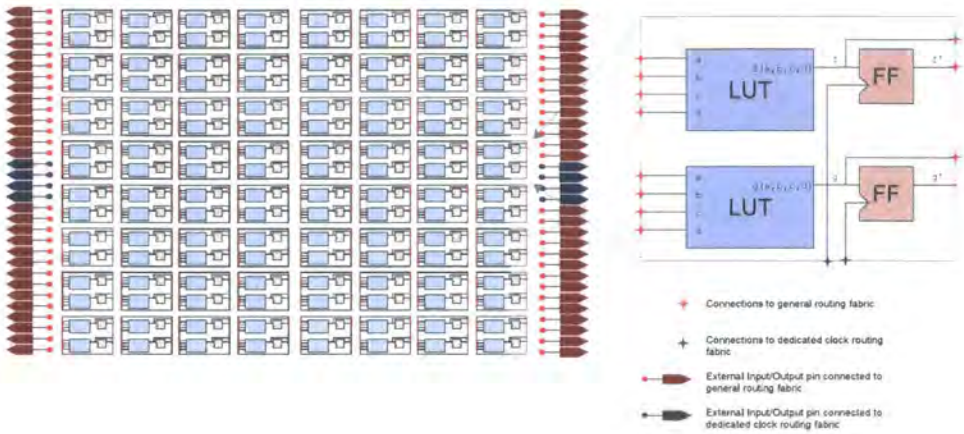


Figure 3.1: Basic structure of Xilinx devices - illustration showing an FPGA as a 2d grid of asynchronous look-up tables (LUTs) and synchronous flip-flops. By programming the contents of the LUTs and the routing between IO pins, LUTs and FFs complex, custom digital logic systems may be created.

input logic functions and registers to build more complex functions, ranging from a simple counter to full blown signals processing systems. The device also offers input/output pins (IO pins) that include programmable logic to control their signalling standards and related behaviour. Finally there are dedicated pins and routing nets for connecting clock signals, as these have a higher fan-out than general logic and benefit from special consideration. A key feature of the LUTs in the Xilinx family is that they are implemented as a 16x1 RAM that is typically programmed when the device is configured, but may be operated within the FPGA design as a 16x1 RAM, with many LUTS being chained to create wider or deeper RAMs as necessary, this feature is described as *distributed ram* or SelectRAM.

Xilinx continued to produce new FPGA families with the same basic LUT/FF structure over several generations as shown in table 3.2 - However the Virtex-5 has changed the fundamental structure of the LUTs with a move to 6-inputs over 4. Until this change each successive generation was marked by an increase in the number of CLBs in a device, increases in the maximum speed at which the logic and routing fabric may operate and the addition of new features.

The Virtex devices represented an increase in LUT/FF count over the XC4000 series. The family also brought the addition of dedicated RAM to the FPGA. Whilst many LUTs may be chained to create RAM, part of the array of CLBs may be replaced by a dedicated RAM store built not from CLBs but directly on microchip offering higher

storage densities and faster speeds. These memories introduced into the Virtex family are called BlockRAM, and have a size of 4096 bits that may be configured as 4096x1, 2048x2, 1024x4, 2048x4, 1024x8 or 512x16. A key feature of these BlockRAMs is that they are true dual port memories, offering two independent read/write ports to the same memory store - particularly useful for a system where real time algorithms and configuration interfaces must share access to a memory.

The Virtex-II family saw BlockRAMs increase in size form 512x16 to 1024x18 with the depth being doubled and two extra bits added to the words, nominally for parity storage but free for any use. Following the precedent of replacing some general logic fabric with dedicated hardware set by the Virtex devices the Virtex-II family adds dedicated single clock cycle multipliers, with one multiplier for each BlockRAM. The multipliers accept two 18 bit signed integers and output a 35 bit signed integer. The prevalence of a large number of independent RAMs and dedicated multipliers allows for highly efficient parallel implementations of algorithms, particularly vector and matrix mathematics.

The Virtex-IIPro range saw the addition of features less useful for low cost embedded control such as multiple dedicated 3.125GBits/sec serial transceivers within the chip and dedicated PowerPC 405 CPU cores to enable complete computer systems to be built within the FPGA. This trend has continued through the Virtex 4 and 5 devices with multiple gigabit ethernet transceivers, 3-10Gb/sec serial transceivers and similar features being included. The Virtex-4 family offers three variants where the ratio of LUTs/FFs to dedicated multipliers and BlockRAM is varied to offer logic orientated devices for general use (LX), an abundance of multipliers for DSP applications (SX) and extra BlockRAM (FX).

The Virtex-5 has introduced changes to the dedicated multipliers replacing them with broader featured DSP blocks and also fundamentally altering the basic CLB structure. However these changes are not reflected in the low cost Spartan family targeted in this work so are not explored further.

The Spartan family of devices are a lower cost range of FPGAs from Xilinx with a decreased range of features - particularly in terms of lack of support for complicated IO standards and faster speeds - compared to the Virtex devices from which they derive. To date their has been a strong correspondence between generations of Virtex and Spartan devices, with the Spartan 3 FPGA having the same CLB structure, and BlockRAM and Multiplier configurations as the Virtex-II family.

Family	Device	LUTs + FFs	BlockRAM (Kb)	SelectRAM (Kb)	Multipliers
XC4000	XC4085XL	6272	0	98	0
Spartan		1568	0	24.5	0
Virtex	XCV1000	24576	128	384	0
Spartan-II	XC2S300E	6144	64	96	0
Virtex-II	XC2V8000	46952	3024	1456	168
Spartan-3	XC3S5000	33280	1872	520	104
Virtex-4 LX	XC4VLX200	89088	6048	1392	96
Virtex-4 FX	XC4VFX140	64512	9936	1008	192
Virtex-4 SX	XC4VSX55	24576	7568	384	512

Table 3.1: Summary of resources for largest device in each Xilinx FPGA family, grouped by device generation

Table 3.2 sumarises the capacity of the largest device from each of these families.

3.3 Programming model and tool flow

As with CPU systems there are many levels at which an FPGA may be programmed, for example at the lowest level each individual LUT and FF and their interconnection may be configured manually to produce some functionality. This is somewhat analogous to programming a CPU in assembler, and offers the most control but with an unpractical amount for work for more than the smallest operations. Further tool support for manual control over routing in FPGAs is normally lacking.

Higher level languages exist under the name of hardware description languages (HDLs) such as VHDL and Verilog. These allow a programmer to describe desired hardware blocks (such as MACs, state machines etc.) and their interconnection in a textual language. At a higher level still packages such as *Simulink HDL Coder* automate the production of VHDL from a more abstract source.

As with a program for a CPU, an FPGA design begins with source code files as shown in figure 3.2. A variety of formats for source code are supported including HDL files, netlists - a simple description of logic operations or fundamental FPGA building blocks and their interconnection and schematic capture. A design typically consists of many source files tied together in a hierarchy, allowing low level descriptions such as the workings of a

numerical divider to be separated from higher level concepts such as the overall data flow through a design.

This design may then be tested under simulation with a simulator or the design may undergo the process of synthesis in which the heirachical design is flattened out and all higher level constructs in the source file are broken down into a combination of FPGA specific features such as BlockRAM and multipliers and basic logic operations, registers etc.- the result of this process is known as a register transfer level (RTL) netlist. Whilst HDLs allow for the inclusion of time delays in the source code, and these will be honoured by the simulation tools, they are ignored by synthesis tools as signal prorogation delay is a consequence of placing the design into hardware, not a desired effect. In practice this means that to have a set of source code that is both synthesisable and simulatable then synchronous logic must be used, where the placing of FFs between logic functions imparts a sense of time to the design without explicitly declaring delays in the source. This disparity between simulation and synthesis must be carefully examined before the use of simulation on areas of logic not explicitly synchronous.

After synthesis the logic operations in the RTL netlist are replaced by low level FPGA hardware such as LUTs and FFs by the translation and mapping tools. The place and route (PAR) tool then takes these components and their connections in conjunction with a constraints file (specifying which physical device pins external connections are made to) and determines where to place each component within the gate array and how to connect them through the routing fabric. Whilst the synthesis operation is loosely analogous to the conversion of a high level language such as C to a low level language such as assembler in CPU programming there is no direct analogue for PAR. The output of the PAR tools is then converted into a *bitstream*, a configuration file for the FPGA. If the speed of all clocks used in the design is specified in the constraints file a timing analysis may be conducted on the post PAR design to ensure that every delay between any two connected synchronous FFs in the design - caused by the propagation times of routing and settling time of LUTs etc - is less than the relevant clock period. If this is not the case then the device will not operate correctly and the design must be revised or the clock slowed.

Each stage prior to bistream generation may produce a model of its output suitable for simulation that incorporates all the changes made by the tools to that stage, for example conversion of logic to LUTs by the 'translate/map' process where high level operations are broken down into a sequence of LUT based functions or the addition of routing delays

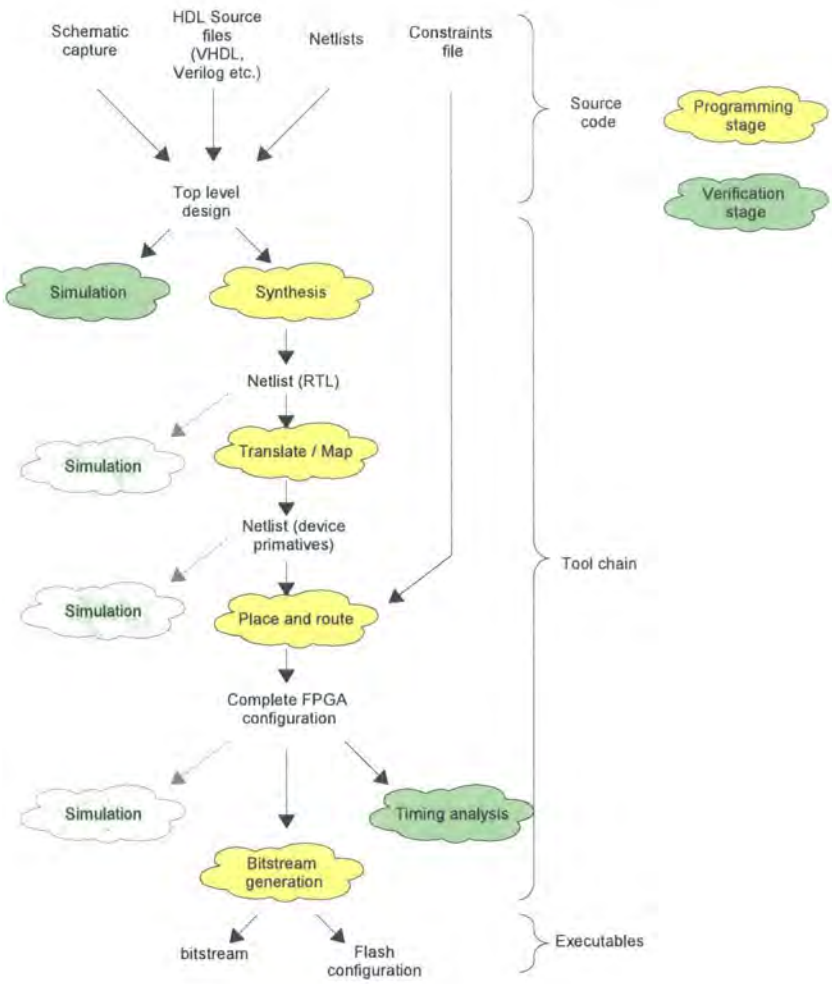


Figure 3.2: Tool flow for programming Xilinx FPGA devices

by placing the LUTs into the layout of a specific FPGA and connecting them through the routing fabric, although when designing with synchronous logic it is typically only necessary to varify correct operation of the design with simulation of the source files, and to varify that timing analysis is passed.

3.4 Existing Applications to AO

In this section we briefly examine other applications of FPGA devices to AO.

The Starfire Optical Range have demonstrated the use of FPGA devices to significantly reduce the latency of a closed loop system[34], in their case by replacing the wavefront gradient processing software with dedicated FPGA modules. However their system was

not concerned with cost reduction, just latency reduction, and used high end Altera FPGA devices to accelerate the parts of the closed loop hosted on custom DSP hardware.

In a similar style the ESO SPARTA AO platform[35] employs high end Xilinx devices for WFS pre- processing and gradient extraction[36], again in the form of high end FPGA modules, with retail prices in the 10-20k range, being added to high end DSP equipment. As with the Starfire work the use of FPGAs is driven by requirements for reduced latency rather than cost reduction.

The ability of FPGA devices to form customised processing engines has also lead to an exploration of their use in the simulation of AO systems[37] for extremely large telescopes (ELTs) where dedicated FPGA processing pipelines may improve the performance of a software simulation over 100x. Particularly helpful here is the ability of FPGA devices to perform high rate FFTs, a key component of optical simulations.

A closed loop AO system based on an FPGA was demonstrated by Rodriguez-Ramos et. al.[38] from the IAC using a 1KHz 8x8 subaperture SH WFS and a 37 channel OKO DM. This system used a high end Xilinx Virtex-4 LX25 FPGA hosted on a Xilinx ML-401 development board which as of January 2007 retails for US$500[39] . The authors state 70% of the Virtex-4 was used by the design. By comparison the Spartan-3 200 as used to construct a closed loop AO system of similar AO capabilities in successive chapters has approximately 17% of the resources of the LX25 (based on LUT capacity).

The IAC system lacks complete real time communications capability with a host PC, the only link between the FPGA and a PC being an RS232 link, which is too slow for frame acquisition and not even well suited to centroid acquisition etc. Further, on-board switches are being used to control some features such as PI control, rather than software commands from a host computer.

WFS image display is performed either by parallel acquisition of the image data by a PC with a high speed frame grabber card or by a VGA output module on the FPGA board. Further the use of such a low speed host link prevents data acquisition (centroids, mirror values etc.) from the system of beyond a few 10s of frames a second. Whilst demonstrating closed loop AO control on an FPGA device this system leaves much room for improvement.

Chapter 4

FPGA/AO Framework

4.1 Introduction

This chapter presents and examines the design of a framework for performing closed loop adaptive optics control on FPGA devices. First a broad overview of a suitable framework is given, followed by a detailed examination of the real time processing components for computing the tasks explored in the theory chapter.

4.1.1 Overview

When starting with a blank FPGA there is a much wider range of possibilities for implementing a control system than when working with a CPU - for example a completely fixed function controller could be implemented that is optimised for one particular imaging array, one geometry of Shack-Hartmann spots and one particular deformable mirror. If correctly implemented such an approach should offer the fastest, or smallest FPGA configuration for controlling that specific AO system, although much effort would be required to adapt the code to a different system configuration.

The opposite of this approach would be to create a generic CPU within the FPGA which could then be reprogrammed in software with relative simplicity to adapt to any number of AO system configurations. However such a system will represent a very inefficient use of resources, as a CPU built on an FPGA will under-perform compared to a commodity CPU implemented directly as an ASIC, so this approach represents an over-complex and inefficient approach.

The approach chosen for this work is to create a framework specifically for the high speed processing of streaming data, with a series of AO modules. Each module is for a

fixed function, such as wavefront sensing or figure reconstruction and so may be implemented efficiently. Nevertheless some degree of flexibility is designed into the modules, for example being able to change the geometry of a Shack-Hartmann sensor without needing to recompile the FPGA code. These modules are interconnected to form the *AO Pipeline*, and are configured and controlled by a wider system, which is refered to as the *AO Framework*.

4.1.2 Target system

The framework is developed for a baseline system consisting of a Shack-Hartmann wavefront sensor of 128^2 pixels with up-to 16 by 16 subapertures, driving a DM of up to 64 channels. The pipeline is intended to fit within a Xilinx Spartan-3 200 FPGA or XC3S200[**40**], which provides 12 BlockRAMs equivalent to 24KBytes of memory, 12 dedicated multipliers and 1920 logic slices for implementing the pipeline logic or for use as SelectRAM - if all slices were used this would form an additional 3.7KBytes. The pipeline is intended to run at 50MHz, and to accept pixels at up-to this rate.

4.2 Framework Overview

As well as the core processing logic for performing AO control within the FPGA there are many other tasks, collectively refereed to as system management, to be performed to enable the system to work. These include providing a means to modify parameters such as the gain of the system or the control matrix and providing a way to offload data from the system in soft real time to a host computer for display and analysis purposes - for example to enable system alignment or calibration.

When working with a CPU/DSP system the basis to do this is already in place through the CPU's peripheral bus and off the shelf peripherals and support libraries, but with an FPGA system there is a blank slate. The simplest option would be to bring all the relevant signals to external pins on the FPGA and to use a digital IO adapter to connect a host computer - for example sending the data from the WFS imaging device out to pins connected to a framegrabber. Such a system is however unwieldy and impractical so normally an interface to perform all the functions listed above is included in the FPGA configuration. This means may be provided by the 'System on a Chip' architecture detailed below.

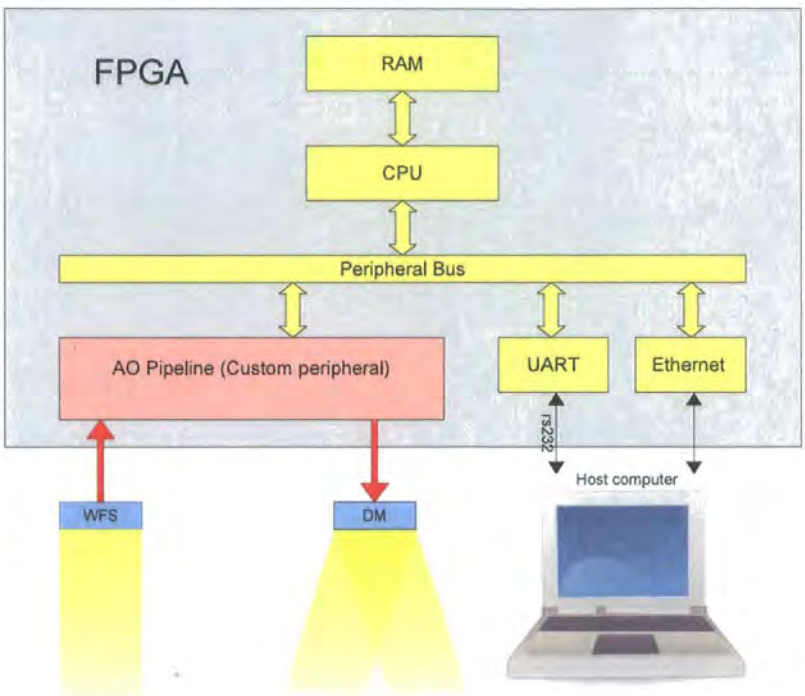


Figure 4.1: Example System on a Chip system for AO - the AO pipeline shown in red performs all the closed loop control algorithms, whilst the CPU arranges data transfer between the pipeline and a host computer over some interface such as ethernet.

4.3 System on a Chip Design

The term 'System on a Chip' refers to a microchip - either a custom ASIC or an FPGA - that contains all the components of a traditional computer system - i.e. the CPU, the program and data memory, peripherals such as RS232 and Ethernet interfaces etc. When a SOC is built on an FPGA it suffers from relatively slow performance (100-200 MHz for an FPGA SOC vs. 2+GHz for a conventional PC) and a small memory if confined to operating only from internal RAM etc. This makes such a SOC a highly inefficient approach for the real time processing component of AO control, although the current performance levels are more than suitable for handling the system management tasks listed above, and use of a CPU makes the system significantly more flexible than dedicated logic for system management, whilst also offering a simpler development environment based around software flows rather than an HDL.

Figure 4.1 illustrates the concept behind this approach. In the figure the 'AO pipeline' is some custom designed logic for performing all the calculations for closed loop AO control.

This module exposes real time interfaces to the WFS and DM, and a non real-time interface to allow for configuration and telemetry access. The remainder of the FPGA device is used to implement a complete system on a chip including a CPU, program memory and interfaces to a host computer such as RS-232 or USB. As with a conventional computer the CPU connects to the external interfaces via some peripheral bus, and the AO pipeline is designed to attach to this also, with the pipeline being designed such that communications with the CPU over the peripheral bus do not detract from or otherwise affect the real time performance - a task much simpler in hardware, due to the inherent parallelism, than in software. In this way both the creation and execution of the real time and system management components are conceptually and physically separated, allowing different approaches for each (typically an HDL for the mathematically intensive real time tasks, and software tools such as Assembler or the C language for the system management tasks) and separating their performance.

4.3.1 Design Choices

In this section the choice of components for the system is examined. Initially the defining choice of which CPU to use is examined, followed by a detailed look at the final SOC architecture including external communications and the link with the AO pipeline.

