NGAO

Real Time Controller

Design Document

Revision History			
Version	Author	Date	Details
1.1	Marc Reinig and Donald Gavel	May 8, 2009	First draft version. Derived from MR's draft version 4.17
Ver.2.2.37	Marc Reinig and Donald Gavel	Dec 1, 2009	Updated Draft
Ver. 3.1.14	Marc Reinig and Donald Gavel	April 28, 2010	Updated Draft

Table of Contents

●

Table of Figures

•

Table of Tables

●

1. Introduction and Organization

This document covers the preliminary design of the Keck Next Generation Adaptive Optics Real Time Controller (NGAO RTC).

This section is an overall introduction and summary of the RTC architecture and concept for implementation. The section also describes the organization of the remainder of this document.

Section 2 describes the overall system characteristics.

Section 3 covers the latency and timing requirements that drive the architecture and implementation decisions.

Sections 4 through 12 cover the individual sub-systems in the RTC grand scope, including the Control Processor (CP), Wave Front Sensors (WFSs), Tomography Engine (TE), DM and Mirror Command Generators.

Subsequent sections are supplementary material covering broader RTC issues or related topics.

For details on the algorithms, see [1].

1.1 *Relationship to the Functional Requirements Document (FRD)*

Each section of this document describes the design of the hardware or software that is intended to satisfy one or more of the functional requirements set forth in the RTC FRD (KAON 573, superseded by the entries in the Contour requirements database).

Where appropriate, an annotation, "**FR-XXXXX**", is noted throughout this document that refers to a specific functional requirement in the FRD that is being addressed.

1.2 *Background and Context*

The design of the Real-Time Processor for the Keck Next Generation Adaptive Optics system represents a radically new computational framework that blazes new territory for astronomical AO systems. The need for a new approach is due to the considerable step up in the amounts of data to be processed and controls to be driven in a multiple guide star tomography system. Furthermore, KNGAO has set ambitious new goals for wavefront correction performance that puts further pressure on processing rates and latency times that traditional compute paradigms have trouble scaling up to. Current processing requirements are in the Terra Operations per second region. Instead, the KNGAO RTC is structured around a massively parallel processing (MPP) concept, where the highly parallelizable aspects of the real-time tomography algorithm are directly mapped to a large number of hardware compute elements. These compute elements operate separately and simultaneously. With this approach, increasing problem size is handled roughly by increasing the number of processor elements, rather than processor speed.

In short, in this design, the structure of the algorithm drives the structure of the hardware. Therefore, it is necessary to understand the algorithm itself to understand the hardware architecture, so please review the companion volume, see Gavel, D., NGAO Real Time Control Algorithms Document, KAON xxx, 2009, which presents the technical details of the massively parallel algorithm for AO tomography.

For the purpose of introducing the hardware design, the algorithm and hardware have three main top-level component steps: wavefront sensing, tomography, and DM control. These physically map to component blocks of hardware, see Figure 1. The actual physical layout of boards and connecting cables mimic this geometry. Inside the blocks, further parallelization is achieved by use of Fourier domain processing, where each spatial frequency component is processed independently. Fourier-domain algorithms have been developed in recent years for wavefront phase reconstruction [2] and for tomography [3].

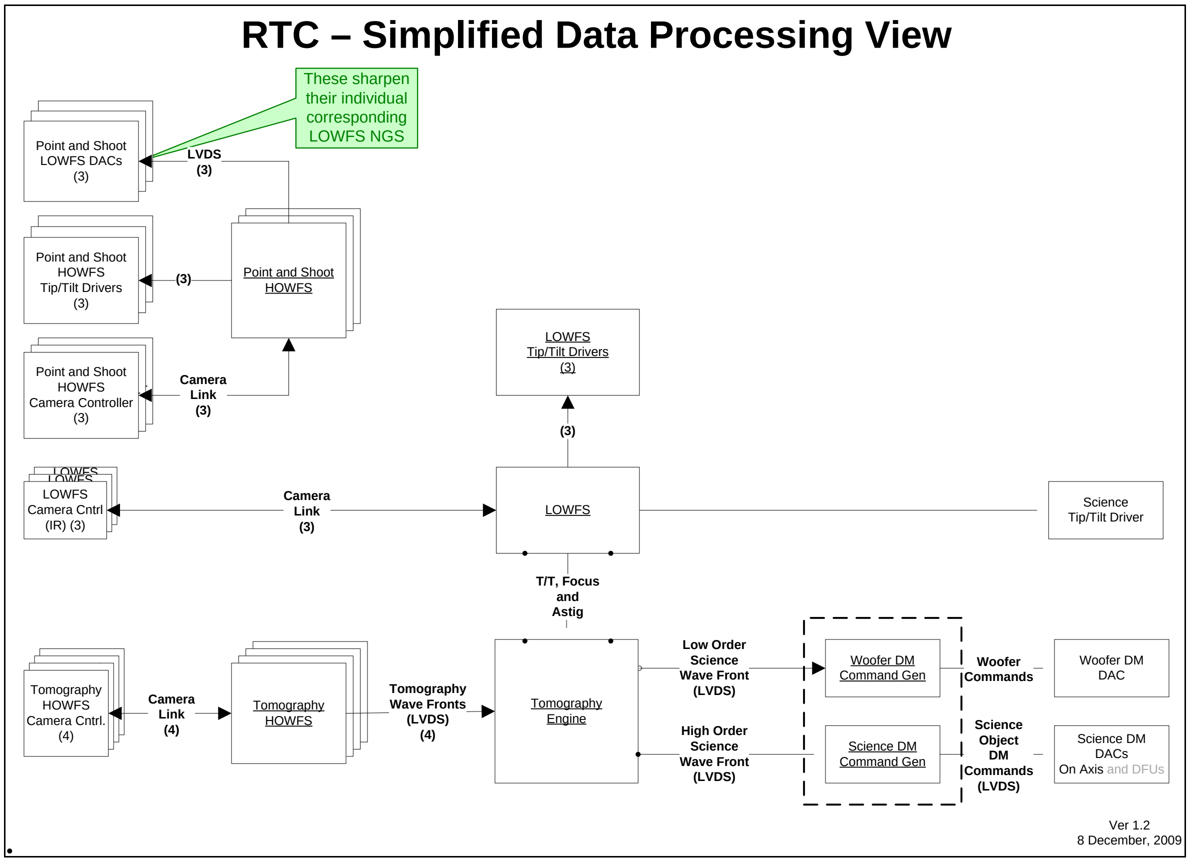

RTC – Simplified Data Processing View
These sharpen their individual corresponding LOWFS NGS
Point and Shoot LOWFS DACs (3)
LVDS (3)
Point and Shoot HOWFS Tip/Tilt Drivers (3)
(3)
Point and Shoot HOWFS
Point and Shoot HOWFS Camera Controller (3)
Camera Link (3)
LOWFS Tip/Tilt Drivers (3)
(3)
LOWFS Camera Cntrl (IR) (3)
Camera Link (3)
LOWFS
Science Tip/Tilt Driver
T/T, Focus and Astig
Tomography HOWFS Camera Cntrl. (4)
Camera Link (4)
Tomography HOWFS
Tomography Wave Fronts (LVDS) (4)
Tomography Engine
Low Order Science Wave Front (LVDS)
High Order Science Wave Front (LVDS)
Woofer DM Command Gen
Science DM Command Gen
Woofer Commands
Science Object DM Commands (LVDS)
Woofer DM DAC
Science DM DACs On Axis and DFUs
Ver 1.2
8 December, 2009

Figure 1 Simplified View of the RTC

As stated, the real-time processing takes place on small compute elements, either Field-Programmable Gate Arrays (FPGAs, Graphical Processing Units (GPUs), or multiple processors and cores.

FPGA devices have been used in the digital electronics industry since the 1980's, when they were first used as alternatives to custom fabricated semiconductors. The arrays of raw unconnected logic were "field-programmed" to produce any digital logic function. They grew in a role as support chips on board computer motherboards and plug-in cards. Recent versions incorporate millions of transistors and some also incorporate hundreds to thousands of on-chip conventional arithmetic processors. Their use is now common in many products from high performance scientific add in boards to consumer portable electronics including cell phones. Today these chips represent a fast growing, multi-billion dollar industry.

Graphical Processing Units were first introduced to offload the video processing for computer displays from the computer's CPU. They have since developed as a key processor for digital displays, which have demanding applications in computer-aided design and video gaming. GPUs specialize in geometric rotations, translations, and interpolation using massive vector-matrix manipulations. Beyond graphics, GPUs are also used in analyzing huge geophysical signal data sets in oil and mineral exploration and other scientific calculations requiring extremely high performance. It is possible that GPUs can be adapted to the wavefront reconstruction problem through these means and, while they remain a possible low-cost option for the front-end video processing, our baseline design uses FPGA processors for this purpose.

The RTC uses a mix of CPUs, GPUs, and FPGAs to best match their individual strengths to the RTC's needs.

1.3 *Design Philosophy*

For efficiency, we are maximizing the use of existing designs and code. The WFS and the DM drivers are based on the heritage of the Villages and GPI adaptive optics systems for wavefront reconstruction and open loop DM control.

To insure the maximum reliability, serviceability, and maintainability, we have maximized the use of commodity parts and industry standard interfaces.

1.4 *Functions of the RTC System (FR-1401)*

1. Measure the atmosphere using 7 laser guide stars (3 point-and-shoot and 4 tomography) and 3 natural guide stars, producing, respectively, high-order wave fronts and low-order wavefronts (tip/tilt, focus, and astigmatism values) based on these measurements. The 3 point-and-shoot guide stars are used only to sharpen the natural guide stars used for T/T, focus and astigmatism). The 4 tomography lasers are used to probe the atmosphere for the tomography engine.
2. Using the measurements from the four tomography WFS, estimate the three dimensional volume of atmospheric turbulence-induced index-of-refraction variations over the telescope (tomographic reconstruction).
3. The point-and-shoot high order wave front sensors (HOWFS) determine the shapes of the multiple deformable mirrors' (DMs) surfaces that will correct for the turbulence along paths to their natural guide stars at their correspondingly assigned positions on the sky. All WFS provide T/T information for its own tip/tilt mirror.
4. Using the desired surface shape, generate the correct voltage commands for each DM and tip/tilt mirror actuator, compensating for any non linearity or influence functions.

5. Capture any diagnostic and telemetry data and send it to the AO Supervisory Control system or the RTC Disk Sub-System as required.
6. The RTC interacts with the telescope system and the AO Control through two Ethernet connections: one connected to the RTC Control Processor (CP) and one connected to the RTC Disk Sub-system.

Additionally the functionality provided will be used to compensate for vibrations in the telescope system.

1.5 *Algorithm*

The mathematical derivation of each component algorithm of the RTC is described in “Gavel, D., NGAO Real Time Control Algorithms Document, KAON xxx, 2009”. The component algorithms consist of:

- Wavefront sensor image processing
- Hartmann sensor wavefront reconstruction – convert slopes to phases
- Inverse Tomography
- Conversion of desired phases to DM and TTM drive voltages
- Offloading

Details are given in **KAON xxx**, but we will briefly go over two key algorithmic steps, wavefront sensing and inverse tomography, here and describe how they map to basic elemental operations in the RTC.

1.5.1 Wavefront Reconstruction Algorithm

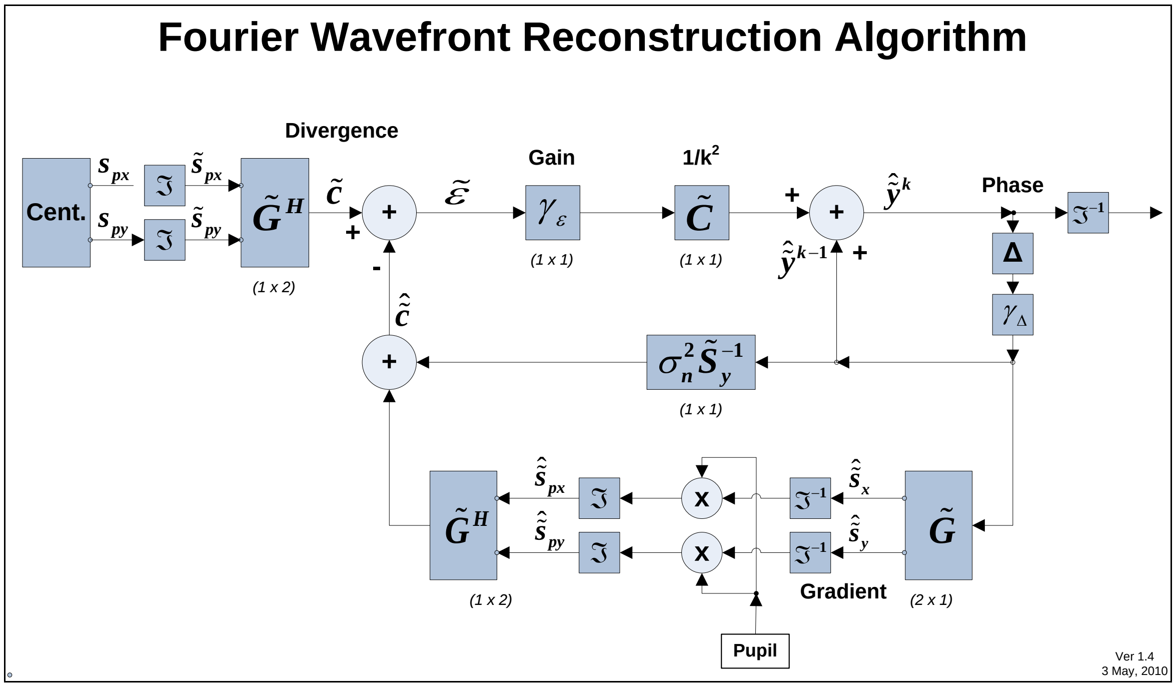


Figure 2 Wave Front Reconstruction Algorithm for HOWFS and NGS

The purpose of the wavefront reconstructor is to convert Hartmann slope data given on the aperture to an estimate of the wavefront phase. The following series of steps are performed for each wavefront sensor in the system:

- The wavefront slope data are compared to the gradient of the current estimate of the wavefront.
- The residual is converted to the Fourier domain where the approximate inverse to the gradient operator is applied.
- For the next iteration of the loop, it is important to compare slopes only within the aperture, where valid measurements exist. This is done by converting back to the spatial domain and selecting, via a pupil mask, the estimate gradients from within the aperture.
- The resulting estimate of wavefront phase is available for the next stage of the process, inverse tomography.

1.5.2 Inverse Tomography

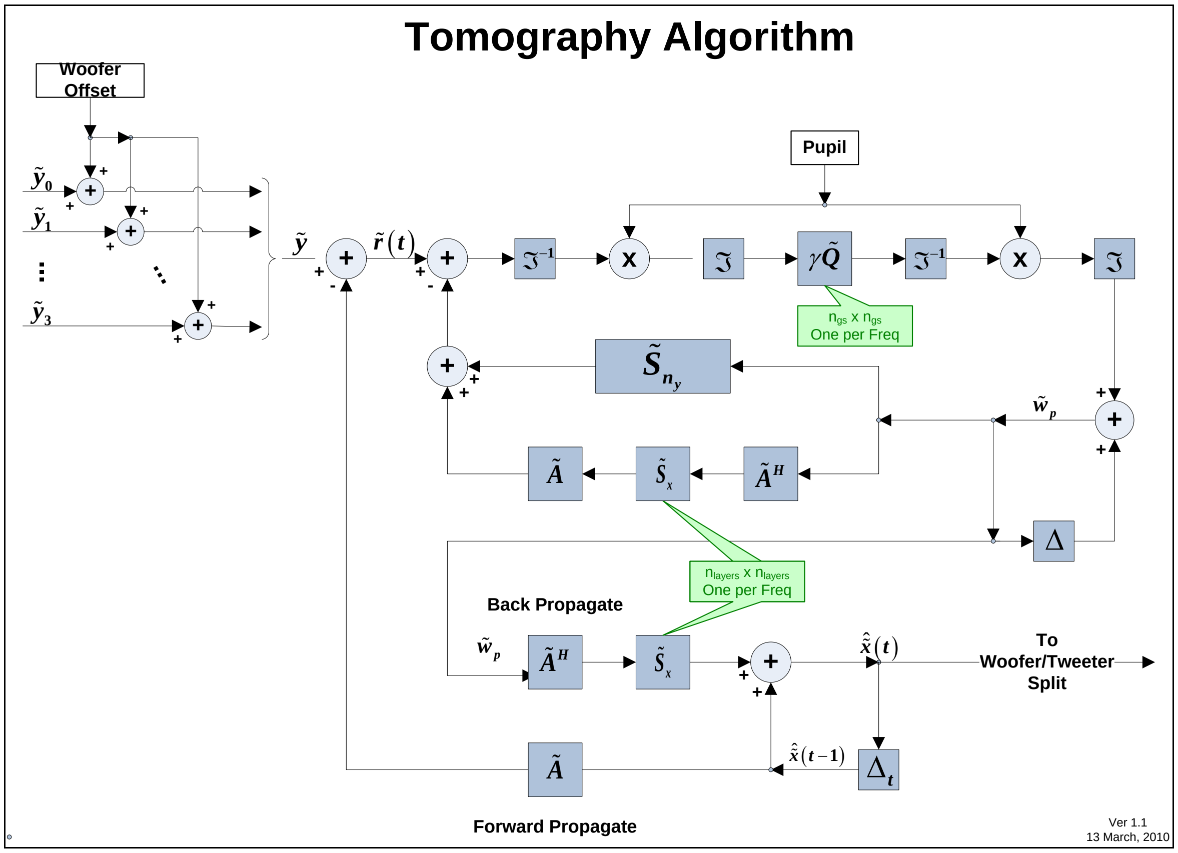


Figure 3 Tomography Algorithm

Inverse tomography consists of this sequence of operations:

- Gather the reconstructed wavefronts from each high-order wavefront sensor (these are already in the Fourier domain)
- Compare the wavefront phases to the prior estimate of what the wavefront phases would be if determined from the modeled atmospheric volume
- With this residual, perform an iteration that determines the preconditioned wavefront phase (see Algorithm Design Document, section 2.2). This step involves converting into and out of the spatial domain and applying the aperture data selection mask (labeled "Pupil" in the diagram).
- Back-propagate the preconditioned wavefront phase residuals and co-add to the volume delta-index estimates.
- The next step is to determine what wavefront is to be placed on the science deformable mirror, given the volume delta-index estimates. This is done by a forward propagating the volume delta-index estimates to the ground, summing all the layers in a mimic of ray tracing.

All the algorithms that need to be performed in real time, including the key functions described above, can be broken down into a set of basic elemental operations that must be performed by the RTC high-speed processor. The elemental operational elements are:

- 2-D Fourier Transform
- Element-wise multiply-accumulate (includes spectral domain filtering)
- Interpolation – along transverse directions in the layer plane
- Matrix multiplication – of small arrays: #guidestars X #guidestars
- Propagation – ray-tracing between layers
- Non-linearity compensation by lookup tables

We have explored the implementation of each of these elemental steps in both FPGA and GPU processor architectures. Comparison results for the Fourier-transform pair, the single most time-consuming computation, is shown in and Appendix B.

1.5.3 Massive Parallelization

Each of the elemental operations involves processing moderate amounts of data but they need to be done at extremely high speed. Much of the operation on these datasets can proceed in parallel, achieving a speed-up benefit that becomes essential to meeting the bandwidth and latency requirements of NGAO. By assigning small compute elements assigned to each spatial domain sample, or to each spatial frequency sample, all of the elemental operations, with the exception of the Fourier Transform, are on the order of *N* times faster than they would be in a corresponding conventional single CPU implementation, where *N* is the number of sample points (*N*~4000 for NGAO).

The Fourier transform is sped up by factor on the order *log*(*N*), using the algorithm described in Appendix G.

1.6 *Functional Architecture*

The NGAO RTC is not a single computer or controller. It consists of many different computers and processors (about 10) each performing different functions simultaneously, but with their inputs and outputs synchronized to perform the overall task of correcting the science wave front.

Figure 1 is a top level view of the overall system and its relationship to the AO Control. Figure 4 depicts the overall design, emphasizing real-time signal flow.

Figure 47 highlights the flow of telemetry, diagnostic, and offload-data.

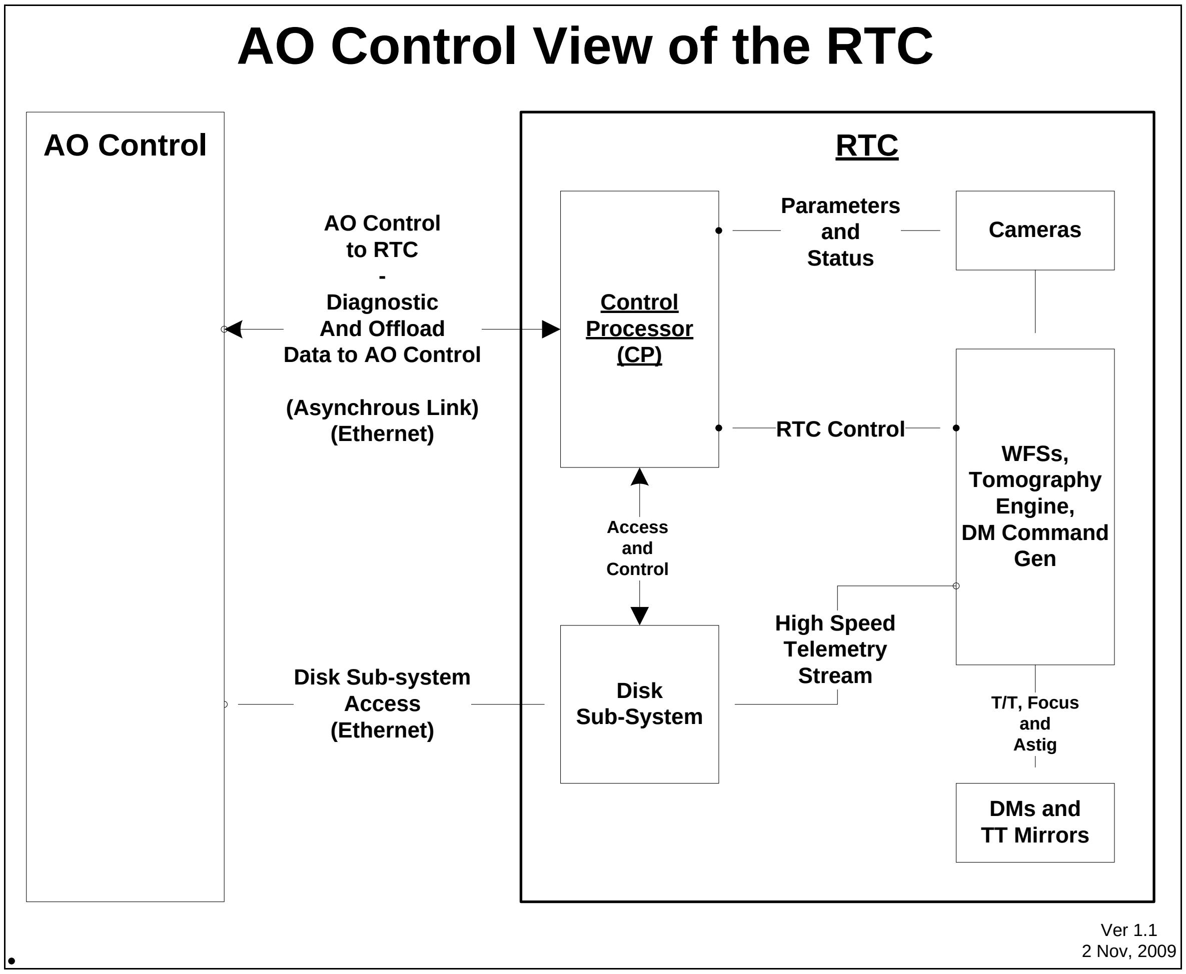


Figure 4 Top-level view of the AO Control's View of the RTC. See Figure 8 for a detailed view.

The RTC System is a real-time digital control system that operates on a fixed clock cycle, referred to as the Frame Clock. Each frame, the RTC takes data from various cameras, processes it, and updates the various mirrors and DMs for which it is responsible.

It runs the same program on each frame, with different data, but the same parameters, repeatedly, until the AO Control causes it to change.

The Control Processor (CP) acts as the synchronizing interface between the AO Control and the rest of the RTC, which is frame-clock driven. The AO Control processor will be implemented with a multi-processor computer running the Linux operating system with a real-time kernel. Synchronization with the frame-clock driven portion of the RTC is described in Sections 4 and 12.

The parameter loading determines the "state" of the RTC. This loading occurs in the background to the real-time cycles. Parameter updates occur synchronized to the frame clock. This insures that all information processed during a frame is processed with the same parameters.

1.6.1 Control Processor (CP)

The RTC Control Processor (CP) provides an asynchronous interface between the RTC system the AO Control. The interface to the AO Control is through a socket interface over Ethernet (See Figure 4 and Figure 18).

It is built on a high performance Linux server with a real-time kernel.

The RTC system is a key component to the overall NGAO system and the rest of the telescope environment. The CP's interface with the AO Supervisory Control and the telescope system will adhere to the component architecture defined in **FR-1418**.

1.6.2 Wave Front Sensors (WFS)

A short overview: There are two types of WFS in the NGAO RTC: low order (LOWFS) and high order (HOWFS).

1.6.2.1 LOWFS

There are 3 LOWFS, which use a IR natural guide stars (NGS) to estimate tip/tilt. One, the TTFA, also captures data for focus and astigmatism analysis.

A low-order wavefront sensor (LOWFS) processor is responsible for converting raw camera pixel data into a tip/tilt signal, plus focus and astigmatism numbers for the one TTFA sensor. For algorithm details, see Gavel, D., NGAO Real Time Control Algorithms Document, **KAON xxx**, 2009. Each LOWFS camera operates in parallel and feeds its data to the LOWFS. The LOWFS processes the data and sends commands to the science tip/tilt actuator on the Woofer and to the tomography engine.

1.6.2.2 HOWFS

There are two types of HOWFS: the Point-and-Shoot HOWFS are used to sharpen the NGS tip/tilt stars and are 32x32 sub-apertures in size; the Tomography HOWFS provide information to perform tomography in the science field of view and are 64x64 sub-apertures in size.

In the NGS mode, one of the Tomography HOWFSs serves as the WFS for the NGS and operates using 32x32 subapertures instead of 64x64. In this mode the other Tomography HOWFSs are not used. The output of the NGS/Tomography HOWFS is passed through the tomography engine unmodified and directly to the science object DM command generator.

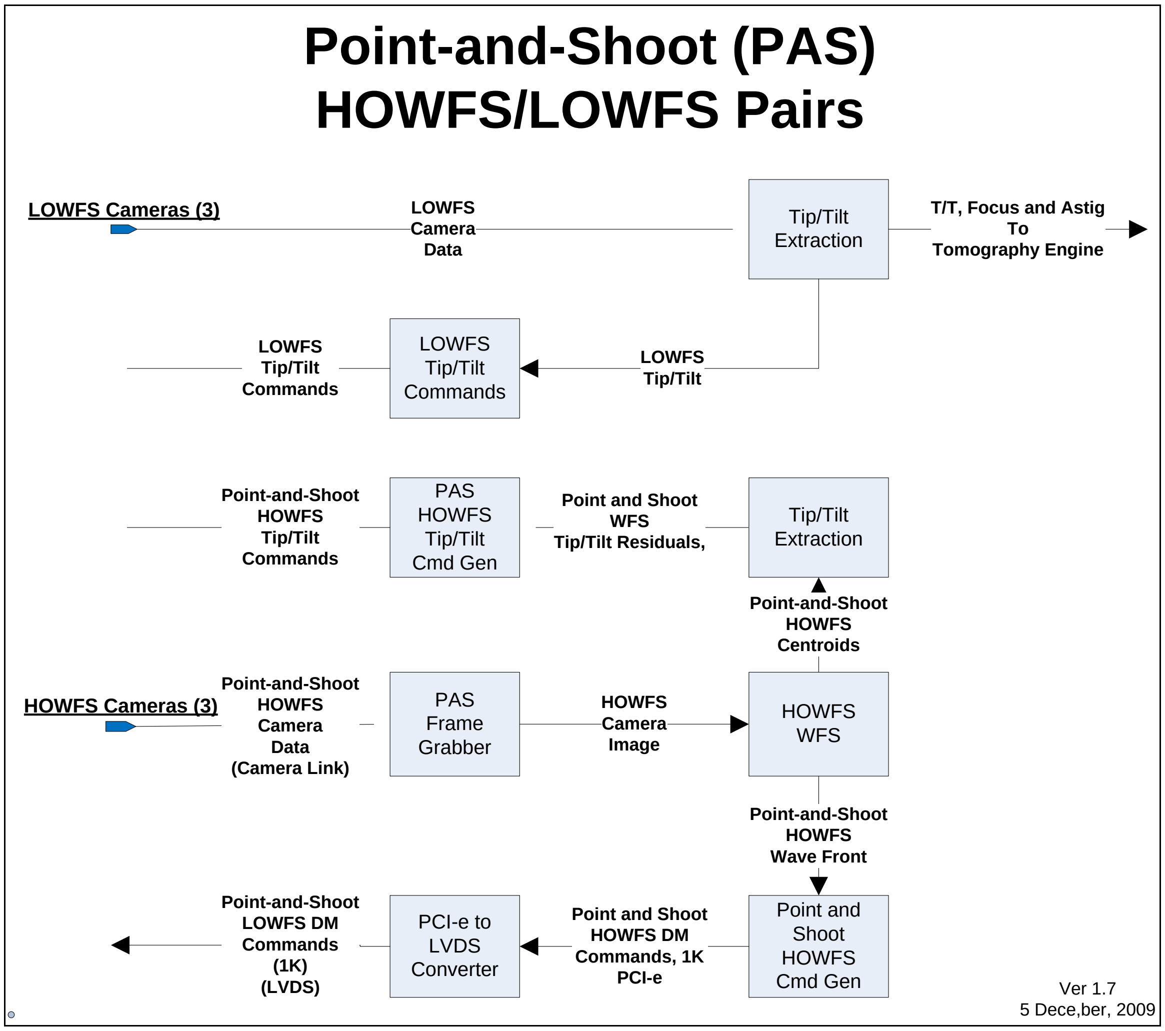

Point-and-Shoot (PAS)
HOWFS/LOWFS Pairs
LOWFS Cameras (3)
LOWFS Camera Data
Tip/Tilt Extraction
T/T, Focus and Astig To Tomography Engine
LOWFS Tip/Tilt Commands
LOWFS Tip/Tilt Commands
LOWFS Tip/Tilt
Point-and-Shoot HOWFS Tip/Tilt Commands
PAS HOWFS Tip/Tilt Cmd Gen
Point and Shoot WFS Tip/Tilt Residuals,
Tip/Tilt Extraction
Point-and-Shoot HOWFS Centroids
HOWFS Cameras (3)
Point-and-Shoot HOWFS Camera Data (Camera Link)
PAS Frame Grabber
HOWFS Camera Image
HOWFS WFS
Point-and-Shoot HOWFS Wave Front
Point-and-Shoot LOWFS DM Commands (1K) (LVDS)
PCI-e to LVDS Converter
Point and Shoot HOWFS DM Commands, 1K PCI-e
Point and Shoot HOWFS Cmd Gen
Ver 1.7
5 Dece,ber, 2009

Figure 5 Point-and-Shoot HOWFS/LOWFS Pair

A high-order wavefront sensor processor is responsible for converting raw Hartmann WFS camera pixel data into an array of wavefront phase measurements spaced on a regular grid in a coordinate system that defined as common throughout the AO system. HOWFS use a laser guide star (LGS) to probe the atmosphere and determine the high order atmospheric distortion in that direction.