4.3.1.1 Choice of CPU

There exist numerous choices for a CPU and associated peripheral busses etc. for use within an FPGA. Concentrating on the Xilinx devices the first choice is whether to use a hard or soft CPU core - a hard core is a CPU that physically exists on the silicon within the FPGA and is connected up to the general logic fabric, allowing it to communicate with RAM and peripherals through the FPGA. A hard core will offer higher performance in terms of speed for a given area of silicon used over a soft core due to the direct implementation. Xilinx offer some devices in the Virtex-IIPro, Virtex-4 and Virtex-5 families with embedded 32-bit PowerPC 405 CPUs [41], and offers useful development tools, namely a GCC compiler tailored for the CPU and their Embedded Development Kit (EDK) [42]) that automates the creation of an FPGA design based around a PowerPC SOC. While highly capable, these CPUs are only available in the high end Virtex families and not the low cost Spartan devices.

Xilinx also offer a soft core CPU called the Xilinx MicroBlaze[43]. As with the hard

PowerPC cores, the MicroBlaze is a 32-bit RISC CPU and is supported by EDK and GCC. Being a soft core this CPU is supported by the Spartan-3 family of devices, although a MicroBlaze would use a significant portion (more than half) of the target XC3S200 device.

Given that all the numerically intensive processing is performed inside the dedicated AO pipeline the required CPU performance is quite low, so significantly smaller CPUs were examined with the Xilinx Picoblaze soft-core being chosen. The Picoblaze is available for both the Spartan and Virtex families, and is described fully in the relevant Xilinx user guide [44]. It is a simple 8 bit CPU with a 1 Kword instruction ROM (implemented using one dual port BlockRAM within the FPGA device), 16 8-bit registers and a 64 byte RAM, with comprehensive logical and arithmetic operations and 256 8-bit input/output ports multiplexed over an IO bus. The Picoblaze core uses less than 2% of the target XC3S200 device's logic fabric and one of the devices twelve BlockRAMs. The Picoblaze may be programmed with a simple assembly language using the KCPSM3.EXE assembler provided by Xilinx. Both the CPU core itself and the assembler are provided charge and royalty free by Xilinx. However the device is not supported by the EDK tool, so the SOC framework, consisting of VHDL source code combining the CPU architecture, the peripheral bus and the peripherals had to be manually written and managed for the AO framework. As well as the Xilinx provided VHDL code that describes the Picoblaze the device itself had to be programmed and this is done using an assembly language and compiler provided by Xilinx. Given the choice of VHDL or Verilog for the FPGA and SOC design VHDL was chosen due to a preference for stronger error checking and clarity of code.

4.3.1.2 External interfacing

To allow interaction with an external computer the system must include some form of external interfacing. A simple byte-level communications protocol was developed that allows the Picoblaze to communicate with software running on a host and to transfer data between the various SOC peripherals such as the AO pipeline, the I^2C and SPI interfaces etc. and the host. This protocol is intended to remain unchanged and run over whatever physical medium is used for the link such as USB or Ethernet, giving flexibility in terms of the link layer.

Various link layers are possible with table 4.3.1.2 summarising some commonly available formats. RS232 and the EPP are both legacy ports found on IBM compatible PCs

that have become increasingly scarce since the advent of Firewire and the Universal Serial Bus (USB). Both are extremely simple to use being very low level interfaces with API calls simply for reading and writing bytes on the host, and simple electronic interfaces for the FPGA to talk to. This simplicity comes at the expense of very low data rates, especially for RS232. However both have been implemented for the AO pipeline, with the RS232 interface providing a simple means of communication with the Picoblaze that proved useful in developing more complicated layers such as USB. To physically connect to RS232 the FPGA requires a small microchip to convert the 12V signals used by RS232 to the 3.3v used by the FPGA. The EPP operates at 5V and may be connected with just series resistors to limit current into the FPGA which is configured to use in-built Zener diodes to clamp the signal down to 3.3v. Both interfaces require minimal logic inside the FPGA to connect to the FIFO bridge.

Firewire and the Universal Serial Bus are the two most prevalent external expansion busses provided by modern PC hardware, and are both differentially signalled serial busses that include an integrated power feed for the peripheral. Firewire is a more complex bus than USB, with the ability to have multiple masters on a bus, each of which can communicate with each other and with multiple peripherals, whereas USB is strictly divided into one master - the host PC - and multiple slaves - the peripherals, with all communication (save interrupts) being strictly initiated by the host.

The serial data-rate of 480MBPS of USB2.0 is too fast for direct connection to the FPGA (via level shifting electronics) for serial/parallel conversion so an external microchip is required for at least this. In practice microchips are available that encapsulate some or all of the protocol stack required for USB devices reducing the burden on the developer - unlike RS232 there is a minimum level of intelligence required by a peripheral to connect to a USB bus. For example a USB peripheral must decode, acknowledge and respond to various commands such as requests for unique peripheral ID codes and inquiries as to which version of the USB specification the device supports. The *EZ-USB* range of devices from Cypress Semiconductor are such devices, including a USB2.0 interface with a serial/parallel converter, high speed programmable FIFO buffers to couple data from this interface into external devices such as an FPGA and an 8051 micro-controller which executes firmware to configure these other components and execute user supplied code, for example responding to various status requests over the USB bus and to marshal data transfers between the host PC and the external FIFOs. The 8051 may be programmed

Interface	Data rate (MBytes/sec)	Power?	Comments	Implemented?
RS-232	0.11	No	legacy port	Yes
Enhanced Parallel Port (EPP)	1	5V, 100ma	legacy port	Yes
USB 1.1	2	5V, 0.5A		Yes
IEEE-1394 (Firewire)	40	12V, 1A		No
USB 2	40	5V, 0.5A		Yes

Table 4.1: Summary of interface options

with a mixture of C code and assembler, with Cypress providing a framework to assist in the development of custom firmware.

4.3.2 AO Framework and SOC Architecture

A detailed view of the design of the AO SOC is shown in figure 4.2. This figure may be viewed as an expansion of figure 4.1 showing the actual components used and including some detail of the AO pipeline. In this diagram the imaging chip from the wavefront sensor feeds data into the AO pipeline which contains all the stages identified in section 2.4 to convert this image into DM commands. The pipeline itself consists of a series of discrete functional modules which are examined further in section 4.4. As well as having streaming data inputs and outputs each module also exposes a system management interface (SMI), in the form of a memory interface. The memory interfaces from each module are amalgamated into a larger memory interface within the pipeline by the 'telemetry access interface'.

The SOC consists of the CPU itself, which directly attaches to the program ROM and controls the peripheral bus, through which the CPU communicates with all other system components, which are explained below.

I2C and SPI Bridges

The *Inter Integrated Circuit* bus, or I^2C, is a simple industry standard 2 wire serial, bidirectional bus often used to communicate between microchips for the purpose of configuration and control. The SOC includes an I^2C bridge to enable the CPU to communicate with I2C devices. Figure 4.2 shows the I^2C bridge being used to communicate with the CMOS imager - this is how the imaging chip in the WFS in the laboratory demonstrator, examined in the following chapters, is configured and controlled.

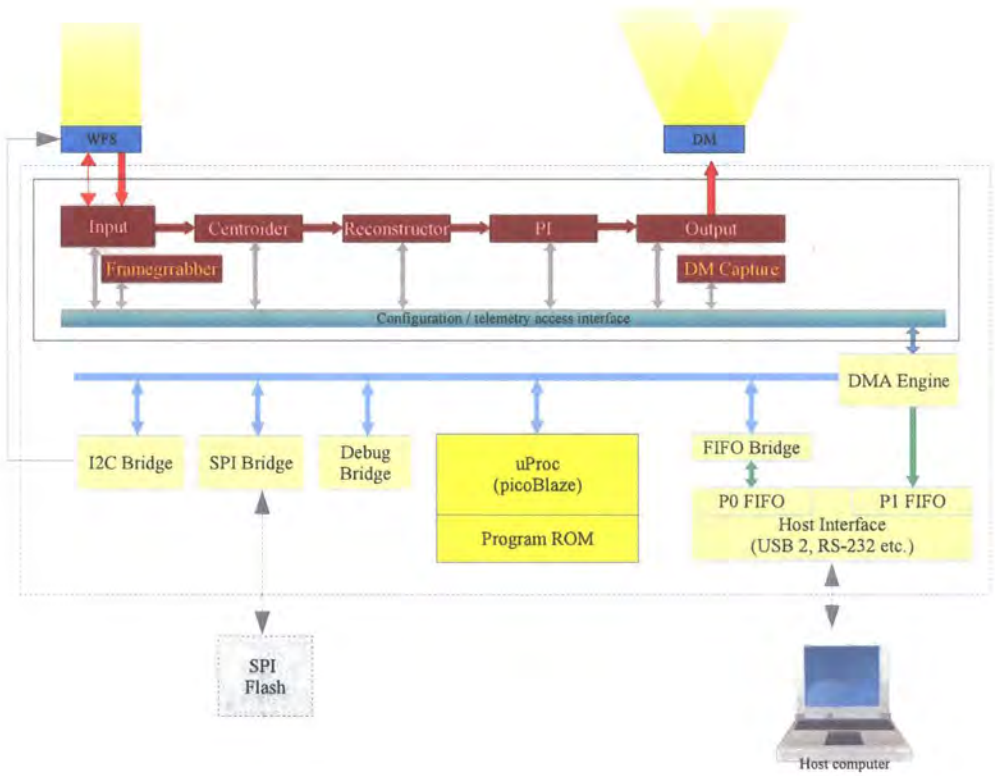


Figure 4.2: Detailed view of the AO SOC - block diagram showing the structure of the AO pipeline (a series of modules such as the centroider) and the other on-chip peripherals (pale yellow) that together with the Picoblaze CPU from the custom SOC

Likewise the *Serial Peripheral Interface Bus* or SPI bus is a simple serial, bidirectional bus that is sometimes used for controlling imaging chips, and is also often used for communicating with serial Flash (non-volatile) memory devices. Here it is shown connected to an external Flash memory chip, from which the CPU may read default start-up data such as interaction matrices etc.

Debug bridge

The debug bridge provides an interface to 8 discreet LEDs and a 4 character 7 segment LED display to enable the CPU to display system status visually, a useful tool in debugging the system and giving feedback for lab work.

FIFO Bridge

First In First Out or FIFIO buffers are a common hardware unit used to help interface different subsystems by providing temporary data storage to smooth and regulate the flow of data between units, and may be implemented asynchronously to guarantee the correct transfer of data between different clock domains, an otherwise fraught task. By connecting the CPU and pipeline to these FIFO buffers and connecting one of a variety of external interfaces to the other side of the FIFOs an external communications link is created.

DMA engine

The DMA engine connects the memory space of the AO pipeline to the Picoblaze peripheral bus, allowing the Picoblaze to read and write data to/from the pipeline. By reading data from the pipeline and then writing it to the 'P0 FIFO' in the external interface data may be transferred from the pipeline to a host computer, with the reverse transaction occurring for a host to pipeline write. Individual and burst transfers are supported, with the DMA engine auto-incrementing pipeline memory addresses to speed up transfers involving the Picoblaze.

Due to the sequential nature of the CPU, any transfer of data from the pipeline to the FIFO buffers/host PC that involves the CPU is limited in bandwidth as the CPU must perform various operations such as retrieving data from the pipeline, checking to see if the FIFO has space for more data, writing the data to the FIFO, checking to see if there are any more transfers to be made etc. In practice a minimum of 6 Picoblaze operations are required per byte transferred, or 12 clock cycles, limiting the bandwidth to 4.6 MBytes/sec. This is only an issue when the bandwidth to the host is faster than this, as with USB2.0. Further this, is a more serious problem for regular transfers of captured data from the pipeline to the host, than for the occasional transfer of configuration data

form the host to the pipeline.

This problem is overcome by a second mode in the DMA engine that can facilitate bulk transfers of data from the pipeline to a second FIFO buffer in the external interface, under control of the DMA engine and not the Picoblaze. The DMA engine created is unidirectional, only supporting accelerated transfers from the pipeline to the host, and can transfer data every second clock cycle, or 25MB/sec at the target 50MHz frequency. The choice of every second cycle over every cycle is not necessary, but somewhat simplifies the coding involved. To perform such a DMA transfer the engine is configured for the transfer by the Picoblaze over the peripheral bus, which then triggers the engine again over the bus. The engine then proceeds to bulk transfer data to the P1 FIFO when the FIFO is able to accept data - i.e. as the host reads the data out. The Picoblaze waits for the DMA engine to signal the transfer complete before resuming control and either configuring the engine for another transfer or performing some other task. In this way the system has the maximum flexibility of having software configurable transfers with full hardware speeds.

4.4 The Adaptive Optics Pipeline

The AO Pipeline is responsible for all operations involved in the AO control loop, and forms the only part of the entire FPGA design that has hard real time requirements. The pipeline reads data from the WFS imaging chip and performs all the calculations shown in figure 2.8 required to convert this into mirror drive commands, which the pipeline outputs. Figure 4.3 shows all these modules created and joined to form a pipeline for closed loop AO processing, with the operation of these modules being detailed below.

4.4.1 Image acquisition

The image acquisition module transfers streaming image data from an external source and clock domain into the pipeline and associated clock domain. The module generates basic information on each frame including a $10\mu s$ accuracy timestamp , a 24-bit rolling frame count or RFC (approximately 16 million frames or 9 hours at 500fps) and a total intensity for the image, all of which are made available to the host via the SMI. The combination of timestamp and RFC allows a host computer to accurately determine the framerate of images arriving into the system.

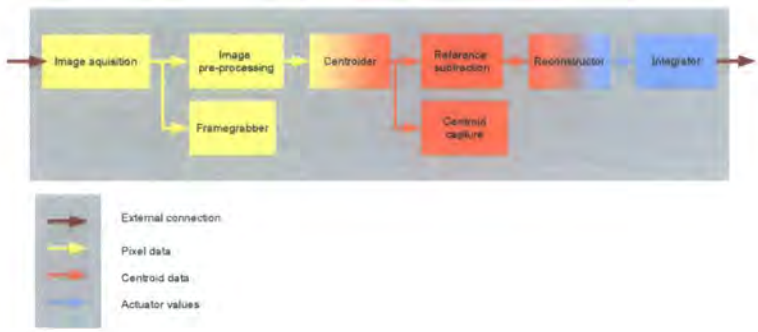


Figure 4.3: AO Pipeline configuration - a detailed breakdown of the processing modules forming the closed lop AO system with the colour of a module indicating whether it operates on pixel, centroid or mirror command data.

Mode	Min	Max	Comments
Host	0	50	Limited by link to host
Internal	0	5,000	External trigger can act as enable
External	0	5,000	Adjustable phase delay and edge sensitivity
Sensor	0	Sensor max	Triggering controlled by external sensor

Table 4.2: Frame capture triggering modes

4.4.2 Frame grabber

The acquisition module provides versatile triggering functionality, in which the module either triggers off timing information included in the external datasource, or in which it can provide the triggering to control the external source. Several triggering modes are available and are listed in table 4.4.1. In *host mode* a single frame acquisition is triggered by a command received from a host computer, whilst for *internal* and *external* triggering modes the trigger signal is generated repeatedly by the module for continuous acquisition. In internal mode a framerate is specified by the host computer, and in external mode by a triggering input.

Although the system is intended to operate primarily as a control loop and not a data acquisition system it is still necessary for the system to record the raw images as seen by the WFS imager and offload these to a host computer, both to enable optical alignment of

the system and to optimise the imager parameters (exposure, gain etc.) for best operation. For display and alignment purposes a frame capture rate of 20Hz is sufficient.

4.4.2.1 Framegrabber

To continuously capture all frames from the imager the external link must be fast enough, and even then if data arrives in bursts that exceed the practical (as opposed to theoretical) maximum external link rate dataflow must be regulated through a suitably large buffer. In practice the bandwidth of any link such as USB to a host computer isn't guaranteed when used with a non real-time operating system such as Windows XP so a buffer is necessary. Typically double buffering is used where one image is acquired into a buffer, and when this is complete this original image is re-transmitted to the host whilst a new image is acquired into the buffer. This system of simultaneous capture and re-transmission allows peak datarates to be averaged out but doubles memory requirements.

The requirements for 8 bit data at 128x128 pixels at 1Khz are 15.6 MB/sec data rate and 16 KB/ frame. To double buffer would therefore require 32KB. however the target device, the XC3S200 has only 12x2048 byte = 24KB memory available (as BlockRAM, SelectRAM made from the logic fabric is also available, but is smaller and it's use would detract processing power) - therefore to acquire a single frame would require 66% of available memory and to do so with double buffering would exceed the target device capacity. Either case is unacceptable as the majority of the memory is required by modules dedicated to AO processing, primarily for storing the control matrix. Therefore even if the bandwidth of the communications link is fast enough it is not possible to capture full images at any rate using only internal memory. If such acquisition is required it is therefore necessary to build a frame buffer in a RAM chip external to the FPGA, for example a modest 2 MB of SRAM could buffer 128 frames of 128x128 pixels, more than is required.

4.4.2.2 Successive Partial Frame Acquisition

For the purposes of the AO control system it was decided to forgo such acquisition to keep the entire system within a single FPGA device - this problem of data volume exceeding on-chip RAM capacity only occurs with images due to their relatively large size compared to the processed centroid and mirror command data.

A viable solution that was implemented is the partial capture of successive frames, in which the image is split into multiple regions and successive regions of successive frames are acquired into on-chip memory and retransmitted to the host computer, where they are

Figure 4.4: Images captured with the successive partial system - (a) shows a normal image of the author taken under natural illumination, whilst (b) shows the striping effects caused by aliasing with the 100Hz mains illumination. Finally (c) shows a tearing effect caused by the motion of a waving hand.

reassembled. The final frame as received by the computer will therefore be a composite of multiple frames, and may show odd effects if there is rapid motion in the field, or stroboscopic effects caused by aliasing with a variable light source, such as the 100Hz flicker of mains lighting. These effects are demonstrated in figure 4.4. However such images are adequate for system alignment etc.

Such a framegrabber was implemented as a pipeline module using one BlockRAM. The module consists of a state machine that controls the sequencing of acquisition and interfaces over the pipeline bus to provide handshaking to ensure one that one partial frame is fully read out over the bus before acquisition of the next commences. This state machine is configured and controlled by a small routine running on the embedded Picoblaze CPU such that the host computer need only issue the relevant command to the CPU to trigger frame acquisition and receive the frame. Whilst not ideal, this system allows frame acquisition to occur with a minimum of hardware overhead and is good enough for an AO system.

4.4.3 Image pre-processing

Whilst image pre-processing may be a detailed module involving per pixel gain corrections and background subtractions all that was implemented for this work is a simple uniform background subtraction routine which accepts a constant background value over the SMI and subtracts this from all pixels, clipping the output at 0 rather than going negative.

NSUBX	No. of subapertures in a row	8 to 16
NSUBX	No. of rows of subapertures	8 to 16
NPIXX	No. of pixels across a subaperture	2 to 8
NPIXY	No. of pixels across a subaperture	2 to 8

Table 4.3: Parameter range for the considered centroider designs

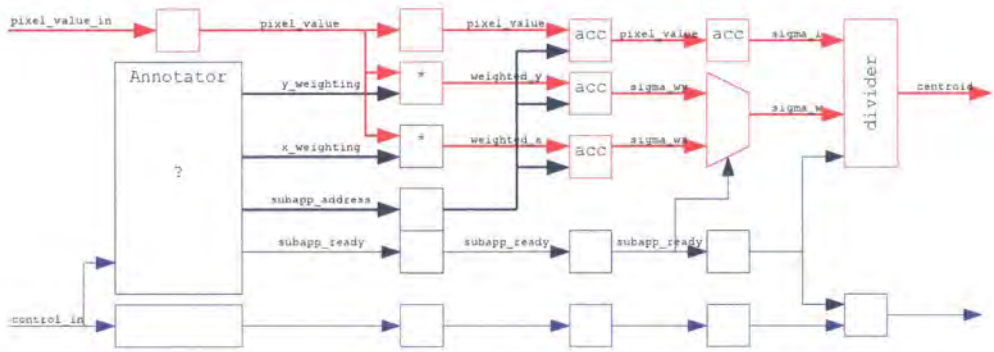


Figure 4.5: Generic hardware based centroider - red lines show data flow, with light blue showing the main control signals associated with the data such as 'start of frame' and dark blue showing control signals generated by and for the centroider. Each box represents a synchronous process registering it's input every clock cycle - all boxes run from the same clock (which is not explicitly shown.) An empty box represents a register, a box with a symbol in represents that operation followed by a register.

4.4.4 Centroider Design and implementation

Here a generic approach to building a flexible hardware centroid processor is presented, followed by a more restrictive, but highly compact implementation. Several abbreviations are used in the descriptions below, and are defined in table 4.4.4.

4.4.4.1 Generic FPGA Centroider

Firstly a generic unit for processing centroids from streaming image data using a center of mass method is presented. An overview of such a design is shown in figure 4.5. This design makes no assumptions about the geometry of the subapertures in the incoming image, other than that no sub-apertures overlap and that the pixel order is fixed between frames, both of which are highly unlikely ever to be untrue of a real system.

Initially each pixel (`pixelvaluein`, shown in red in the figure) is annotated with several numbers describing which subaperture it lies in, it's position within the subaperture and whether it is the last pixel to arrive for that subaperture (`x`, `yweighting`, `subappaddress`, `subappready` in blue in the figure.)