The Point-and-shoot HOWFS first generates a wave front to correct light in the direction of the LGS. This wave front is then converted to DM commands that are used to determine the shape of a DM. This DM is used to sharpen their associated LOWFS NGS.

The tomography HOWFSs provide probing of the atmospheric volume containing the science object and generate a wave front in the direction of the LGS. These wave fronts are processed by the tomography engine and used to estimate the volume in the direction of the science object. The estimate of the volume is in turn used to provide a wave front to correct light in that direction. This wave front is passed to a DM command generator to place the desired shape on the science DM.

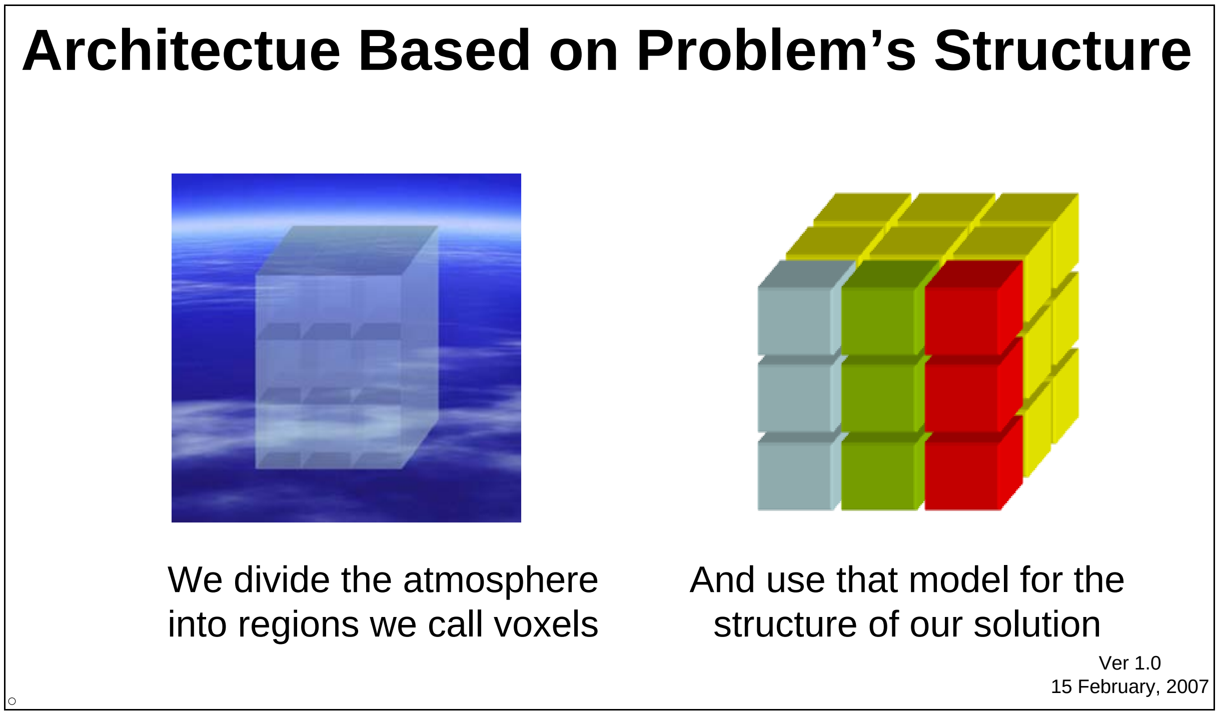
Each tomography HOWFS wavefront sensor has an associated wavefront sensor processor. These operate in parallel then feed their aggregate results to the tomography engine.

The architecture for the hardware for all WFSs follows the massively parallel approach. First, each wavefront sensor has an associated wave front sensor processor card, operating in parallel with but otherwise independent of (other than the frame synchronization clock) every other processor card.

The baseline design for a WFS processors card uses FPGAs. We have investigated the use of GPUs as an option, and they look to be a promising alternative, however for the remainder of this document, and the description of the baseline architecture and processing, we assume these are FPGA based cards.

1.6.3 **Tomography Engine**

The tomography engine's Processing Elements are mapped "horizontally" over the aperture, and "vertically" over the layers of the model atmosphere. A given processing element (PE) on this map is assigned a piece of the aperture and alternates processing a portion of either the spatial or Fourier domain of it.



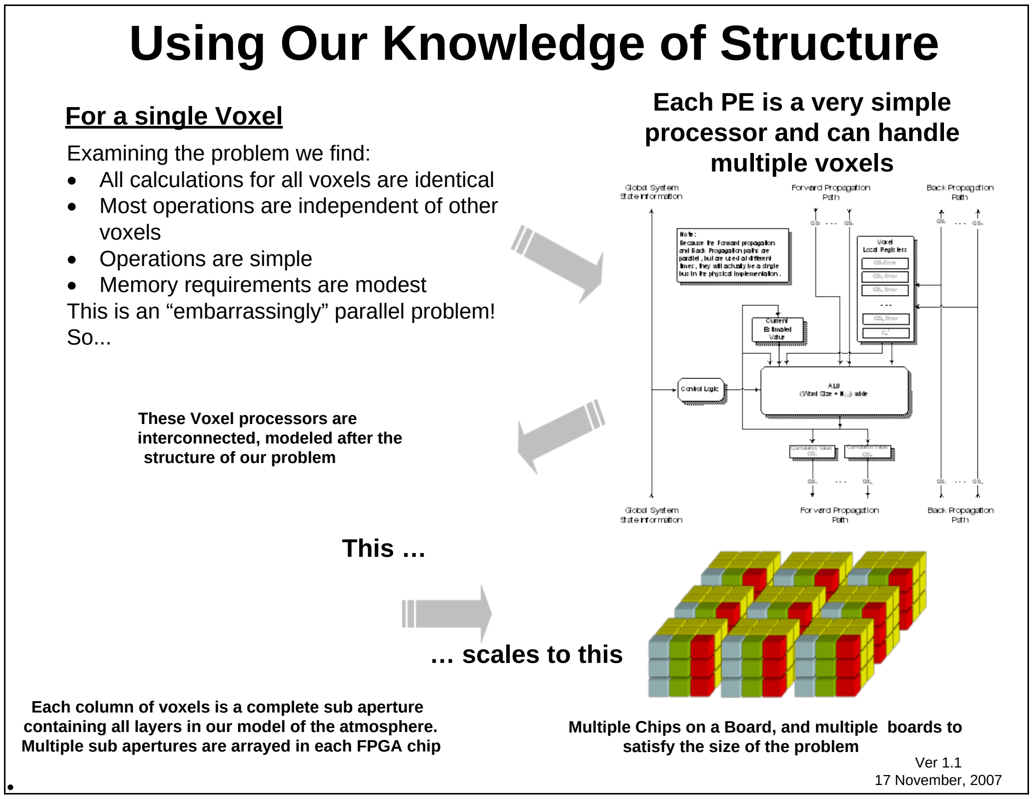
Architectue Based on Problem's Structure

We divide the atmosphere into regions we call voxels

And use that model for the structure of our solution

Ver 1.0
15 February, 2007

Figure 6 Processing Elements Mapping to the Atmospheric Model



Using Our Knowledge of Structure

For a single Voxel

Examining the problem we find:
- All calculations for all voxels are identical
- Most operations are independent of other voxels
- Operations are simple
- Memory requirements are modest

This is an “embarrassingly” parallel problem!
So...

These Voxel processors are interconnected, modeled after the structure of our problem

Each PE is a very simple processor and can handle multiple voxels

This …

… scales to this

Each column of voxels is a complete sub aperture containing all layers in our model of the atmosphere. Multiple sub apertures are arrayed in each FPGA chip

Multiple Chips on a Board, and multiple boards to satisfy the size of the problem

Ver 1.1
17 November, 2007

Global System State Information
Forward Propagation Path
Back Propagation Path
Note: Because the Forward propagation and Back Propagation paths are parallel, but are used at different times, they will actually be a single bus in the physical implementation.
Voxel Local Registers
Current Estimated Value
Control Logic
ALU (Word Size + N_GS) wide
Cumulative Value GS_1
Cumulative Value GS_n

Figure 7 Mapping Processing Elements to the Problem

All computational elements run the identical program, albeit with different parameters and data. Each processor in the tomography engine is connected to its four neighboring elements (representing the next spatial frequency over in both directions) because is it necessary to shift data to neighbors in order to implement the Fourier transform and interpolation steps in the tomography algorithm.

1.6.4 DM Command Generators

The Point-and-Shoot HOWFS’s output and the tomography engine’s output wavefronts are sent to the deformable mirror command generators for their DMs. These are dedicated units, one per DM. They take desired wavefronts in and produce DM command voltages that result in the correct wavefront, taking into account actuator influence functions and nonlinearities of each DM, see Figure 1.

1.6.5 Disk Sub-System

The RTC Disk Sub-System (see Figure 4) is an extremely high performance, large storage system that is designed to capture system Telemetry data that can be generated by the RTC. This data can be generated at over 2 GBytes/sec. This system is designed to be able to capture this data for an extended period, but it is not intended to be an archival system for long-term storage or access. The system has a capacity to store up to 60 terra bytes of data, which could be filled in a matter of days of heavy use.

Likewise, no data base facilities are provided beyond normal directory services.

1.6.6 Global RTC System Timing

Accurate time stamping of data will be provided (to 0.1 msec) for the telemetry stream. This will provide timing information for the various telemetry and other data products produced by the RTC. It is important that any science data be synchronized to this time stamp, see Appendix C.

1.6.7 Diagnostics and Telemetry Stream

All major data signals can be sent to the Telemetry/Diagnostic stream at the request of the AO Control. Any data in this stream can be sent to either-or-both the AO Control, for diagnostics, or the RTC disk sub-system, for storage. (See: Section 9)

Data that can be saved through Telemetry or viewed through Diagnostics include (See Section 9):

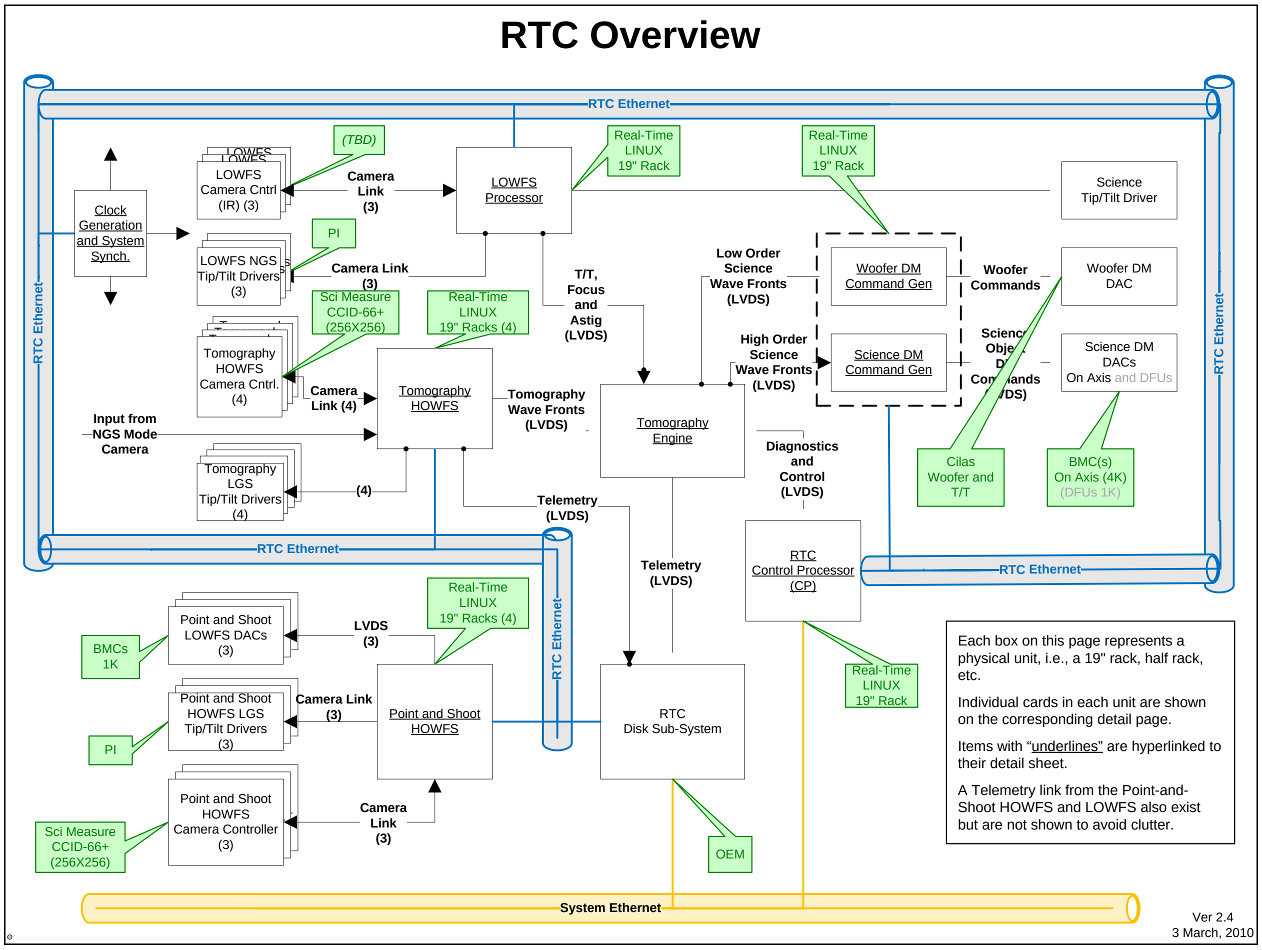
1. Raw Camera Data for HOWFS and LOWFS
2. HOWFS Centroids
3. HOWFS Wave Fronts
4. Tip/Tilt for all WFS + Focus and Astigmatism for the TTFA LOWFS
5. Science on-axis T/T
6. Tomographic Layers
7. Science on-axis unfiltered Wave Front
8. Science on-axis High Order Wave Front
9. Science on-axis Woofer Wave Front
10. Science on-axis High Order Commands
11. Science on-axis Woofer Commands
12. RTC Current Parameter Settings

All Diagnostic information is time stamped accurate to one Frame Time.

1.7 *Physical Architecture*

The RTC consists of over 15 computers, require several cubic meters of space, and dissipate over 5KW. To minimize the heat and space requirements on the Nasmyth platform, only the equipment which requires proximity to the AO optics are placed on the Nasmyth. The rest are housed in the computer room below the telescope.

Figure 8 shows the RTC and all sub-systems. The RTC is physically divided between the Nasmyth and the computer room below the telescope. The cameras, their controllers, the DMs, and their controllers are located on the Nasmyth, see Figure 9. The rest of the RTC, the WFS, tomography engine, disk sub-system, DM command generators, control processor, and clock generator are located in the computer room. The two sections are connected by high-speed fiber links.


RTC Overview
RTC Ethernet
(TBD)
Real-Time LINUX 19" Rack
Real-Time LINUX 19" Rack
LOWFS Camera Cntrl (IR) (3)
Camera Link (3)
LOWFS Processor
Science Tip/Tilt Driver
Clock Generation and System Synch.
PI
LOWFS NGS Tip/Tilt Drivers (3)
Camera Link (3)
T/T, Focus and Astig (LVDS)
Low Order Science Wave Fronts (LVDS)
Woofer DM Command Gen
Woofer Commands
Woofer DM DAC
Sci Measure CCID-66+ (256X256)
Real-Time LINUX 19" Racks (4)
High Order Science Wave Fronts (LVDS)
Science DM Command Gen
Science Object DM Commands (LVDS)
Science DM DACs On Axis and DFUs
Tomography HOWFS Camera Cntrl. (4)
Camera Link (4)
Tomography HOWFS
Tomography Wave Fronts (LVDS)
Tomography Engine
Input from NGS Mode Camera
Diagnostics and Control (LVDS)
Cilas Woofer and T/T
BMC(s) On Axis (4K) (DFUs 1K)
Tomography LGS Tip/Tilt Drivers (4)
(4)
Telemetry (LVDS)
RTC Ethernet
Telemetry (LVDS)
RTC Control Processor (CP)
RTC Ethernet
RTC Ethernet
Real-Time LINUX 19" Racks (4)
Point and Shoot LOWFS DACs (3)
LVDS (3)
BMCs 1K
Real-Time LINUX 19" Rack
Point and Shoot HOWFS LGS Tip/Tilt Drivers (3)
Camera Link (3)
Point and Shoot HOWFS
RTC Disk Sub-System
PI
Point and Shoot HOWFS Camera Controller (3)
Camera Link (3)
Sci Measure CCID-66+ (256X256)
OEM
System Ethernet
Ver 2.4 3 March, 2010

Each box on this page represents a physical unit, i.e., a 19" rack, half rack, etc.

Individual cards in each unit are shown on the corresponding detail page.

Items with "underlines" are hyperlinked to their detail sheet.

A Telemetry link from the Point-and-Shoot HOWFS and LOWFS also exist but are not shown to avoid clutter.

Figure 8 Overview of the RTC sub-systems

Note: Since this is an overview, not all interfaces and data paths are shown.

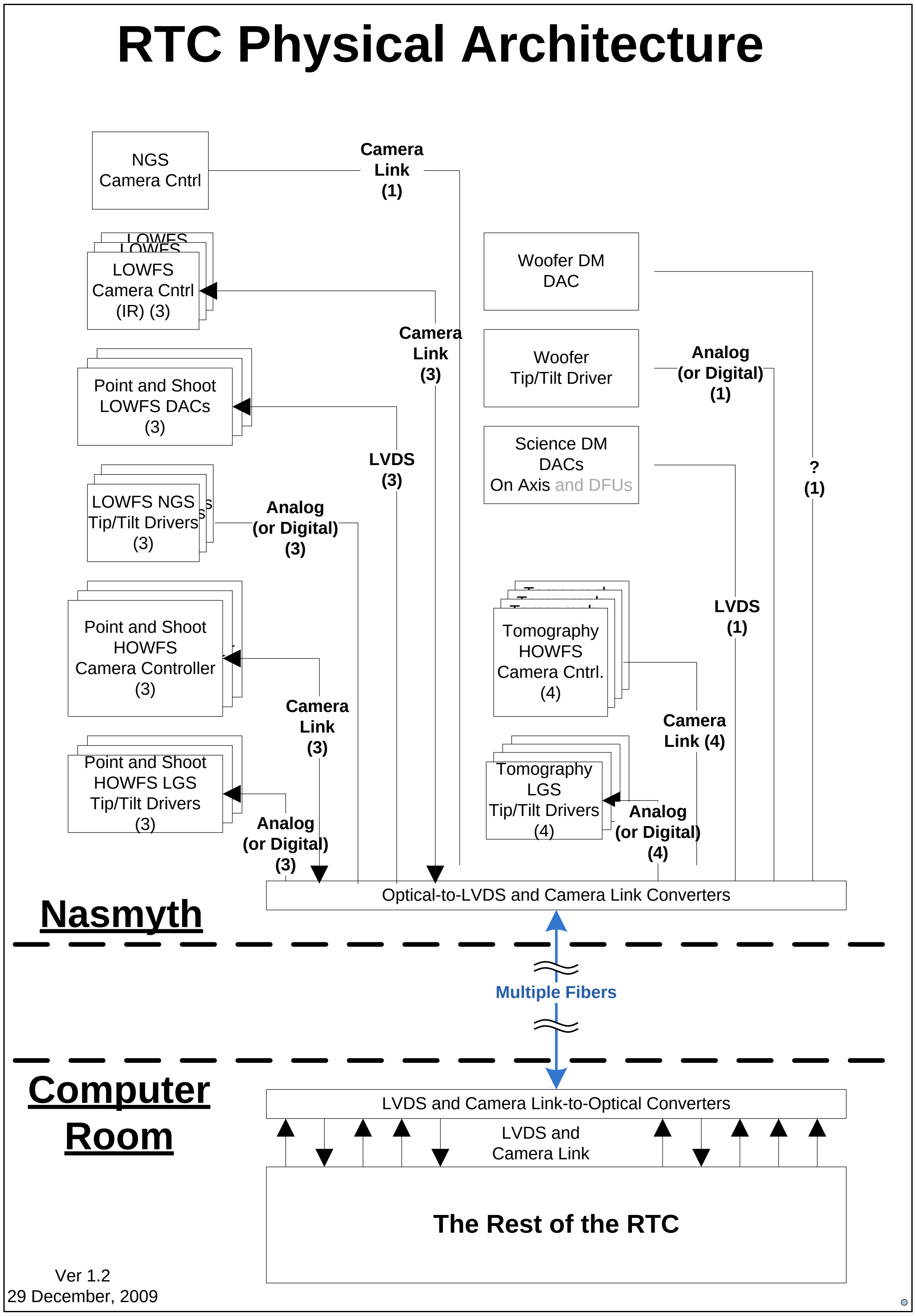

RTC Physical Architecture
NGS Camera Cntrl
Camera Link (1)
LOWFS Camera Cntrl (IR) (3)
Woofer DM DAC
Camera Link (3)
Woofer Tip/Tilt Driver
Analog (or Digital) (1)
Point and Shoot LOWFS DACs (3)
LVDS (3)
Science DM DACs On Axis and DFUs
? (1)
LOWFS NGS Tip/Tilt Drivers (3)
Analog (or Digital) (3)
LVDS (1)
Point and Shoot HOWFS Camera Controller (3)
Tomography HOWFS Camera Cntrl. (4)
Camera Link (3)
Camera Link (4)
Point and Shoot HOWFS LGS Tip/Tilt Drivers (3)
Tomography LGS Tip/Tilt Drivers (4)
Analog (or Digital) (3)
Analog (or Digital) (4)
Optical-to-LVDS and Camera Link Converters
Nasmyth
Multiple Fibers
Computer Room
LVDS and Camera Link-to-Optical Converters
LVDS and Camera Link
The Rest of the RTC
Ver 1.2
29 December, 2009

Figure 9 RTC Physical architecture, showing the divide between the Nasmyth and the computer room

Include NGS

1.8 *Implementation Alternatives*

As part of the design process, we have investigated a number of alternative architectures and hardware implementation alternatives. For the sake of the design, we are recommending that FPGA boards be used for wavefront sensing and tomography in the frame-clock driven portion of the RTC. The decision for this baseline is based on one of conservativeness: the approach is well characterized and has been demonstrated in low-level simulators, the timing requirements are clearly met, and the process of specifying and building these boards is a clearly established technology within the industry. The drawbacks are the additional cost of custom board design and sparing, so for this reason we have investigated the use of commercial off-the-shelf alternatives: multi-core CPUs and GPUs. We describe our findings below.

1.8.1 Multi-core CPUs

Multi-core CPUs are the traditional method of implementation. The support parallelization and have very high clock rates. However, maximum parallelization is limited to the number of cores per CPU and the number of CPUs.

1.8.2 FPGAs

FPGAs are containers of a very large amount of unconfigured logic (millions of logic gates) that can be personalized to the needs of the users. This gives extreme flexibility to design logic that is not available as a commercial off the shelf product.

In our application of FPGAs to the tomography engine, we are able to create many 100's of simple interconnected processors per chip to execute the tomography algorithm.

1.8.3 GPUs

GPUs are extremely efficient at operations that can be parallized and stay within the GPU itself. Since they are co-processors processors to the CPUs in a system, care must be taken to synchronize the two and minimize transfer times, see Appendix B.

1.8.4 Recommended implementations

We have examined different candidate technology implementations for the key stages of the RTC. Here are our findings

- Each WFS reconstructor can be implemented using one conventional multi-core multi-CPU solution (for example the GPI processor, which is uses 4 quad-core Xeon processors, one per WFS), or one custom FPGA card, or one advanced-technology GPU card (potentially). The GPU driver would need a custom driver to allow direct memory access for loading of camera data, something that present GPUs don't offer. Although we are continuing to hold the alternatives as potential cost-savers, the baseline WFS reconstructor is implemented on an FPGA based card.
- The Tomography Engine can only meet requirements if it is implanted with a multiple card system based on FPGAs. The NGAO requirements are met with a 9-FPGA systolic array arrangement, based on the XILINX Vertex-6 chip as a baseline.

- The DM Command Generation can be implemented using logic wrapped around a static memory on a custom board, or on a GPU. FPGA implementation is rejected because the two-entry nonlinear lookup table needed for open-loop MEMS DM control requires a large amount of memory on each computing element, more than current FPGAs provide. We have therefore established commercial GPUs, one per deformable mirror as the baseline DM command generator. GPUs provide both the necessary speed and memory. Communications are the bottleneck, but the necessary data rates are achieved via the PCI-bus interface available on current commercial GPU cards. Both the input, from the tomography engine, and the output, to the DM driver, are via the PCI-bus interface.

2. System Characteristics

This section provides the basic RTC system characteristics and assumptions for the operating conditions it is designed to operate in. The assumptions are taken either from the Functional Performance Requirements (FPR) database, or from the error budget spreadsheets (with the FPRs taking precedence).

2.1 *Assumptions and Performance Requirements*

Item No.	Assumptions	Value	Units
1	Wind	9.5	m/sec
2	Sub Aperture Size	16.6	cm
3	Max Zenith Angle	54	degrees
4	Frame Rate	2	KHz
5	Stare Time HOWFS	500	µsec
6	Stare Time LOWFS	500	µsec
7	Sub Apertures Across Primary	60	
8	Sub Apertures Used in Calculations	88	
9	Number of Tomography Layers	5	
10	Number of Tomography WFSs	4	
11	Number of WFSs for Tip/Tilt, focus, and astigmatism	3	
12	Number of Science Objects	1	

Table 1 NGAO RTC System Assumptions

2.1 *RTC States Visible to the AO Control*

There are multiple levels of states in the AO control system including the supervisory control. This section describes only those states that the RTC itself needs to keep track of. These states are limited and insure only that the RTC cannot damage itself, will not give invalid data, and that it can be controlled to move between states become functional or be shut down.

Figure 10 shows the states visible to the AO control.

Finer grained information may be available for diagnostics, but are not presented here.