The second stage multiplies the pixel value by the weightings values whilst pipelining signals not yet required to maintain coherency. The third stage contains several addressable accumulators that maintain the total weight x, weighted y and intensities for each of the subapertures. These totals are built up pixel by pixel as the data streams in. Therefore each accumulator needs a memory of $NSUBX.NSUBY$ words.

The final stage is a divider to calculate the centroids from the total intensities and weighted intensities, with the preceding fourth stage sequencing the values for the y and then the x centroid through the same divider. The `subappready` signal generated by the annotator is used to indicate when the totals are valid and ready for processing, and also selects between the x and y values by controlling the stage 4 multiplexer. The entire design is able to operate on one new pixel on every clock cycle - although the x and y data are interleaved when input to the divider, requiring two cycles, this does not present a problem as a subaperture must contain at least two pixels, providing the required two cycles. In practice a serial divider may be used that minimises FPGA resource requirements by using more cycles to perform the divisions - in such a case care must be paid to the rates of data arrival at the divider, and if this is too fast a FIFO buffer is inserted into the divider to smooth datarates out over the frame. Each stage has an unspecified latency in this centroider - typically it will be 1 for any stage except for the divider, which has a latency of $n + 1$ where n is the number of bits in the output signal.

This design exploits minimal parallelism by processing x and y centroids in parallel, and significantly more parallelism through pipelining, with annotation, multiplication, accumulation (and associated memory fetch/stores) and division occurring simultaneously for different pixels through the pipeline. Two differing designs for the annotator designs are presented below, a generic version as used in larger astronomical systems and a much smaller implementation enabled by certain assumptions and implemented in this work.

4.4.4.2 Arbitrary Geometry Centroider

This section presents a design for an annotator intended to be used with the generic FPGA centroider described above that is simple to implement and allows for a highly

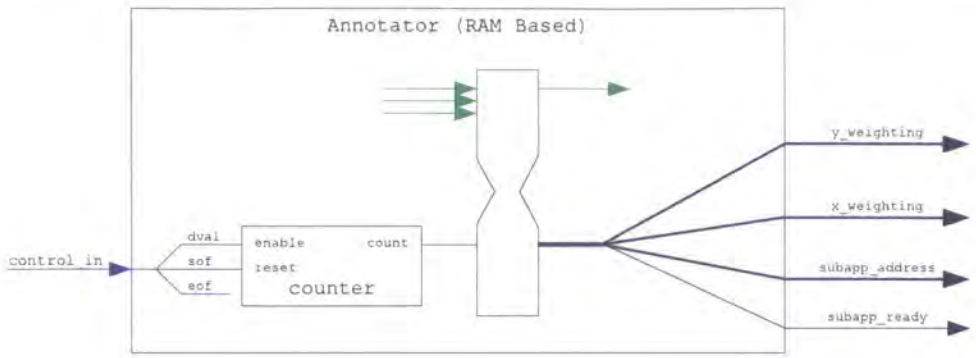


Figure 4.6: RAM annotation for an arbitrary geometry - as with the previous diagram blue lines represent data-flow. Here the data valid (dval) and start and end of frame (sof, eof) signals are used to count the pixels as they stream in. The count is then used to index a dual port BlockRAM, from which all the required coefficients such as the x weighting are retrieved. The green lines on the second RAM port represent a configuration interface allowing the contents of the RAM to be modified.

flexible range of subaperture geometries. However this comes at the cost of large RAM requirements as demonstrated below.

The simple approach taken by this aribtrary geometry centroider (AGC) is shown in figure 4.6. Pixels are enumerated by a simple counter which addresses a BlockRAM acting as a lookup table for all the required data for each pixel in the image. For the maximum specifications listed in table 4.4.4 the x and y weightings would each require 3 bits for a range of 0 to 7, the maximum number of subapertures is NSUBX*NSUBY or 256 for the sample specifications, requiring 8 bits for the subapp address. Finally the subapp ready signal requires 1 bit. Therefore the lookup memory needs a total of 15 bits for each pixel - for implementation this is effectively 16 bits or 2 bytes as RAM devices will usually order their storage into groups of 8 or 16 bits, meaning 16 would be used with one spare. For a 128x128 pixel image sensor there are a total of 16,384 pixels, requiring 32,768 bytes of memory. This exceeds the BlockRAM available in the target Spartan III device, which has 12 BlockRAMs each of 2048 bytes. For larger imagers the RAM requirements will scale worse than a linear factor as the number of RAM words increases linearly with image size, but the number of bits in each word also increases as the logarithm of the total number of

subapertures (due to the increasing size of subappaddress). Such a design is however very simple to implement and may run at high speeds. It is well suited to a system with a large external RAM device connected, for example the ESO SPARTA[**36**] project, and allows great flexibility for various geometries by changing the RAM contents, as illustrated in figure figure 4.7 and listed below.

1. Square geometry with guard pixels
2. Square geometry, close packed.
3. Square geometry with a circular pupil and central hole as typical of a telescope fed system
4. Two interleaved square arrays as used by SLODAR turbulence profiling systems[**45**, **46**]
5. Hexagonal geometry - similar to a square with alternate rows offset in the X dimension
6. Hexagonal geometry with the offset in Y dimension
7. Circular geometry - occasionally used when driving a bimorph mirror from an SH as for example in the Vulcan laser amplifier system [**47**]
8. Pyramid sensor - in this case each 'supaberture' is a quad cell with it's four pixels spread between four pupils (see section 2.2.2.4) - in this diagram each coloured box represents an individual pixel, with each subaperture split between of the four pupils present as indicated by colour.
9. This illustrates a square geometry with pupil etc., in which the positions of the subapertures have been slightly perturbed, perhaps to accommodate strong static aberrations in the system, or perhaps for use with a dithered microlens array used to reduce alisaing effects[**48**].
10. Non-square subapertures - perhaps for use where the WFS is at a strong angle to the incoming beam, or for use in non-conventional applications such as particle tracking in optical tweezing experiments.

Whilst small compared to the RAM required for the annotator look-up, if the accumulators are to accept any geometry of subapertures then no guarantees can be made

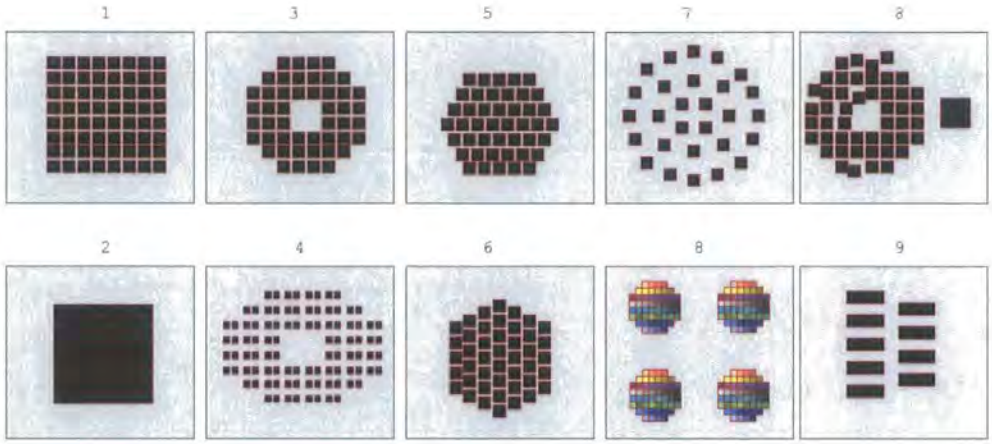


Figure 4.7: Some possible subaperture geometries for the AGC

that the readout of some subapertures will be completed before others begin, so the x, y and total intensity accumulators all need RAMs large enough for the total number of subaperturess. For 256 SAs in the example system this means 3x256 or 768 words are required, with this consuming another BlockRAM.

The AGC may be adapted to an imager with multiple readout ports by multiplexing the pixels from each readout port through the one pipeline, as long as the system clock is sufficiently fast compared to the pixel clock, as is normally the case. This is possible due to the fact no specific order is required between pixels due to the arbitrary RAM lookup.

4.4.4.3 Partially Confined Geometry Centroider

Whilst the arbitrary geometry centroider presented above is a fully capable system and one that the author has recommended for the ESO SPARTA platform, the heavy RAM requirements require the use of either a much larger FPGA - typically found only in the significantly more expensive device families, or external RAM. Either of these requirements impact the cost and physical size of the wider RTC system, making it unsuitable for the stated goal of a low cost, embedded RTC.

By enforcing several restrictions (hence the title partially confined) on the format and layout of the Shack-Hartmann microlens array and hence the WFS image a significant reduction in the RAM requirements for the annotator and interleaved accumulators may

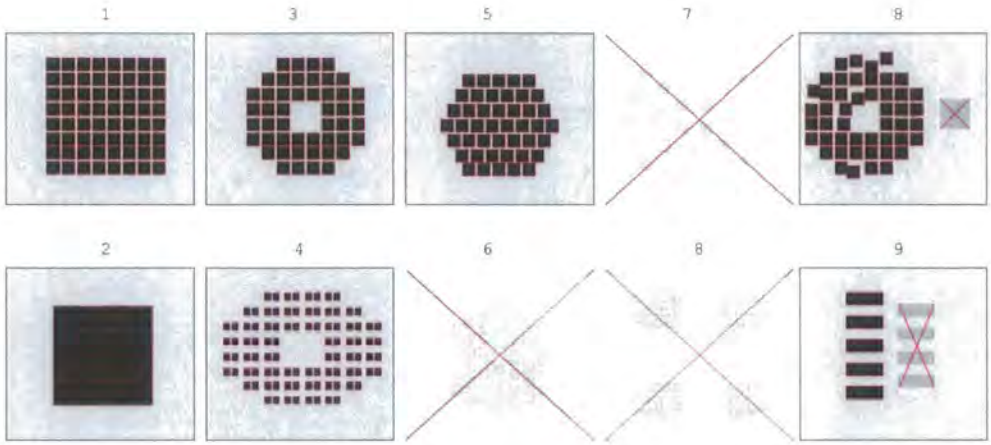


Figure 4.8: Limitations imposed by restricted centroider - Hexagonal geometries are only possible where the subapertures are horizontally aligned, circular geometries (7) and pyramid systems (8) are no longer support, and neither are systems with mismatched subaperture sizes (8).

be realised. The two principal restrictions on geometry this introduces are as follows:

(1) - Subapertures lie in rows of NSUB or less across the WFS image

This ensures that one row of subapertures is read out completely before readout of the next row of subapertures begins. This allows the memory requirements on the interleaved accumulators to be reduced from enough for the total number of subapertures in the image to just enough for one row of subapertures. This is possible as one row will always complete before readout of the next begins, meaning the memory can be reused. In practice this reduces the requirement of 768 words for the example system to 48 words, which is small enough to be implemented as SelectRAM in the FPGA meaning that a BlockRAM is no longer required.

(2) - Each subaperture is of the same size, and each subaperture in a row begins on the same row of pixels on the imager. The effects if these limitations are shown in figure 4.8

Given these constraints it is possible to produce an annotator that uses a simple algorithm executed by a state machine that is aware of certain WFS parameters (`NSUBX`, `NSUBY`, `NPIXX`, `NPIXY`) to generate the pixel annotations from a much reduced store of

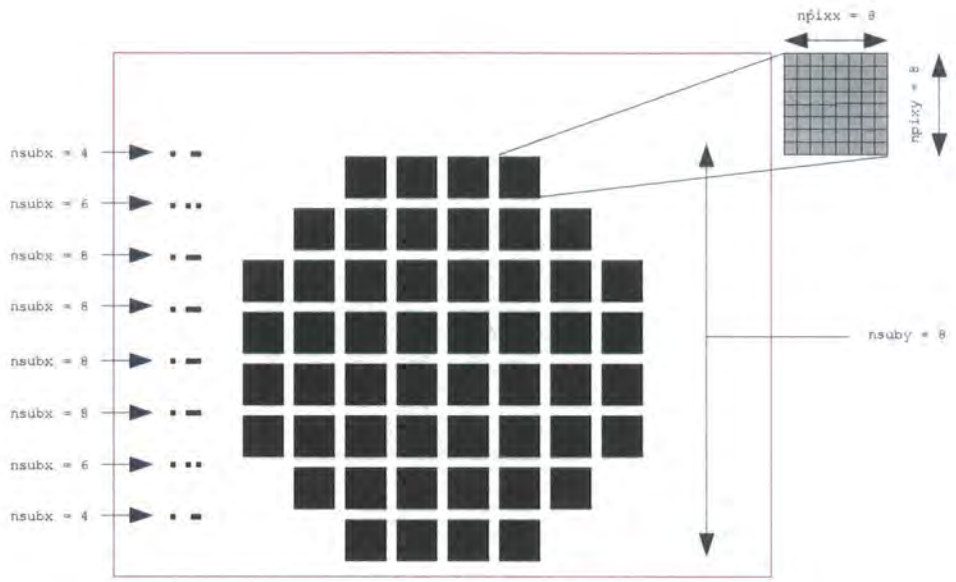


Figure 4.9: Compressed annotation RAM for horizontally confined geometries

information consisting of just 1 bit per pixel indicating if it is within a subaperture or not. By changing this 1-bit pixel map in the RAM and the WFS parameters fed to the state machine different geometries may be accommodated. Indeed the 1-bit lookup table is itself a viable candidate for compression further reducing the RAM requirements of the system, perhaps to enable a larger WFS to operate from one BlockRAM.

Although the vertical position of the subapertures within a row is fixed by this approach the horizontal position is fully variable allowing for the use of a hexagonal geometry or two interleaved square patterns with a variable offset as may arise in a SLODAR system. It should be noted that in order to use a hexagonal array the microlens and the WFS imager must be aligned so that a row of subapertures is aligned with the readout direction of the imager.

Modification for non square array of subapertures

The concept for a lightweight centroider as presented so far assumes that `NSUBX`, the number of subapertures in a row, is a constant. However most optical systems operate with a circular pupil, and some with a central obstruction to the pupil, leading to a reduced number of sub-apertures around the edges of the image - the image itself will still be rectangular. This leads to non-illuminated subapertures, which will output signals of either 0 or arbitrary data depending on noise levels in the system. This calculation of

unnecessary data does not slow the system down or waste resources due to the pipelined nature of the system, but it does need removal from the final WFS vector, otherwise the size of the vector and therefore control matrix, will be increased. The method employed to achieve this is to repeatedly encode `NSUBX` in the 1-bit bitmap instead of supplying it over the SMI. Figure 4.9 shows how these measurements are encoded. A subaperture must consist of at least two pixels, so a single active pixel in the map can be used for other purposes, in this case signifying that the following pixels encoded a value for NSUBX in binary rather than a subaperture.

4.4.5 Reference subtraction and centroid capture

Reference subtraction and centroid capture modules were created, each based around a single BlockRAM giving the capability to store and subtract up to a 2048 element reference vector and to capture up to a 2048 element centroid vector. These modules are able to operate at a frequency of 1 vector element per clock cycle. However a 2048 element **wfs** is significantly larger than required for many AO systems, so a third module was created that uses a single BlockRAM for both reference storage and capture, limiting the size of **wfs** to 1024 elements, enough for a 22^2 Shack-Hartman system.

4.4.6 Reconstructor Design and Implementation

As outlined in section 2.2.4 the reconstructor converts the wavefront sensor vector **wfs** into mirror drive commands **DM** by multiplying the WFS vector by some pre-calculated control matrix, **C**, i.e. $\mathbf{DM} = \mathbf{C}.\mathbf{WFS}$.

The general design of a hardware MVM unit depends upon the order in which the calculations are to be performed, i.e. whether the matrix is accessed row by row or column by column. For a row-wise system the unit requires the entire input vector before processing can begin, whereas in the AO system the WFS vector streams in in-step with the completion of readout of a row of sub-apertures from the imaging chip, so this method would require an additional latency. For this reason a column-wise approach is used. In this approach each element of the input vector is multiplied by the corresponding centroid to form part of the row total, and requires intermediate storage for partial totals in the output vector unlike the row-wise approach. This decision of row-wise or column-wise applies equally to CPU/DSP and FPGA based systems, and the desire to operate on streaming data leads to column-wise operation in both cases. This is one of the areas

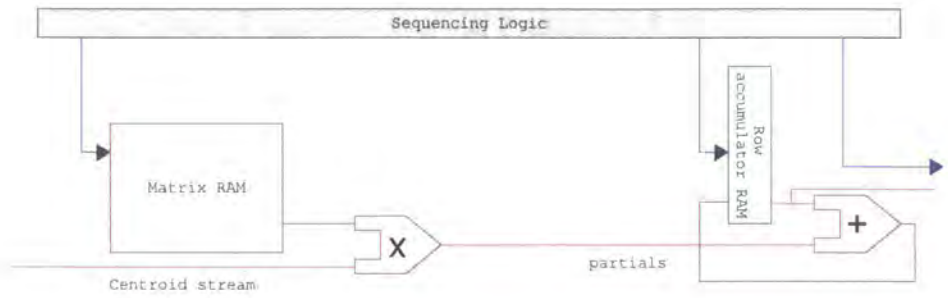


Figure 4.10: Generic FPGA based reconstructor

where the FPGA architecture excels, as physically separate memories can be used for the matrix storage and the partials storage, as shown in figure 4.10, a diagram of a generic FPGA based reconstructor.

In this figure the centroids forming the WFS vector stream into the system, with each vector element being sequentially annotated with each element for the corresponding column in the matrix, as retrieved from the Matrix RAM. After its retrieval a matrix element is multiplied with the centroid to yield a partial term whilst the next element is retrieved etc. The corresponding total from the Column Accumulator RAM is fetched and the partial term is added to this. The *Sequencing Logic* is aware of the shape of the matrix and vectors, and uses this information to generate the correct addresses for the two RAMs and control the writing of data to the column RAM. Further once a complete MVM operation is complete and the final DM vector resides in the column RAM the sequencing logic reads and streams this data out of the module sequentially and resets the values of the accumulator RAM to 0 in anticipation of the next MVM operation. This generic design uses two memories, one for storing the matrix coefficients and one for storing the partials of the output vector - which form of ram is used (external, internal BlockRAM or SelectRAM) will depend on the size of the matrix and the number of subapertures.

4.4.6.1 Design of an efficient reconstructor for the Xilinx Virtex-II architecture

For the target specification of 40 channels and a 10x10 Shack-Hartman with no more than 64 active spots, the matrix size is $40x2x64 = 5120$ elements. To deliver this data to a reconstructor at the target frame rate of 1000Hz and to deliver an MVM latency

of less than 5% of the frame time equates to a bandwidth of 102e6 elements / second. Using 16-bit fixed precision numbers for the matrix element a single Virtex-II/Spartan-III BlockRAM offers 1024 words of storage, so the reconstructor requires 5 BlockRAMs. For the accumulation of the partial terms in the output vector only 40 storage elements are required, a number small enough that SelectRAM build from the logic fabric may be used in place of BlockRAM, helping to keep the design small. A generic approach was taken to the implementation of the MVM unit for this reconstructor, with individual MVM 'slices' being created, each on based around a single BlockRAM containing 1024 matrix elements and a 16 word SelectRAM for the partial terms. Each slice corresponds to a part of the control matrix, corresponding to all WFS channels but only a subset of the DM channels in the matrix. These slices are tiled to form an MVM of the required size as shown in figure 4.11, and are all controlled by a common sequencing unit linked to the SMI. Additionally a readout unit connects to all the slices, on the right in the figure. This consists of a series of registers that are loaded with output values from the MVM slices' row accumulators. Multiplexers then convert these registers into one long shift register for sequential readout. The use of a shift register represents very large space savings over a data multiplexer approach due to the very inefficient nature of wide multiplexers implemented in FPGAs, and furthermore offers linear scaling of resource utilisation with size unlike a multiplexer approach.

Significant use was made of automatic code generation abilities available through using looping constructs such as `generate` in the VHDL language to simplify the construction of this module, and as such the number of slices may be varied between 1 and 16 in the VHDL code with only one parameter change in the source code files. The sequencing unit is capable of operating any number of MVMs with any matrix geometry permitted by the system size, with the necessary parameters for correct operation being loaded into the module at runtime by the SMI.

4.4.7 Gain and Integrator

The closed loop gain and integration are provided by the module illustrated in figure 4.12. As the elements of the $\nabla\mathbf{DM}$ vector are streamed in, they are multiplied by an integer gain, a 10-bit constant supplied by the pipeline's SMI. The result of this is then divided by 2^{10} to convert the integer gain from a range of 0-1024 to 0-1. This provides a gain with a resolution of $1/1024 = 0.001$. After multiplication with the gain the streaming elements

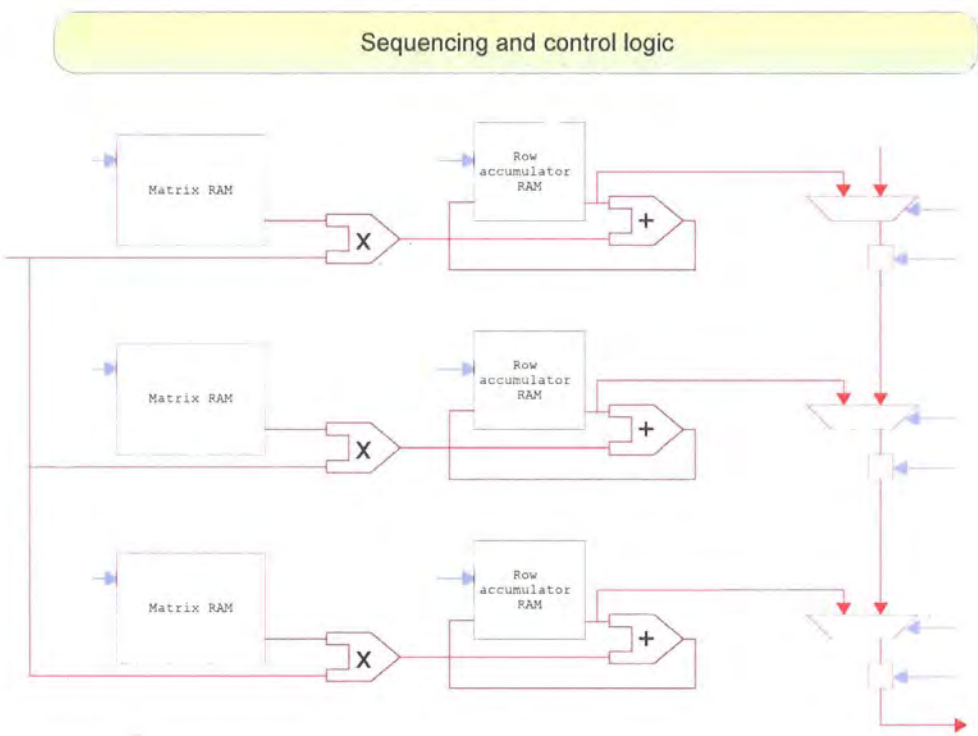


Figure 4.11: Reconstructor designed for all-internal operation

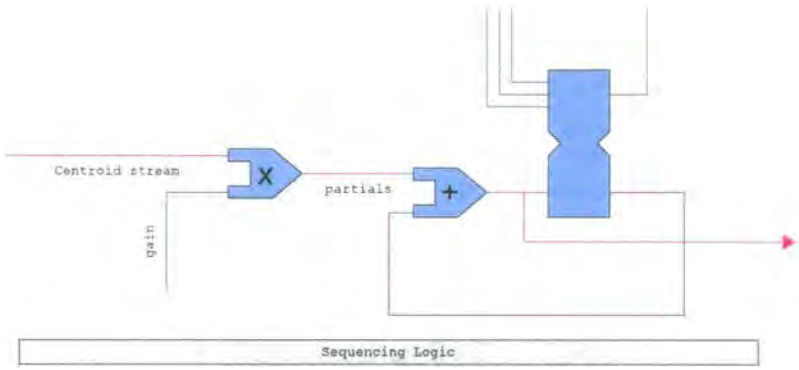


Figure 4.12: Gain and integration

are then added to the elements of the **DM** vector stored in, and fetched from a dual port BlockRAM. After the delta is added the value is output to the next pipeline module and re-stored in the BlockRAM. The use of a BlockRAM allows for an arbitrary number of channels up to the RAM's size of 1024. The second port on the BlockRAM is connected to the system management interface, allowing a host computer to program the default **DM** vector by writing it to the BlockRAM, and to monitor the values at runtime by reading from the device.

4.4.8 Code implementation

The majority of the AO pipeline modules and the wider SOC were written in the VHDL language as part of this work, with the exception of the Picoblaze CPU which whilst still written in VHDL is provided by Xilinx. The total amount of VHDL source written is approximately 0.5MBytes and 30,000 lines of code. A further 500 lines of code were written in assembler for the Picoblaze CPU.

4.4.9 Software architecture

An extensive set of software was written for use on the host computer using the Python programming language. Python is an object-orientated, bytecode interpreted language that runs on many common platforms such as Microsoft Windows, Mac OS X, UNIX and Linux etc. The code is heavily modular, separating low level communications with the AO RTC, configuration of RTC modules and user interface code.

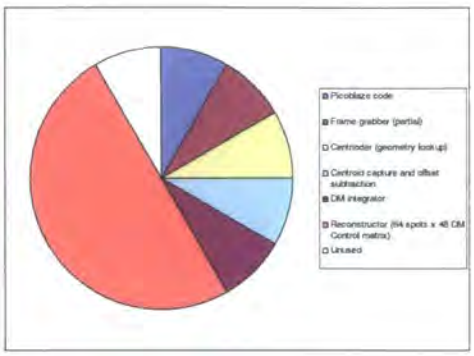


Figure 4.13: Breakdown of BlockRAM usage within the FPGA/AO Framework - half the available memory is used to store the reconstructor matrix with the rest being split between all remaining tasks.

4.5 Design performance and summary

The FPGA/AO framework presented in this chapter represents a body of code for use with a Xilinx Spartan 3, Virtex-II, 3 or 4 series FPGA for performing closed loop AO control. The system has been designed for WFS imagers of up to 128^2 pixels with up to a 16^2 Shack-Hartman array and pixel rates of up to 50MHz. The framework consists of a generic SOC for supporting streaming data processing and links to a host computer, and a custom pipeline for AO control. When compiled into an FPGA bitstream the tools report a usage of 85% of the target XC3S200 FPGA, with 11 of the 12 BlockRAMs being used, with the majority of these forming the reconstructor memory - a breakdown of BlockRAM usage is shown in figure 4.13. In reality the tools report the design will run correctly at speeds of up to 82MHz, somewhat faster than the goal of 50MHz. Subsequent chapters examine the realisation of this framework in a laboratory demonstrator.

Chapter 5

Laboratory Demonstrator 1 - Design and construction

5.1 Introduction

In order to test the FPGA/AO framework in operation a laboratory demonstration AO system was constructed. The primary aim of this system is to allow the testing of a real time controller (RTC) based on the FPGA/AO framework with real components including a wavefront sensor and deformable mirror etc. in a realistic environment, and not to produce a complete AO system for any particular application.

A simple overview of the laboratory demonstrator is shown in figure 5.1 with the system consisting of four modules; a diffraction limited light source, a system to provide turbulence emulation, the corrective AO system itself and finally a camera to quantify the operation of the system. In this chapter a detailed view is presented of these four modules and their interconnection, the realisation of the FPGA/AO framework used for the laboratory demonstrator and finally a summary of the system specifications.

The laboratory demonstrator consists of an optical setup including the DM, WFS etc. built on a laboratory bench and a collection of electronic systems and computers required for controlling these components. Firstly the overall optical design and the AO electro-optics are reviewed in section 5.2 before the AO system electronics and their integration with the optics are presented as section 5.3.

Figure 5.1: Demonstrator AO System - a high level functional breakdown of the laboratory demonstrator components for testing the FPGA/AO framework

5.2 Optical Layout

The detailed optical layout of the demonstration system is shown in figure 5.2 with the four basic modules from the overview figure of 5.1 identified.

5.2.1 The Source Module

A diffraction limited source is provided by a 633nm HeNe laser (HeNe) combined with a spatial filter (A1). The beam from the spatial filter is collimated to a diameter of 3.3mm by L1 and finally folded off two adjustable mirrors (M1 and M2) intended to provide flexibility in the system alignment.

5.2.2 Turbulence emulation

Turbulence emulation in the system is provided by means of a Displaytech 256x256 pixel FLC device. This is a binary (0 or π) phase only SLM that may be programmed with deformed diffraction gratings to generate continuously varying analogue wavefronts as described by Neil et. al[**49**]. The 3.3mm beam from the source fully illuminates the width of the SLM, producing many beams corresponding to both multiple diffraction orders from the structure of the device itself and to the many orders generated by the programmed grating. A pair of transform lenses and a spatial filter (L2, L3, A2) are used to select the desired diffractive order, and simultaneously to expand the beam from the 3.3mm diameter of the FLC to 12mm. The FLC is very inefficient in terms of light usage due to the binary nature leading to many diffractive orders, so for simple system tests the FLC may be replaced by a mirror whilst retaining the lenses for beam expansion.

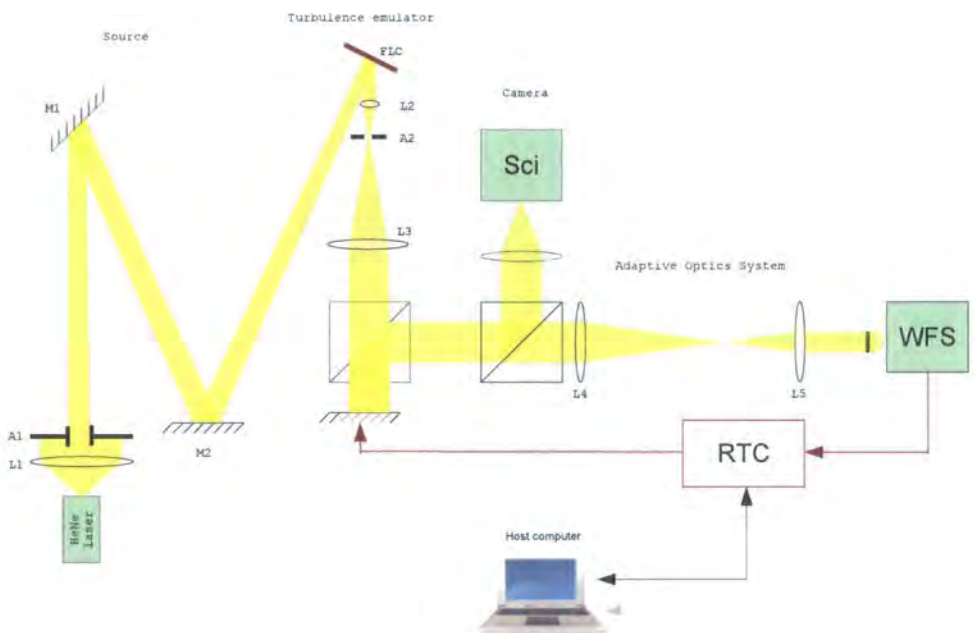


Figure 5.2: Optical bench configuration for the AO demonstrator - the left hand box illustrates the diffraction limited light source module, followed by a turbulence emulation modlue incorporating a binary FLC SLM and subsequent spatial filter, with the AO system itself forming the third module and finally a simple imaging camera module to quantify system operation.

5.2.3 Adaptive Optics System

The AO system consists of an OKO 37 channel MMDM placed at a conjugate plane to the FLC in the turbulence emulator, followed by a beam splitter providing a 'science' beam for measuring the system operation and a second beam for wavefront sensing. The lenses L4 and L5 shrink the beam from the 12mm diameter of the mirror to 1.2mm that illuminates a 200μm pitch, 6.3mm focal length microlens, leading to an array of 8x8 SH spots. The lenses are placed such that the microlens is conjugate to the DM and therefore also the turbulence source.

Due to the demonstration nature of the system it was possible to operate with a surplus of light, therefore allowing the deformable mirror to be illuminated on axis, using a beamsplitter to separate the incoming and outgoing light beams - the 75% waste in light this incurs is of no great significance. This is in contrast to many practical AO systems where the DM must be illuminated off axis to allow for the separation of incoming and outgoing beams by geometry rather than a beamsplitter. The DM and WFS are examined in more detail below.

5.2.3.1 DM

The deformable mirror used in this system is a 37 channel micro-machined deformable mirror from Flexible Optical B.V., formerly known as OKO Technologies. This device is an electrostatic deformable mirror as reviewed in section 2.2.3.2, with a clear aperture of 15mm and a 12mm wide hexagonal array of 37 control actuators centred under the aperture. Two standard Flexible Optical 20 channel high voltage driver PCBs were used to supply the 0-250V signals required to make full use of the mirror range. The mirror was operated at a bias of 125V to offer deflection in both directions.

5.2.3.2 WFS

Imaging chip

For the wavefront sensor a Kodak KAC-9630[50] CMOS imager sensor was used for image acquisition. This device is a single microchip that includes a 128 x 101 pixel sensor array along with all the digital logic required to sequence readout and an analogue to digital converter to digitise the output. The device was chosen both because of its high maximum frame rate - 580fps, the simplicity of use as only two digital connections are required to the FPGA - a bidirectional serial bus (I^2C) for configuring imager parameters

Figure 5.3: The 37 Channel OKO Deformable Mirror - photograph of the 37 channel MMDM from OKO/FlexibleOptik BV as used in the demonstrator

Parameter	Value	Units
Number of pixels	101 x 128	Pixels
Pixel pitch	20	μm
Array size	2.56 x 2.00	mm^2
ADC resolution	8	Bits / pixel
Max frame rate	580	fps
Expsure (min)	10	μs
Exposure (max)	20,000	μs

Table 5.1: Summary of the features of the Kodak KAC-9630 CMOS imager

such as exposure and gain, and an eight bit parallel data-bus from the device to the FPGA for reading digitized image and framing data, and also for the low cost of the device, with individual microchips retailing for under 10US$, significantly cheaper than the Dalsa cameras previously used by low cost AO systems. The device specifications are summarised in the table 5.1.

Initially this chip was produced by National Semiconductor under the name LM9630, with the transfer of the devices to Kodak occurring in January 2005 when Kodak aquired National Semiconductors imaging product line. [51]. Prior to this transition an evaluation PCB housing the imager device and required external components was produced and sold by National Semiconductor under the name *LM 9630 Headboard*. This is a small PCB (aproximatly 2cm x 2 cm) that contains an LM-9630 image sensor and all the associated

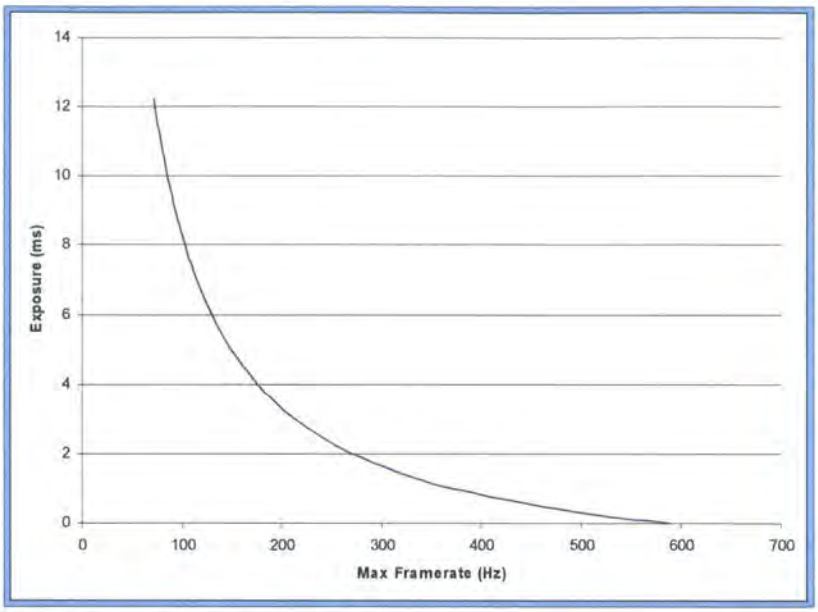


Figure 5.4: Maximum framerate vs Exposure for the KAC-9630 image sensor - as the sensor is not able to simultaneously expose and readout subsequent frames there is an exponential drop-of in maximum possible exposure with increasing framerates

electronics needed to operate it, including discrete components and a 10MHz clock source, providing a convenient platform for testing the device in a WFS.

Whilst The KAC-9630 provides global shuttering of all pixels to synchronise exposure, it does not implement any form of double buffering or charge storage, so is unable to expose one frame whilst simultaneously reading out another. This means that the maximum possible exposure decreases inversely with exposure, as detailed in the device's datasheet[**50**] and illustrated in figure 5.4.

Micro-lens array and geometry

The microlens array used in the WFS has a 200μm pitch on a square geometry, with a 6.3mm focal length. The array was manufactured by Adaptive Optics Associates. This corresponds to 10 detector pixels / subaperture, so the desired 8x8 Shack-Hartman will require 80x80 detector pixels, which combined with the number of microlenses in the array (over 100 across the array) means that precise registration and alignment of the two are not necessary.

Before use in the system the microlens array was illuminated with a laser and the resulting focal plane spots were imaged with a high resolution scientific camera, as shown

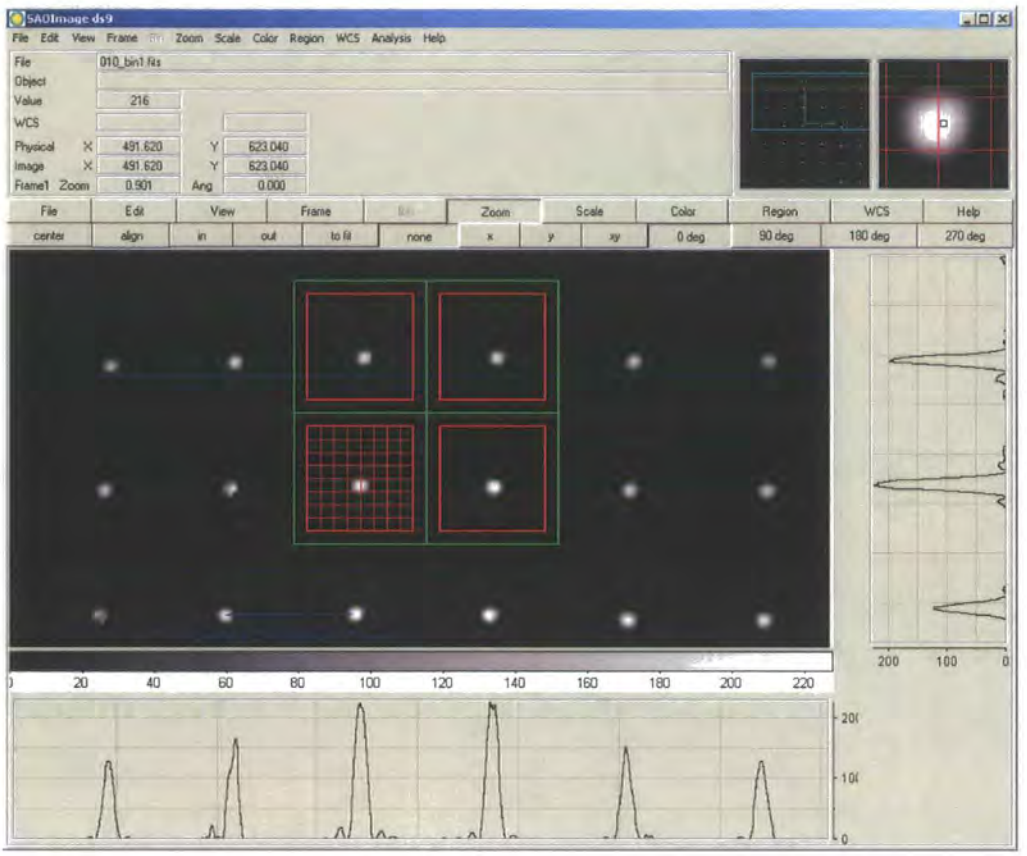


Figure 5.5: Shack-Hartmann spot pattern - focal plane spots produced by the microlens array as recorded with a high resolution camera. Overlaid in red are the pixels forming individual subapertures as would be seen by the low resolution KAC-9630 used in the WFS. The microlens is diffraction limited with the Airy discs of individual spots being visible in the horizontal and vertical slices plotted in the figure.

in figure 5.5. Overlaid on the figure in red is a grid showing the pixel size on the KAC-9630 used in the WFS. The spots are of a similar size to an individual WFS pixel, which combined with the poor fill factor of the device would lead to non-linear sensing. To overcome this when the microlens is being aligned to the KAC-9630 they are separated by more than the focal length such that the defocused spots falling on the detector have a full width half max of at least two pixels, guaranteeing linear sensing.

Assembly

The WFS was constructed by placing the microlens array inside a small metal barrel which may be held in a standard 25mm adjustable lens mount, to which the LM9630 Headboard is fixed as shown in figure 5.6. The lens mount provides for control of microlens

Figure 5.6: Wavefront sensor assembly - Photographs showing the LM9630 camera headboard screwed to the rear of a lens mount (left), which is clamping the microlens array held in a small barrel mount (right). In the later photograph the active area of the KAC9630 imager is visible through the microlens array.

rotation and microlens-imager separation, and the whole assembly may be directly used on an optical bench or placed on a translation stage for fine positioning. In practice no translation stage was necessary.

5.2.4 Imaging Camera

For imaging the output of the AO system a simple lens (L6) was used in conjunction with a QImaging Retiga 1300 scientific camera, which is connected to the host computer for data recording by a FireWire link. The QImaging camera may be used independently of the RTC host, or by using the vendor supplied API may be integrated into the host software for tasks such as static abberation correction.