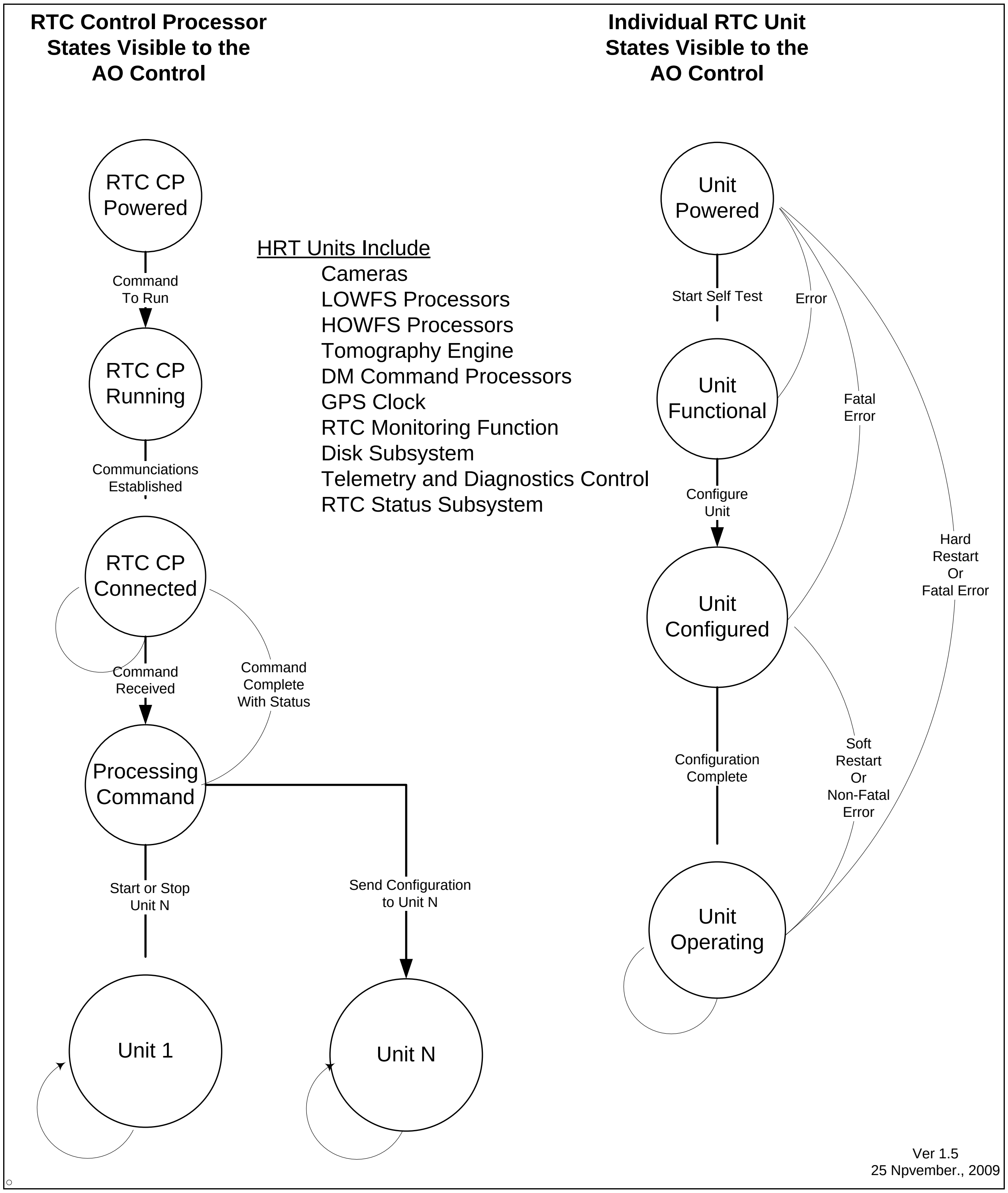


Figure 10 RTC and Sub-System States Viewable by the AO Control

The RTC CP will not honor a request to do something that is not valid, given its current state, and will return an appropriate error to the AO Control.

Examples:

1. If a sub-system has not been configured and the AO Control attempts to start the sub-system, the RTC will return an error indicating "Not Configured".
2. If the AO Control attempts to set a parameter with a value that is out of the allowable bounds for that parameter, the RTC will return and error indicating "Out of Bounds" with an additional message that indicates the allowable bounds. This is not really a "state" issue, but is mentioned here for clarity.
3. However, if the AO Control attempts to set a parameter with a valid value but one that does not make sense in the presence of other parameters that have been set, the RTC will simply set that parameter. The states represented by various combinations of settable parameters have meaning to a higher level of control than the RTC and should be controlled at that level.

2.2 *Timing and Control of Internal States of the RTC*

See Section 2.1 for a discussion of the States of the RTC visible to the AO Control.

Several sub-systems of the RTC run at different rates or start at different times during operation. Both the LOWFSs and the HOWFSs can run at approximately 2 KHz. The basic frame rate of the RTC is determined by the current rate of the camera with the fastest frame rate. All cameras and Tip/Tilt data must be synchronized to an integer multiple of this rate

Commands to change parameters come from the AO Control asynchronously. The RTC synchronizes these changes in parameters or data to occur at the start of the above frame rate, ~2 KHz. This ensures that all processing that occurs during a frame is done with the same parameters and input data.

Anything arriving too late to be applied at the start of a frame will be applied at the start of the next frame.

2.3 *Reliability and Recoverability*

The RTC will have a failure rate of less than 1 per 200 hours of operation. It constantly monitors itself for errors and will notify the AO Control with appropriate messages, logs, and alarms as determined by the AO Control. The primary cause of errors will be Single Event Upsets due to cosmic rays and related events. We constantly monitor for these events and can recover from them in less than 5 minutes. This can be either automatically or under control of an operator. A hardware signal is available to close the science shutter when an error is discovered and can be opened when the problem is resolved, either manually or automatically as preferred.

2.3.1 SEUs

An SEU is a single event upset in the system, usually caused by neutron showers from cosmic rays in the atmosphere. Alpha particles from concrete or other materials can also cause them.

When these particles strike electronic circuitry, the result can be to switch the state of some elements, such as memory.

SEU events normally leave no lasting damage to the circuitry, so that once detected, we can remedy their effect by reloading the memory with the correct value.

3. Error Budgets, Latency, and Data Rates

Table 2 shows the major latency components for the RTC.

RTC Latency Summary	(µSec)	Notes
Camera Stare	250	1
CCD Read	500	2
WFS	300	3
Tomography	450	4
Command Generation	200	5
DM Hold	250	6
Latency Contingency (~5%)	100	
Total Latency	**2050**	

Table 2 NGAO RTC Latency Summary

Scrub this for consistency and add derivation for clarity.

Notes:

1. This is one half the time for a single camera to capture an image from the LGS. It is a function of the CCD and camera controller settings.
2. This is the time to read the CCD. It can be partially overlapped with the WFS calculations and this is already included in the timing.
3. This is the time that includes reading the CCD, Centroiding, reconstructing the wave front, and transferring it to the Tomography Engine. Reading the CCD, grabbing the frame, and computing the centroids may be overlapped in some implementations.

   These values are based on analysis and measurements done for the GPI AOC [4]. This is a similar sized problem.
4. This is the time to combine the wave fronts from the tomography HOWFS, compute the estimate of the atmosphere, forward propagate the science object, and transfer the data to the low and high order command generators for the science object DMs.

   This is based on simulations and calculations summarized in Appendix D.
5. This is the time to generate the commands for the science object DMs and transfer them.
6. This is one half the time the commands are held on the DM. It is the same as the camera stare time.

3.1 *Error Budgets*

The error budgets in this document come from the NGAO Requirements Database. These numbers are expressed in nanometers or milliarcseconds.

They can be the result measurement noise or resolution, computational accuracy, algorithms, A/D accuracy, accuracy of our DM model, or other factors or of latency, which is the primary source or error in the RTC control system.

3.1.1 Accuracy

The accuracy of a system is determined by several factors. In our system, they are the accuracy of the:

- Camera data
- Parameters used in calculations
- Algorithms used
- Arithmetic operations
- Output hardware

3.1.1.1 Camera Data

The camera data supplied is 16 bits. We have assumed that the camera data is accurate to 12 bits.

3.1.1.2 Parameters

All fixed parameters and arrays used in calculations are represented as 18-bit numbers for real values with an added 18-bit imaginary part if they are complex. Examples of the parameters are C_n^2, Fourier coefficients, Kolmogorov filter coefficients, asterism parameters, DM modeling tables, etc.

3.1.1.3 Algorithms

Both the wavefront reconstruction and tomography steps require iteration. Upon convergence, the wavefronts at the output of the wavefront reconstruction and the output of the tomography engine, have achieved the match to data to the accuracy of the machine word length. The word length is selected so that digitization error is less than other physical limits, such as wavefront measurement error due to photon noise. The current error budget has this set at a few nanometers, and given that starting wavefronts are several microns RMS, the 18 bit word length baseline for use in the RTC was deemed sufficient.

The Hartmann centroiding error depends on the photon-counting and read-noise properties of the wavefront sensor. The digitization error, or photon noise, whichever is dominant, carries over to the error in the reconstructed wavefront. The 12-bit camera pixel data is sufficient to process the 1000 detected photo-electrons per Hartmann spot per frame to a level well below its inherent 33:1 signal to noise ratio set by photon noise.

The tomography algorithm is designed to converge to a minimum variance solution. Again, the digitization word length is chosen to represent numbers well below the expected residual tomography wavefront error. The residual error is expected to be in the 10 nm regime, and since the whole wavefront is on the order of 10's of microns rms, a signal to noise of at least 1000:1 (10 bits) needs to be maintained in all calculations, in particular the subtraction in the iteration loop. 18 bit calculations are the baseline.

DM command generation requires the use of a non-linear functional lookup. Because nonlinearities change the bit accuracy between input to the table and output, the D/A converter word length is designed to maintain the resulting wavefront control accuracy to below the assigned error budget number. At the fasted part of the MEMS nonlinear response curve, the wavefront is changing at about 100 nm/V, so ~100mv out of a total of 240 V accuracy is needed on the D/A, or about 12 bits.

The nonlinear model for open loop DM control is accurate to about 30 nm wavefront (15 nm surface), so a signal to noise on the order of 1000:1 should be maintained during the process of non-linear function fit. The baseline design is to use the GPU to implement 100-coefficient spline functions to map mirror displacements and forces to MEMS drive voltages. Accuracy of this technique well exceeds the 30 nm inherent accuracy of the model. The model is established with a calibration process performed on the MEMS DM. The calibration visits 400 force and displacement points in a 20x20 array [reference Morzinski paper], recording the voltages at each of these points. The resulting voltage surface is quite smooth and is accurately modeled by the 100-coefficient spline method. The GPU must now complete this operation for all the actuators on the DM within the margin of the latency time allocated to it.

3.1.1.4 Arithmetic Operations

Individual arithmetic operations take 18-bit values and accumulate a 48-bit result.

3.1.1.5 DACs

The DM's are supplied with 14-bit values.

3.2 *Latency Summary*

The most significant element to overcome in an AO system is the latency between sampling the atmospheric turbulence and applying the compensation for it. See Sect 4.1.1 of the Algorithm document for calculations on the impact of latency components on the wavefront error.

It and the spatial sampling determine the rate at which calculations and data transfers must be made to achieve a given level of compensation.

These in turn determine the size of the computation engine that is needed to handle the problem. For the current NGAO specification, this is a terra operation per second problem, which directly affects and limits the choices of algorithms, architectures, and implementations.

The total RTC latency is made up of several components, illustrated in Figure 11. The time to transfer data between processing units is included in the times for that unit.

- Camera Stare is the time the CCD is collecting light for a given frame.

  Only half the time of camera stare is counted as latency because the value measured at the end of the stare time is the integrated value over the entire time.

- CCD Read is the time to read the CCD while it is collecting the light for the next frame
- WFS is the time to perform centroiding and reconstruction of the wavefront
- Tomography is the time to take the 4 tomography wavefronts, update the estimate of the atmosphere and forward propagate the correct wavefront correction in the science direction.
- DM Command generation is the time to take the science wavefront and determine the correct voltages to place on the DM to achieve the desired shape.
- DM Hold time is the amount of time the shape remains on the DM and is the same as the Camera Stare time.

  Only half the time of the DM hold is counted as latency because the effect of the shape applied is integrated over the time it is on the mirror.

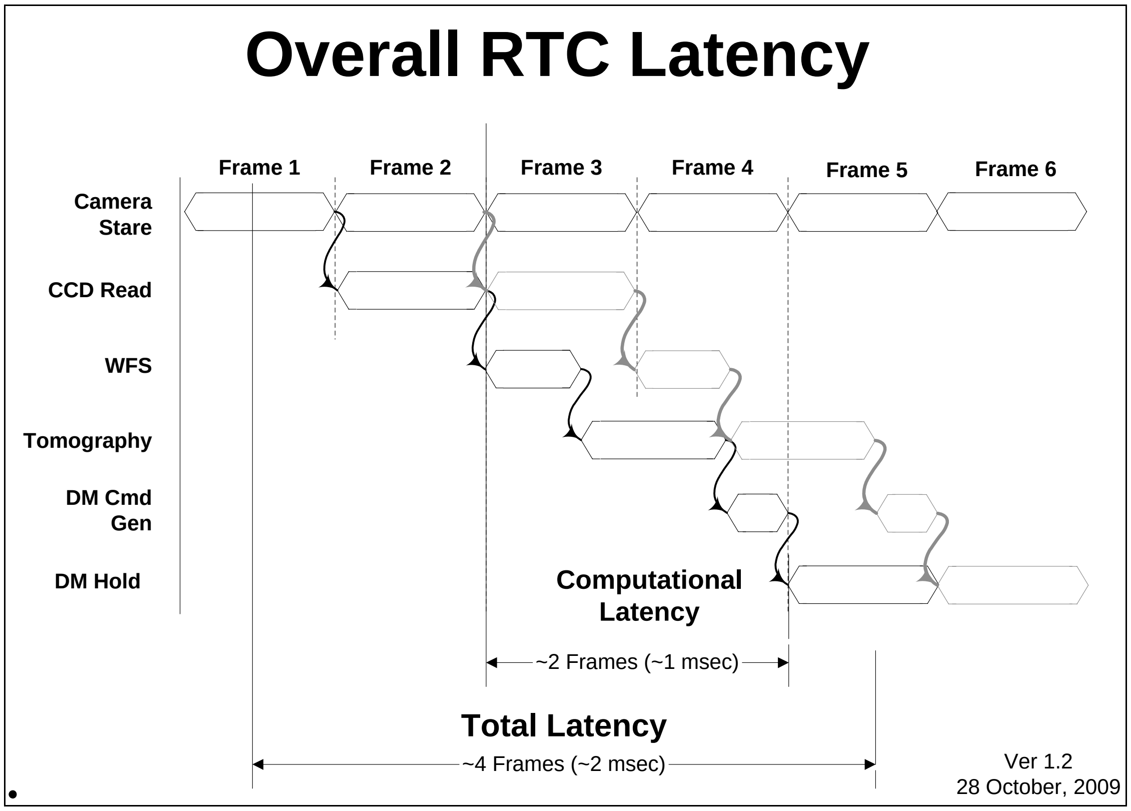


Figure 11 Overall RTC Latency Components (not to scale)

The total latency is calculated from the midpoint of the camera stare time to the midpoint of the subsequent DM hold-time. Computational Latency includes only the portion of time due to our actual processing of the data.

3.2.1 Architectural Issues Affecting Latency

Processing system architectures can be divided into two major categories: non-pipelined and pipelined. The following discussions attempts to highlight the differences between the two. One is not better than the other except in the context of the requirements and limitations of the system to which they are applied.

In the two examples that follow, it is assumed that the total processing time is the same in each case. The effects on the minimum total latency is examined. The example timings are for 900 µsec msec of computational latency, 400 µsec for the longest element of the computational latency, and 500 µsec to read the CCD. The time to perform centroiding can be overlapped with the CCD read and are ignored. However, the reconstruction and tomography cannot effectively start until all the centroids have been processed.

3.2.1.1 The effect of wind prediction and compensation on latency errors

Computations and operations to calculate the DM commands to place on the mirrors to reduce the effect of the atmospheric turbulence add to our latency. However, there is only so much we can do to reduce them. The estimate we create of the turbulence on each layer is old by the time it is placed on the DM and the layers have now moved.

However, at the frame rates at which we operate, the turbulence is primarily frozen flow with various layers of “fixed” turbulence blown across our aperture. If we know the speed of the wind in each layer, we can easily shift our estimate to be consistent with the how much the layer has moved since we measured it.

Measuring and compensating for the wind in the layers can dramatically reduce the effective latency of the RTC and hence the errors attributed to it.

The RTC has the capability to compensate for the wind layer by layer at virtually no added cost in latency or hardware. Wind induced shift – given the wind velocity at each layer in the tomography – is implanted with a simple shift factor, a multiplication in the Fourier domain. However, the wind velocity estimation portion of the problem needs to be a separate calculation.

Fortunately, wind estimation can be done off-line, in parallel with the RTC calculations. A method advance by Johnson [reference Luke Johnson’s OSA 2009 paper] utilizes the telemetry data from the AO system to optimally estimate the wind velocity, assuming this velocity vector is changing slowly relative to the time scale of the frame rate.

Latency error could potentially be reduced significantly using the information, since pre-shifting the prior wavefront estimate by the estimated frozen-flow amount is tantamount to increasing τ_0. This is true even though the wind velocity estimates lag the RTC measurements, from which they were generated, by many frames. As long as the wind is not changing faster than the lag, it can benefit.

3.2.1.2 Non-Pipelined Architectures

The following describes an example of a non-pipelined MOAO control system. It does not represent the current RTC timing and is provided for illustration only.

In a non-pipelined system, data is brought in, processed by a single processing unit, the results are used, and the cycle is started again. In our case, we have a computational latency of 1 msec and a CCD read time of 500 µsec. Since there the processing unit is busy processing the last frame, no processing can occur during the next CCD read. The resultant frame rate is 1.5 msec and the total latency is 3 msec.

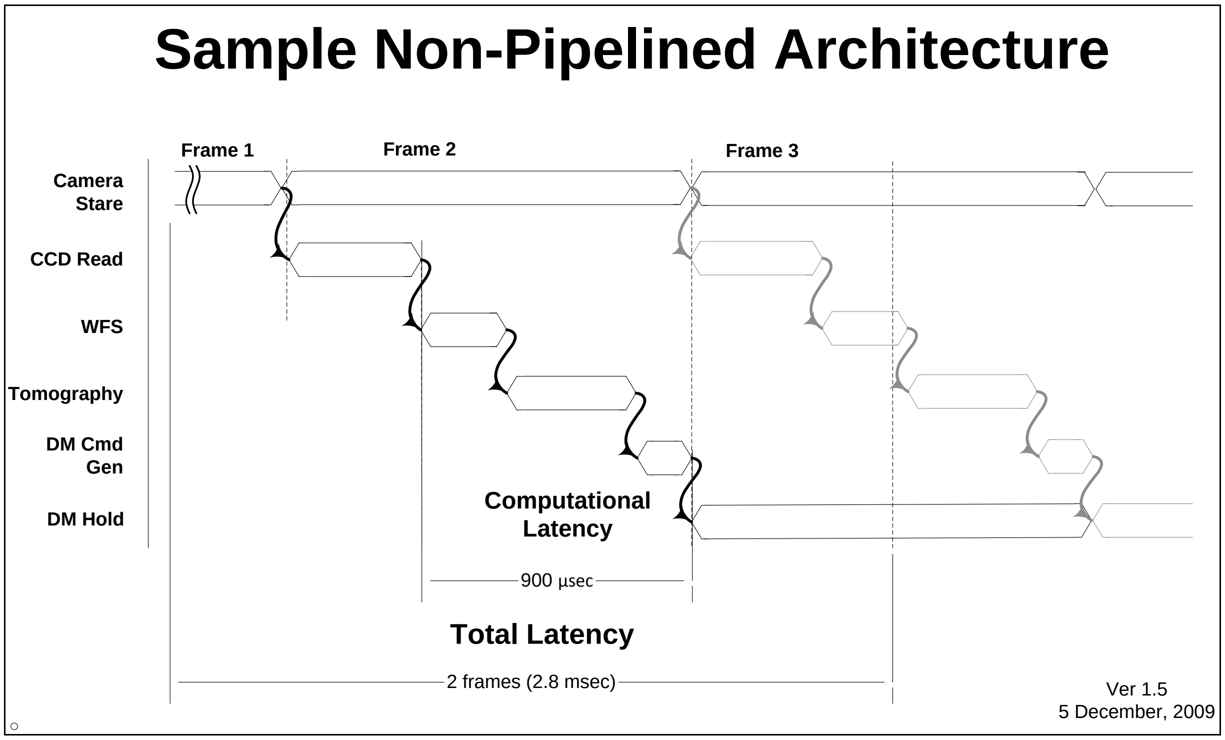


Figure 12 Non-Pipelined Latency (not to scale)

In a non-pipelined system, the frame rate is determined by the total latency of handling the data. In this case, it is the time to read the CCD plus the computational latency.

All calculations must be finished before new calculations can start. The total latency is two frames (1 frame to handle the data, plus ½ a frame on the front end to account for the integration time of the camera and ½ a frame at the end to account for the integrated effect of the DM hold time).

3.2.1.3 Pipelined Architectures

The following describes an example of a pipelined MOAO control system. It does not represent the current RTC timing and is provided for illustration only.

In a pipelined system, the processing is divided amongst several units. Each unit processes the data and passes it on to the next unit. The total computational Latency is still 1 msec, as above (assuming transfer times between units are negligible). However, it can be seen that new data can be brought in as soon as the first unit is through processing its last data, and the output can be updated as soon as the last unit has processed its data. The frame rate here is determined by the largest time spent by any individual unit. Consequently, this rate can be considerably faster than that in the non-pipelined case.

Note that the camera frame rate is the same as the DM frame rate. The DM rate is synchronous but has a different phase than the camera's.

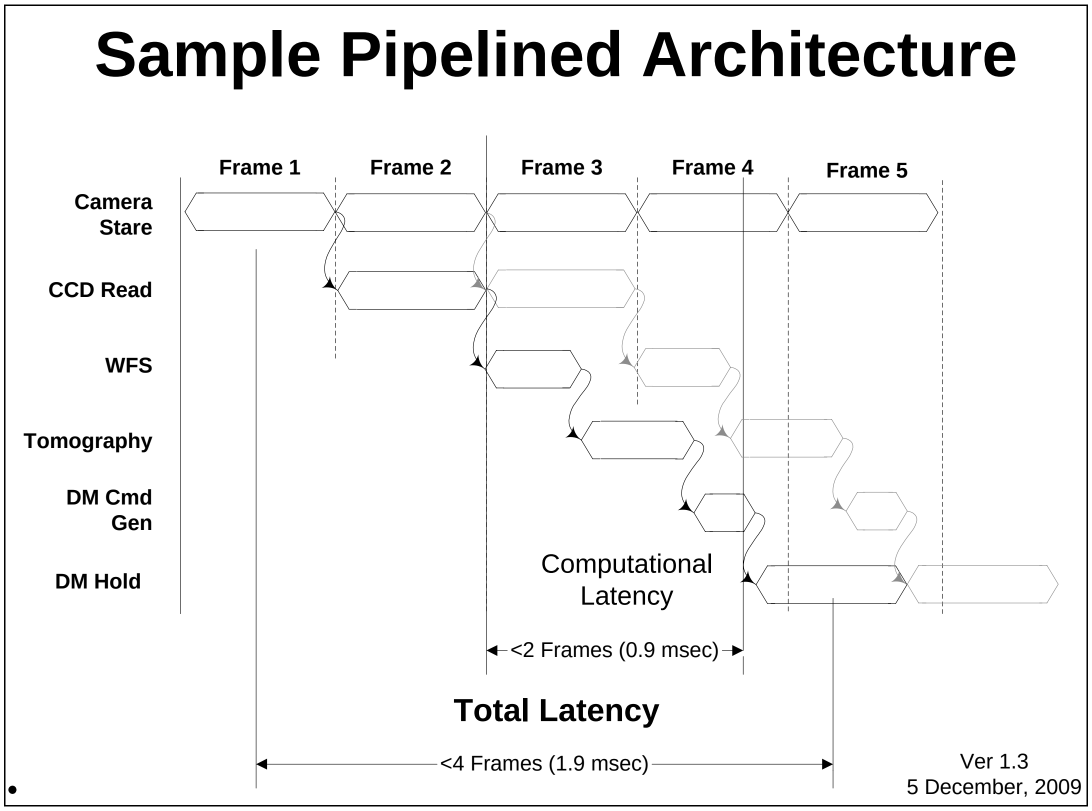


Figure 13 Pipelined Architecture Latency (not to scale)

In this system, each element of the data processing is handled by separate hardware. This allows us to have several frames of camera data being processed somewhere in the pipe at the same time, just not in the same piece of hardware. The maximum frame rate is determined by the longest time it takes for any single element to process its data.

In our situation, the frame rate is determined by the CCD read time, which happens to be 500 µsec. All other individual computational elements are less than this. The time to handle the data is the time to read the CCD plus the Computational Latency , <3 Frames (1.4 msec). The total latency is 1.9 µsec, <4 frames (3 frames to handle the data, plus ½ a frame on the front end to account for the integration time of the camera and ½ a frame at the end to account for the integrated effect of the DM hold time).

Figure 13 shows that the computational Latency does not end on a frame boundary. While the DM commands are applied synchronous to the frame clock they are applied at a time that does not occur on a frame clock boundary.

Assuming the time to handle the data is the same for both the pipelined and the non-pipelined case, the pipelined architecture will provide less total latency. It can require more hardware, however.

3.2.1.4 Comparison of Non-Pipelined vs. Pipelined Architectures

A pipelined system takes more hardware and is more complex to implement than a non-pipelined system. However, the benefits are that we can decouple the frame rate from the processing time and achieve a lower latency time. In our case, the latency difference is 2.8 msec vs. 1.9 msec. a savings of 900 µsec or 30%.

For the NGAO RTC, we use a pipelined architecture.

3.2.2 Latency Calculations for NGAO

The times for executing the RTC algorithms were derived as follows:

- Detailed FPGA simulations to verify that the basic tomography loop functioned correctly and at the desired clock rate.
- Extending the algorithm for additional functionality and counting clocks needed to execute the code for these functions, see Appendix D.
- Timing tests and analysis for the Gemini Planet Imager AO Control were used for CPU operations [4].
- Timing tests and analysis for the GPU were used for CPU operations.

3.2.2.1 Latency Categories

Latency can be divided into three categories: transfer, compute, and fixed latencies.

Transfer latency is the result of moving data from one place to another, usually between physically distinct hardware.

Compute latency is due to the time it takes to apply an algorithm on the data, and includes any minor transfer latency that may occur during that process.

Fixed latency is due to system architecture issues that are separate from the above. In the case of the RTC, it is the result of the value of the frame clock and the time read the data out of the CCD subsequent from the actual time to transfer it to the WFS frame grabber.

Detailed latency analysis can be found in the sections below.

Table 3 shows the system parameters used in the following latency analysis.

Sub Apps is the number of subapertures across the primary.

Extended Sub Apps is the total number of subapertures we use in computations to account for the FOV and zenith angle and is larger than the number of subapertures across the primary.

Frame Rate (Hz)	Primary Sub Apps	Tomography Ext Sub Apps
2,000	60	88

Table 3 NGAO RTC System Parameters (Note: the Point-and-Shoot WFS use 30 and 44 sub apps. Instead of 60 and 88)

The frame rate of 2 KHz is driven by the maximum bandwidth error allowed (see Requirements Database **FR-1436.**

There are 60 subapertures across the primary aperture.

In order to avoid artifacts due to wrapping while processing Fourier data, the number of apertures used internally is 88.

The Point-and-Shoot HOWFS is a complete open loop control system. We calculate the latency from the mid-point of the stare time to the mid-point of the DM hold (see Figure 14). For these WFSs, there are 30 subapertures across the primary and similarly to the Tomography HOWFS we use 44 apertures internally to avoid wrapping artifacts.

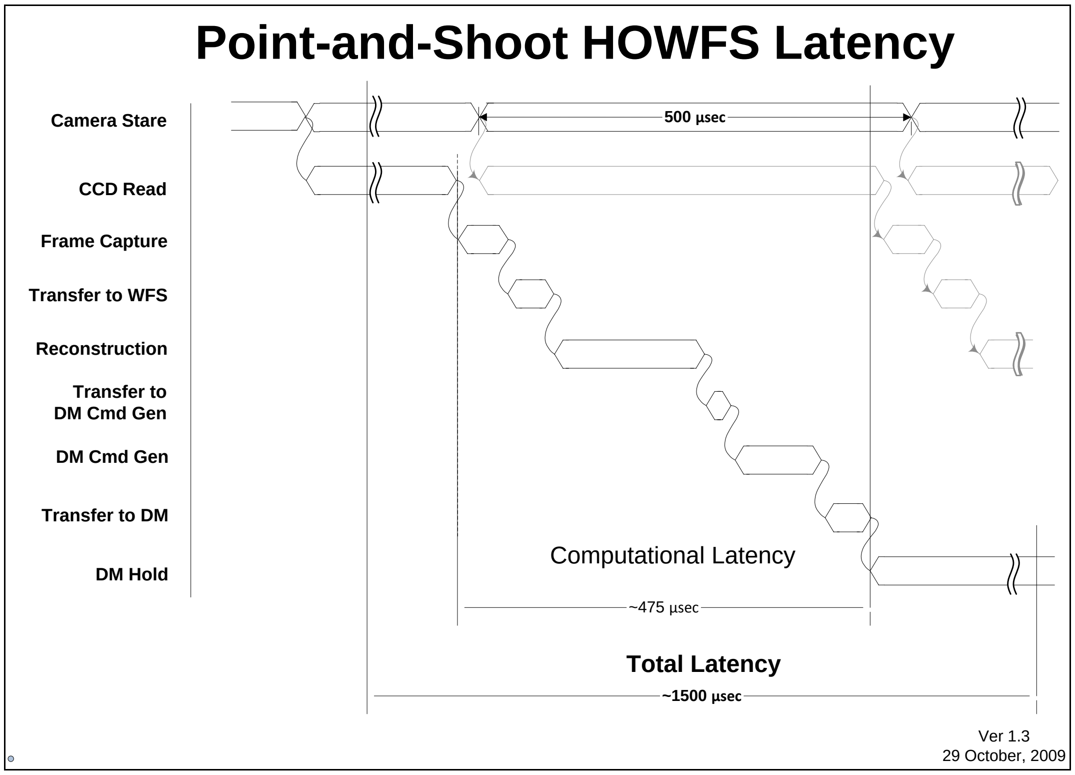


Figure 14 Point-and-Shoot HOWFS Latency

The Tomography HOWFS is different from the Point-and-Shoot HOWFS, since its latency is only part of the overall science control loop latency. Consequently, its latency is calculated from the end of the stare time to the start of the tomography calculations (see Figure 15).