5.3 Real time controller

This section presents the overall configuration of PCBs used to implement the AO control system. The core FPGA/AO framework is implemented on a Xilinx Spartan-3 200 FPGA, one of the smallest devices in the family that retails for us$25 as of January 2007. This device is hosted on a *NuHorizons Spartan 3 Starter Kit*[52], a compact (12cm by 8cm) PCB that contains the FPGA, configuration flash, power regulation etc, along with standardised

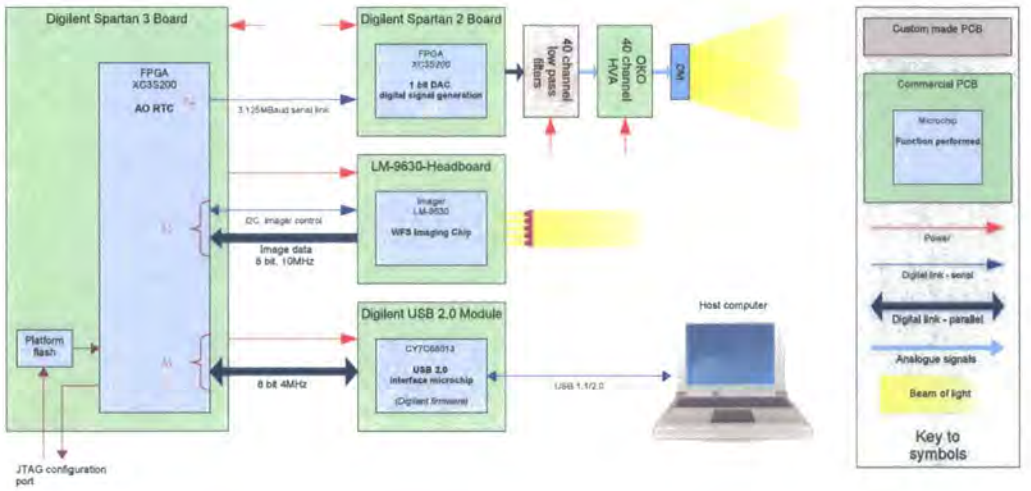


Figure 5.7: Electronics configuration - a block diagram of the electronics modules for the AO test bench, showing the interconnection of the various PCBs used in the system

IO connectors and retails for $US150. The Spartan 3 Starter Kit PCB also includes various LED displays which were connected to the debug bridge outlined in section 4.3.1.2, and numerous other features such as 1MB of SRAM, a VGA graphics port and an RS-232 serial port that are not used in this work.

This AO pipeline running within the FPGA on this board communicates with three more PCBs, firstly the actuator commands for the mirror are output through a simple custom 3 wire 3.125MHz serial link over the A2 header to an older generation Spartan 2 evaluation board from Digilent - this board does not perform any processing but acts as the DAC controller, as there were insufficient IO pins left on the Spartan 3 board to perform this directly. Secondly the FPGA communicates over the B1 header with the KAC-9630 imaging chip on the headboard to aquire WFS images. Finally a *Digilent USB 2.0* module is used - this is a small PCB from Digilent which contains a Cypress EZ-USB interface microchip running custom firmware developed by Digilent, and bridges a USB 1 or 2 link from a host computer to a simple bidirectional FIFO interface connected to the FPGA via the A1 header. Whilst the Cypress devices is claimed to be able to operate at the maximum practical bandwidth of the USB 2 link of 40MBytes per second, the Digilent module is limited to approximately 4MBytes per second by Digilent's decision to use a legacy FIFO interface for communications with the FPGA, and omitting the synchronous clock from this interface.

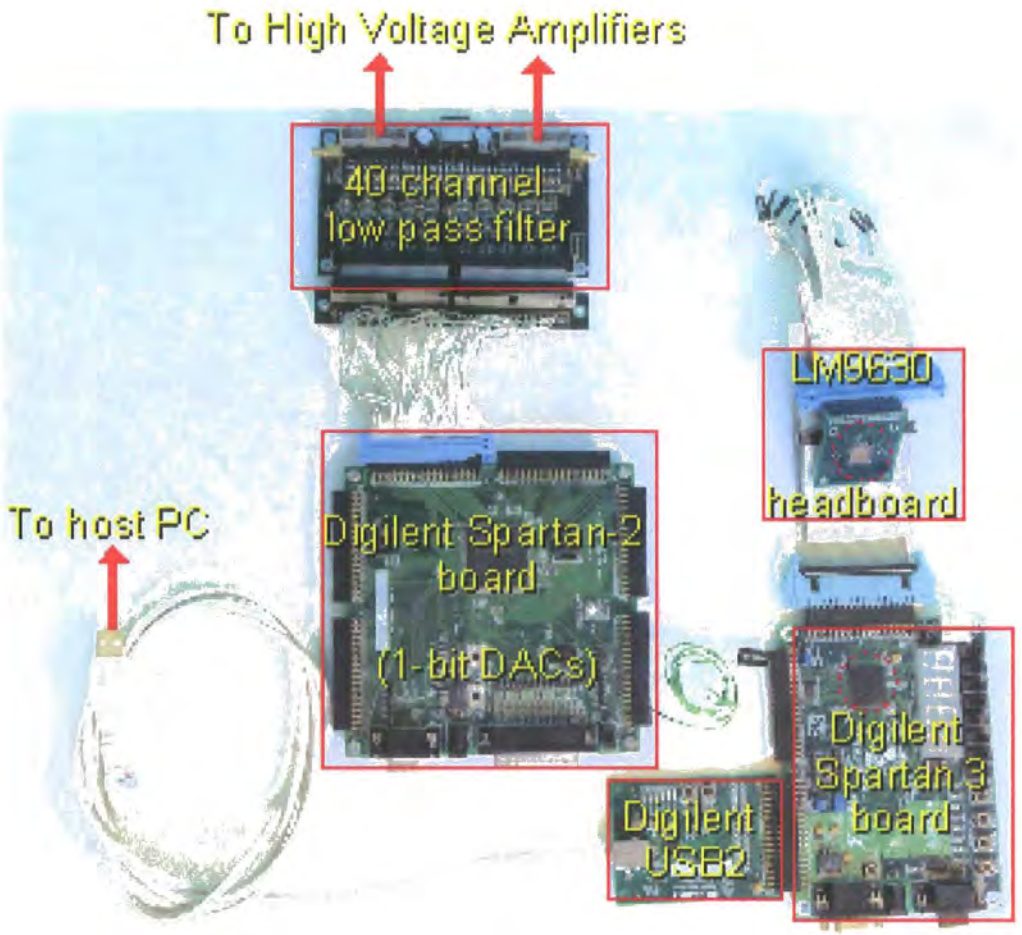


Figure 5.8: Photograph of PCBs used in the demonstrator - this photograph shows the configuration of boards used to implement the FPGA/AO framework and RTC for the laboratory demonstrator, overlaid with identifying captions. Each PCB corresponds to one shown in the block diagram in figure 5.7. Clockwise from top right there is the LM9630 headboard used for the WFS here the imaging chip is circled. This communicates with the Digilent Spartan 3 board, where an XC3S200 (circled) runs the entire closed loop control. This board integrates with a Digilent USB2 board for communications to a PC and transmits DM drive commands over a 2-wire serial interface to the Digilent Spartan-2 board which combined with the 40 channel low pass filter boards and high voltage amplifiers (not shown) forms the entire AO system drive electronics.

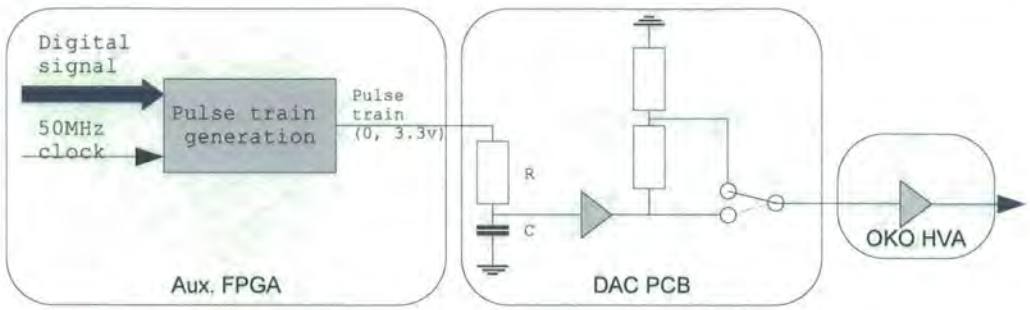


Figure 5.9: DM Drive electronics - shown here is a block diagram of the drive electronics for a single DM channel. When replicated 37 times this drives the OKO DM. Initially the auxiliary FPGA generates a digital pulse train, which is then low pass filtered by a simple RC filter on the DAC PCB. An amplifier boosts the signal from 0-3.3v to 0-10v before a potential divider that may be bypassed lowers the range to 0-5V. Finally this range is amplified by an OKO supplied HVA.

5.3.1 DM Drive electornics

For driving the DM several stages of electronics were used as shown in figure 5.7. Initially an auxiliary FPGA (the XC2S200 on the Digilent Spartan 2 Board in figure 5.8) converts the digital actuator commands - the **dm** vector - into a series of pulse trains, with one for each channel. The average on/off ratio of each channel is set by the FPGA such that when the pulse train is averged by the RC filter on the *DAC PCB* an analogue voltage is produced, with a 100% on ratio in the pulse train yielding 3.3V (the logic level of the FPGA) and a 0% ratio leading to 0V. This is then amplified by on the DAC PCB to a range of 0-10V, before a potential divider allows the signal to be reduced to a range of 0-5V if desired. This is to allow for operation of the DACs with various HVAs. For the OKO HVAs the range of 0-5V was selected, as with the gain of 53 provided by the OKO cards this provides a final range of 0-250V, ideal for use with the OKO MMDM. This approach was taken over the use of discrete DAC devices to demonstrate the use of simple RC filters and FPGA logic for the DAC stage - the amplifier and the potential divider on the DAC PCB could both be omitted if the HVAs were designed to provide a gain matched to the voltage range of the 1-bit DACs, allowing a low cost system using an FPGA to omit DACs entirely and simply use RC filters fed from digital pulse trains on the inputs to the HVAs.

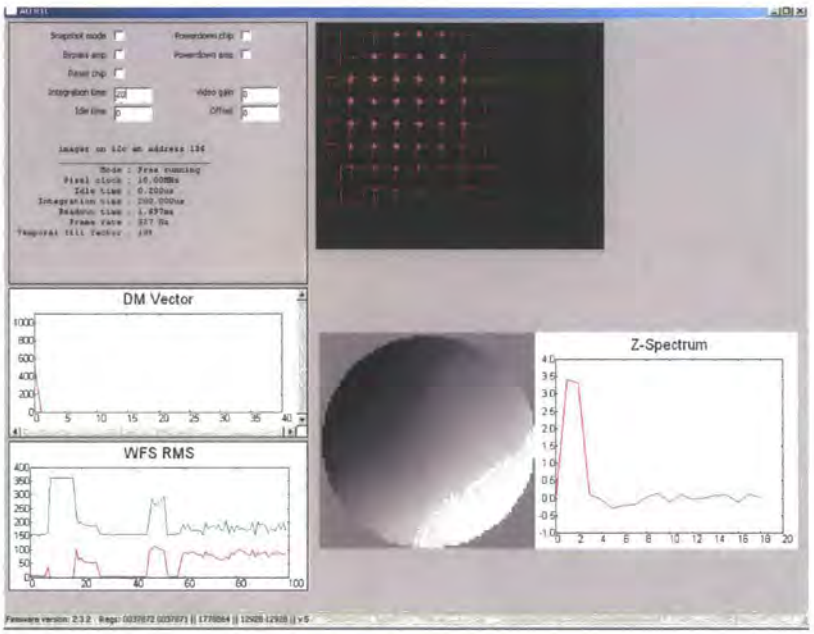


Figure 5.10: Screenshot of the AO GUI created for monitoring the AO RTC - GUI components include, clockwise from the top left, a control panel for the KAC-9630 imager, WFS image display with a graphical centroid plot overlaid, open loop reconstruction of WFS Zernike spectrum and corresponding phase plot, RMS centroid value and the DM actuator command vector.

5.3.2 FPGA/AO framework implementation

An FPGA configuration was created using the custom system on a chip (SOC) and AO pipeline as presented in chapter 4 with specific *image aquisition* and *output* modules from figure 4.3 being created for the KAC-9630 image sensor and the custom serial link to the DM electronics. Also a specific *Host interface* from figure 4.2 was created for the Digilent USB 2.0 module to provide a host interface. For calibrating the system a series of Python scripts were produced, including scripts that override the control of the FPGA/AO RTC implementation to generate interaction matrices and perform static aberration correction, and scripts for the processing of the interaction matrix into a control matrix.

For observing closed loop operation a simple GUI was created as shown in figure 5.10. The GUI uses the cross-platform wxWidgets[53] GUI library through the wxPython bindings. Heavy use of Python's multi-threading abilities is made within the code. This GUI allows control of the imager parameters and provides live displays of data from the AO pipeline running within the FPGA. Control of AO loop parameters such as gain and

DM figure was not included in this GUI but they are accessible from a Python command line shell which may be run as an extra thread.

Chapter 6

Laboratory Demonstrator 2 - Results

This chapter presents a series of experimental results taken with the laboratory demonstrator showing the calibration and operation of the system described in chapter 5. Results are presented of the testing of the WFS and reconstructor components of the system, followed by the calibration matricies and static aberration correction necessary for closed loop operation. Finally the system is shown to operate in closed loop correcting optical turbulence.

6.1 Open Loop Measurements

The first measurements taken with the system consisted of open loop WFS measurements. These were conducted before the construction of the full laboratory system shown in figure 5.2 was complete, with both the FLC from the turbulence emulator and the DM being replaced with fold mirrors, leaving a system in which the source laser illuminates the wavefront sensor, the images from which were processed into centroids by the RTC. The KAC-9630 image sensor was configured to acquire images at 500fps, with the FPGA processing all images and measuring the centroids at the same rate. These centroids were captured by a host laptop computer over the Digilent/USB2.0 interface.

Two sets of measurements are presented with the first in figure 6.1 showing the centroids recorded against time whilst one of the alignment mirrors was manually scanned to adjust the wavefront slope in the x-axis at the WFS. The x-centroid against time of a single subaperture is also plotted beneath. The x-axis centroids oscillate negative and

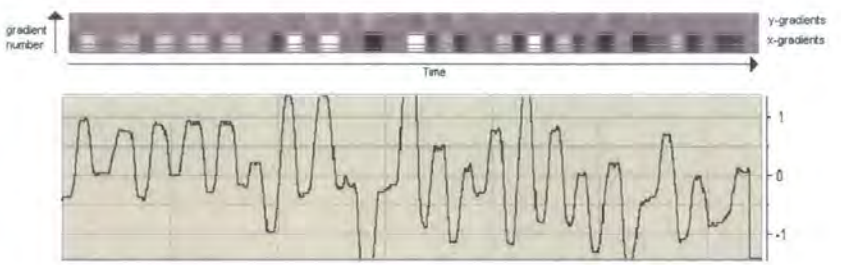


Figure 6.1: high speed open loop sensing of global tilt - shown here (top) are 15,000 WFS vectors plotted against time representing the centroids of the Shack-Hartmann spots whilst an oscillating global wavefront tilt is applied. The data is shown as a greyscale plot where light and dark represent the extreme positive and negative slope values. The plot (bottom) shows the response of a single centroid against time.

positive as the wavefront tilt is adjusted. The y-axis centroids show a slight ghost of the x-axis motion due to a rotation of several degrees on the microlens array. The second set of measurements are shown in figure 6.2 and consist of a 30 second log of the centroids (top) and a zoomed in view of 3 seconds (bottom) recorded when a hot soldering iron was placed below the beam inducing turbulence.

In both cases the RTC is processing 128x101 pixel images at 8 bits/pixel into 128 8-bit centroids, compressing the data transferred to the PC by a factor of a hundred, decreasing the datarate required to the host from 6.1MB/sec for the raw images to 0.06MB/sec for the centroids. Such a capability is useful for a continuously running data acquisition system as either the processing burden or the disk space requirements on the host computer are reduced. For a system with high framerates and more pixels the benefit increases, especially if the raw pixel datarate exceeds the bandwidth of the host link.

6.2 System Calibration

Having demonstrated basic operation of the WFS, the DM and the turbulence emulator were aligned into the optical setup leading to the complete demonstrator system as shown in figure 5.2.

Figure 6.2: high speed open loop turbulence sensing - this plot shows 15,000 WFS vectors recorded whilst a soldering iron induces turbulence in the beam (top) with a zoomed view (bottom) showing 1,500 WFS vectors corresponding to 3 seconds of data. Centroid values are plotted as greyscale with light and dark representing extremes of slopes. In the lower plot a diagonal structure is clearly visible caused by relatively fixed aberrations translating across the beam.

The turbulence emulator was configured to provide a static, unaberated beam. This beam was used to record the null wavefront, and as the light source for the generation of the interaction matrix. This is achieved by the host computer commanding the RTC to sequentially actuate the DM control channels and record the resultant WFS vectors in accordance with section 2.4.1. The generated matrix is shown in figre 6.3(a) with the corresponding control matrix in (b).

6.2.1 Reconstructor performance

Having verified the operation of the WFS and achieved a control matrix it is now possible to verify the correct operation of the figure reconstructor. This was tested by uploading the control matrix into the RTC and running the WFS and reconstructor open loop, with the DM figure kept static. Both the WFS and reconstructed DM vectors were logged against time in the presence of a turbulence source. Figure 6.4 shows a plot of the DM vector as reconstructed against time by the RTC on the left. Using the WFS vector that was also logged a simple reference software reconstructor using floating point arithmetic was run on the host PC in non real-time yielding a reconstructed DM figure that should correspond to that produced by the RTC. The right side of the figure shows the difference between the RTC and software produced DM figures - they are in agreement within the limited rounding error of the fixed point RTC reconstruction, which at less than 0.06 DAC units is significantly smaller than required to maintain operation indistinguishable from the software reconstructor.

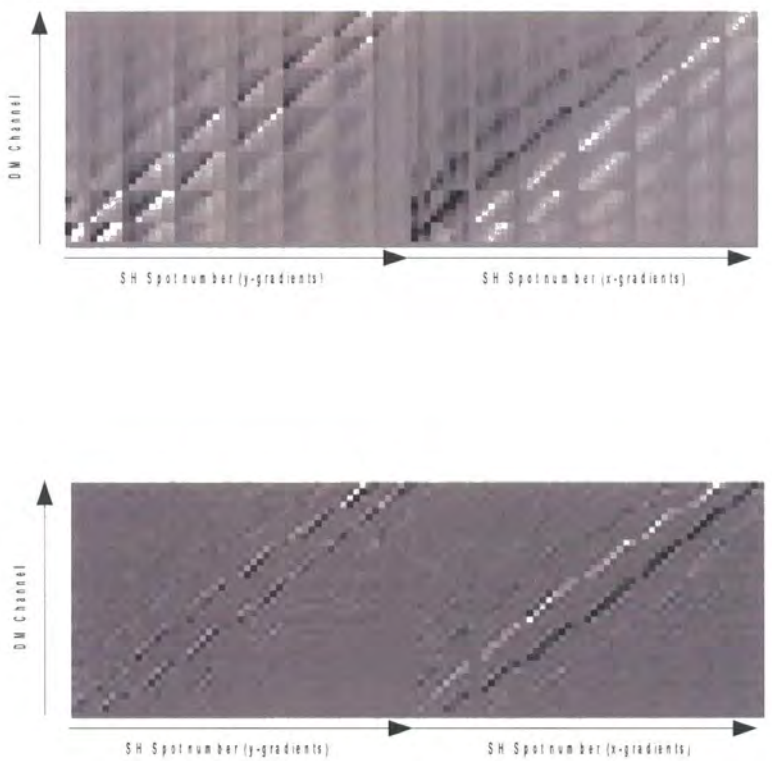


Figure 6.3: Interaction Matrix (top) and the derived Control Matrix (bottom) for the AO Laboratory Demonstrator

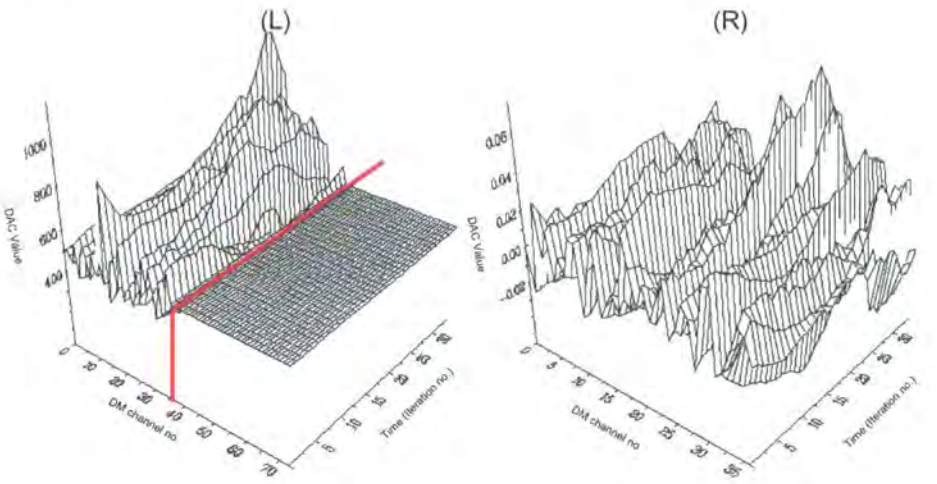


Figure 6.4: FPGA based reconstructor test - the WFS observes time evolving turbulence, for which the DM command vector is reconstructed and plotted against time, both by the FPGA based RTC (left, channels 0-36) and by a non-real time software implementation, with the difference between the two being shown to scale (left, channels 37-75) and magnified (right)

6.2.2 Self correction tests

Various factors of alignment and configuration my affect the closed loop operation of the system - for example all the DM control channels must be well observed by the WFS, and any poorly sensed modes should have been removed in the creation of the control matrix. One simple way to test this is to run the system closed loop in the absence of turbulence and to occasionally modify the position of an actuator distorting the mirror figure. If the control matrix is well conditioned then the system should return any actuator to the default position.

This process was carried out experimentally using the interrupt capability of the Picoblaze CPU in the FPGA to distort successive actuators on the DM every 30 frames, equivalent to every 60ms at the 500Hz frame rate. Whilst this was occurring the DM vector was logged for each frame and is plotted against time in figures 6.5(a) and (b). Two DM figures are shown in the slices, with (c) showing the flat mid-range DM vector at time 0 when the loop is first closed, and a slightly irregular vector at frame 30 where noise in the system has propogated into the DM figure and channel 0 has just been moved from a mid-range position to a DAC value of 0. Finally the DAC value of this actuator

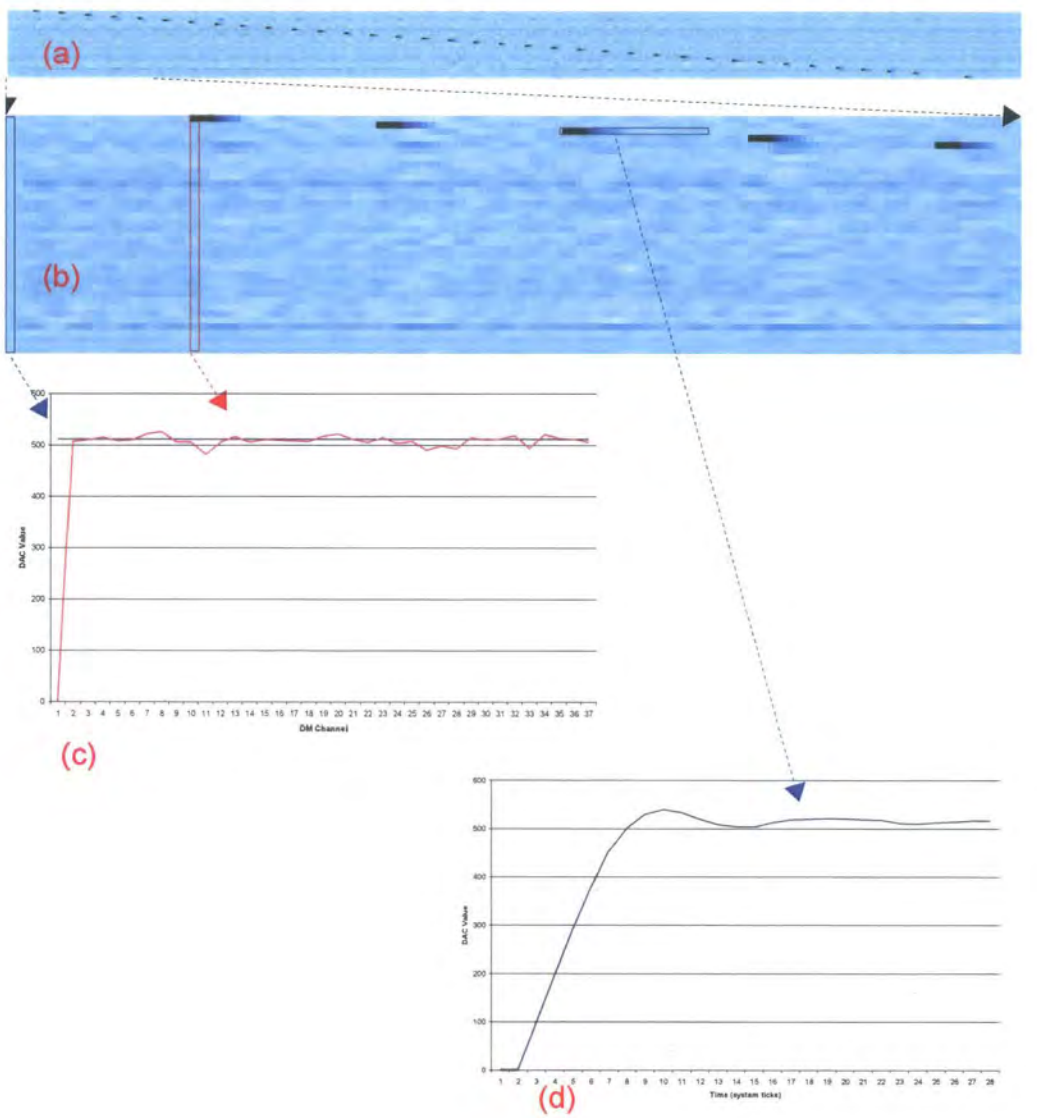


Figure 6.5: Closed loop step response testing of the AO system - the control loop is closed in the absence of turbulence and successive DM channels are actuated. The system acts to return these channels to their default values. This is shown in the plots of DM vector against time in (a) and (b), with (c) and (d) plotting the indicated slices through the data.

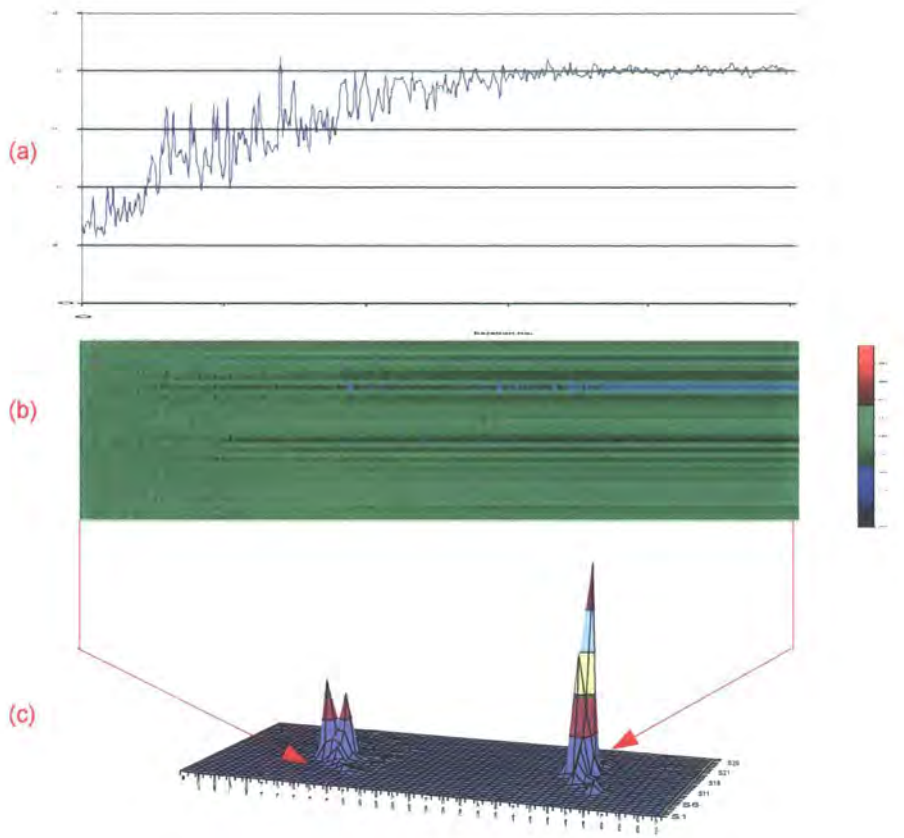


Figure 6.6: Static aberration correction - plots showing the use of the Simplex algorithm to reduce the effect of static aberration on the system, with (a) showing the metric used as a basis of the optimisation, (b) showing the evolution of the DM vector against time and (c) showing initial and final PSFs.

is shown against time in (d) as the control system corrects the mirror figure. As can be seen in (a) each channel correctly returns to the mid-range DAC value with no instability appearing in the DM figure.

6.2.3 Static aberration optimization

Before closed loop correction of turbulence could be demonstrated it was necessary to correct for severe static aberrations primarily due to the poor surface quality of the DM. This was achieved by using an image metric minimization method based on the simplex algorithm, where the metric M was calculated by $M = \sum_{x,y} I(x,y)^2 / \sum_{x,y} I(x,y)^2$. The

Simplex increased the peak intensity of the PSF by a factor of approximately 3. Figure 6.6 plots the PSF derived metric used by the simplex and the DM vector against time, and shows the initial (statically aberrated) and resultant (corrected) PSFs

6.3 Closed Loop

To test the system with realistic high bandwidth turbulence the FLC was used. This generated dynamic wavefronts caused by translating the FLC pupil across a larger phas-escreen, with the turbulence having $\tau_0 = 50ms$ and average D/r_0 values of 4 and 8. A 5 second circular buffer was played on the FLC for each strength of turbulence whilst a continuous series of short exposure images were acquired with the 'science' camera, the QImaging Retiga. These were then averaged to produce a long exposure image.

Figure 6.7 shows these long exposure images for when the AO system is on and off. For the flat wavefront the closed loop AO system has no effect on the PSF. For $D/r_0 = 4$ and there is a marginal improvement in peak intensity and reduction in width of the PSF, with a much more visible corrective effect for $D/r_0 = 8$.

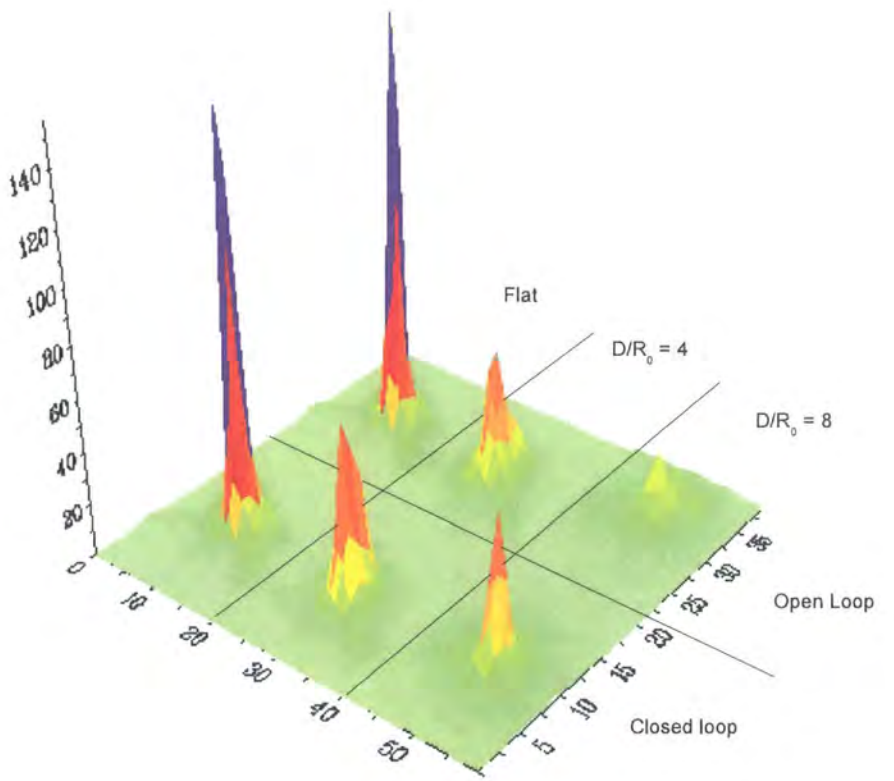


Figure 6.7: Closed loop AO correction - long exposure PSFs with the AO system off and on for various strengths of turbulence.

Chapter 7

Further Applications

7.1 Introduction

In this chapter we present alternative applications of the body of work in which the FPGA framework and prototype hardware developed for the AO Real Time Controller (RTC) is developed into an enhanced prototype and subsequently employed in different contexts, those of controlling a tip/tilt loop and of performing open loop data acquisition and processing in real time.

Initially an enhanced prototype of the AO RTC is examined, followed by presentations of this hardware applied to tip/tilt control in a free space optics (FSO) environment and to the construction of a 'smart camera' for use in optical tweezing.

7.2 Enhanced Prototype

A set of custom PCBs were designed and then fabricated with the help of SIRA Technologies Ltd. This was with the aim of producing a compact, low cost demonstration of FPGA based AO control.

The revised boards used in the closed loop AO environment of the laboratory demonstrator are illustrated in figure 7.1 which may be compared to figure 5.7 where commercial PCBs were used. The custom designed RTC board integrates the FPGA and support circuitry with the Cypress CY7C68013 USB2 interface chip. The extra control over the interfacing of the FPGA and the USB device compared to the commercial boards, combined with custom USB firmware development enabled measured data rates to exceed 20MB/sec, approximately 4 times as fast as the solution using the Digilent boards. The

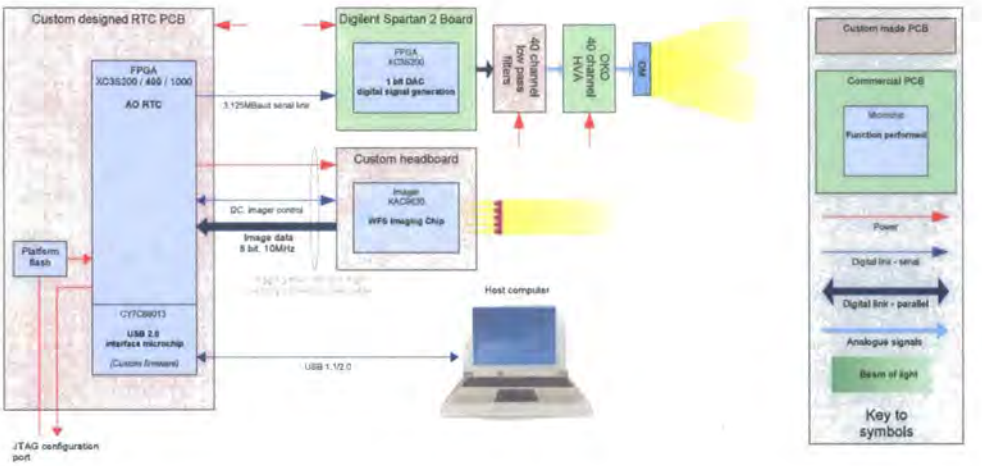


Figure 7.1: Custom PCBs developed for AO control - the main RTC PCB integrates the Xilinx FPGA and Cypress USB2 interface devices used in the laboratory demonstrator, and has a dedicated interface connector to the custom imager PCB. This reduces the size and complexity of the laboratory demonstrator, as may be seen by comparing this figure to 5.7.

front face of the custom RTC board - before population with components - is shown in figure 7.2(a).

Furthermore a custom headboard was designed to replace the 'LM-9630 headboard' (from National Semiconductor) for the imager, both to reduce the size (from 45mm by 38mm to 40mm by 30mm) and due to supply problems with the commercial board. Finally matching high density connectors were used on the main board and the imager headboard, allowing them to be connected with IDC terminated ribbon cable, removing all soldering and providing a significantly neater alternative to that seen in figure 5.6(a).

To protect these boards a mechanical enclosure was designed that provides a Linos optical rail system on which the imaging headboard is mounted in conjunction with a microlens array and/or a C-Mount lens adapter. This enclosure and the PCBs are shown photographed in 7.3 with the lid removed to show the PCBs and optics.

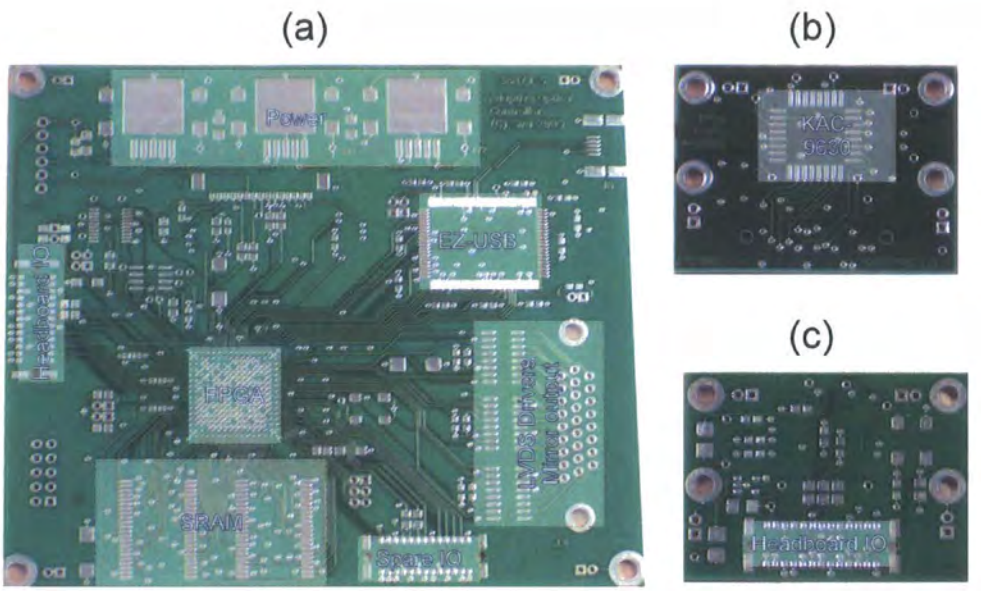


Figure 7.2: Photographs of the custom PCBs for AO control - (a) shows the front face of the RTC board with principle components labelled, (b) and (c) shown the front and rear faces of the imager headboard.

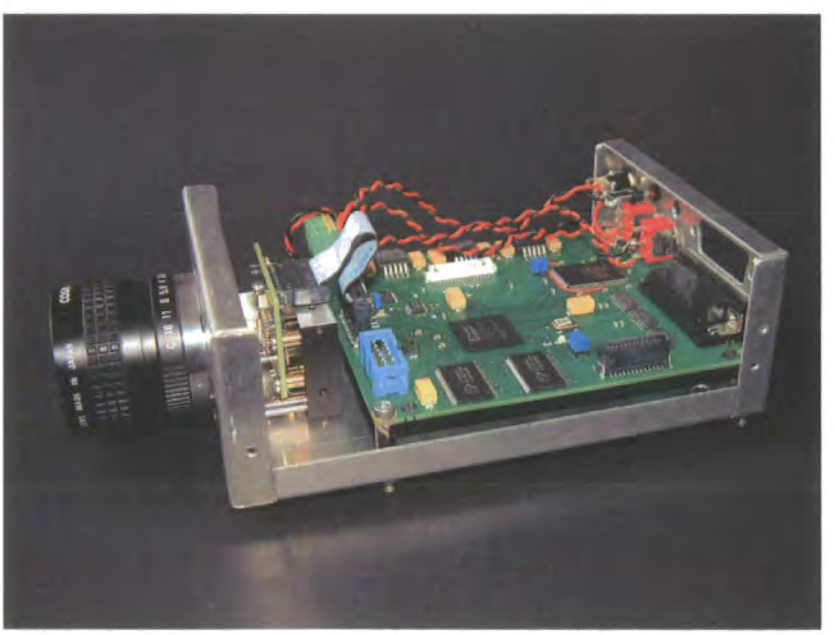


Figure 7.3: Photograph of the enhanced prototype AO controller

7.3 Dedicated tip/tilt controller

As part of Durham's contribution to the DTI-funded Adaptive Long Wave Free Space Optical Networks Solutions (ALFONSO) project an embedded tip/tilt sensing and control system was produced, with the intention of this being included in a technical demonstration system of long distance, high speed FSO communications. The optical configuration of this demonstrator is shown in figure 7.4, and is based around the use of a custom Ritchie Chrtien telescope attached to an optical system contained on the baseplate. This system uses a Physik Instrumente S-325[54] three actuator PZT stage to remove tip and tilt from the received beam, to compensate for atmospheric turbulence and motion of the communications units.

A modified real time pipeline was created for running the tip/tilt control loop. By replacing the AO pipeline in the real time controller shown in figure 4.2 with a new tip/tilt only pipeline as show in figure 7.5 a dedicated tip/tilt controller is produced. As with the AO pipeline the design begins with the camera interface, although the high order WFS has been replaced by a series of modules forming a low order WFS for tracking just one spot, but with more flexibility than the high order WFS. The first of these modules is the 'pixel annotator' that is aware of the image sensor geometry and uses this to tag each pixel with its co-ordinates (X,Y in the figure.) The second module in the WFS in the 'ROI Isolator' that is configured at run time with the region to be centroided for guiding the tip/tilt loop. This module isolates just this portion of the image and renumbers the pixels to begin at (0,0) in the top left corner of the ROI. Finally the data from this module are processed by a centroider. The ROI may fall anywhere on the imager and be of 2x2 to 32x32 pixels in size, with these parameters being changeable by the host computer at run time.