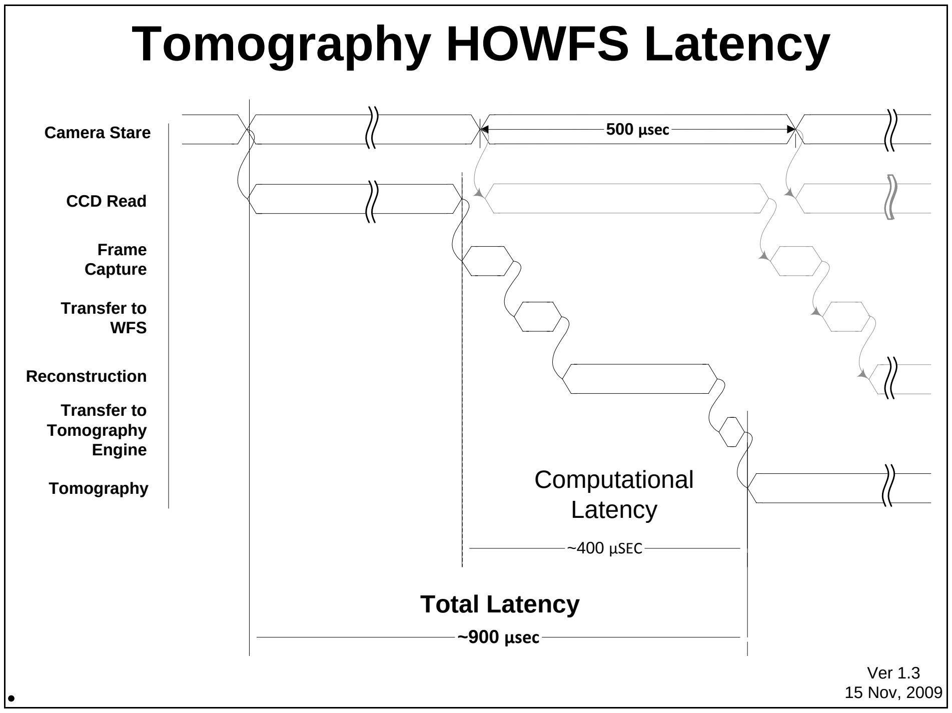


Figure 15 Tomography HOWFS Latency

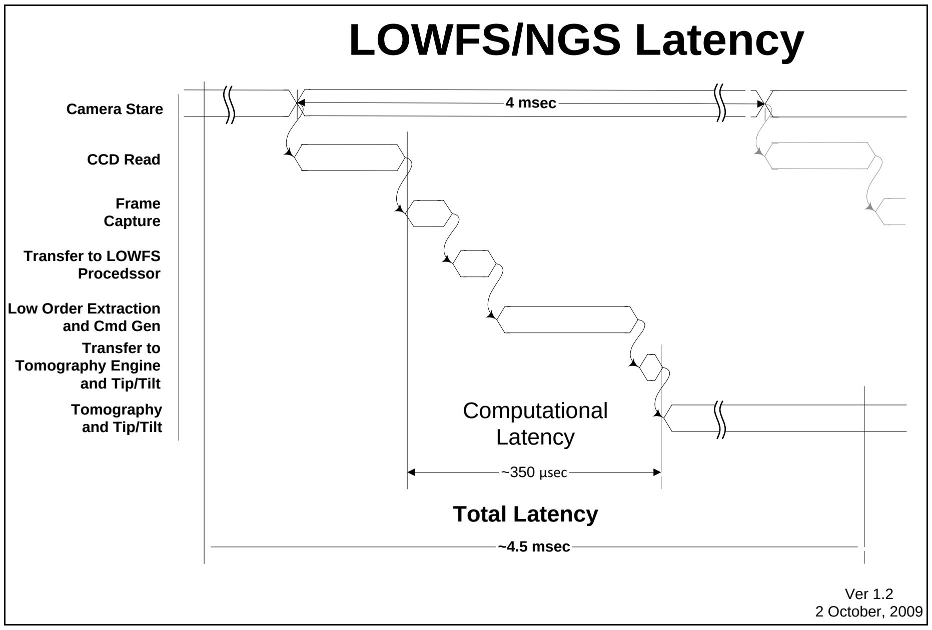


Figure 16 LOWFS/NGS Latency

3.2.2.2 Transfer Latency

Table 4 through Table 7 show the transfer latencies that affect the total latency of the RTC. Also shown are the data rates needed to support the required Telemetry of system data during operation.

Ver 1.2	1 Jan, 2010	Tomographic Data Rates And Latency Calculations							
Transfer Latencies	Number of Items	Size per Frame (B)	Rate for Diag/Telem (MB/Sec)	Total Diag/Telem Rate (MB/Sec)	Operational Xfer Latency Budget (µSec)	Operation Rate (MB /Sec)	Transfer Rate (GB/Sec)	Total Op. Rate (MB/Sec)	Notes
WFS CCD Read Out	1	131,072	N/A	N/A	500	N/A	N/A	N/A	
HOWFS Camera to Frame Grabber	4	131,072	262.14	1,049	50	2,621	N/A	10,486	1
HOWFS Frame Grabber to Centroider	4	131,072	262.14	N/A	50	2,621	N/A	10,486	2
HOWFS Centroider to Telemetry	4	30,976	61.95	248	N/A	N/A	N/A	N/A	3
HOWFS Wave Front	4	15,488	30.98	124	8	2,000	2.00	8,000	4
Tomography Layer	5	15,488	30.98	155	8	2,000	2.00	10,000	5
DM Cmd	4	7,200	14.40	58	50	N/A	N/A	N/A	6
Total Tomographic Transfer Time				1,633	665			38,972	

Table 4 Tomography Transfer Latencies and Rates

Ver 1.2	1 Jan, 2010	Point-and-Shoot HOWFS Data Rates And Latency Calculations							
Transfer Latencies	Number of Items	Size per Frame (B)	Rate for Diag/Telem (MB/Sec)	Total Diag/Telem Rate (MB/Sec)	Operational Xfer Latency Budget (µSec)	Operation Rate (MB /Sec)	Transfer Rate (GB/Sec)	Total Op. Rate (MB/Sec)	Notes
WFS CCD Read Out	1	131,072	N/A	N/A	500	N/A	N/A	N/A	
HOWFS Camera to Frame Grabber	3	131,072	262.14	786	50	2,621	N/A	7,864	1
HOWFS Frame Grabber to Centroider	3	131,072	262.14	N/A	50	2,621	N/A	7,864	2
HOWFS Centroider to Telemetry	3	30,976	61.95	186	N/A	N/A	N/A	N/A	3
HOWFS Wave Front	3	15,488	30.98	93	8	2,000	2.00	6,000	4
DM Cmd	4	7,200	14.40	58	50	N/A	N/A	N/A	6
Total NGS Transfer Time				1,123	658			21,729	

Table 5 Point-and-Shoot HOWFS Transfer Latencies and Rates

Ver 1.0	1Mar, 2010	NGS Mode Data Rates And Latency Calculations							
Transfer Latencies	Number of Items	Size per Frame (B)	Rate for Diag/Telem (MB/Sec)	Total Diag/Telem Rate (MB/Sec)	Operational Xfer Latency Budget (µSec)	Operation Rate (MB /Sec)	Transfer Rate (GB/Sec)	Total Op. Rate (MB/Sec)	Notes
WFS CCD Read Out	1	131,072	N/A	N/A	500	N/A	N/A	N/A	
HOWFS Camera to Frame Grabber	3	131,072	262.14	786	50	2,621	N/A	7,864	1
HOWFS Frame Grabber to Centroider	3	131,072	262.14	N/A	50	2,621	N/A	7,864	2
HOWFS Centroider to Telemetry	3	30,976	61.95	186	N/A	N/A	N/A	N/A	3
HOWFS Wave Front	3	15,488	30.98	93	8	2,000	2.00	6,000	4
DM Cmd	4	7,200	14.40	58	50	N/A	N/A	N/A	6
Total NGS Transfer Time				1,123	658			21,729	

Table 6 NGS Mode Transfer Latencies and Rates

Ver 1.0	1Mar, 2010	LOWFS Data Rates And Latency Calculations							
Transfer Latencies	Number of Items	Size per Frame (B)	Rate for Diag/Telem (MB/Sec)	Total Diag/Telem Rate (MB/Sec)	Operational Xfer Latency Budget (µSec)	Operation Rate (MB /Sec)	Transfer Rate (GB/Sec)	Total Op. Rate (MB/Sec)	Notes
WFS CCD Read Out	1	131,072	N/A	N/A	500	N/A	N/A	N/A	
HOWFS Camera to Frame Grabber	3	131,072	262.14	786	50	2,621	N/A	7,864	1
HOWFS Frame Grabber to Centroider	3	131,072	262.14	N/A	50	2,621	N/A	7,864	2
HOWFS Centroider to Telemetry	3	30,976	61.95	186	N/A	N/A	N/A	N/A	3
HOWFS Wave Front	3	15,488	30.98	93	8	2,000	2.00	6,000	4
DM Cmd	4	7,200	14.40	58	50	N/A	N/A	N/A	6
Total NGS Transfer Time				1,123	658			21,729	

Table 7 LOWFS Data Rates and Latency Calculations

Notes:

Transfer times in the processing flow of the data must take place in a small part of a frame (~50 µsec, ~10% of a frame) whereas Diagnostic/Telemetry transfers may take an entire frame (~500 µsec). This leads to a much higher rate for Operational transfers.

The two key parameters calculated are the "Total Diagnostic/Telemetry Rate" and the "Operational Transfer Time". The former determines the characteristics of the RTC Disk Sub-System and the later is part of the Total RTC latency calculation.

1. This is the time for a single camera to transfer the camera data to the Frame Grabber. A time of 50 µsec has been allocated which is consistent with a transfer using Camera Link full configuration. Also shown is the total data rate needed to save the camera data for all 7 cameras through the Diagnostic/Telemetry port.
2. If the frame grabber is a separate piece of hardware from the rest of the WFS, this is the time to transfer the camera data from it to the next stage of the WFS, the centroider. Since camera data has already been saved at the camera output if requested, there is no load associated with saving it to the Diagnostic/Telemetry port in the WFS processor.

   This time may be overlapped with the CCD read time, depending on the exact implementation of the WFS. It is included here for conservativeness.
3. If it is desired to save the centroids, this number is the amount of time to transfer the data over the Diagnostic/Telemetry port. No additional operational transfer load is required.

   This time may be overlapped with the CCD read time, depending on the exact implementation of the WFS. It is included here for conservativeness.
4. After the WFS has calculated the wave front, these numbers are the amount of time needed to transfer it to the tomography engine and the Diagnostic/Telemetry Port.

   After the Tomography Engine has estimated the atmospheric volume, the data needs to be transferred to the DM Command Generator. If it is also desired to save the tomographic layer information, the amount of time to transfer the data over the Diagnostic/Telemetry port is also given.
5. The DM information from the Tomography Engine is in a spatial measure and the DM Command Generator generates the correct DM actuator voltages to best match the desired DM shape. These numbers are the amount of time it takes to transfer the correct shape to the DM. Additionally, the data rate for the Diagnostic/Telemetry Port is shown if the data is to be saved.

3.2.2.3 Non-Transfer Latencies

Non-Transfer Latencies	**(µSec)**	**Type**
Camera Stare	500	Fixed
WFS	300	Compute
Tomography	450	Compute
DM Cmd Gen	200	Compute
DM Hold	500	Fixed
Total Fixed Time	**500**	
Total Compute Time	**950**	

Table 8 Non-Transfer Latencies

3.2.2.4 Total Latency for Tomography Operations

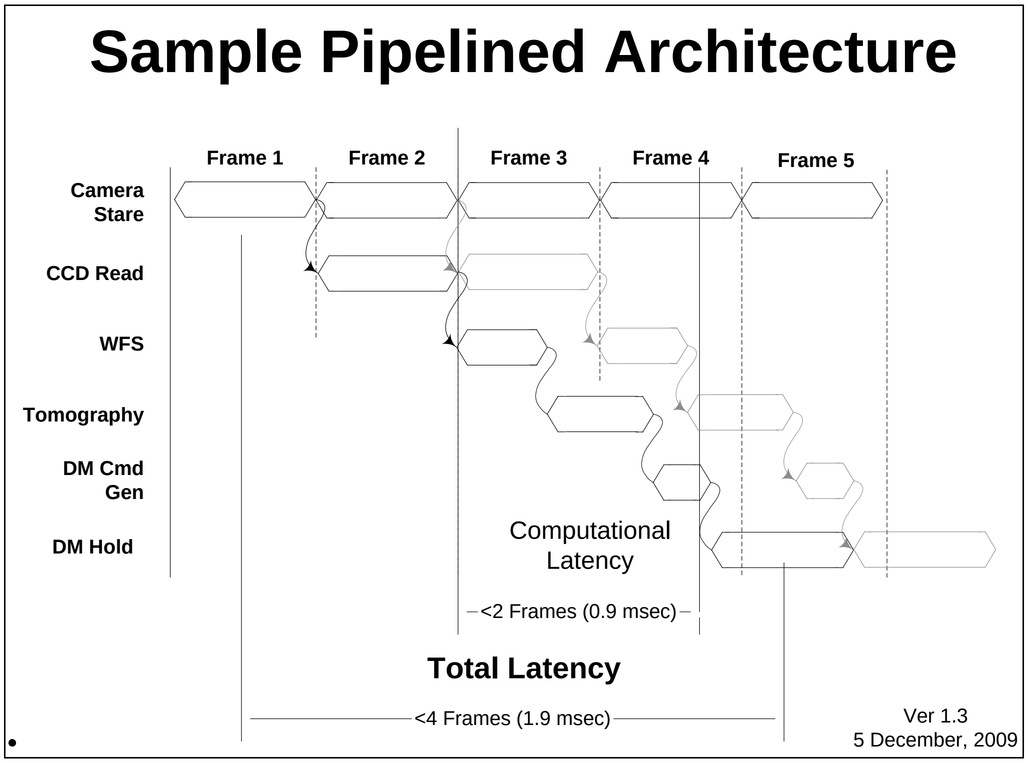


Figure 17 RTC Latency

Total Tomographic Latency Calculations	
Total Computation Time	950
Total Fixed Time	500
Total Op. Xfer Time	665
Total Tomographic Latency	**2,115**

Table 9 Total Latency Calculations

3.2.2.5 Total Latency for NGS Operations

Total Point-and-Shoot Latency Calculations	
Total Computation Time	500
Total Fixed Time	500
Total Op. Xfer Time	658
Total HOWFS/NGS Latency	**1,658**

Table 10 Total NGS Latency Calculations

3.2.3 Calculating Error Values Due to Latency

The calculation of error values due to latency are covered in the RTC Algorithm Document [1].

3.3 *Data Rates*

In addition to the latency associated with acquiring the guide star image, processing the AO data, and controlling the actuators, the NGAO system needs to be able to save key system telemetry data. This feature allows subsequent processing of the science data, atmospheric and system analysis, and the display real time diagnostics information.

The amount of data it is possible to save is huge (> 2 GB/sec) and a very large, fast disk sub-system is needed to capture this data prior to analysis, see Table 11 and Section 11.

Item	# per side	Total items	Bytes per Wd	Total Bytes	Telem Time (µsec)	Telem Rate (MB/sec)	Total Telem Rate (MB/sec)	Transfer Time (µsec)	Transfer Rate (MB/sec)	# of Instances	Total Xfer Rate (MB/sec)
Tomography WFS Camera	256	65,536	2	131,072	500	262	1,049	50	2,621	4	10,486
Point-and-shoot Camera	128	16,384	2	32,768	500	66	197	50	655	3	1,966
T Cent	60	7,200	2	14,400	500	29	115	50	288	4	1,152
P Cent	30	1,800	2	3,600	500	7	22	50	72	3	216
T WF	60	3,600	3	10,800	500	22	86	50	216	4	864
P WF	30	900	2	1,800	500	4	11	50	36	3	108
T Layer	88	7,744	3	23,232	500	46	232	50	465	5	2,323
Sci WF	88	7,744	3	23,232	500	46	46	50	465	1	465
Sci WF HO	88	7,744	3	23,232	500	46	46	50	465	1	465
Sci WF LO	88	7,744	3	23,232	500	46	46	50	465	1	465
Tweeter	64	4,096	2	8,192	500	16	16	50	164	1	164
Woofer	20	400	2	800	500	2	2	50	16	1	16
Telemetry Effect											
Total rate (MB/sec)							**1,869**				
Total data per 6 hours (TB)							**40**				

Table 11 Data Rate Summary

Add NGS

4. RTC Control Processor (CP)

The RTC Control Processor (CP) provides an asynchronous interface between the RTC system the AO Control. The interface to the AO Control is through a socket interface over Ethernet (See Figure 4 and Figure 18).

It is built on a high performance Linux server with a real-time kernel.

The RTC system is a key component to the overall AO system and the rest of the telescope environment and the CP's interface with the AO Control and the telescope system will adhere to the component architecture defined in [**FR-1418**].

4.1 *Architecture and Design*

As stated, the CP is built on a high performance Linux server. This will be augmented by a real-time kernel by Wind River, to synchronize its interface to the rest of the RTC where necessary.

The CP controls all aspects of the RTC's configuration and overall operation. However, it does not control the real-time state transitions. The sole need for its real-time capabilities is to provide synchronization of data and commands between the AO Control and the rest of the RTC.

It also contains a "Camera Simulator." This simulator is used to supply dummy camera data to the WFS for diagnostic purposes.

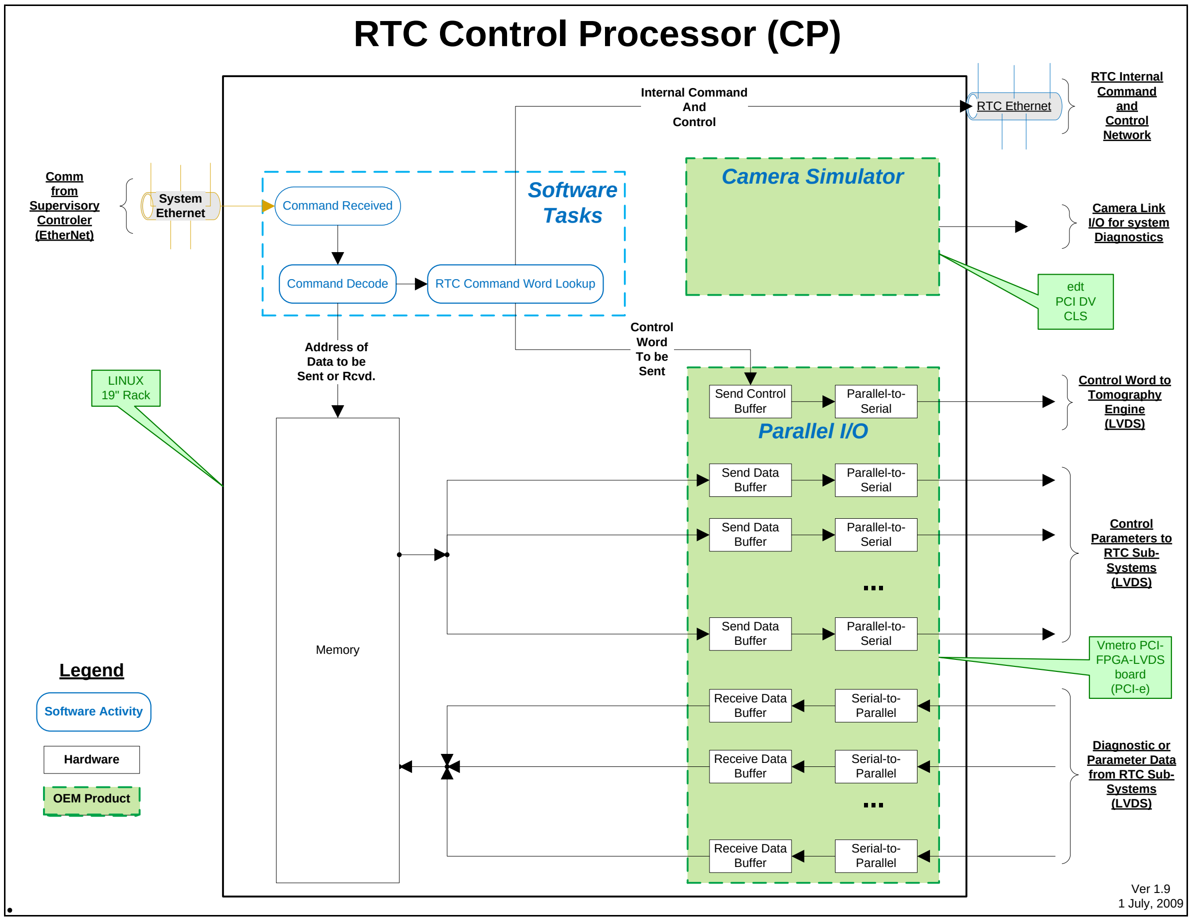

RTC Control Processor (CP)
Internal Command And Control
RTC Ethernet
RTC Internal Command and Control Network
Comm from Supervisory Controler (EtherNet)
System Ethernet
Software Tasks
Command Received
Command Decode
RTC Command Word Lookup
Camera Simulator
Camera Link I/O for system Diagnostics
edt PCI DV CLS
Control Word To be Sent
Address of Data to be Sent or Rcvd.
LINUX 19" Rack
Send Control Buffer
Parallel-to-Serial
Parallel I/O
Control Word to Tomography Engine (LVDS)
Send Data Buffer
Control Parameters to RTC Sub-Systems (LVDS)
Memory
Vmetro PCI-FPGA-LVDS board (PCI-e)
Receive Data Buffer
Serial-to-Parallel
Diagnostic or Parameter Data from RTC Sub-Systems (LVDS)
Legend
Software Activity
Hardware
OEM Product
Ver 1.9
1 July, 2009

Figure 18 RTC Control Processor (CP)

Parameters to control the tomography engine and other sub systems must be loaded at specific times, using specific hardware interfaces, and protocols. Commercially available products residing in the CP are used to accomplish this.

These include Ethernet, LVDS interfaces for the tomography engine, but also a Camera Link camera simulator for support of diagnostics and analysis.

4.2 *AO Control System Feature Support*

See the AO Control Interface Document for details on these features (GAOControlSWArchitecture_rev1, Erik Johansson, et. al.).

The following are handled by the CP.

4.2.1 Commands

The CP accepts commands from the system AO Control (or alternatively from a telnet session) over sockets. Data is passed as part of the command or as a system address from which the CP will fetch the data.

4.2.2 Monitoring

This provides the ability to monitor the attributes of any components for changes.

4.2.3 Events

Events within the RTC are generated by the individual sub-systems and monitored by the CP. The CP in turn generates events for the AO Control as required by the AO Control.

4.2.4 Alarms

Alarms are propagated using from components in the RTC and include things such as over temperature and fatal errors

4.2.5 Log Messages

These are information that are recorded by the AO Control in a persistent store and may contain configuration, environment, or other data of interest.

4.2.6 Configuration

The CP under control of the AO Control handles low-level configuration.

4.2.7 Archiving

AO data is saved short term in the RTC disk sub-system. The telescope archiving systems provides long term archiving.

4.2.8 Diagnostic Data Stream

Prior to transfer to the AO Control over the diagnostic stream, the CP will provide boxcar averaging, filtering, or sampling of camera frame or other data using the Offload Processor. This data will include the 12 values measured by the LOWFS processor (see 4.3.4 of the Algorithm Document). The Offload Processor works as a slave processor controlled by the CP.

4.2.9 Computational Load

In general, the computational load of the CP is moderate and should be well supported by a multi-CPU/multi-core server.

The processing in 4.2.8, which is handled by the Offload Processor, may be handled by the CP depending on the power of the servers available at the time of design. If not, the Offload Processor will remain a separate slave unit to the CP. An analysis is ongoing to determine the exact implementation and hardware/software impact.

4.3 *Interfaces and Protocols*

4.3.1 Interface between the CP and the AO Control

The interface is over Ethernet using sockets and the details of this interface are covered in a separate document, see NGAOControlSWArchitecture_rev1.

The compute load of the CP is expected to be relatively low. Therefore, it is possible that the AO Control functions and the CP functions could be integrated into a single hardware platform if desired.

4.3.2 Interface between the CP and the rest of the RTC

The CP's interface to the rest of the RTC is through Gbit Ethernet, LVDS, or Camera Link busses as appropriate.

4.3.2.1 Cameras

The interface between the CP and the cameras will be over Camera Link. The will provide commands to the cameras directly for control and status.

4.3.2.2 HOWFS

There are two interfaces between the CP and the HOWFS: Camera Link is used to present dummy camera data during diagnostics and tests; Ethernet is used for HOWFS control and status.

4.3.2.3 LOWFS

There are two interfaces between the CP and the LOWFS: Camera Link is used to present dummy camera data during diagnostics and tests; Ethernet is used for LOWFS control and status.

4.3.2.4 Tomography Engine

The interface between the CP and the Tomography Engine is over LVDS for control and status.

4.3.2.5 DM Command Generators

The interface between the CP and the DM command generators is over Ethernet for control and status. LVDS is used for diagnostic data interfaces.

4.3.2.6 Tip/Tilt Command Generators

The interface between the CP and the Tip/Tilt command generators is over Ethernet for control and status.

4.3.2.7 Clock Generation Sub-System

The interface between the CP and the Clock Generation Sub-system is over Ethernet for control and status.

4.3.2.8 Disk Sub-system

The interface between the CP and the disk sub-system will be over the RTC internal Gbit Ethernet for control and status.

5. Cameras

5.1 *HOWFS (Tomography and Point-and-Shoot)*

5.1.1 Interfaces and Protocols

The interface between the HOWFS camera sub-systems and the RTC will be through Camera Link™ using the Camera Link™ Full configuration.

5.1.2 Data Flow, Rates and Latency

The following are the specifications the camera systems must meet in order for the RTC to meet the required error budget.

5.1.2.1 Camera Frame Rate

The HOWFS camera must be able to sustain a 2 KHz frame rate, Ref FR-1436.

5.1.2.2 CCD Read Time Latency

After each frame is exposed, the CCD transfers the frame data to holding registers and starts exposing the next frame. The CCD must be able to transfer this data to the camera controller at the camera frame rate without affecting the simultaneous science object acquisition.

5.1.2.3 Camera Transfer Latency

The data latency between the last byte transferred from the CCD and the last byte sent to the frame grabber must be less than 50 µsec. See Table 4.

5.2 *LOWFS (IR)*

In order to accommodate differing natural guide star's (NGS) brightness, the rate of the camera clocks can be set differently on each wavefront sensor with respect to the tomography engine. This is useful to allow overall system optimization by trading individual wavefront or tip/tilt/focus sensor signal-to-noise ratio with sensor bandwidth. However, all sensor sample rates must be less than or equal to the tomography sample rate in order for the system to remain synchronized at its maximum rate. The LOWFS sample period must be an integer multiple of the system frame rate. The master frame clock is derived from one of the HOWFS cameras.

5.2.1 Interfaces and Protocols

The interface between the camera sub-systems and the RTC will be through Camera Link™ using the Camera Link™ Full configuration.

5.2.2 Data Flow, Rates and Latency

The following are the specifications the camera systems must meet in order for the RTC to meet the required error budget.

5.2.2.1 Camera Frame Rate

The LOWFS IR cameras must be able to sustain a 2 KHz frame rate.

5.2.2.2 Camera Read Time Latency

After each frame is exposed, the camera transfers the frame data to holding registers and starts exposing the next frame. The camera must be able to transfer this data to the camera controller at the camera frame rate without affecting the simultaneous science object acquisition.

5.2.2.3 Camera Transfer Latency

The data latency between the last byte transferred from the camera and the last byte sent to the frame grabber must be less than 50 µsec. See Table 7.

5.3 *Master Clock Generation for Camera Synchronization*

Synchronization between cameras is required so the Tip/Tilt can be synchronized with the tomography calculations and the all camera telemetry can be synchronized with the frame from which it came.

Cameras will be able to synchronize to an external Frame Clock or be able to generate a Frame Clock to which other cameras can be synchronized. One camera will be designated as the master RTC camera. All other cameras will be synchronized to this camera. An optional mode of operation is for the Frame rate to be determined by the RTC Clock Synchronization and Control System.

5.3.1 Inter Camera Capture Jitter

The time between the synchronization signal and the start of the camera exposure will be a Maximum of 500 psec for cameras of the same frame rate.

5.4 *Control*

The AO Control will control all camera parameters through facilities provided by the CP. The following controls will be supported be each camera:

5.4.1 Gain

The analog gain of the camera can be set.

5.4.2 Frame rate

The frame rate is the frequency at which new frames are captured.

The RTC will support whatever rates are required by the system, i.e. 2 KHz, 1 KHz, 500 Hz, 100 Hz.

5.4.3 Frame Transfer Rate

The frame transfer rate is the speed of the link between the camera and the frame capture card.

5.4.4 Internal Camera Clock Speeds and Parameters

These parameters set the detailed internal workings of the camera and are generally not set by non-maintenance personnel. The AO Control will package these into some manageable set for the users.

6. Wave Front Sensors (WFSs)

There are seven physical WFSs in the system, divided into two types: 3 32x32 Point-and-Shoot and 4 64x64 Tomography.

Ideally, all WFS processors would have a common architecture and implementation. This would bring the advantage of lower spares cost, easier troubleshooting, development, and faster debug time during operation. However, due to the differing size and speed requirements, this is unlikely to be possible.

The RTC will provide an input to the focus offload loop controlled by the AO Control function.

The RTC will provide a tip-tilt input to the tip-tilt offload loop controlled by the AO Control function.

6.1 *NGS Mode*

In NGS mode, one of the Tomography WFS will operate in 32x32 mode and will take its input from the NGS camera. It will execute the same centroiding and reconstruction algorithm as the Point-and-Shoot WFSs, and will pass its output to the tomography engine and tip/tilt information to the science Tip/Tilt mirror system. The rest of the Tomography and Point-and-Shoot HOWFSs and the LOWFS will do nothing in this mode. The Tomography Engine will pass the reconstructed wave front unmodified to the Woofer/Tweeter process and on to the DM command generators.

6.2 *Algorithms*

The WFSs will support several centroiding and reconstruction algorithms:

6.2.1 Centroiding Algorithms

1. Center of mass
2. Quad-cell
3. Binned quad-cell

All of these “algorithms” are established by weights on a 4 by 4 grid of pixels. There are infinite variations based on weighting settings that can be set by the supervisory control. See the Algorithms Design Document for additional details.

6.2.2 Reconstruction and Tomography Algorithms

The reconstruction and tomography algorithms are described in the NGAO Real Time Control Algorithms Document [1].

6.3 *WFS Interfaces*

The interfaces to the WFSs are shown in the following figures: LOWFS in Figure 19, Point-and-Shoot HOWFS in Figure 23, and Tomography HOWFS in Figure 21. Input will be via the machine-vision industry’s standard Camera Link cabling and communications protocol using the full configuration, unless otherwise noted. Output will connect to the tomography engine also via Camera Link communication. The 24 bits data width in this standard can represent two camera pixels each with a dynamic range of up to 2^12 = 4096 counts per pixel, which is above the maximum counts anticipated in NGAO wavefront sensing. The computed result from the WFSP does not increase the measurement limited signal-to-noise ratio so this 24 bit word width is also sufficient for transfer to the tomography engine, even though the tomography engine will ultimately be using a 36 bit internal word width to maintain computational accuracy.

The Camera Link interface cable is serialized low-voltage differential signaling (LVDS) pairs on an engineered standard cable (e.g. 3M's Mini D Ribbon). The base configuration allows a throughput of up to 2.04 Gbit/s. For the 256x256 WFS chip running at 2 kHz frame rate, and 12 bits per pixel, the pixel data rate is 1.573 Gbit/s, which is under the base configuration limit and so does not demand an enhanced bandwidth configuration. Since the bit rate of Camera Link is generally faster than the processor clock (in the case of FPGAs), standard serializer/deserializer (SERDES) transceivers are needed to terminate the connections on each board.

Wavefront sensor processor cards are assigned one per wavefront sensor, while the tomography engine processors are mapped over the aperture, it is necessary for the four tomography WFSPs (each containing a full aperture's worth of data) to distribute their results over the tomography engine processors. The distribution is accomplished using a specialized interface located along the left side of the tomography engine array, as shown in Figure 37. This interface makes sure the correct data are shifted in along each row of tomography processors. The cable connection from the WFSPs to the tomography engine uses the same LVDS / Camera Link 2 Gbit/s standard as used for the input to the WFSPs, only this time there is far less data to transfer. The full-frame transfer of 64x64x2x12bit numbers (an extra factor two for complex Fourier coefficients) is accomplished in 50 microseconds, about 10% of a 2 KHz frame cycle. A doublewide Camera Link interface can be used if this is deemed to take too much of the latency budget.

Wave front sensors will accept the following data to adjust the camera data and centroids:

1. A dark current array containing values to adjust the camera data

   This array will be a combined value of back ground and dark current to adjust the camera data. It is calculated by the AO Control.

2. A centroid zero point array containing reference centroids for the centroid calculations.

   This array will be a combined value of reference centroids, offsets to compensate for non common path errors and long term systematic errors. It is calculated by the AO Control.

3. A flat field array containing data to adjust the individual pixel gains.

The RTC has sufficient configurability for diagnostics to set up all of the calibration data-taking modes envisioned. Any input of any of any sub computation (e.g. pixel processing, centroiding, wavefront reconstruction, tomography, DM command generation) can be driven to a setting (e.g. flatten the DMs), and any output can be captured. The averaged background data (or averaged centroid data, averaged wavefront data, etc.) are available through he offloading ports, with settable averaging times. The RTC will act as a dumb "slave" to the calibration scripts kept by the AO Controller, which in turn uses the information gathered to create data that must be loaded into the RTC, such as the reference centroids, sky backgrounds, etc.

6.4 *Data Flow and Rates*

See Table 11.

6.5 *LOWFSs (Low Order Wave Front Sensors) (for T/T, Astigmatism and Focus)*

The LOWFSs take inputs from the 3 IR Tip/Tilt/focus and astigmatism cameras. The cameras feeding the LOWFSs are focused on natural guide stars (NGS). They generate low order control for the on-axis correction of the light for both science and AO paths.

6.5.1 Architecture and Design

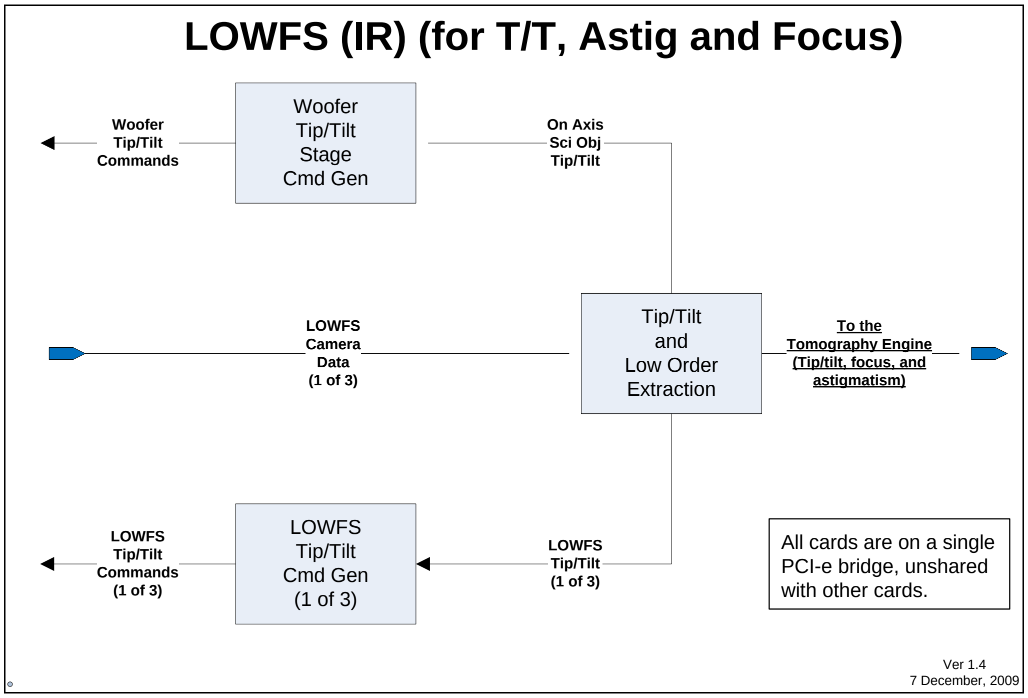

LOWFS (IR) (for T/T, Astig and Focus)
Woofer Tip/Tilt Commands
Woofer Tip/Tilt Stage Cmd Gen
On Axis Sci Obj Tip/Tilt
LOWFS Camera Data (1 of 3)
Tip/Tilt and Low Order Extraction
To the Tomography Engine (Tip/tilt, focus, and astigmatism)
LOWFS Tip/Tilt Commands (1 of 3)
LOWFS Tip/Tilt Cmd Gen (1 of 3)
LOWFS Tip/Tilt (1 of 3)
All cards are on a single PCI-e bridge, unshared with other cards.
Ver 1.4
7 December, 2009

Figure 19 LOWFS

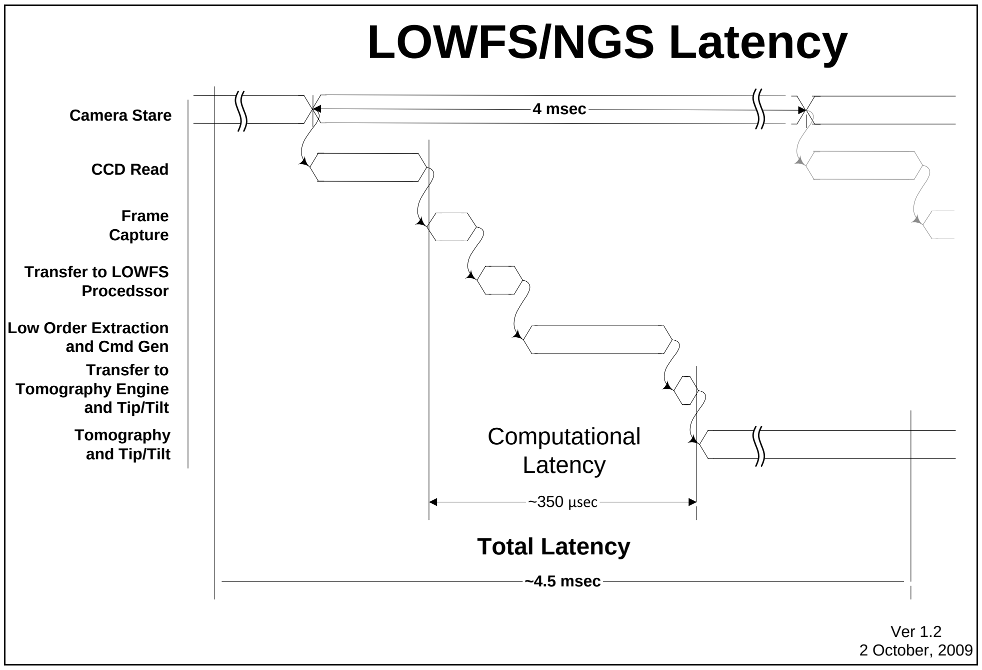


Figure 20 LOWFS Latency (not to scale)

6.5.2 Interfaces and Protocols

6.5.2.1 Inputs

The Inputs are from 3 IR cameras over Camera Link base configuration at up to 2 KHz frame rate: **FR-1410**

- Two tip/tilt cameras
- One tip/tilt, focus/astigmatism camera

6.5.2.2 Outputs

The outputs are:

- An analog X/Y controls to the on-axis Tip/Tilt stage of the Woofer at a 2 KHz frame rate.
- 3 analog X/Y controls to the tip/tilt mirrors associated with the 3 LOWFS cameras at a 2 KHz frame rate.
- Digital Tip/tilt, focus, and astigmatism information sent to the tomography engine at a 2 KHz frame rate to synchronize with the Tomography's frame rate.

6.5.3 Data Flow, Rates and Latencies

6.5.3.1 Camera Data

The LOWFS detector size and LOWFS data rates have not yet been determined. The specifications on the LOWFS to RTC data flow interface is work that will have to follow this determination by the LOWFS design team.

The LOWFS IR cameras have frame rates of from 25 Hz to 2 kHz (**FR-1410**).

6.5.3.2 Tip/Tilt Signals

The tip/tilt signals are analog voltages < ±10 V, with a rise/fall time of < 10 µsec, and a frame rate of 2 KHz.

6.5.3.3 Astigmatism and Focus

The astigmatism and Focus data are sent to the tomography engine as digital data over 1 LVDS port.

6.5.4 Low Order Command Generation

Figure 19 shows the process of extracting tip, tilt, focus, and astigmatism from the Hartmann information coming from the TTFA LOWFS. This is accomplished in a dedicated RTLinux CPU, as shown in Figure 8.

6.6 *HOWFSs (High Order Wave Front Sensors)*

Seven HOWFSs generate seven wave fronts used by the AO RTC system.

- Four tomography HOWFSs generate wavefronts used by the tomography engine that in turn generates on axis wavefronts used by the woofer and high order DM(s) to correct for atmospheric aberration affecting the science object(s). Currently there is only one science object, but we have the possibility of supporting more if needed.
- Each of three point-and-shoot HOWFSs compensate for atmospheric aberrations affecting a corresponding LOWFS NGS used for T/T, astigmatism, and focus.

To generate these wave fronts, the HOWFSs take inputs from the tomography or point-and-shoot cameras. These cameras are focused on laser guide stars (LGS).

The images are first corrected for dark current and background. They are then thresholded, and processed to produce centroids corrected for reference centroids and offsets.

Dark current and background values are combined by the AO Control prior to sending to the RTC. References to "dark current" include the combined background values.

Reference centroids and offsets are combined by the AO Control prior to sending to the RTC. References to "reference centroids" include the combined offset values.

Tip/Tilt are then extracted from the centroids and subtracted from them.

The Tip/Tilt removed centroids are then processed by the wavefront reconstruction algorithm to produce a corresponding tip/tilt removed wavefront.

The tip/tilt value is used to control the tip/tilt mirror associated with the HOWFS

The tip/tilt removed centroids are sent to a separate non-RTC focus processor to extract focus. This will be used by the focus processor to sense any change in the sodium layer or persistent system focus. It will use this information to compensate for the focus by moving the HOWFS to a more correct focus position and/or updating the offsets and consequently the reference centroids. The path for this data to the focus processor is through the telemetry path. The RTC is not involved in changing the focus of the HOWFSs other than to supply information to the focus processor.

The focus processor(s) are not part of the real time system.

6.6.1 Centroider and Wave Front Reconstruction Engine

6.6.1.1 CPU Implementation

This would be a conventional implementation using vector-matrix multiplication for the reconstruction.

Currently, it is not clear that a CPU implementation will meet our requirements for the Tomography HOWFS. Consequently, we are examining both a GPU and an FPGA implementation for all HOWFSs.

6.6.1.2 GPU Implementation

A 64 subaperture across (88 across for zero padding) reconstructor has been implanted on a single Nvidia Graphics Processor card. This presents a simple solution for the HOWFSs that could be built with standard off the shelf components. It would potentially provide lower latency than the CPU implementation and fewer racks of equipment since multiple HOWFSs could be supported by a single computer.

The GPU would reside on a PCI-E backplane slot on a CPU motherboard. The CPU can be used to configure and run the GPU. Additional PCI-E slots would house Camera Link interface cards, which in turn connect to the WFS camera for input and to the tomography engine for output.

We are currently analyzing whether we can utilize the GPU in the system at a 2 KHz frame rate under a real-time OS.

6.6.1.3 FPGA Implementation

Using FPGAs is the base line implementation. It is possible to implement a centroider and reconstructor in a small number of chips that can implement a variety of algorithms. These would require custom boards but have the potential for the lowest latency implementation.

There is a small architectural difference between a Tomography HOWFS and a Point-and-Shoot HOWFS.

6.6.2 Tomography HOWFS

The Tomography HOWFS sends the reconstructed wave front to the Tomography Engine. See Figure 21.

Table 8 is the source of the computational Latency shown in Figure 22.

See Figure 2 for the Tomography HOWFS algorithm.

6.6.2.1 Architecture and Design

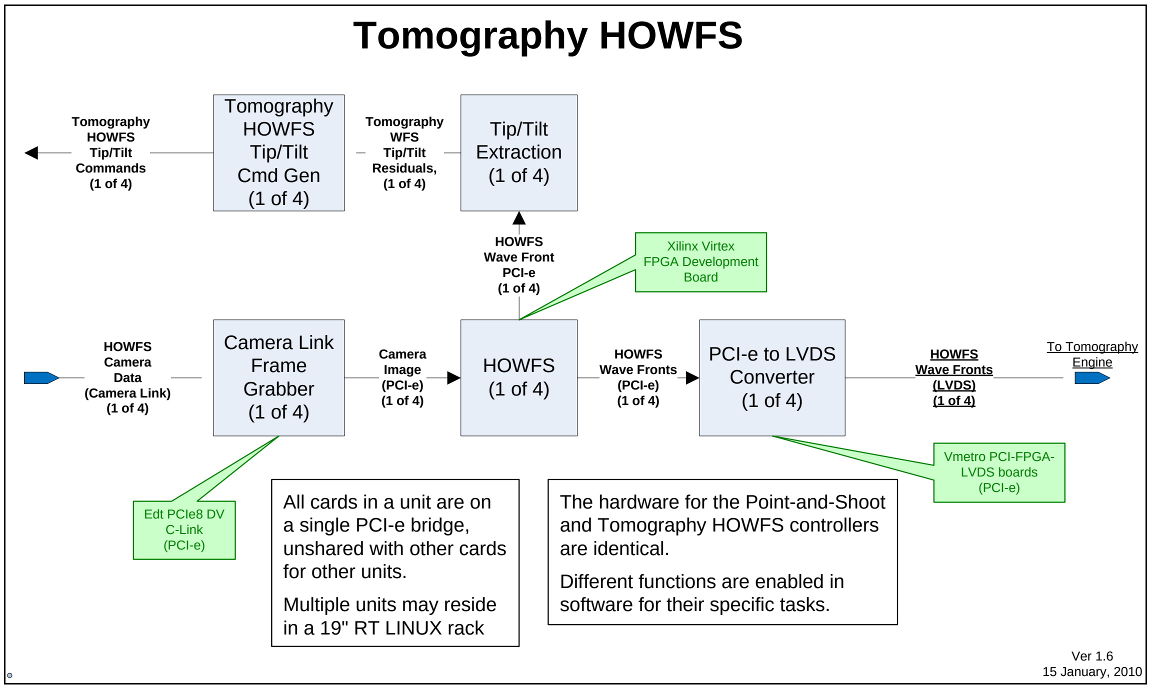

Tomography HOWFS
Tomography HOWFS Tip/Tilt Commands (1 of 4)
Tomography HOWFS Tip/Tilt Cmd Gen (1 of 4)
Tomography WFS Tip/Tilt Residuals, (1 of 4)
Tip/Tilt Extraction (1 of 4)
HOWFS Wave Front PCI-e (1 of 4)
Xilinx Virtex FPGA Development Board
HOWFS Camera Data (Camera Link) (1 of 4)
Camera Link Frame Grabber (1 of 4)
Camera Image (PCI-e) (1 of 4)
HOWFS (1 of 4)
HOWFS Wave Fronts (PCI-e) (1 of 4)
PCI-e to LVDS Converter (1 of 4)
HOWFS Wave Fronts (LVDS) (1 of 4)
To Tomography Engine
Vmetro PCI-FPGA-LVDS boards (PCI-e)
Edt PCIe8 DV C-Link (PCI-e)
All cards in a unit are on a single PCI-e bridge, unshared with other cards for other units.
Multiple units may reside in a 19" RT LINUX rack
The hardware for the Point-and-Shoot and Tomography HOWFS controllers are identical.
Different functions are enabled in software for their specific tasks.
Ver 1.6
15 January, 2010

Figure 21 Tomography HOWFS

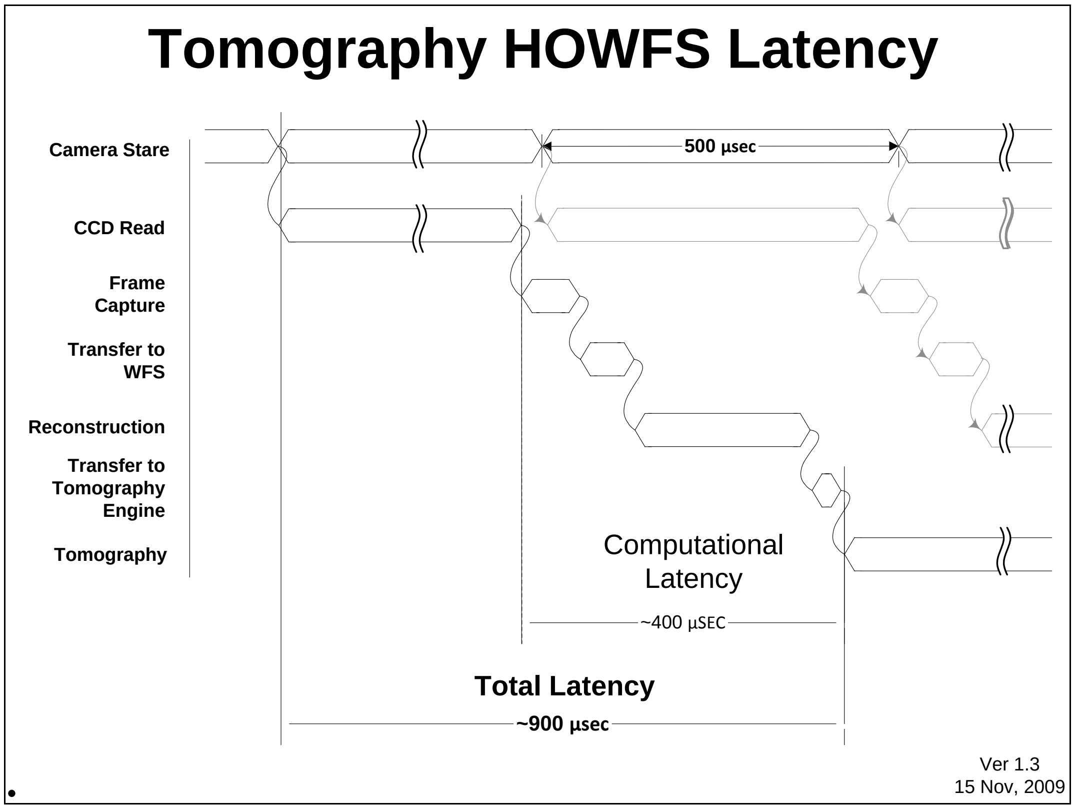


Figure 22 Tomography WFS Latency (not to scale)

6.6.3 Point-and-Shoot HOWFS

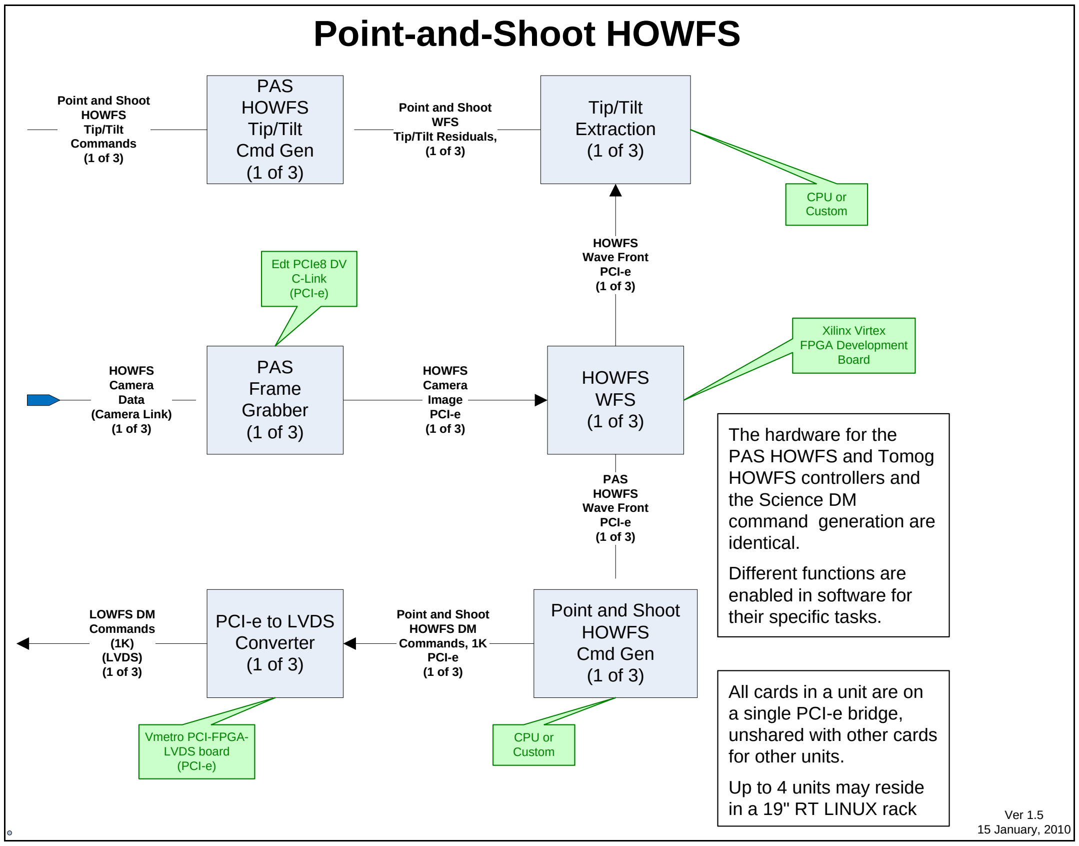
The Point-and-Shoot HOWFS, however, further processes the reconstructed wave front to generate DM commands, which are sent to a DM that corrects the laser spot associated with its corresponding camera, see Figure 23 and Section **8**. Figure 23 illustrates the Point-and-Shoot HOWFS latency.

There is currently no requirement for the Point-and-Shoot latency

See Figure 2 for the Point-and-Shoot HOWFS algorithm.

6.6.3.1 Architecture and Design

For the Point-and-Shoot HOWFS, we have a selection of implementations because of their smaller problem size. This could be FPGA, GPU or CPU. Currently there is no latency requirement for the Point-and-Shoot HOWFS and this will ultimately determine the final implementation.


Point-and-Shoot HOWFS
Point and Shoot HOWFS Tip/Tilt Commands (1 of 3)
PAS HOWFS Tip/Tilt Cmd Gen (1 of 3)
Point and Shoot WFS Tip/Tilt Residuals, (1 of 3)
Tip/Tilt Extraction (1 of 3)
CPU or Custom
HOWFS Wave Front PCI-e (1 of 3)
Edt PCIe8 DV C-Link (PCI-e)
Xilinx Virtex FPGA Development Board
HOWFS Camera Data (Camera Link) (1 of 3)
PAS Frame Grabber (1 of 3)
HOWFS Camera Image PCI-e (1 of 3)
HOWFS WFS (1 of 3)
PAS HOWFS Wave Front PCI-e (1 of 3)
LOWFS DM Commands (1K) (LVDS) (1 of 3)
PCI-e to LVDS Converter (1 of 3)
Point and Shoot HOWFS DM Commands, 1K PCI-e (1 of 3)
Point and Shoot HOWFS Cmd Gen (1 of 3)
Vmetro PCI-FPGA-LVDS board (PCI-e)
CPU or Custom
Ver 1.5
15 January, 2010

The hardware for the PAS HOWFS and Tomog HOWFS controllers and the Science DM command generation are identical.

Different functions are enabled in software for their specific tasks.

All cards in a unit are on a single PCI-e bridge, unshared with other cards for other units.

Up to 4 units may reside in a 19" RT LINUX rack

Figure 23 Point-and-Shoot HOWFS

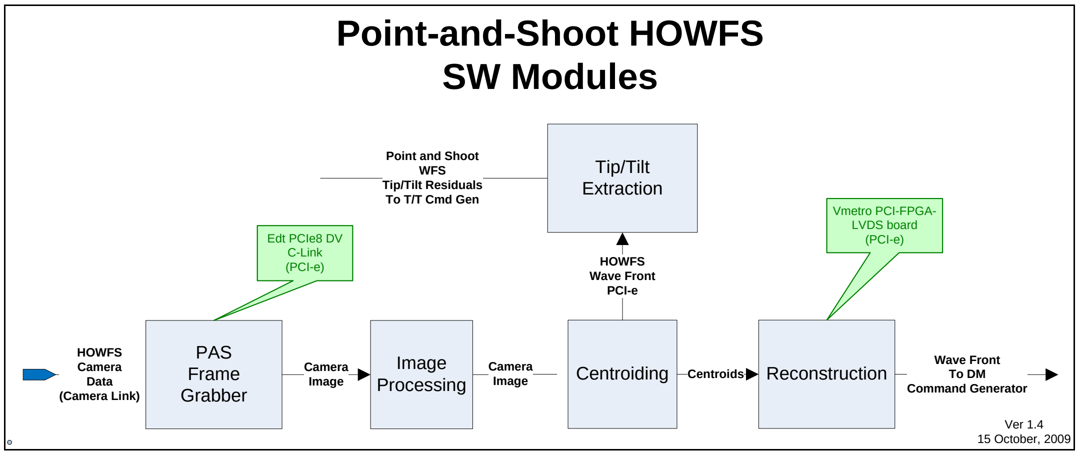


Figure 24 Point-and-Shoot HOWFS SW Modules

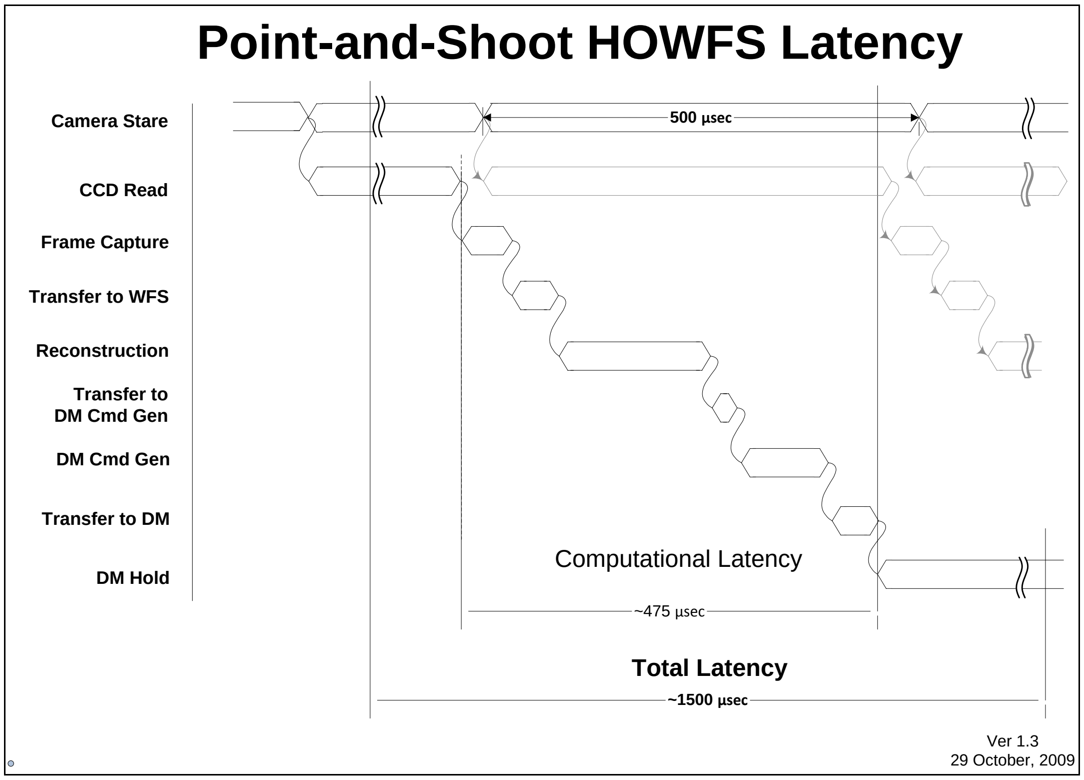


Figure 25 Point-and-Shoot HOWFS Latency (not to scale)

6.7 *NGS Wave Front Sensor*

In NGS mode, one of the Tomography WFS will operate in 32x32 mode and will take its input from the NGS camera. It will execute the same centroiding and reconstruction algorithm as the Point-and-Shoot WFSs, and will pass its output to the tomography engine and tip/tilt information to the science Tip/Tilt mirror system. The rest of the Tomography and Point-and-

Shoot HOWFSs and the LOWFS will do nothing in this mode. The Tomography Engine will pass the reconstructed wave front unmodified to the Woofer/Tweeter process and on to the DM command generators.