To enable the tip/tilt system to work with a three actuator mirror a reconstructor matrix is still needed to converted the XY signals from the centroider into commands for the mirror. The use of a reconstructor has a second benefit of allowing an arbitrary rotation to exist between the axes of the sensor and the mirror. As with the high order AO loop the output of the reconstructor is integrated to form a closed loop system . Finally a de-pistoning operations is performed, in which any global piston is removed by subtracting the average actuator value from each actuator value. This is necessary as any pistoning of the actuator is unsensed by the WFS and will reduce the range available to the actuators.

The reconstruction, integrator and de-pistoning represent only a few mathematical

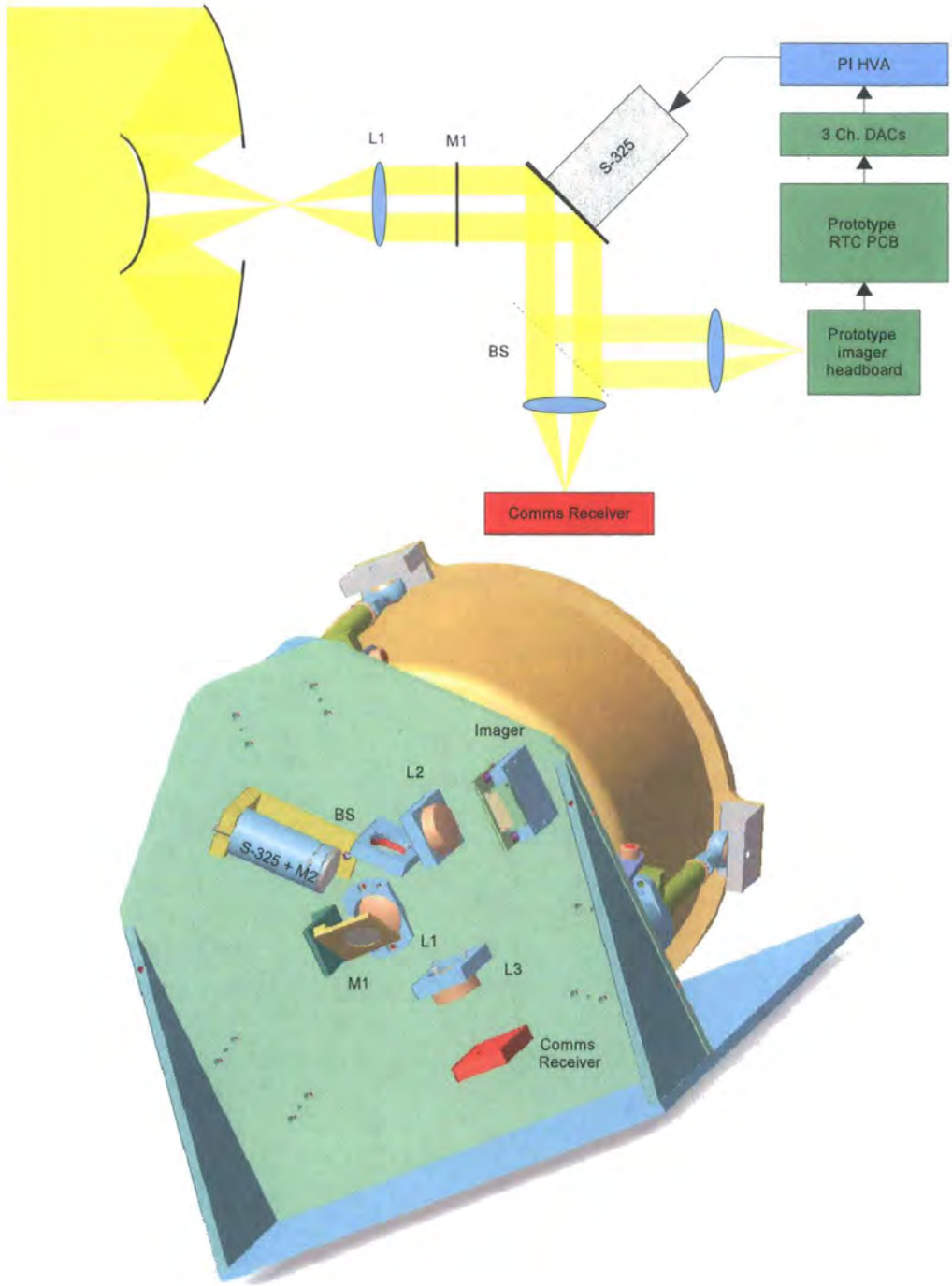


Figure 7.4: FSO receiver with tip/tilt system - Both the electro-optical design (top) and the opto-mechanical design (bottom) are shown. In both figures a telescope (t1) collects light from a distant source. This is collimated (L1) before passing of a passive fold mirror (M1) and an actuated mirror (mirror M2 bonded to a S-325 actuated stage). This stabilised beam is separated by a beamsplitter (BS) into a beam for the communications receiver and for sensing and closed loop control using a series of electronics modules shown in green.

operations that are performed once for every frame, so the total amount of computation required here is minimal. The equations for these operations are as follows:

1. **Reconstruction**; where $\nabla M(0...2)_{n+1}$ is the reconstructor output for the mirror actuator x, CX, CY are the measured centroids and R is the reconstructor matrix.

$$\begin{bmatrix} \nabla M0_{n+1} & \nabla M1_{n+1} & \nabla M2_{n+1} \end{bmatrix} = \begin{bmatrix} CX_n & CY_n \end{bmatrix} * \begin{bmatrix} r_{0,0} & r_{1,0} & r_{2,0} \\ r_{0,1} & r_{1,1} & r_{2,1} \end{bmatrix}$$

2. **Integration**, where g is the closed loop gain and $M(0...2)'$ are intermediate actuator values

$M0'_{n+1} = M0_n + g.\nabla M0_{n+1}$

$M1'_{n+1} = M1_n + g.\nabla M1_{n+1}$

$M2'_{n+1} = M2_n + g.\nabla M2_{n+1}$

3. **De-pistion**; calculates the actuator commands M(0...2) by subtracting the average of the intermediate actuator values form each value. $M0_{n+1} = M0'_{n+1} - \langle M0'_{n+1} + M1'_{n+1} + M2'_{n+1} \rangle$

$M1_{n+1} = M1'_{n+1} - \langle M0'_{n+1} + M1'_{n+1} + M2'_{n+1} \rangle$

$M2_{n+1} = M2'_{n+1} - \langle M0'_{n+1} + M1'_{n+1} + M2'_{n+1} \rangle$

Rather than implement dedicated hardware blocks for each of these tasks, they were implemented as software on the embedded Picoblaze CPU core with an interrupt being used to request processing. The execution of this code requires the execution of approximately 1200 CPU instructions, the count being so high primarily because all the multiplications must be implemented in software. Given the Picoblaze execution rate of one instruction every two cycles and the 50MHz clock, this equates to a latency of 50μs, significantly higher that the sub-microsecond times possible through using dedicated code units build with the FPGA, but more than adequate given the requirements for a 1ms frame time. The actuator values and all coefficients, such as the gain and the matrix, are stored in the CPU's built-in 64 byte Scratchpad RAM, which may be accessed by the host computer to monitor or modify values. By implementing these algorithms in the already-present CPU the resource requirements for the RTC were reduced to an extent that the design will run in a Xilinx Spartan-3 50, the smallest and cheapest device in the low cost Spartan-3 range of devices.

7.4 A Smart Camera for particle tracking

This section outlines a final set of experimental work based around the AO RTC, in which it performs open loop data processing and acquisition in an optical tweezing experiment,

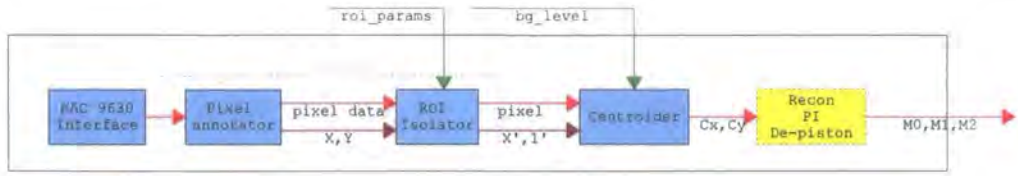


Figure 7.5: Tip/tilt data path - shown here is a block diagram of the data processing pipeline used for dedicated tip/tilt sensing and control. The final module, 'Recon, PI and De-piston' is implemented not as FPGA code but as interrupt driven software on the embedded Picoblaze processor.

offloading the image processing from the host to the FPGA in the RTC, forming a device that may be termed a 'smart camera'. This is conceptually similar to the operation of the RTC for open loop wavefront sensing as presented in section ??, but with different data processing code.

7.4.1 Background

Single beam optical trapping in which a small dielectric particle may be confined within 3 dimensions by the focus of a single beam of light was first demonstrated in 1986[55]. Typically this is a biological cell or small manufactured bead contained within a medium such as water. By moving the focus of the light beam it is possible to move the trapped particle, a process known as optical tweezing. A single particle held in a trap is subject to perturbation by the random Brownian motion of the carrier medium and restricted by the restorative force of the trap. As such a study of the dynamics of trapped particles can reveal information on the force of the trap or the activity of the medium. Typically the dynamics are studied by observing the transmitted laser trap light with a quadrant detector, as any lateral motion of the trapped bead steers the transmitted laser light, or by a high speed camera imaging the bead under a separate white light source.

When multiple particles are trapped in close proximity their motions may become coupled to a degree by fluid dynamics. In order to measure the dynamics of two particles using the quad cell method the trap light for each particle must be separated. This is possible using polarisation states as has been demonstrated by Quake and Meiners [56] for two particles. However for more than two the polarisation method ceases to be helpful, rendering the quad cell method of little use. This leaves high speed cameras, however these

are unpractical in use as their datarates are significantly faster than a PC may handle, leading to the use of a limited several second acquisition buffer. This limited acquisition ability combined with the length of time required to post process the images extracting the position of particles poses challenges where data must be acquired over longer periods of time.

Initial studies of more than two particles have focused on beads in a more viscous medium[57] of glycerol solution where the dynamics are slowed down sufficiently that a normal video frame rate camera may be used. To enable dynamics measurements with a water based medium - more typical in biological applications - the AO RTC was employed using modified programming to image one or more beads in optical traps at high speed, matching the flexibility of a high speed camera, but also to process the position of the beads into simple coordinates in real time and return only these positions to a host PC, matching the low data rates and therefore ease of use of the quad cell method.

Below the implementation of the AO RTC for this application, dubbed a 'smart camera' is examined, followed by the experimental setup and results, initially comparing measurements of a single bead taken with the smart camera and a QPD, then results obtained by examining multiple particles with the smart camera.

7.4.2 Smart Camera Implementation

The custom engineered PCBs and packaging shown in the previous section were used, with the microlens array removed to convert the device into a plain imaging camera rather than a Shack-Hartmann system. Whilst a series of beads as seen by a camera resemble a series of SH spots and may be centroided with a centre of mass algorithm[58], their freeform geometry rules out the use of the partially confined geometry centroider examined in section 4.4.4.3 and developed for the AO RTC.

An alternative data processing pipeline was created that derives from the tip/tilt controller in that an individual region of interest representing one bright spot, or bead, is isolated from the image data and then centroided by the same modules as the tip/tilt system. However, compared to the tip/tilt system the isolation and centroiding pipeline has been repeated eight times, enabling eight parallel regions or beads to be centroided simultaneously. Additional register files are created to simplify the configuration and readout of the many signals this creates.

The final pipeline is able to process up to 8 independently positioned regions, each of

an independently variable size of between 2x2 and 32x32 pixels. By reducing the 12.6KB images to just 16 centroids each with 10 bits - padded to 2 bytes - of precision the datarate is reduced 32 bytes per frame, or by a factor of over 400.

Whilst being rapid to implement and test by way of being highly derived from the tip/tilt work, this did not prove to be an efficient use of the FPGA, with more than 86% of the target XC3S400 FPGA's LUTs being required by a complete system covering just 8 regions, significantly more than are required for an AO system including SH processing for perhaps 64 spots or regions and a figure reconstructor. Primarily this is due to the duplication of logic, especially the numerical division units, between each centroider and the large size of the output register file which itself requires 8% of the device. If further development of the smart camera for this application is perused a more efficient redesign would enable more particles to be tracked simultaneously or for a lower capacity FPGA device to be used.

A GUI was created to configure and acquire data from the smart camera, and is shown in figure 7.6. The collapsible panels on the left provide control over the imager parameters, the colour mapping used to display images, the recording of data etc. The main display area in the centre shows a full frame live image from the KAC-9630 chip, with 8 regions of interest that may be dragged round the image by the user - doing so programs the corresponding ROI processor in the FPGA pipeline. Finally on the right each individual region is displayed, with sliders to control the background subtraction level used in the corresponding centroiding calculations. Any data that is below the subtraction threshold is zeroed by the FPGA for calculation purposes, and is coloured in red on the GUI as a visual queue to the user.

7.4.3 Experimental setup

7.4.3.1 Single particle

For comparative imaging of a single particle with the smart camera and the QPD a dilute sample of 2μm diameter silicon beads in distilled water was prepared, and a drop of this placed between two glass slides. These slides were placed in the sample tray of a microscope as shown in figure 7.7, where a 532nm continuous wave laser is brought to a focus within the sample by the microscope objective. The light from the laser continuous to propagate through the sample, and is collimated by the condenser lens before being directed onto the QPD by a beamsplitter. Simultaneously the sample is illuminated through the condenser

Figure 7.6: Screenshot of the Smart Camera GUI tracking 8 particles - the GUI consists of control panels (left) for various features, a live video display (centre) on which regions of interest for centroiding may be moved by mouse, and zoomed views of the region of interest, showing centroids by cross-hairs, and with sliders to ajust background subtraction levels (right).

by a white light source, enabling the beads to be imaged by the smart camera, mounted on a viewport on the microscope.

7.4.3.2 Multi-particle setup

The setup used for trapping multiple particles is shown in figure 7.8 and is similar to that used for one particle. However a HoloEye LCR 2500 SLM and associated spatial filters are placed in the laser feed. The Gerchberg-Saxton hologram creation algorithm[**59**, **60**] was used to generate a pattern for the SLM such that 8 simultaneous foci are produced in the same horizontal plane in the sample cell in an octagonal alignment.

7.4.4 Results

7.4.4.1 Single particle

The position of a bead in the single trap was measured over the same 30 second period by both the QPD and the smart camera, each at an acquisition rate of 500Hz. It was not possible to synchronise the camera triggering with the separate acquisition system for the QPD so direct comparisons of the position data are not possible. However the form of the power spectrum of the bead position is well defined both theoretically and

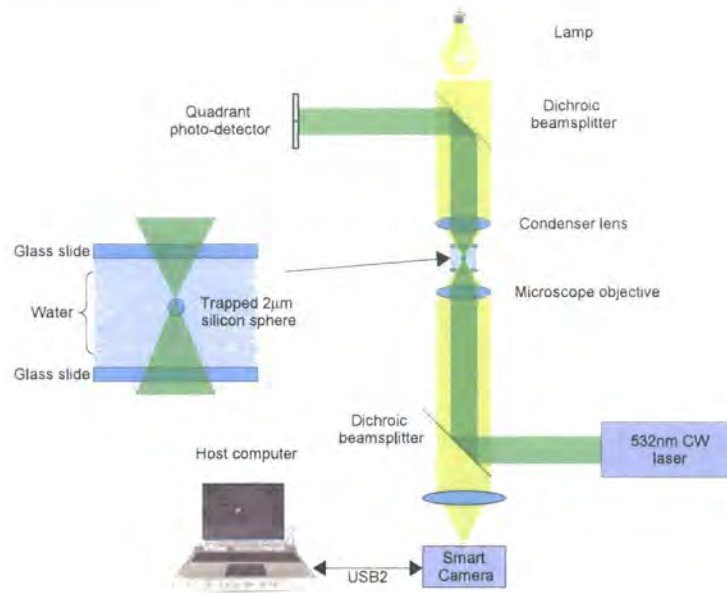


Figure 7.7: Experimental setup used for the single particle experiment - a continuous wave 532nm laser is brought to focus within a sample creating an optical trap, with transmitted trap light being directed onto a QPD and the sample cell imaged by the smart camera.

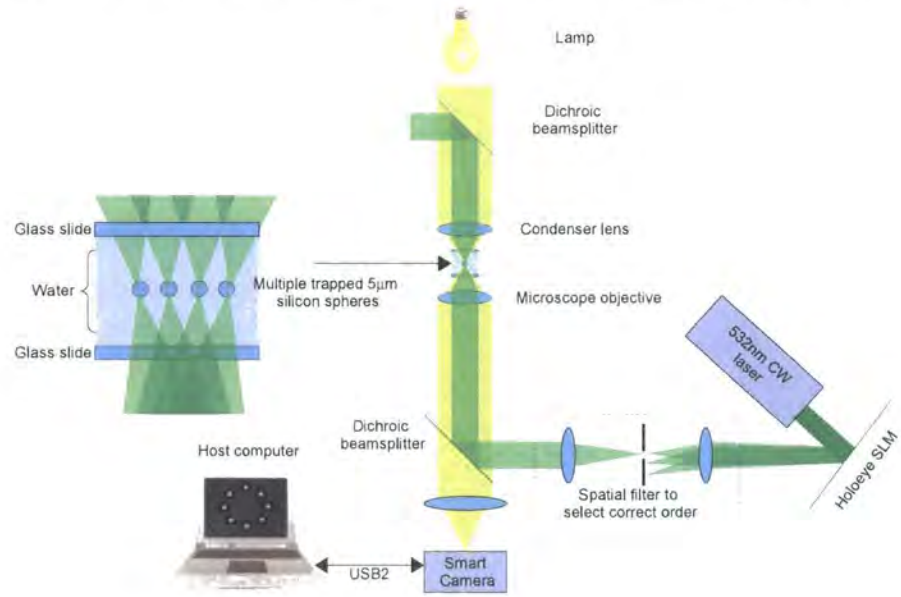


Figure 7.8: Experimental setup used for the hydrodynamic coupling experiments - a similar setup to the single particle experiment, but with the QPD removed and an SLM and associated spatial filters used to generate multiple traps, each of which is filled with a bead.

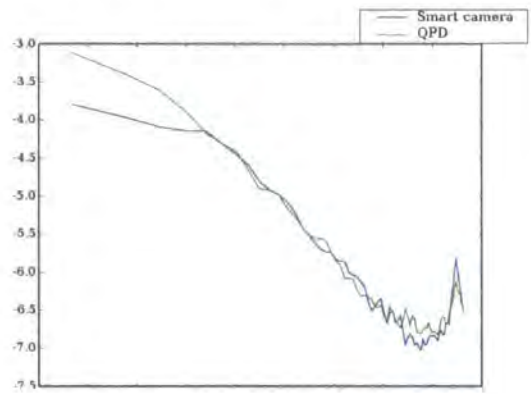


Figure 7.9: Power spectra of 1 optically trapped bead - a log / log plot showing a comparison of positional power spectrum data measured for a single bead with the smart camera and a QPD. Both methods show similar power spectra, consisting of a flat low frequency region (the confinement effect of the optical trap) and a linear roll-off at high frequencies corresponding to Brownian motion. Further on the far right are noise spikes caused by vibration of the equipment.

experimentally[61] and should be invariant of synchronisation errors. Figure 7.9 shows a log/log plot of the power spectrum of the bead position as measured by both the smart camera and the QPD, showing the devices to be in agreement.

7.4.4.2 Multiple particles

The experimental configuration was changed to that in figure 7.8 and the smart camera configured to process and log the position of 8 beads simultaneously. Further, efforts were taken to reduce the amount of vibration affecting the experiment by switching off all unnecessary equipment with moving parts. The camera was able to do this for an effectively indefinite length of time. The power spectrum of the positions of 8 beads was processed for 10 minutes of recorded data, and this is shown in a log/log plot in figure 7.10, showing the spectrum of all 8 traps to conform to the expected shape, also seen in the single particle case, and showing good uniformity between the trap strengths and a reduction in the noise on the far right of the plot. In these experiments the use of the RTC to form a smart camera has enabled continuous data acquisition in a regime not previously possible. Further joint work with Glasgow is in progress assessing these results.

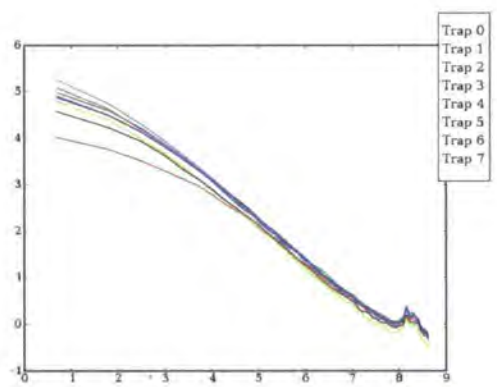


Figure 7.10: Power spectrum of 8 beads in optical traps

Chapter 8

Concluding Remarks

8.1 On Work Undertaken

8.1.1 Adaptive Optics

Having defined the scope of a closed loop adaptive optics (AO) system and presented the mathematics underlying its operation in chapter 2, approaches to building a real time controller (RTC) for AO were examined in chapter 3 along with a review of previous low cost RTC designs.