See Figure 21 for the architecture.

There is no current requirement for the NGS WFS latency.

See Figure 2 for the NGS algorithm.

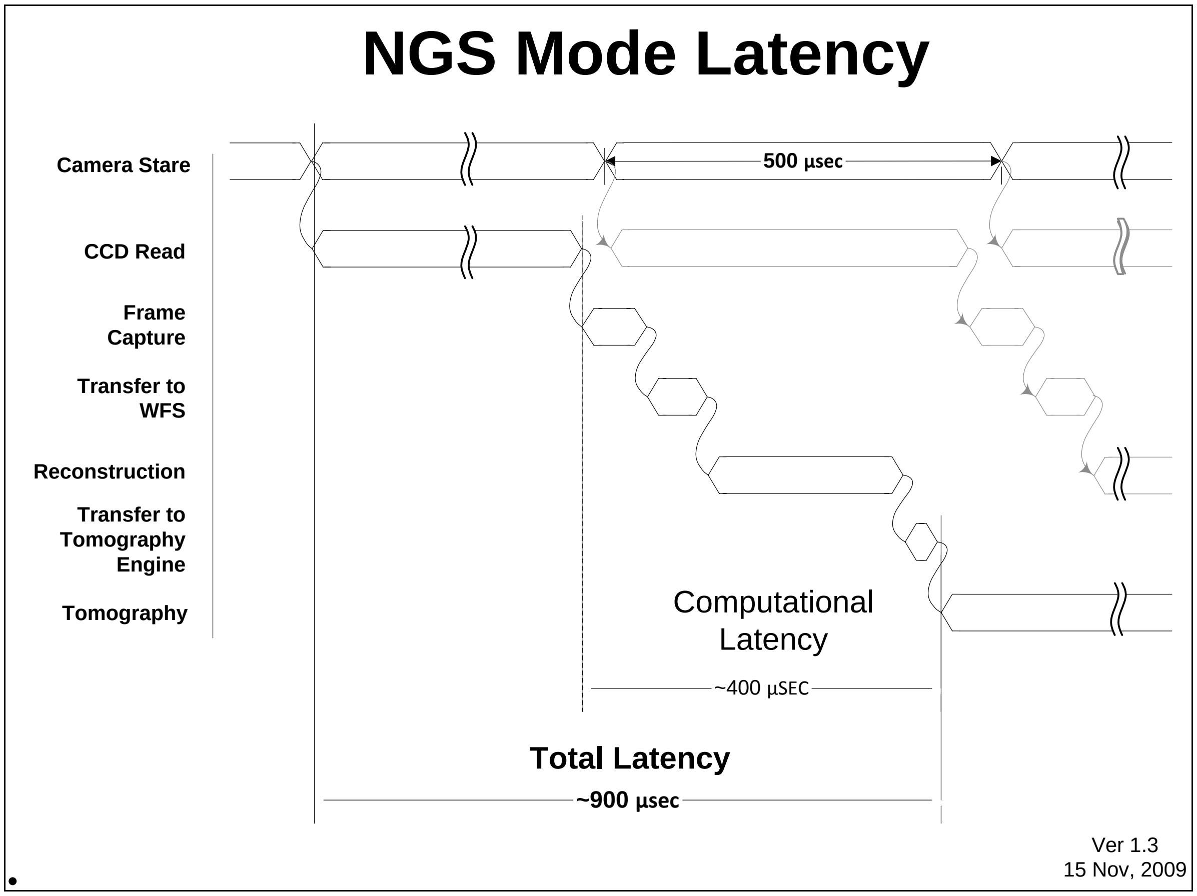


Figure 26 NGS Mode Latency

Simple description and diagram

7. The Tomography Engine (TE)

The tomography engine takes wave fronts from several tomography HOWFS and recreates an estimate of the volume of the atmosphere, in the field of view, that would have created those wave fronts. It does this using tomographic techniques similar to those of a CAT scan. In our case it estimates the optical path delay (OPD) of volume elements of the atmosphere instead of the density of the volume elements of the body.

The wave fronts from the tip/tilt, focus, and astigmatism point-and-shoot HOWFSs are not sent to the Tomography Engine as a result of the “Build-to-Cost” decisions, see “Wizinowich, P. *Design Changes in Support of Build-to-Cost.* Keck Next Generation Adaptive Optics Project : KAON 642, 2009”.

The volume estimate is divided up vertically into layers and horizontally into subapertures. The intersection of a layer and a sub aperture is a volume element called a Voxel.

Once this estimate has been generated, it can be used to correct for the atmospheric aberrations in the direction of a science object within the field of view. The path to the science object is propagated through the estimate of the atmosphere. This generates a wavefront of the atmospheric distortion along the science path and this wave front is sent to the deformable mirror (DM), which corrects for the distortion.

The tomography engine is implemented as a 3-dimensional systolic array [5] of processing elements (PEs) that are mapped to the physical volume of the atmosphere above the telescope. Each processing element handles the processing of several neighboring voxels in a layer.

This is not a typical tower or rack-mounted PC. It is specifically designed to implement very high-speed vector/matrix operations with very small latency requirements. The total throughput rate is in the terra operation per second range with latencies of less than 500 microseconds.

A Note on Systolic Arrays

> “A systolic system is a network of processors which rhythmically compute and pass data through the system.
>
> “Physiologists use the word ‘systole’ to refer to the rhythmically recurrent contraction of the heart and arteries which pulses blood through the body.
>
> “In a systolic computing system, the function of a processor is analogous to that of the heart. Every processor regularly pumps data in and out, each time performing some short computation so that a regular flow of data is kept up in the network.”
>
> *Kung and Leiserson*

This implementation is a scalable, programmable, and configurable array of processors operating in unison.

It is a system can be easily configured (by replication of rows, columns, and layers of PEs) to be able to work for any sized problem, from the small to the large.

It is programmable, so that changes in the algorithm can be loaded at run time without changing the hardware. Additionally, even the hardware is programmable in its architecture, interconnect and logic.

It is configurable so that the parameters can be easily changed to adapt to changing needs for a given system: number of sub apertures, guide stars or atmospheric layers; values for $\mathbf{C_n}^2$; mixes, heights and positions of natural and laser guide stars; height and profile of the sodium layer; height of turbulent layers; etc.

The algorithm we are executing allows all PEs to execute identical code, simultaneously on their respective data

The normal operation of the tomography engine (TE), ignoring initialization and specialized diagnostics, is frame based, with a maximum frame rate of 2 KHz, see. At the start of each frame, wave fronts from tomography HOWFS are sent to the tomography engine in parallel.

Simultaneously, data for the DMs, which was calculated by the TE in the previous frame, is sent to the Woofer and Tweeter DM command generators.

After the wave fronts have been loaded, the tomography algorithm creates a consistent 3D estimate of the atmospheric volume. This volume estimate is arranged by sub apertures and layers.

When the estimate is complete, the science object is forward projected through the estimated atmosphere and low and high order information is extracted from the wavefront. This is the end of a frame.

The low and high order wavefronts are then sent to the woofer and tweeter command generators respectively while the new wave fronts are loaded, starting a new frame.

7.1 *Processing the Data*

The tomography algorithm is not only a parallelizable algorithm, but it is what is called an embarrassingly parallelizable one. If we have N data elements to calculate for, we can actually use N processors in parallel with minimal interaction between them to reduce the total calculation time.

The algorithm also is an iterative one and converges step by step to a minimum variance estimation of the atmosphere based on the wave fronts from the tomography WFS, see Arithmetic Reconstruction Tomography (ART) [6] and [1]. A simplified diagram is presented in Figure 27.

In executing each iteration of the algorithm, we need to perform several tasks that are computationally intensive:

1. Apply a Kolmogorov filter to match our estimates to know atmospheric statistics
2. Account for the positions of our guide stars by shifting our forward and back projections
3. Precondition our errors before back projecting them
4. Scale the forward and back projections by layer to compensate for the cone effect

Each of these tasks can be done either in the spatial or Fourier domains. However, they are much less computationally intensive in the Fourier domain. Consequently, we perform most of our iterative loop in the Fourier domain.

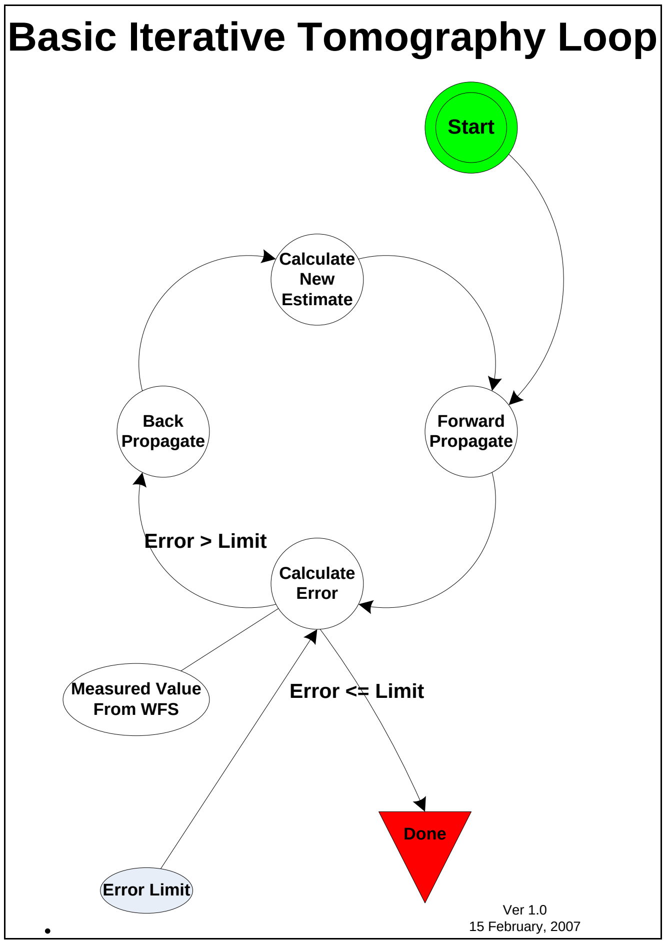


Figure 27 The basic iterative tomography algorithm

Because we solve our problem in the Fourier domain, we may get wrapping errors when paths are propagated through the replicated domains. This could result in forward propagated values in regions outside our aperture being back propagated into our estimated atmosphere. To avoid this, we may have to force those values to zero.

To do this we would take our forward propagated Fourier domain data back to the spatial domain and reapply our aperture, forcing values outside it to zero. Then we would apply the Fourier transform to the apertured spatial domain data so we could again back propagate it. If we precondition the data, we may have to apply the aperture again after preconditioning.

The augmented data flow for the Fourier domain is shown in Figure 28.

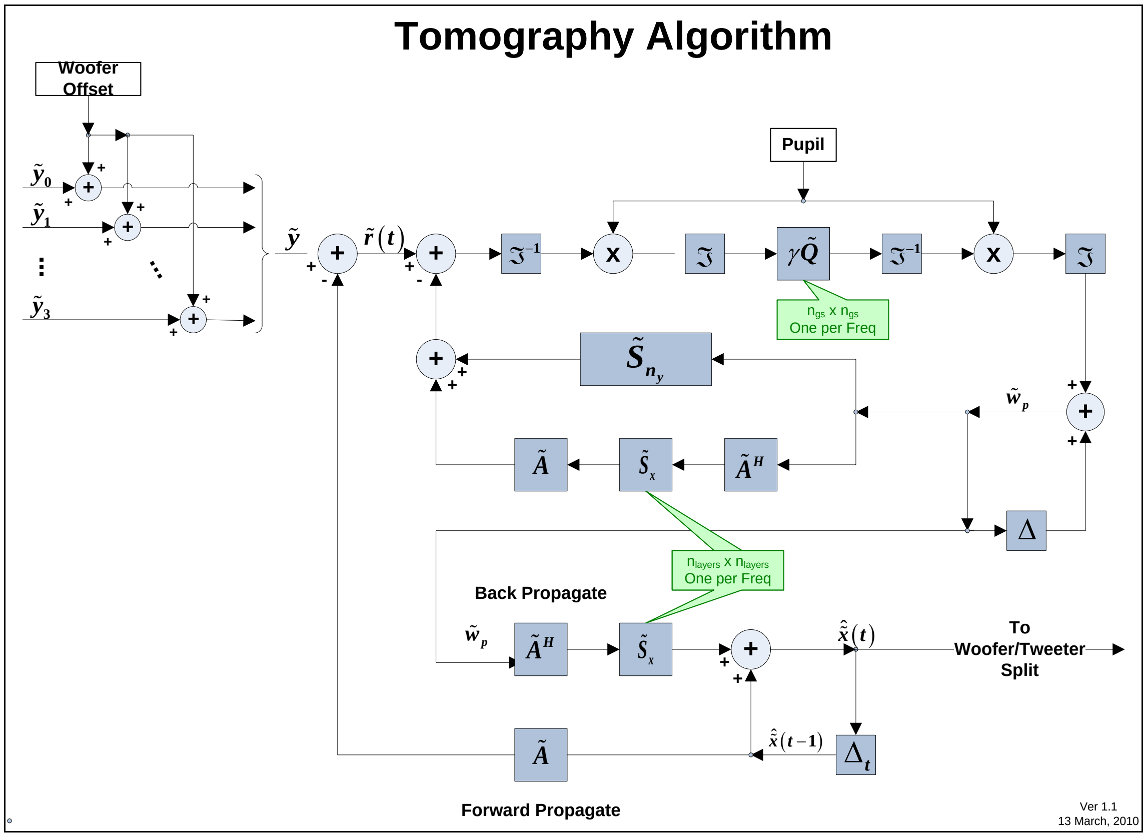


Figure 28 Actual Fourier Domain Data Flow with Aperturing

Applying the aperture in the iterative loop is the dominant element time-wise. The relative time to do the two 2D DFT^{-1}/DFT pairs in order to apply the aperture can be seen in Figure 29.

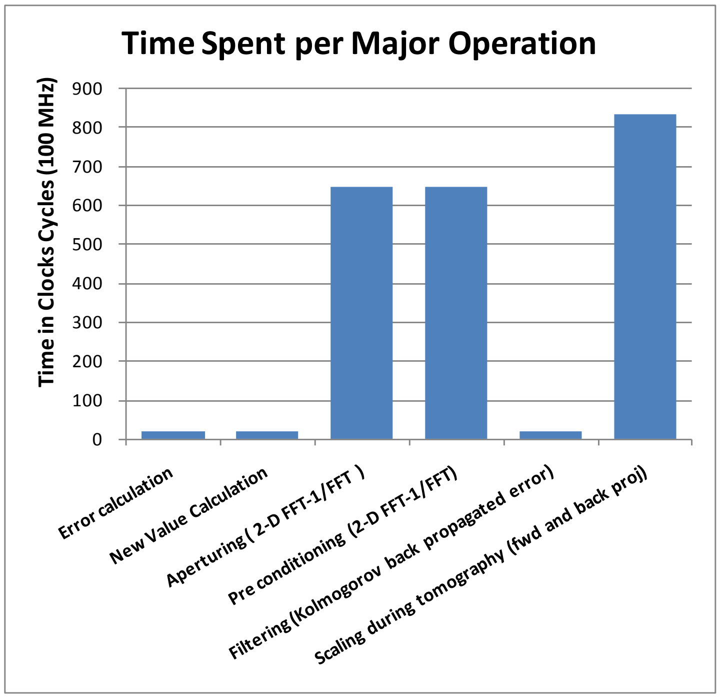


Figure 29 Relative time of components of iterative loop

However, the iterative algorithm will converge to a minimum variance solution without applying the aperture, perhaps taking more iterations. We are currently evaluating the accuracy, stability, and relative speed of the loop both with and without applying the aperture.

7.1 ***Architecture***

We have modeled the implementation of our algorithm after the tomography problem itself. We divide the atmosphere into sub-domains called voxels. We use an iterative algorithm to calculate the influence of each voxel. These calculations are highly parallel, very simple, have minimum memory requirements, and minimum communication requirements with other voxels. Our implementation of the algorithm follows this model. We assign a very simple processor called a processing element (PE), see Figure 30, to a small number of adjacent voxels. We place a large number of these processors on a field programmable gate array (FPGA). Each FPGA contains all of the voxels for one column of the atmosphere above the telescope. Multiple FPGAs, then, are used to hold all of the voxels for the space above the telescope.

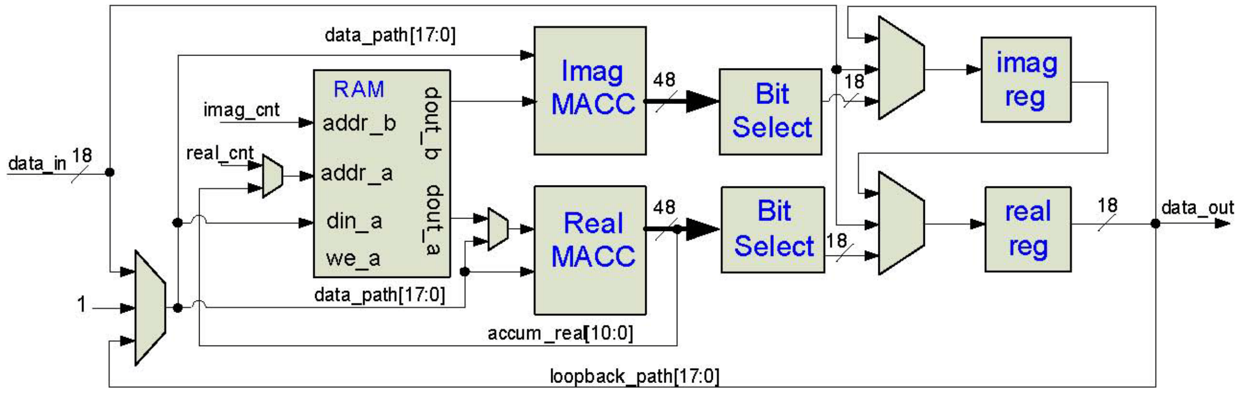


Figure 30 Processing Element (PE) diagram

This implementation maintains the visibility of all aspects of the original problem through the various phases of the solution.

As stated above, we solve our problem using domain decomposition. We have 'N' elements in our domain, we have 'm' processing elements (PEs), and we divide the task into 'm' parts with each part handling N/m elements.

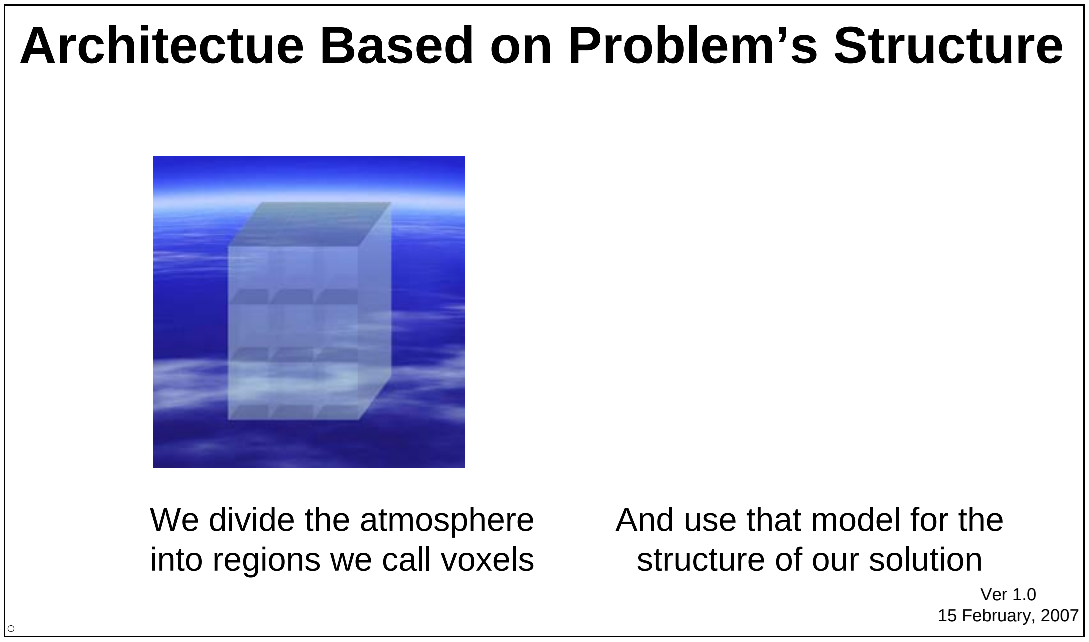
Memory is completely local to the Processing Elements (PEs), so no global memory access is required by any PEs.

All PEs use exactly the same algorithm and lines of code, with no branching. Therefore, all PEs are executing the same line of code at the same time and so finish at exactly the same time. Consequently, the program could be located in a single external controller [7]. However, for timing and performance reasons and in order to minimize the busing resources used, we duplicate the control logic for each sub domain: one controller per FPGA with hundreds of PEs on each FPGA served by that controller.

The control logic for these domains is very small; and does not impact the density (PEs per chip) of the implementation significantly.

7.2 *Macro Architecture*

At the top level is the domain of the entire 3D atmospheric volume we are estimating. This is further divided into layers and sub apertures.



Architectectue Based on Problem's Structure

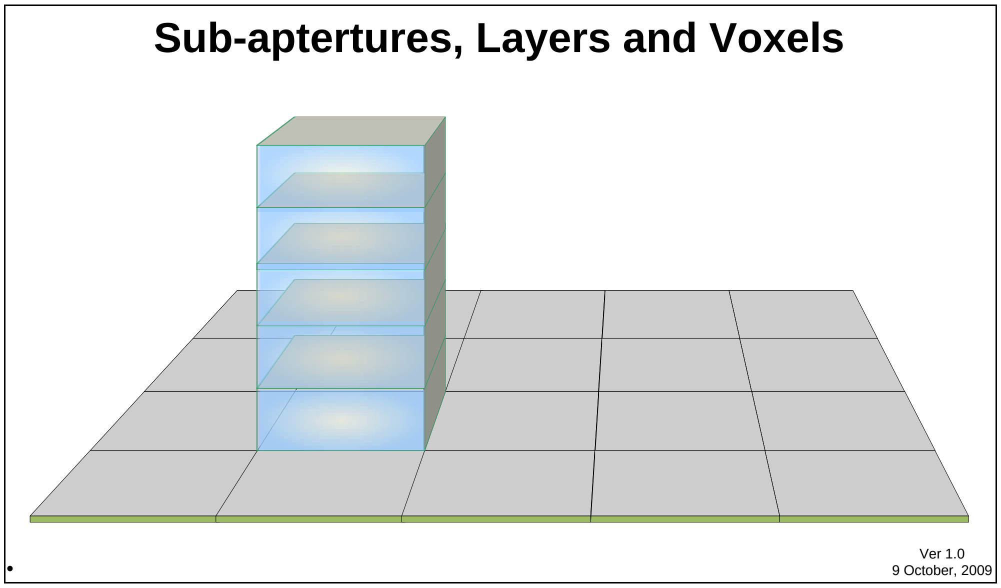
We divide the atmosphere into regions we call voxels

And use that model for the structure of our solution

Ver 1.0
15 February, 2007

Figure 31 Modeling the solution after the form of the problem

For our purposes, we divide the atmosphere into columns. Each column is one sub aperture square at the ground and extends vertically through the various layers of the atmosphere, see Figure 32. The intersection of a column and a layer defines a voxel and we assign a processor to each voxel to calculate its OPD (<u>O</u>ptical <u>P</u>ath <u>D</u>elay). The voxel is the smallest sub domain we deal with.



Sub-aptertures, Layers and Voxels

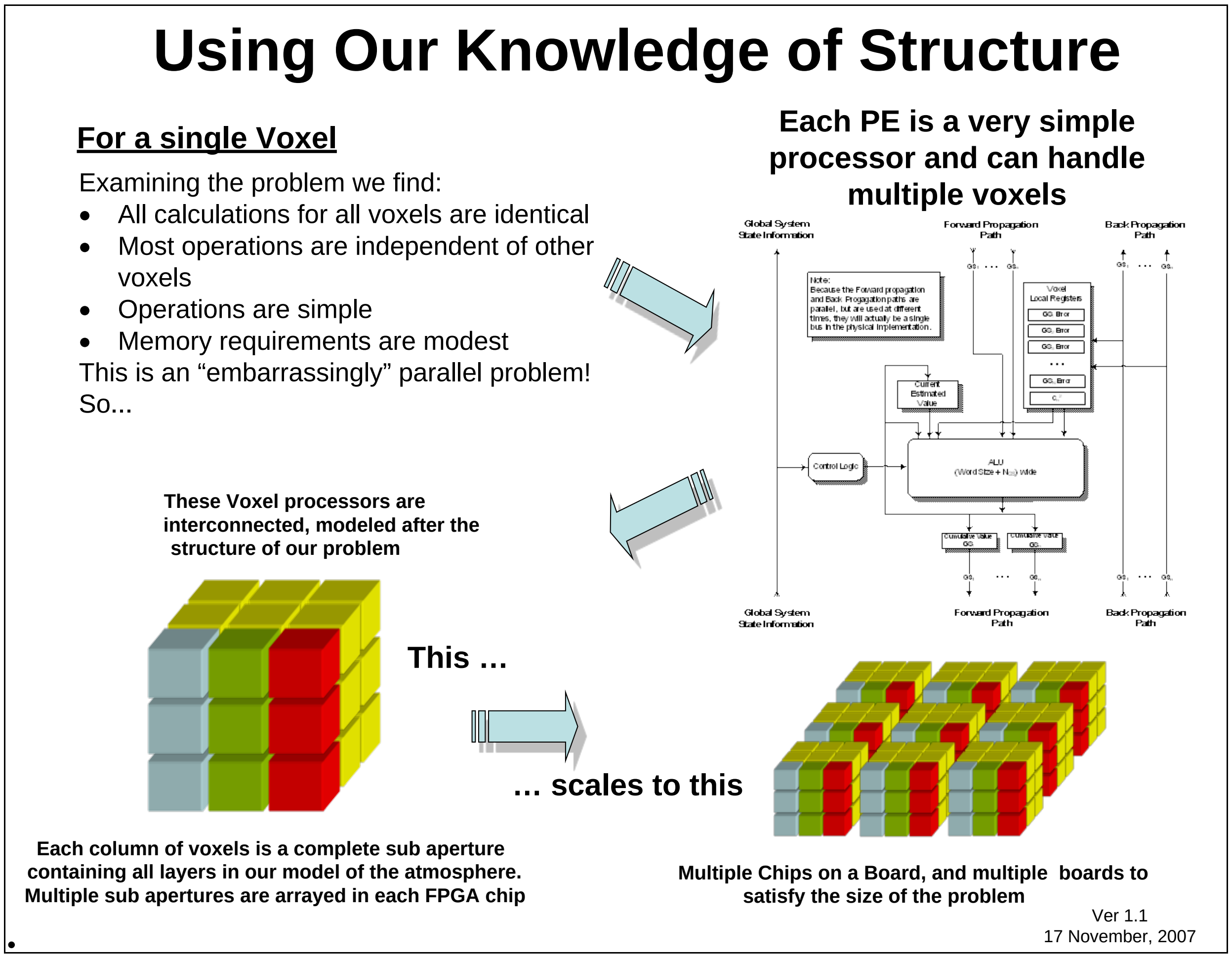
Ver 1.0
9 October, 2009

Figure 32 A column of combined layer and guide star processors form a sub aperture processor

We assign one processor to handle all computations for some number of voxels in a layer arranged in a square. If we have 5 layers, a column of 5 processors might handle a single column of voxels for a single sub aperture or it might handle all processing for a column of 9 sub apertures (45 voxels). Each FPGA handles a sub-domain consisting of a square region of adjacent columns, which as described contain all layers of the atmosphere for that region.

We take an array of many of these adjacent regions and place them in an FPGA, which can support hundreds to thousands of processors. Each FPGA then supports calculations for some larger portion of the atmosphere.

We then array these FPGAs on a circuit board to support a larger volume of the atmosphere and in turn array these circuit boards to support the total volume necessary.



Using Our Knowledge of Structure

For a single Voxel

Examining the problem we find:
- All calculations for all voxels are identical
- Most operations are independent of other voxels
- Operations are simple
- Memory requirements are modest

This is an "embarrassingly" parallel problem!
So...

Each PE is a very simple processor and can handle multiple voxels

These Voxel processors are interconnected, modeled after the structure of our problem

This ...

... scales to this

Each column of voxels is a complete sub aperture containing all layers in our model of the atmosphere. Multiple sub apertures are arrayed in each FPGA chip

Multiple Chips on a Board, and multiple boards to satisfy the size of the problem

Ver 1.1
17 November, 2007

Global System State Information
Forward Propagation Path
Back Propagation Path
Note: Because the Forward propagation and Back Progagation paths are parallel, but are used at different times, they will actually be a single bus in the physical implementation.
Voxel Local Registers
GS Error
Current Estimated Value
Control Logic
ALU (Word Size + N_GS) wide
Cumulative Value GS

Figure 33 Using our knowledge of structure

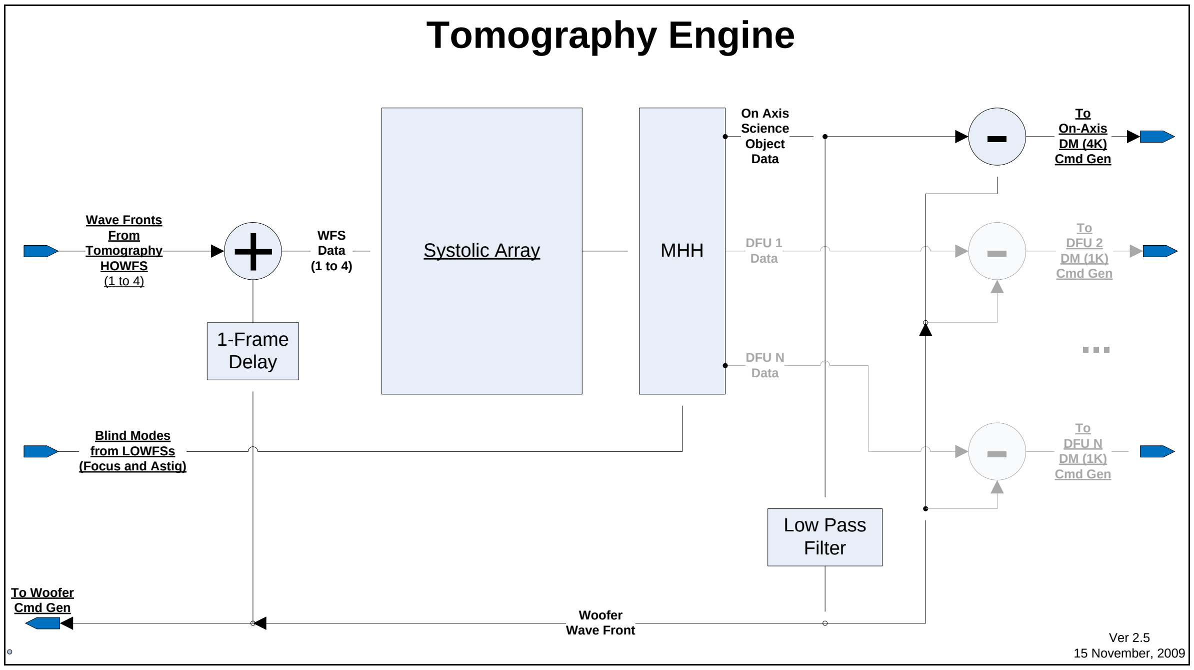


Figure 34 Tomography Engine Block Diagram

7.2.1 Inter Element Communications

In a 3D systolic array, each processor in the array must communication with its six orthogonal neighbors: north/south, east/west, up/down. Within an FPGA, there are an abundance of routing resources and this is not a problem. However, to communicate between the voxels on one FPGA and the neighboring voxels on another, we have a limit set by the I/O pins available on the FPGA. This also holds for the voxels at the boundaries between adjacent printed circuit boards.

Figure 35 shows the array interconnects for a layer. The bandwidth across all interfaces must be the same to insure efficient operation. However, the data path width and transfer clock rate across chip and board boundaries can be different to accommodate the physical realities of the system.

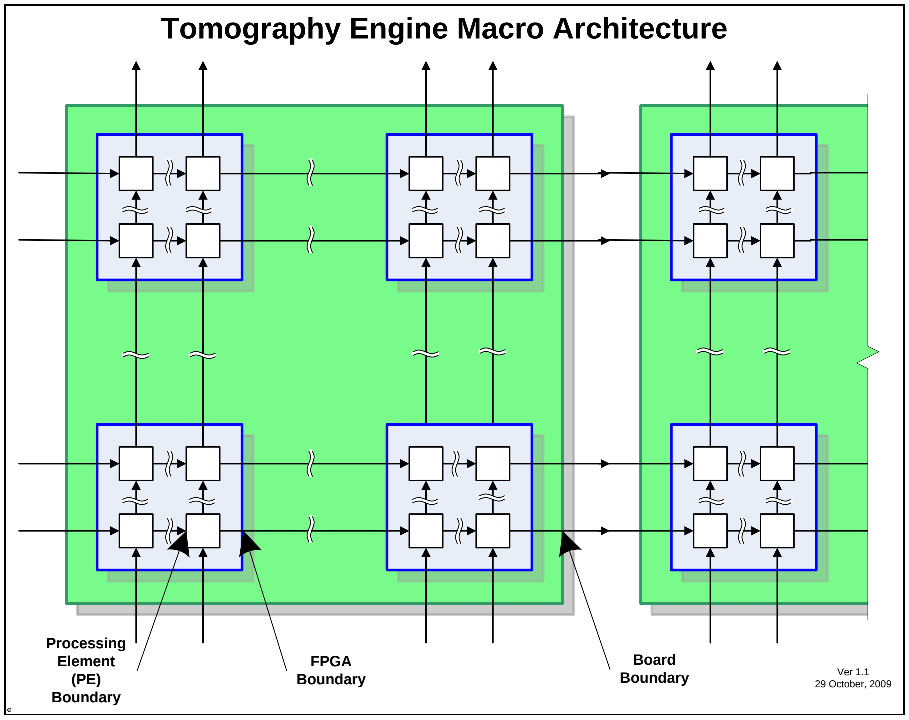

Tomography Engine Macro Architecture
Processing Element (PE) Boundary
FPGA Boundary
Board Boundary
Ver 1.1
29 October, 2009

Figure 35 The Systolic Array, showing boundaries for different elements

Sub aperture processors (black), chips (blue), and boards (green)

Each processor for a sub aperture must talk to each orthogonal neighbor.

Each FPGA has **P** I/O pins. For L layers and m sub apertures per side of the square area the FPGA covers we can calculate the number of I/O pins available per voxel. This gives each voxel a bus size of $\frac{P}{(4Lm)}$ I/O pins to communicate with each neighbor on a different FPGA.

An example could be an FPGA that has 600 pins we can use for this communication. Each device handles five layers and nine sub apertures. Therefore, the bus size between adjacent chips would 10 bits per voxel. This allows eight pins for data and two for clock and control. However, each voxel handles a complex value. So we need to transfer 2 18-bit words. As a result, we serialize the data across the chip boundaries, while internally, we use a wide bus.

The number of iterations in a frame is set to a fixed number. This number is set to guarantee that all iterations will complete before the start of the next frame. During anomalous situations, such as initial startup or an obscuration traveling rapidly across the aperture, the system may not be able to converge to our lower error limit in a single frame and must continue at the start of the next frame. We don’t want the frame sync to occur in the middle of an iteration, leaving the machine in an uncertain state.

Given the immense data bandwidth and compute requirements, we use a Harvard architecture (program and data memory are separate), SIMD model processor for the basic processing element (PE) throughout the array. Independent communications paths for code and data and allows us to perform operations on all data elements in the volume simultaneously on each instruction.

As described, the system is implemented using direct communications only between the immediately adjacent cells in the x, y and z directions: four horizontally orthogonally adjacent voxels and the voxel above and below.

These sub aperture columns are arranged in a square on an x,y-grid, which corresponds directly to the sub apertures on the telescope. The actual number of sub apertures used is larger than the number of sub apertures on the instrument, since we must allow for the fact that our volume will typically be a trapezoidal and be wider at the top, see Figure 36.

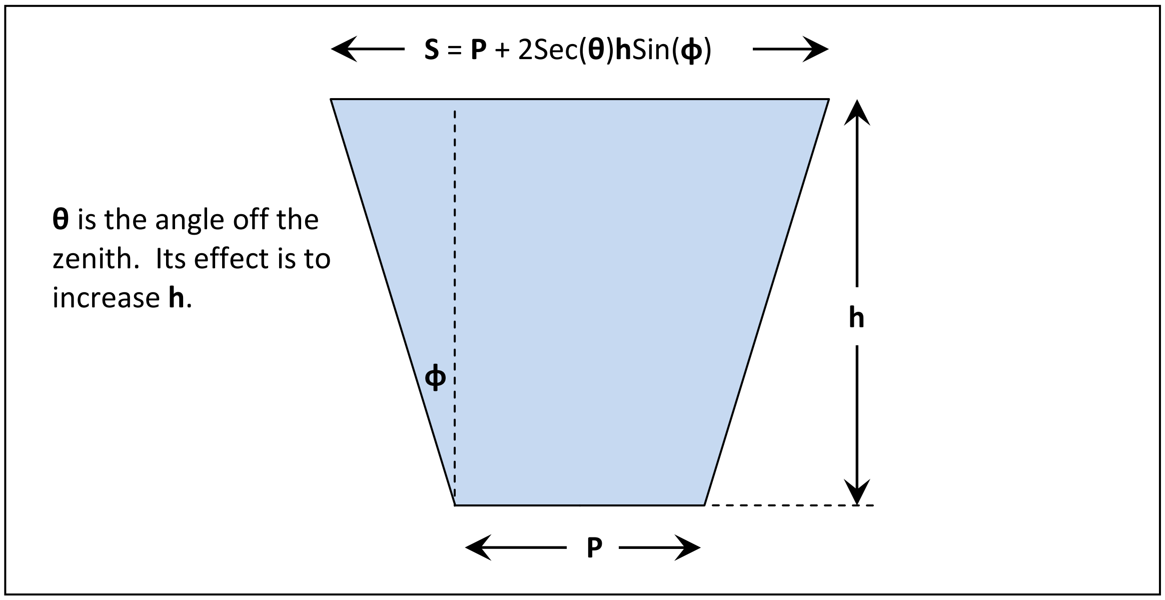

$\mathbf{S} = \mathbf{P} + 2\mathrm{Sec}(\boldsymbol{\theta})\mathbf{h}\mathrm{Sin}(\boldsymbol{\phi})$
θ is the angle off the zenith. Its effect is to increase **h**.
ϕ
h
P

Figure 36 Extended Sub Aperture due to FOV

7.2.2 Layer-to-layer and Guide-Star-to-Guide-Star Communications

As stated, we divide our problem into sub-domains. Each FPGA handles a sub-domain consisting of a region of all layers in m x m spatial or spatial-frequency sub-apertures.

We create a three-dimensional grid of N X N X L processors, where N is the number of sub apertures and L is the number of layers. Actually, L is MAX(Layers, guide stars); however, since we will normally have more layers than guide stars (to insure an underdetermined problem), we use L here for convenience.

Consider a vertical projection of a single sub aperture through our volume. It consists of the area/volumes of each layer above it in our model. It also corresponds to a particular sub aperture in each of our WFSs.

Each processor for a sub aperture must talk to each orthogonal neighbor.

Each FPGA has **P** Input/Output pins. This gives each processor a bus size of $\frac{P}{(4Lm)}$ Input/Output pins to communicate with each neighbor. Our device (a Xilinx® Virtex-5 or Virtex-6) will have 600 or more pins we can use for this communication between chips. Each device will handle five layers and nine or more sub apertures. Therefore, our bus size between adjacent chips is approximately 10 pins. This allows eight pins for data and two for clock and control.

This communication path is used for DFT calculations, loading and unloading data, and performing interpolation to compensate for the cone effect for laser guide stars.

7.3 *Program Flow Control*

Each FPGA has a simple control processor that supplies the control words for all the PEs on the chip. The structure of the control word is a wide-word, with minimal local decoding at the PE level. All control processors, one for each FPGA, are globally synchronized at the start of each frame.

For robust operations, if a state machine finds itself in an invalid state, a log of the event is kept and a signal is made available outside the system. System logic or the operator can determine the best action based on frequency or location of the occurrence.

There is no need (or ability) for each voxel processor (or sub aperture processor) to have branch capability or individual register addressability within the array. All PEs execute the same code on exactly the same register address in synchronism.

However, at global level, we do need to:

- Determine what diagnostic programs to run in the time between the end of convergence and the start of a new frame
- Synchronize control logic at the start of a frame and determine when we are too close to the end of a frame to start another iteration.

Each frame starts with the result of the previous frame, and in general, we will converge to a new estimate before the end of a frame. The remainder of each frame can thus be used for diagnostic and monitoring processes. Diagnostic and monitoring functions can be interrupted at the end of a frame and continued at the end of the next frame and so on until they complete.

Our algorithm requires that we convert between the spatial and Fourier domains. In our architecture, we do this by N shifts across rows to accomplish the transform. During this transformation, each CP calculates the running sum-of-squares for the error. When the transform is over, if the value is below our criteria, we stop iterations and start or continue any diagnostic or monitoring processes that need to be run.

We have three basic tasks that our architecture needs to support: I/O, DFT (both highly parallelizable) and a linear solver (embarrassingly parallelizable).

No local or global memory access is required to other Processing Elements (PEs), so memory can be completely local to the PEs.

Since all PEs execute the same program, the architecture is SIMD (Single Instruction Multiple Data) and the program could be located in a single external controller [7]. However, for timing and performance reasons and in order to minimize the busing resources used, we duplicate the control logic for each sub domain (individual FPGAs). As a side effect, this also gives us the ability to implement MIMD algorithms if needed.

The control logic for these domains is very small; and does not affect the density (cells per chip) of the implementation significantly.

7.4 *Implementation*

7.4.1 General Operation

The Tomography Engine is a 3D, systolic array of processor cells that calculates a 3-D estimate of the index of refraction of a volume of the atmosphere. The volume is broken into layers and each layer contains a number of voxels (volume elements that are one layer thick and one sub aperture square).

Since the calculations for the tomography are embarrassingly parallel, we dedicate a single processor for each voxel. Each processor is very simple and has enough local memory to do all its processing without needing access to global or a neighbor's memory. The nature of the parallel algorithms implemented determines the exact amount of this memory required.

The overall operation is on a frame basis, with a frame having a nominal duration of 500 µSec. The operations during each frame are identical to those of all other frames.

- At the beginning of a frame, data is shifted into one side of the array from the WFSs. As the WFS data is shifted in, the processed data is simultaneously shifted out of the other side for DM command generation. Thus the data load and unload portion of a frame are completely overlapped.
- During the rest of a frame, data circulates through the array for processing, shifting from cell to cell, by rows or columns during operation. The shifting of data during operation uses the same data paths that were used for the load/unload.

There is no global communication between the processors. All communication is local between adjacent processors.

7.4.2 The Systolic Array

The iterative solver in the tomography engine is easily parallelized. As soon as input data become available, calculations can be performed concurrently and in-place as data flows through the system. This concept was first defined by Kung at Carnegie-Mellon University:

A systolic system is a network of processors that rhythmically compute and pass data through the system. Physiologists use the word "systole" to refer to the rhythmically recurrent contraction of the heart and arteries that pulses blood through the body. In a systolic computing system, the function of a processor is analogous to that of the heart. Every processor regularly pumps data in and out, each time performing some short computation, so that a regular flow of data is kept up in the network [8].

At first, systolic arrays were solely in the realm of single-purpose VLSI circuits. This was followed by programmable VLSI systolic arrays [9] and single-purpose FPGA systolic arrays [10]. As FPGA technology advanced and density grew, general-purpose “reconfigurable systolic arrays” [11] could be put in single or multiple FPGA chips. The capability of each processing element in early FPGA systolic array implementations was limited to small bit-level logic. Modern FPGA chips have large distributed memory and DSP blocks that, along with greater fabric density, allow for word-level 2’s complement arithmetic. The design goals for our systolic array are:

- Reconfigurable to exploit application-dependent parallelisms
- High-level-language programmable for task control and flexibility
- Scalable for easy extension to many applications
- Capable of supporting single-instruction stream, multiple-data stream (SIMD) organizations for vector operations and multiple-data stream (MIMD) organizations to exploit non homogeneous parallelism requirements [12]

Because of tasks, such as multiple 2-D DFTs per Frame, the number of compute operations drastically outnumbers the I/O operations. The system is therefore “compute-bound” [13].

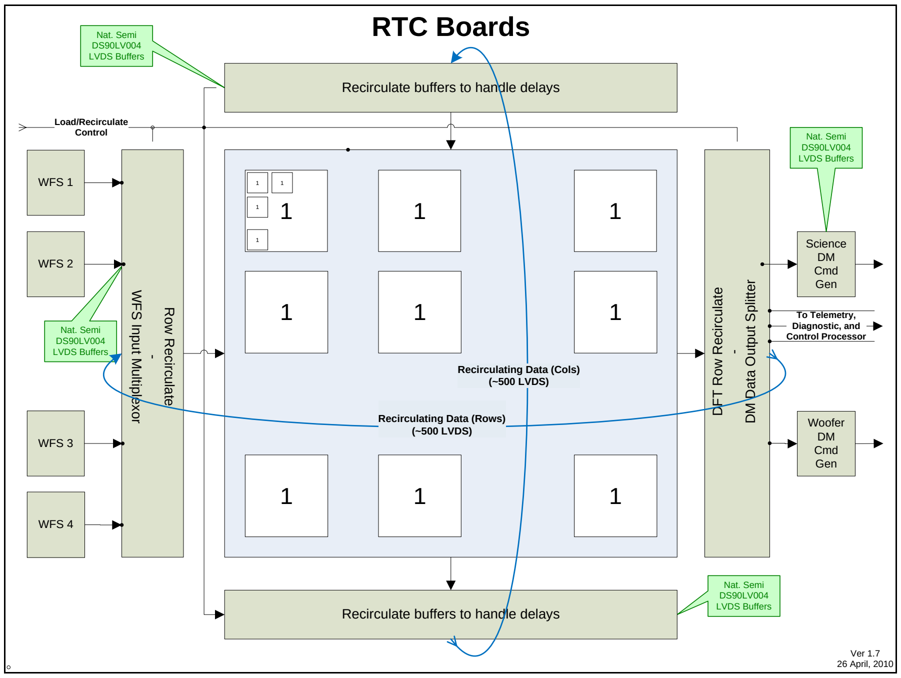

RTC Boards
Nat. Semi DS90LV004 LVDS Buffers
Recirculate buffers to handle delays
Load/Recirculate Control
WFS 1
WFS 2
WFS 3
WFS 4
WFS Input Multiplexor - Row Recirculate
1
Recirculating Data (Cols) (~500 LVDS)
Recirculating Data (Rows) (~500 LVDS)
DFT Row Recirculate - DM Data Output Splitter
Science DM Cmd Gen
To Telemetry, Diagnostic, and Control Processor
Woofer DM Cmd Gen
Ver 1.7 26 April, 2010

Figure 37 Tomography Engine Boards

7.4.3 The processing element (PE)

The heart of our systolic array is the Processing Element (PE). This is a very powerful cell, which can perform 18-bit pipelined operations at 500MHz. Each PE uses two multiply-accumulate units (MACs) for the complex arithmetic of the Fourier transform. A single FPGA chip can have over 500 of these MACs and allows us to have over 250 PEs per chip.

This is the power of the FPGA. While each FPGA might only be processing data at 100 MHz, each chip can contain >250 processors for a combined processing capability of >25 G Operations per second.

For the Xilinx chips we use, these MACs are called DSP48s. They have 18-bit MACs, with a 48-bit accumulate register to prevent overflow on long MAC operations.

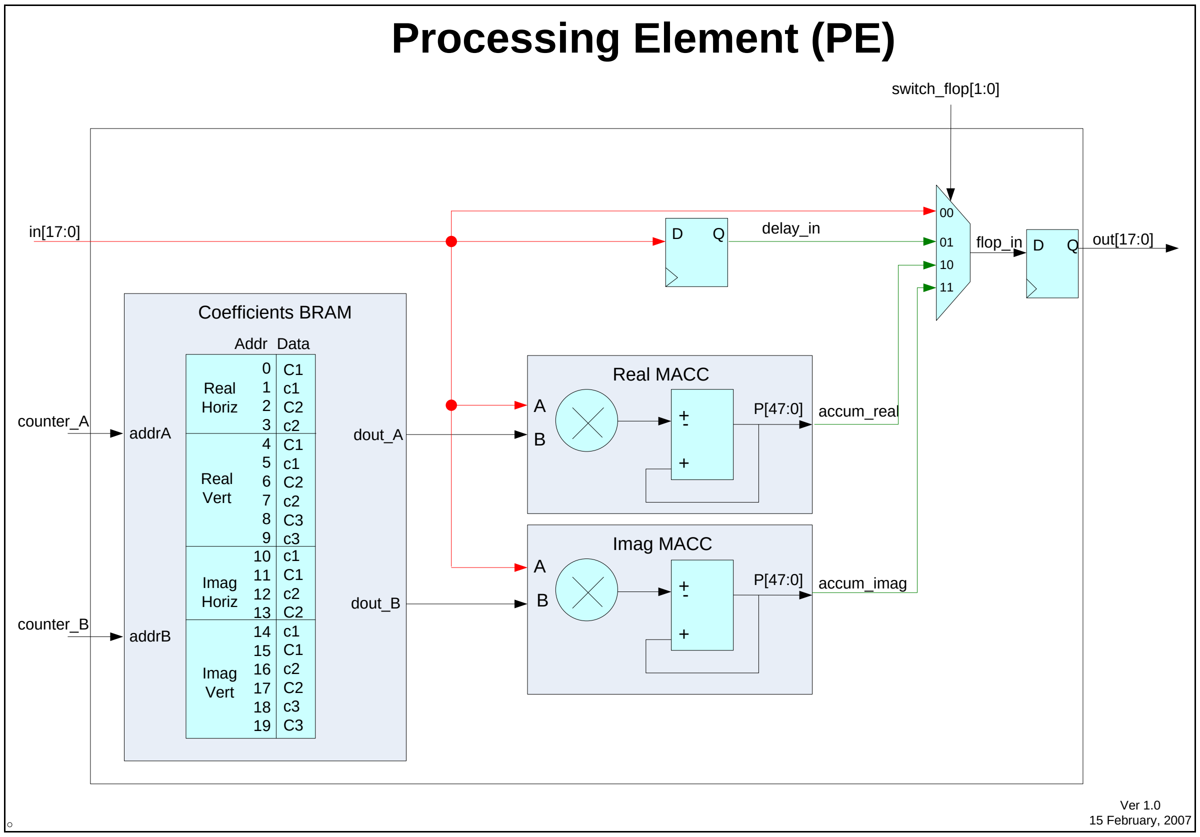


Figure 38 Processing Element (PE)

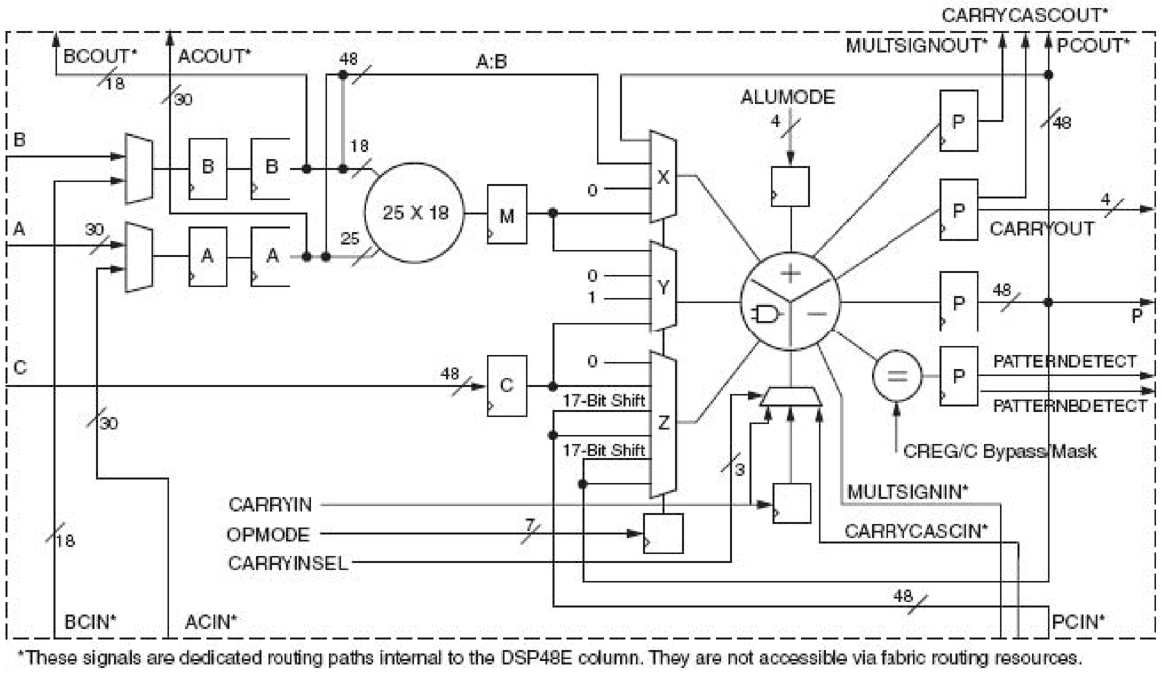


Figure 39 The Xilinx DSP48E architecture [14]

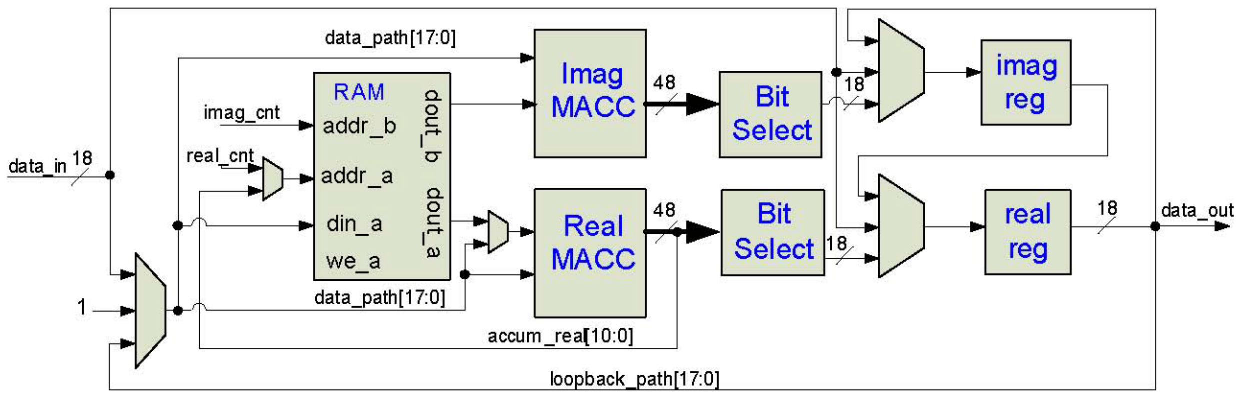
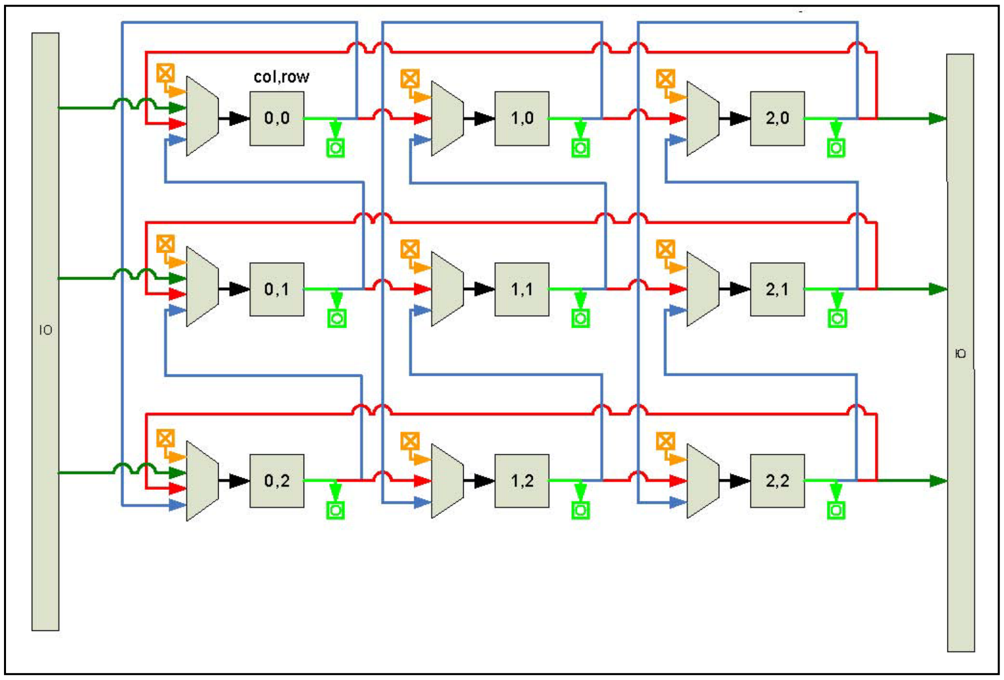


Figure 40 A single Processing Element (PE)

Two DSP48s are employed, one for the real and one for the imaginary part of the complex number. As shown in Figure 38, the PE also contains a dedicated Block RAM memory, an18-bit register for the real part, and an 18-bit register for the imaginary part. Multiplexors control whether these registers receive data from their respective MACC or if data are just snaked through them to the next PE. Note that there is only a single 18-bit input and single 18-bit output. This is because when data is flowing through the mesh, the real part is transferred on even clock edges and the imaginary part on odd edges. This is particularly important for instructions types such as the DFT where complex multiplication is performed on inputs split across two clock cycles.