A system for processing high speed streaming images with a field programmable gate array (FPGA) device was developed, consisting of a custom system on a chip (SOC) design presented in chapter 4 that includes comprehensive frame sequencing and acquisition control and options for communication with a host computer such as a high bandwidth USB2 link. Chapter 4 also presents various processing modules for use with this framework that perform the mathematics outlined in earlier chapters needed for closed loop AO control, enabling the construction of an FPGA based AO RTC.

Chapter 5 presents a complete laboratory demonstration AO system using a low cost CMOS sensor, the Kodak KAC-9630 imager, for wavefront sensing and the RTC from chapter 3 implemented on a low cost gate array, the Xilinx Spartan-3 200. With these two devices being the most complex components of the control system and each costing under 15 it is clear that the system delivers extremely low cost wavefront sensing and control.

A summary of experimental results obtained with the system including open loop wavefront sensing and closed loop wavefront control results are presented in chapter 6, demonstrating the data-reduction capabilities of the system when used for open loop

data-processing and the ability of the system to correctly drive a wider AO system. A modification of the RTC for tip/tilt only sensing and correction is presented briefly in chapter 8.

The RTC developed and presented in this thesis represents a significantly smaller and cheaper approach to AO control than any the author is previously aware off - the use of a single, small FPGA based system board for the entire control loop represents a significant advance over previous low cost AO work based on using commodity PCs[13, 12] and represents the first AO control system directly suitable for use in an embedded environment.

8.1.2 Smart Camera

Whilst intended as a closed loop control system the RTC developed and explored in earlier chapters was shown to be capable of capturing images at high speed and performing open loop data processing from these images, transmitting the results of this processing to a host computer - for example the open loop turbulence sensing data from the results chapter.

Chapter 8 presents a 'Smart Camera' designed and constructed to act in such a way, drawing on the AO RTC and performing open loop data acquisition and processing for optical tweezing experiments.

8.2 Future Directions

8.2.1 Future AO control systems

Having demonstrated closed loop AO control with an FPGA through a series of evaluation PCBs, and then having shrunk these somewhat with the enhanced prototype it would now be possible to produce a detailed design for a significantly smaller AO RTC, perhaps a single circuit board of less that 40mm x 40mm size that contains a KAC9630 imager and an XC3S200 FPGA, with all required support circuitry including the USB device. Excepting the analogue drive circuitry required by a deformable mirror this one circuit board would represent all that is required for a stand-alone sensing and control system for AO. Given the rapid evolution of microchip devices and short life-cycle of specific components, there seems little point in designing such a PCB until a commercial need arises.

A further possible evolution for the AO RTC is integration with alternative imaging devices. There are several reasons for this including the obsolescence of the KAC-9630

device, it's inability to readout and expose simultaneously leading to poor light efficiency and its relatively low pixel count and framerate.

8.2.2 High Order AO Control

The use of fixed point arithmetic in figure reconstruction, as examined in chapter 3 and verified in the results chapter bears further investigation, as if the technique extends to the larger (many thousand control channel) matrix operations of ELT scale astronomical AO then the complexity and memory bandwidth benefits this brings compared to floating point arithmetic may bring a significant reduction in cost and complexity to their design.

8.2.3 Smart Camera

The alternative use of the lightweight FPGA/AO framework for other imaging purposes, generically termed a 'smart camera' is an open-ended question that is ripe for further investigation.

Bibliography

[1] H. W. Babcock. The Possibility of Compensating Astronomical Seeing. *PASP*, 65: 229, October 1953.

[2] J. Liang, D. R. Williams, and D. T. Miller. Supernormal vision and high-resolution retinal imaging through adaptive optics. *Journal of the Optical Society of America A*, 14:2884–2892, November 1997.

[3] A. Roorda and D. R. Williams. The arrangement of the three cone classes in the living human eye. *Nature*, 397:520–522, February 1999.

[4] W. Jiang, N. Ling, Y. Zhang, Z. Yang, X. Rao, C. Guan, C. Wang, L. Xue, and E. Li. Medical and industrial application of adaptive optics in Institute of Optics and Electronics, Chinese Academy of Sciences. In W. Jiang, editor, *5th International Workshop on Adaptive Optics for Industry and Medicine. Edited by Jiang, Wenhan. Proceedings of the SPIE, Volume 6018, pp. 1-16 (2005).*, pages 1–16, December 2005.

[5] C J Hooker, E J Divall, W J Lester, Moutzouris K, Reason C J, and Ross I N. A closed-loop adaptive optical system for laser wavefront control. In *Central Laser Facility Anual Report, 1998 - 1999 , p. 153 - 155*, pages 199–200, 1998. Accessed from **www.clf.rl.ac.uk/Reports/1998-1999/pdf/86.pdf** on Jan 15th 2007.

[6] R. M. Sova, J. E. Sluz, D. W. Young, J. C. Juarez, A. Dwivedi, N. M. Demidovich, III, J. E. Graves, M. Northcott, J. Douglass, J. Phillips, D. Driver, A. McClarin, and D. Abelson. 80 Gb/s free-space optical communication demonstration between an aerostat and a ground terminal. In *Free-Space Laser Communications VI. Edited by Majumdar, Arun K.; Davis, Christopher C.. Proceedings of the SPIE, Volume 6304, pp. 630414 (2006).*, September 2006.

[7] M. Lloyd-Hart, P. Wizinowich, B. McLeod, D. Wittman, D. Colucci, R. Dekany, D. McCarthy, J. R. P. Anel, and D. Sandler. First results of an on-line adaptive optics system with atmospheric wavefront sensing by an artificial neural network. *APJL*, 390:L41–L44, May 1992.

[8] M. H. Cohen, M. A. Vorontsov, G. W. Carhart, and G. Cauwenberghs. Adaptive wavefront correction: a hybrid VLSI/optical system implementing parallel stochastic gradient descent. In A. Kohnle and J. D. Gonglewski, editors, *Proc. SPIE Vol. 3866, p. 176-182, Optics in Atmospheric Propagation and Adaptive Systems III, Anton Kohnle; John D. Gonglewski; Eds.*, pages 176–182, December 1999.

[9] G. D. Love, T. J. D. Oag, and A. K. Kirby. Common path interferometric wavefront sensor for extreme adaptive optics. *Optics Express*, 13:3491–+, May 2005.

[10] M. Watanabe, H. Takami, N. Takato, S. Colley, M. Eldred, T. Kane, O. Guyon, M. Hattori, M. Goto, M. Iye, Y. Hayano, Y. Kamata, N. Arimoto, N. Kobayashi, and Y. Minowa. Design of the Subaru laser guide star adaptive optics module. In D. Bonaccini Calia, B. L. Ellerbroek, and R. Ragazzoni, editors, *Advancements in Adaptive Optics. Edited by Domenico B. Calia, Brent L. Ellerbroek, and Roberto Ragazzoni. Proceedings of the SPIE, Volume 5490, pp. 1096-1104 (2004).*, pages 1096–1104, October 2004.

[11] R. J. Dorn, B. E. Burke, and J. W. Beletic. A CCD-Based Curvature Wavefront Sensor for Adaptive Optics in Astronomy. In P. Amico, J. W. Beletic, and J. E. Belectic, editors, *Scientific Detectors for Astronomy, The Beginning of a New Era*, pages 319–323, 2004.

[12] C. U. Keller, C. Plymate, and S. M. Ammons. Low-cost solar adaptive optics in the infrared. In S. L. Keil and S. V. Avakyan, editors, *Innovative Telescopes and Instrumentation for Solar Astrophysics. Edited by Stephen L. Keil, Sergey V. Avakyan . Proceedings of the SPIE, Volume 4853, pp. 351-359 (2003).*, pages 351–359, February 2003.

[13] C. Paterson, I. Munro, and J. C. Dainty. A low cost adaptive optics system using a membrane mirror. *Optics Express*, 6:175–+, April 2000.

[14] W. H. Southwell. Wave-front estimation from wave-front slope measurements. *Opt. Soc. Am.*, 70(8):998–1006, 1980.

[15] Y. Carmon and E. N. Ribak. Phase retrieval by demodulation of a Hartmann-Shack sensor. *Optics Communications*, 215:285–288, January 2003.

[16] R. Ragazzoni. Pupil plane wavefront sensing with an oscillating prism. *Journal of Modern Optics*, 43:289–293, February 1996.

[17] T. Y. Chew, R. M. Clare, and R. G. Lane. A comparison of the Shack-Hartmann and pyramid wavefront sensors. *Optics Communications*, 268:189–195, December 2006.

[18] I. Iglesias, R. Ragazzoni, Y. Julien, and P. Artal. Extended source pyramid wave-front sensor for the human eye. *Optics Express*, 10:419–+, May 2002.

[19] S. R. Chamot, C. Dainty, and S. Esposito. Adaptive optics for ophthalmic applications using a pyramid wavefront sensor. *Optics Express*, 14:518–526, January 2006.

[20] F. Roddier. Curvature sensing and compensation: a new concept in adaptive optics. *Applied Optics*, 27:1223–1225, 1988.

[21] C. R. Benn, M. Blanken, C. Bevil, S. Els, S. Goodsell, T. Gregory, P. Jolley, A. J. Longmore, O. Martin, R. M. Myers, R. Ostensen, S. Rees, R. G. M. Rutten, I. Soechting, G. Talbot, and S. M. Tulloch. NAOMI: adaptive optics at the WHT. In D. Bonaccini Calia, B. L. Ellerbroek, and R. Ragazzoni, editors, *Advancements in Adaptive Optics. Edited by Domenico B. Calia, Brent L. Ellerbroek, and Roberto Ragazzoni. Proceedings of the SPIE, Volume 5490, pp. 79-89 (2004).*, pages 79–89, October 2004.

[22] T. G. Bifano, P. A. Bierden, H. Zhu, S. Cornelissen, and J. H. Kim. Megapixel wavefront correctors. In D. Bonaccini Calia, B. L. Ellerbroek, and R. Ragazzoni, editors, *Advancements in Adaptive Optics. Edited by Domenico B. Calia, Brent L. Ellerbroek, and Roberto Ragazzoni. Proceedings of the SPIE, Volume 5490, pp. 1472-1481 (2004).*, pages 1472–1481, October 2004.

[23] S. Kendrew, P. Doel, D. Brooks, C. Dorn, C. Yates, R. M. Dwan, I. M. Richardson, and G. Evans. Development of a carbon fiber composite active mirror: design and testing. *Optical Engineering*, 45:3401–+, March 2006.

[24] S Hawkes, J Collier, C Hooker, C Reason, C Edwards, C Hernandez-Gomez, I Ross, and C Haefner. Adaptive optics trials on Vulcan. In *Central Laser Facility Anual Report, 2000 - 2001 , p. 153 - 155*, pages 155–153, 2001. Accessed from **www.clf.rl.ac.uk/reports/2000-2001/pdf/69.pdf** on Jan 15th 2007.

[25] F. Forbes, F. Roddier, G. Poczulp, C. Pinches, and G. Sweeny. Segmented bimorph deformable mirror. *Journal of Physics E Scientific Instruments*, 22:402–405, June 1989.

[26] R. Arsenault, R. Biasi, D. Gallieni, A. Riccardi, P. Lazzarini, N. Hubin, E. Fedrigo, R. Donaldson, S. Oberti, S. Stroebele, R. Conzelmann, and M. Duchateau. A deformable secondary mirror for the VLT. In *Advances in Adaptive Optics II. Edited by Ellerbroek, Brent L.; Bonaccini Calia, Domenico. Proceedings of the SPIE, Volume 6272, pp. (2006).*, July 2006.

[27] H. Hofer, L. Chen, G.-Y. Yoon, B. Singer, Y. Yamauchi, and D. R. Williams. Improvement in retinal image quality with dynamic correction of the eye's aberrations. *Optics Express*, 8:631–+, May 2001.

[28] A. Sivaramakrishnan and B. R. Oppenheimer. Deformable mirror calibration for adaptive optics systems. In D. Bonaccini and R. K. Tyson, editors, *Proc. SPIE Vol. 3353, p. 910-916, Adaptive Optical System Technologies, Domenico Bonaccini; Robert K. Tyson; Eds.*, pages 910–916, September 1998.

[29] R. V. Digumarthi and N. Mehta. Simplex optimization method for adaptive optics system alignment. In P. B. Ulrich and L. E. Wilson, editors, *Propagation of high-energy laser beams through the earth's atmosphere II; Proceedings of the Meeting, Los Angeles, CA, Jan. 21-23, 1991 (A93-24526 08-36), p. 136-147.*, pages 136–147, May 1991.

[30] R. M. Myers, A. J. Longmore, C. R. Benn, D. F. Buscher, P. Clark, N. A. Dipper, N. Doble, A. P. Doel, C. N. Dunlop, X. Gao, T. Gregory, R. A. Humphreys, D. J. Ives, R. Østensen, P. T. Peacocke, R. G. Rutten, C. J. Tierney, A. J. A. Vick, M. R. Wells, R. W. Wilson, S. P. Worswick, and A. Zadrozny. NAOMI adaptive optics system for the 4.2m William Herschel telescope. In P. L. Wizinowich and D. Bonaccini, editors, *Adaptive Optical System Technologies II. Edited by Wizinowich, Peter L.; Bonaccini, Domenico. Proceedings of the SPIE, Volume 4839, pp. 647-658 (2003).*, pages 647–658, February 2003.

[31] John W. Hardy. *Adaptive Optics for Astronomical Telescopes*, pages 18–20. Oxford University Press, first edition, 1998.

[32] A. Ghedina, W. Gaessler, M. Cecconi, R. Ragazzoni, A. T. Puglisi, and F. De Bonis. Latest developments on the loop control system of AdOpt@TNG. In D. Bonaccini Calia, B. L. Ellerbroek, and R. Ragazzoni, editors, *Advancements in Adaptive Optics. Edited by Domenico B. Calia, Brent L. Ellerbroek, and Roberto Ragazzoni. Proceedings of the SPIE, Volume 5490, pp. 1347-1355 (2004).*, pages 1347–1355, October 2004.

[33] Xilinx. Xc4000xla fpgas description. 2007. Accessed from **www.xilinx.com/bvdocs/publications/ds015.pdf** on March 3, 2007.

[34] T. S. Duncan, J. K. Voas, R. J. Eager, S. C. Newey, and J. L. Wynia. Low-latency adaptive optics system processing electronics. In P. L. Wizinowich and D. Bonaccini, editors, *Adaptive Optical System Technologies II. Edited by Wizinowich, Peter L.; Bonaccini, Domenico. Proceedings of the SPIE, Volume 4839, pp. 923-934 (2003).*, pages 923–934, February 2003.

[35] E. Fedrigo, R. Donaldson, C. Soenke, R. Myers, S. Goodsell, D. Geng, C. Saunter, and N. Dipper. SPARTA: the ESO standard platform for adaptive optics real time applications. In *Advances in Adaptive Optics II. Edited by Ellerbroek, Brent L.; Bonaccini Calia, Domenico. Proceedings of the SPIE, Volume 6272, pp. 627210 (2006).*, July 2006.

[36] S. J. Goodsell, E. Fedrigo, N. A. Dipper, R. Donaldson, D. Geng, R. M. Myers, C. D. Saunter, and C. Soenke. FPGA developments for the SPARTA project. In R. K. Tyson and M. Lloyd-Hart, editors, *Astronomical Adaptive Optics Systems and Applications II. Edited by Tyson, Robert K.; Lloyd-Hart, Michael. Proceedings of the SPIE, Volume 5903, pp. 148-159 (2005).*, pages 148–159, August 2005.

[37] A. G. Basden, F. Assémat, T. Butterley, D. Geng, C. D. Saunter, and R. W. Wilson. Acceleration of adaptive optics simulations using programmable logic. *MNRAS*, 364: 1413–1418, December 2005.

[38] L. F. Rodriguez-Ramos, T. Viera, J. V. Gigante, F. Gago, G. Herrera, A. Alonso, and N. Descharmes. FPGA adaptive optics system test bench. In R. K. Tyson and M. Lloyd-Hart, editors, *Astronomical Adaptive Optics Systems and Applications II. Edited by Tyson, Robert K.; Lloyd-Hart, Michael. Proceedings of the SPIE, Volume 5903, pp. 120-128 (2005).*, pages 120–128, August 2005.

[39] Xilinx. Prototyping and development boards : Ml401 overview. Accessed from **http://www.xilinx.com/products/boards/ml401/index.htm** on March 3, 2007.

[40] Xilinx. Spartan-3 fpga family: Complete data sheet. Accessed from **http://www.xilinx.com** on March 3, 2007.

[41] Xilinx. Accessed from **http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/capabilities/powerpc.htm** on March 3, 2007.

[42] Xilinx. Platform studio and the edk. Accessed from "**http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm**" on March 3, 2007.

[43] Xilinx. Microblaze soft processor core. Accessed from "**http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=micro_blaze**" on March 3, 2007.

[44] Xilinx. Ug129: Picoblaze 8-bit embedded microcontroller user guide. Accessed from **www.xilinx.com/bvdocs/userguides/ug129.pdf** on March 3, 2007.

[45] R. W. Wilson. SLODAR: measuring optical turbulence altitude with a Shack-Hartmann wavefront sensor. *MNRAS*, 337:103–108, November 2002.

[46] R. W. Wilson, J. Bate, J. C. Guerra, N. N. Hubin, M. Sarazin, and C. D. Saunter. Development of a portable SLODAR turbulence profiler. In D. Bonaccini Calia, B. L. Ellerbroek, and R. Ragazzoni, editors, *Advancements in Adaptive Optics. Edited by Domenico B. Calia, Brent L. Ellerbroek, and Roberto Ragazzoni. Proceedings of the SPIE, Volume 5490, pp. 758-765 (2004).*, pages 758–765, October 2004.

[47] J Collier, C Hooker, S Hawkes, and C Edwards. Adaptive optics for the petawatt upgrade. In *Central Laser Facility Anual Report, 2001 - 2002 , p. 181 - 182*, pages 181–182, 2001. Accessed from **www.clf.rl.ac.uk/Reports/2001-2002/pdf/86.pdf** on Jan 15th 2007.

[48] O. Soloviev and G. Vdovin. Hartmann-Shack test with random masks for modal wavefront reconstruction. *Optics Express*, 13:9570–+, November 2005.

[49] Wilson T. Neil M.A.A., Booth M.J. Dynamic wave front generation for the characterization and testing of optical systems. *Optics Letters*, 23(23):1849–1851, Dec 1998.

[50] KODAK. Device performance specification - kodak kac-9630 cmos image sensor. *web*, 2006. Accessed from **http://www.kodak.com/ezpres/business/ccd/global/plugins/acrobat/en/datasheet/cmos/KAC-9630LongSpec.pdp** on Jan 22nd 2007.

[51] KODAK. Webpage: Kodak image sensor solutions. *web*, 2005. Accessed from **http://www.kodak.com/US/en/dpq/site/SENSORS/name/ISSpr20050110_DigiKey** on Jan 4nd 2007.

[52] Digilent. Spartan-3 starter kit board user guide. Sourced from **http://www.digilent.com** on March 3, 2007.

[53] wxwidgets homepage. 2007. Accessed from **www.wxwidgets.org** on March 3, 2007.

[54] Physik Instrumente. S-325 high-speed piezo tip/tilt platform and z positioner. *web*, 2006. Accessed from **http://www.physikinstrumente.de/pdf/S325_Datasheet.pdf** on Jan 22nd 2007.

[55] A. Ashkin, J. M. Dziedzic, J. E. Bjorkholm, and S. Chu. Observation of a single-beam gradient force optical trap for dielectric particles. *Optics Letters*, 11:288–290, May 1986.

[56] Jens-Christian Meiners and Stephen R. Quake. Direct measurement of hydrodynamic cross correlations between two particles in an external potential. *Phys. Rev. Lett.*, 82 (10):2211–2214, Mar 1999.

[57] M. Polin, D. G. Grier, and S. R. Quake. Anomalous Vibrational Dispersion in Holographically Trapped Colloidal Arrays. *Physical Review Letters*, 96(8):088101–+, March 2006.

[58] B. C. Carter, G. T. Shubeita, and S. P. Gross. Tracking single-particles: a user-friendly quantitative evaluation. *Physical Biology*, 2:60–72, 2005.

[59] R. W. Gerchberg and W. O. Saxton. A practical algorithm for the determination of the phase from image and diffraction plane pictures. *Optik*, 35:237–246, 1972.

[60] R. Di Leonardo, F. Ianni, and G. Ruocco. Computer generation of optimal holograms for optical trap arrays. *Optics Express*, 15(4):1913–1922, 2007.

[61] K. Berg-Sørensen and H. Flyvbjerg. Power spectrum analysis for optical tweezers. *Review of Scientific Instruments*, 75:594–612, March 2004.