7.4.4 PE interconnect

The switching lattice for a simple 3x3 layer of the tomography engine is shown in Figure 41. For a large telescope, this lattice could be 64x64 or larger. The individual PEs are labeled by their column and row position in the mesh. Each PE has a multiplexor on its input to route data orthogonally from a neighboring horizontal PE, vertical PE, or next layer PE. External I/O only takes place at the mesh boundary on the right and left sides. Information shifts in a circular fashion along columns, rows, or layers. All data paths are 18-bits wide to match the fixed 18-bit inputs of the DSP48E block.



⊡ = Output to layer above ⊠ = Input from layer below

Figure 41 A layer of PEs, showing the interconnect and I/O

7.4.5 I/O bandwidth

In a multi-chip system, an important question is how many PEs can be partitioned for each chip. The system has I/O bandwidth and logic resource limits. In Appendix I, I/O requirements and resource availability for the threeVirtex5SXFPGAs are compared. It is important to note that not all of the DSP48E resources were used because total chip bandwidth runs out at a certain point for different FPGAs. Serialization/deserialization logic are used for the interconnect.

7.4.6 SIMD System Control

Most control signals are single bit control, but some, like the address inputs to the Block RAM, function as index counters to the stored coefficient data. In addition to fine-grained control at the PE level, control signals also manipulate the multiplexors at the switch lattice level. These control signals are global and can be issued by a single control unit or by distributed copies of the control unit. The control unit requires very few resources, so even if multiple copies are distributed, the cost is minimal. The control communication overhead of larger arrays can also be avoided with multiple control processors per FPGA.

7.4.7 Cycle Accurate Control Sequencer (CACS)

When the tomography algorithm is mapped to hardware, it can be controlled by linear sequences of control bits, specific clock counts of idle, and minimal branching. A Cycle Accurate Control Sequencer module (CACS), shown in Figure 42, was architected to be a best of both worlds solution that would (1) borrow single cycle latency benefits from finite state machines and (2) use programmability aspects of a small RISC engine. Because the CACS logic consists of only an embedded Block RAM module and a few counters, it has both a small footprint and a fast operational speed.

The control sequence up counter acts as a program counter for the system. The following types of instructions can be issued from the control Block RAM:

1. Real coefficient address load: Bit 22 signals the real coefficient up counter to be loaded with the lower order data bits.
2. Imaginary coefficient address load: Bit 21 signals the imaginary coefficient up counter to be loaded with the lower order data bits.
3. Control sequence address load: Bit 20 and status bits control whether or not a conditional branch is taken. If a branch is to be taken, then the control sequence up counter is loaded with the address contained in the lower order data bits.
4. Idle count: Bit23 loads adown counter with a cycle count contained in the lower order data bits. This saves program space during long instruction sequences where control bits do not have to change on every cycle. When the down counter reaches zero, the idle is finished and the control sequence up counter is re-enabled.
5. Control bus change: When the high order bits are not being used for counter loads, the low order bits can be changed cycle by cycle for the control bus registers.

Three types of low-level instructions are presented in Figure 43 to show how a sample control sequence in a text file is compiled by script into Block RAM content. First the single bit outputs are defined, then an idle count command creates a pause of “cols*2-1” number of times. Note that “cols” is a variable dependant on the number of east/west columns in the systolic array. The instructions are flexible because they incorporate these variables. Single bit changes are done on the subsequent two cycles and finally the real and imaginary coefficient counters are loaded.

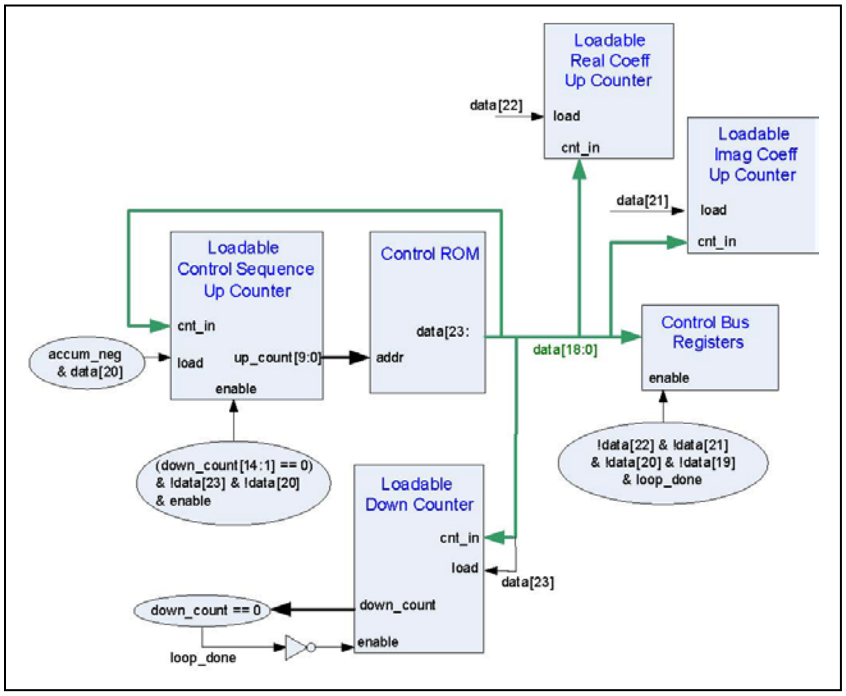
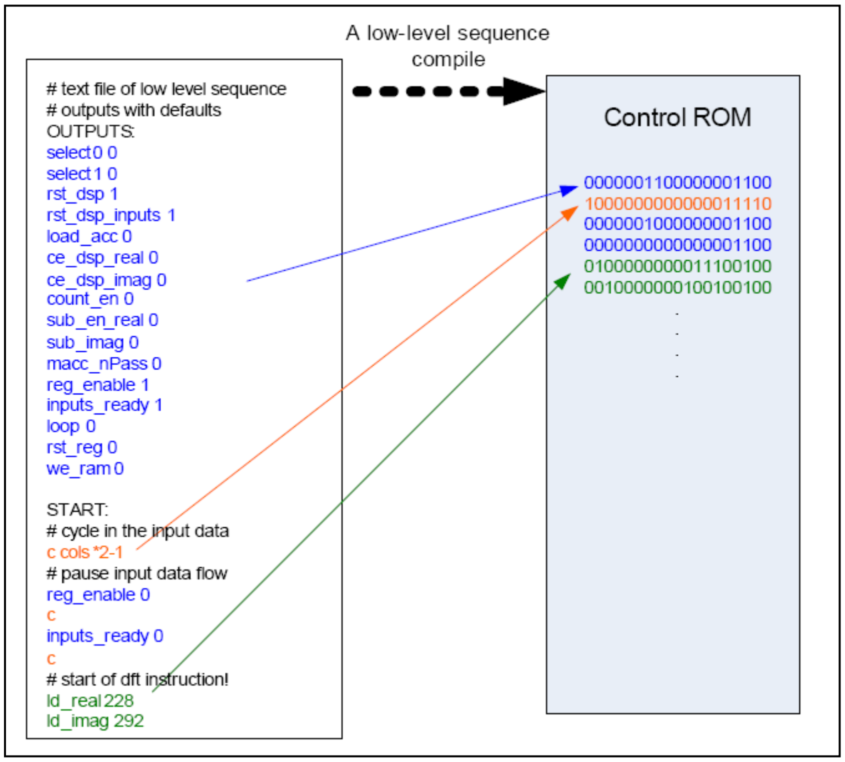


Figure 42 CACS Architecture



```
# text file of low level sequence
# outputs with defaults
OUTPUTS:
select0 0
select1 0
rst_dsp 1
rst_dsp_inputs 1
load_acc 0
ce_dsp_real 0
ce_dsp_imag 0
count_en 0
sub_en_real 0
sub_imag 0
macc_nPass 0
reg_enable 1
inputs_ready 1
loop 0
rst_reg 0
we_ram 0

START:
# cycle in the input data
c cols*2-1
# pause input data flow
reg_enable 0
c
inputs_ready 0
c
# start of dft instruction!
ld_real 228
ld_imag 292
```

```
0000001100000001100
1000000000000011110
0000001000000001100
0000000000000001100
0100000000011100100
0010000000100100100
.
.
.
.
```

Figure 43 Sample CACS ROM content

7.4.8 Timing and Events

The TE operates on a frame basis. All events that occur during a frame are signaled at the end of the frame.

7.4.9 Control and Orchestration of Processor States

The parameters and data being processed combine to form the TE's state.

7.4.9.1 On-Line Parameter Loading

Parameters that change during operation, such as C_n^2, are loaded at the end of a frame after data processing has completed. The next frame will reflect the current parameter set. This is controlled by the CACS, see 7.4.7.

7.4.9.2 Off-Line Parameter Loading

The CP loads parameters, such as Fourier coefficients that do not change during operation, during configuration, see 7.5.2.1.

7.5 *Design*

7.5.1 Physical Architecture of the Tomography Engine

The systolic array tomography engine is composed of many FPGA chips on multiple boards.

7.5.2 FPGA Design

The design of the TE uses the Xilinx Virtex 6 chips.

7.5.2.1 FPGA Configuration

FPGA configuration will be done by the CP. Configuration files which have been previously loaded into the Disk Sub-system by the AO Control, will be automatically loaded at power up.

7.5.2.2 System Monitor

The System Monitor monitors the core temperatures and voltages of each FPGA. They are linked serially, so that the health of each FPGA in the tomography processor can be monitored independent of the operation state of the system.

7.5.2.3 Position Detection

Each FPGA will have a pin dedicated to identifying its physical location on the board. This pin will be connected to a serial EPROM with the position identification and other relevant information as deemed necessary. This will facilitate uniquely identifying devices when diagnostics are being performed.

7.5.2.4 Heat Removal

Each FPGA will have a high profile heat sink that will be capable of keeping the junction temperature, (θ_J), below 60 °C with an ambient temperature of 20 °C.

7.5.3 Board Design

7.5.3.1 Board Stack Up

All metal layers will have a thickness of 0.7 mil (0.5 oz Copper). The dielectric material will be FR4.

Layer #	Function
1	Sig
	0.0035
2	Gnd Pin
	0.01
3	Sig
	0.01
4	Gnd Pin
	0.006
5	Sig
	0.006
6	Pwr Pin
	0.01
7	Gnd Pin
	0.01
8	Gnd Pin
	0.01
9	Gnd Pin
	0.006
10	Sig
	0.006
11	Gnd Pin
	0.0035
12	Sig

Table 12 Board Stack Up

7.5.3.2 FPGA Configuration

Configuration will be done with a single connector to each board. This will transfer the configuration to all FPGAs on board.

7.5.3.3 Power Regulators

Each board will have its own power regulator for the critical core voltages.

These voltages must be regulated to < ±5% for all FPGA's on the board.

7.5.3.4 Voltage Monitors

All on board voltage regulators will have voltage monitors that will report the voltage over an SPI link to the CP.

7.5.3.5 Temperature Monitor

Each board will have a centrally located temperature monitor that will report the temperature over an SPI link to the CP.

7.5.3.6 Position Detection

Each board will have a connector fixed at its location that will identify the boards unique position among the other identical boards. This will be used by the voltage and temperature monitors to identify the board on which they reside.

7.5.4 System (Multi-Board)

7.5.4.1 Tomography Engine Interface boards

Simple descriptions and diagrams

7.5.4.2 Board to Board Data Communications

Each board will communicate with its orthogonal neighbors through high speed LVDS serial links, see Figure 35.

Data will be shifted into and out of the array or recirculated through the array by multiplexor on the boundaries, see Figure 37.

.

7.5.4.3 Communications to the Telemetry Stream

The Tomography Engine communicates to the Telemetry Stream through the Recirculating boards on the boundaries of the array. These boards determine if data is to be recirculated during operations such as a DFT. At the start of the frame new wave fronts are shifted in on the left. At the end of computation, the science wave front is shifted out on the right. Between the end of computation and the start of a new frame, new parameters are shifted in on the left and telemetry data is shifted out on the right.

7.6 *Other Architectural Issues*

7.6.1 Program Flow Control

Since we must convert the forward propagated values to the spatial domain and apply the aperture to them, we calculate the error in the spatial domain. Then we transform the error back to the Fourier domain for back propagation. In our architecture, we do this by N shifts across rows to accomplish the transform.

The total number of iterations in the frame is fixed. This limit is set to guarantee that each iteration will complete before the start of the next frame. During anomalous situations, such as initial startup or an obscuration traveling rapidly across the aperture, the system may not be able to converge to our lower error limit in a single frame and must continue at the start of the next frame. We do not want the frame sync to occur in the middle of an iteration, leaving the machine in an uncertain state.

Diagnostic programs to run after the tomography iterations have completed.

7.6.2 Arithmetic Errors and Precision

A DFT has O(N^2) operations for each transform, while an FFT has O(Nlog(N)) operations. Using exact arithmetic, they have equal accuracy. However, when using finite math, the DFT has the potential to incur more round-off errors than an FFT.

Special care must be taken to avoid this error from accumulating over many transforms. We use pre-scaling of our coefficients and post-scaling of the transformed data to minimize this.

8. DM and T/T Command Generation

The DM and T/T command generator sub-systems convert wavefront information into mirror voltage commands to result in the desired wavefront.

8.1 *Algorithms*

See NGAO Real Time Controller Algorithms Design Document for details.

8.2 *Tip/Tilt Mirrors*

The RTC controls 10 tip/tilt mirrors (one for each WFS) and 1 on-axis tip/tilt actuator on the Woofer:

- 3 T/T mirrors associated with the 3 LOWFS cameras
- 7 T/T mirrors associated with the 7 HOWFS cameras
- 1 T/T actuator on the Woofer

Each mirror or actuator is driven by its own driver. The driver is supplied by a tip/tilt command generator with a value that corresponds to the desired tip/tilt.

8.2.1 Tip/Tilt Command Generators

The tip/tilt command generators take a desired tip/tilt angle from a WFS and convert that angle to a signal to the tip/tilt mirror drivers or actuators that will set the mirrors to the correct position.

The tip/tilt information for each HOWFS is derived from the centroids calculated for that HOWFS.

The tip/tilt information for each LOWFS is derived from LOWFS camera data.

The on-axis tip/tilt information is derived from the tip/tilt information supplied by the three LOWFS.

8.2.2 Systematic Vibration Compensation (FR-XXXX, no current functional requirement)

The on-axis tip/tilt actuator will be capable of compensating for systematic vibrations through a TBD algorithm.

8.2.3 Interfaces and Protocols

The controlling signal provided to the tip/tilt mirror drivers will be either an analog voltage or a digital value. The decision as to which is used will be reserved for the build phase.

8.2.4 Data Flow and Rates

8.2.4.1 LOWFS and On-axis T/T

The LOWFS and the On-axis tip/tilt are updated at up to 2 KHz. This rate is determined by the LOWFS frame rate.

8.2.4.2 HOWFS T/T

The HOWFS T/T mirrors and the on-axis T/T actuator are updated at 2 KHz. This rate is determined by the HOWFS frame rate.

8.3 *DM Command Generators*

The DM Command Generators convert wave fronts to segment voltage commands necessary to achieve the desired shape, taking into account the non linear response and influence functions of the DM.

8.3.1 Low Order (Woofer) (On-axis DM, closed loop operation)

The low order DM (woofer) operates closed loop. All WFS and the science object are affected by the shape of the Woofer.

8.3.1.1 Algorithm

See NGAO Real Time Controller Algorithms Design Document for details.

8.3.1.2 Interfaces and Protocols

The wavefront supplied to the Woofer command generator by the Tomography Engine will be an array of 400 displacement values. These values will be in scaled nanometers.

The electronics that drive the Woofer have not been specified at this time. Therefore the interface and protocol to the DM cannot be specified. However, it is not expected to be a limiting issue.

8.3.1.3 Data Flow and Rates

The Woofer is sent commands from the Tomography engine at the frame rate of the system (~2 KHz). These commands are sent synchronously with the tweeter commands.

The electronics that drive the Woofer have not been specified at this time. However, a reasonable estimate of the time to transfer the commands to the approximately 400 elements of the Woofer is less than 50µsec, which is the time that the current tweeter system (4092 elements) takes to transfer the its commands.

8.3.2 High Order (Tweeter) (Science and LOWFS DMs, Open Loop Operation)

The high order on-axis science DM and the LOWFS DMs all operate open loop.

The high order science DM removes high order aberrations from the science image. The shape placed on it is calculated by forward projecting the science object through the tomographic estimate of the atmosphere.

The LOWFS DMs correct aberrations in the light from the NGSs. The NGS LOWFSs sense tip/tilt, focus, and astigmatism that the HOWFS cannot. The shape that is placed on the LOWFS DM is calculated by sensing the wave front from an adjacent point-and-shoot LGS through its HOWFS.

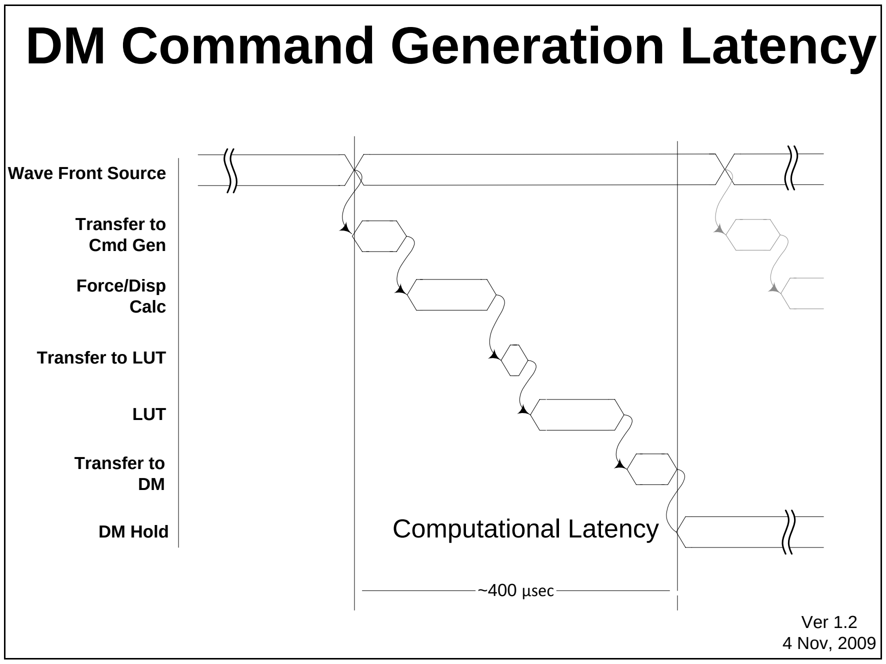


Figure 44 DM Command Generation Latency (not to scale)

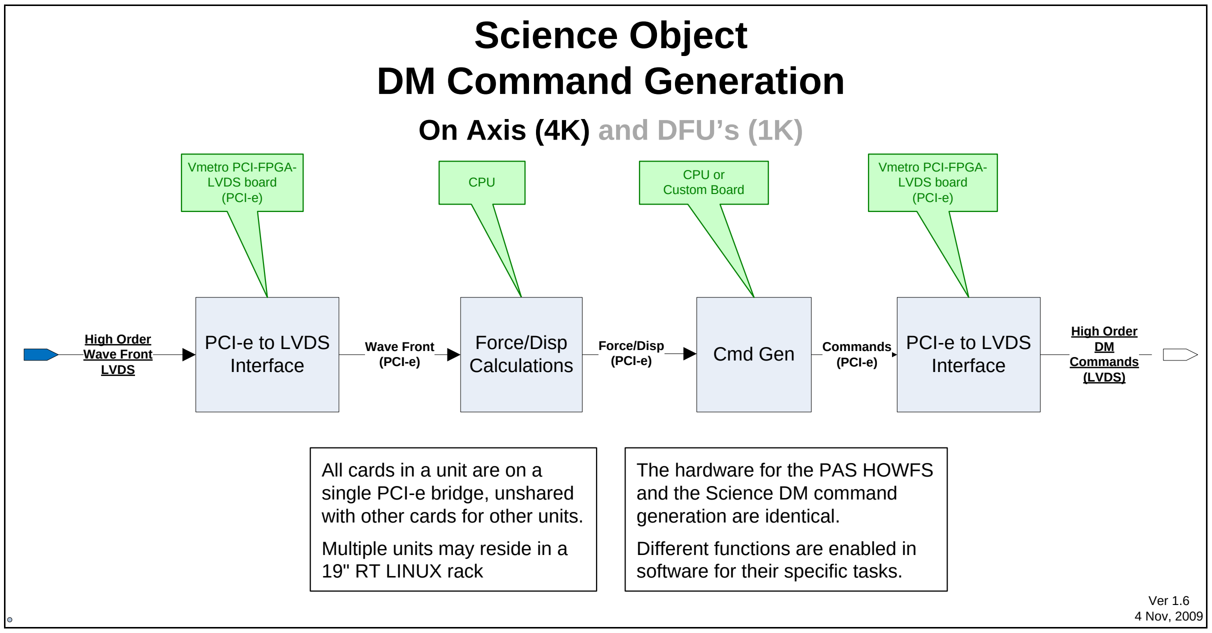

Science Object
DM Command Generation
On Axis (4K) and DFU's (1K)
Vmetro PCI-FPGA-LVDS board (PCI-e)
CPU
CPU or Custom Board
Vmetro PCI-FPGA-LVDS board (PCI-e)
High Order Wave Front LVDS
PCI-e to LVDS Interface
Wave Front (PCI-e)
Force/Disp Calculations
Force/Disp (PCI-e)
Cmd Gen
Commands (PCI-e)
PCI-e to LVDS Interface
High Order DM Commands (LVDS)
All cards in a unit are on a single PCI-e bridge, unshared with other cards for other units.
Multiple units may reside in a 19" RT LINUX rack
The hardware for the PAS HOWFS and the Science DM command generation are identical.
Different functions are enabled in software for their specific tasks.
Ver 1.6
4 Nov, 2009

Figure 45 Science Object DM command generation

8.3.2.1.1 Interfaces and Protocols

The drive electronics for the Command Generator are housed in a real-time Linux system running on a multi-processor multi-core system.

The Tomography Engine sends the Command Generator an array of displacement values (in nanometers) over an LVDS cable using a simple clocked protocol.

The Command Generator sends the DM driver the resultant commands over an LVDS cable (32 pairs) using a proprietary but simple protocol. The cable is driven by a general purpose I/O board from VMETRO. This board is supported by a RT Linux driver supplied by the DM Driver manufacturer, Cambridge Innovations.

NOTE:

> The drivers for the LVDS cable must be on the same AC ground as the Cambridge Innovations DM controller.
>
> Currently, the Command Generator will be in the computer room and the DM Controller will be on the Naismith (see Figure 9). In this configuration, the Command Generator will drive an LVDS-to-optical converter and the signals will be brought to the Naismith through optical cables. At the Naismith, the optical cables will drive an optical-to-LVDS converter that will drive the DM Controller. It is this converter that must be on the same AC ground as the DM Controller.

8.3.2.1.2 Data Flow and Rates

The Tweeter Command Generator is sent an array of displacements (4K elements) from the Tomography engine at the frame rate of the system (~2 KHz). It processes these and sends the commands (4K) to the Tweeter. These commands are sent synchronously with the Woofer commands.

The electronics for the Tweeter are able to write to all 4K elements in less than 50 µsec. All elements must be written sequentially so there is no ability to update only a single element.

8.3.2.2 High Order DM for the LOWFS NGS Command Generation

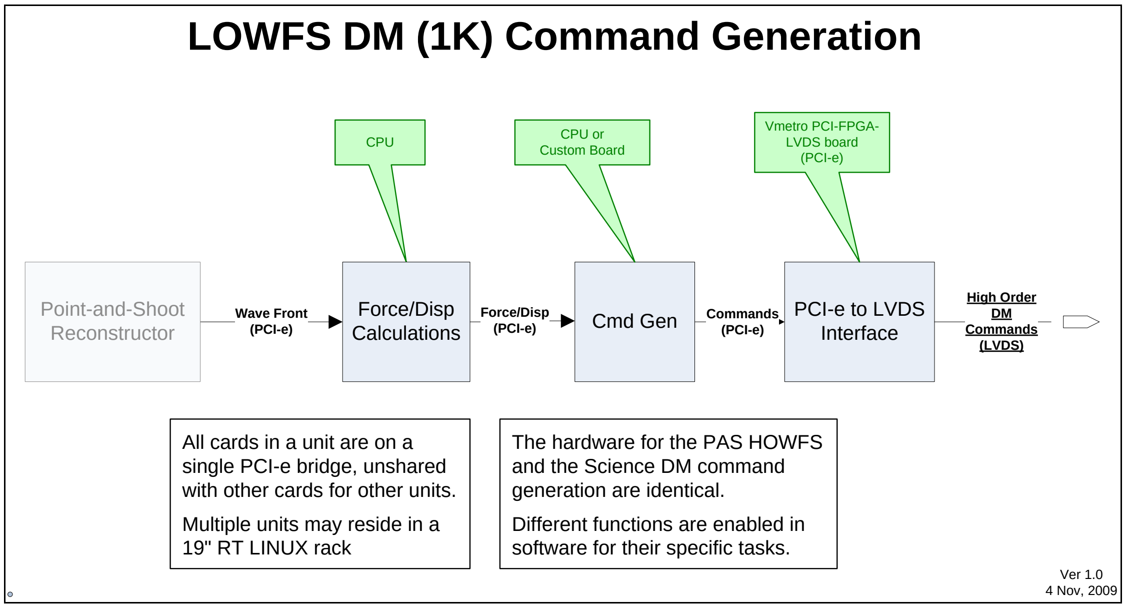

LOWFS DM (1K) Command Generation
CPU
CPU or Custom Board
Vmetro PCI-FPGA-LVDS board (PCI-e)
Point-and-Shoot Reconstructor
Wave Front (PCI-e)
Force/Disp Calculations
Force/Disp (PCI-e)
Cmd Gen
Commands (PCI-e)
PCI-e to LVDS Interface
High Order DM Commands (LVDS)
All cards in a unit are on a single PCI-e bridge, unshared with other cards for other units.
Multiple units may reside in a 19" RT LINUX rack
The hardware for the PAS HOWFS and the Science DM command generation are identical.
Different functions are enabled in software for their specific tasks.
Ver 1.0
4 Nov, 2009

Figure 46 LOWFS Command Generation

8.3.2.2.1 Interfaces and Protocols

The drive electronics for LOWFS DM Command Generator are part of the real-time Linux system that comprises the Point-and-Shoot HOWFS. No external transfer mechanism is needed to transfer the desired wavefront.

The Point-and-Shoot HOWFS sends the Command Generator an array of displacement values (in nanometers).

The Command Generator sends the DM driver the resultant commands over an LVDS cable (32 pairs) using a proprietary but simple protocol. The cable is driven by a general purpose I/O board from VMETRO. This board is supported by a RT Linux driver supplied by the DM Driver manufacturer, Cambridge Innovations.

NOTE:

The drivers for the LVDS cable must be on the same AC ground as the Cambridge Innovations DM controller.

Currently, the Command Generator will be in the computer room and the DM Controller will be on the Naismith (see Figure 9). In this configuration, the Command Generator will drive an LVDS-to-optical converter and the signals will be brought to the Naismith through optical cables. At the Naismith, the optical cables will drive an optical-to-LVDS converter that will drive the DM Controller. It is this converter that must be on the same AC ground as the DM Controller.

8.3.2.2.2 Data Flow and Rates

The LOWFS DM Command Generator is sent an array of displacements (1K elements) from the HOWFS at the frame rate of the system (~2 KHz). It processes these and sends the commands

(1K) to the LOWFS DM. These commands are sent synchronously with the Woofer and Tweeter commands.

The electronics for the LOWFS DM are able to write to all 1K elements in less than 50 µsec. All elements must be written sequentially so there is no ability to update only a single element.

9. NGS Mode Operation

Simple description

10. Telemetry, Diagnostics, Display and Offload Processing

There are several types of data that may be supplied by the RTC: raw data, displayable data (offload processed), settable parameters, environmental parameters, logs, notifications and errors.

In this document:

- Telemetry refers to raw data sent unprocessed to the disk sub-system.
- Diagnostics refers to raw data that has been filtered by the offload processor (sampled or temporally filtered) and sent at a low bandwidth to the AO Control. Such data may be used for on line r_0 calculations or for PSF analysis.
- Settable parameters are any parameter which the AO Control can set during normal operations. These comprise the RTC's current operating state.
- Environmental parameters include temperature, residual errors, airflow, voltages, vibration of components of the RTC, etc.

All major data signals can be sent to the Telemetry/Diagnostic data path at the request of the AO Control. Any data in this path can be sent to either-or-both the AO Control, for diagnostics, display, or the RTC disk sub-system, for storage. (See: Section 8)

These signals include:

1. Raw Camera Data for 4 tomography WFSs, 3 point-and-shoot WFSs, and 3 NGS cameras
2. Centroids all HOWFS
3. Reconstructed wave fronts for all HOWFS
4. T/T commands for all cameras
5. Calculated Focus and Astigmatism for LOWFS
6. Tomography Layers
7. Science on-axis High Order Wave Front produced by the Tomography engine after forward propagating from the science object through the tomographic estimate of the atmosphere. This is the open loop control wave front for the High Order DM
8. Science on-axis Woofer Wave Front produced by the Tomography engine after forward propagating from the science object through the tomographic estimate of the atmosphere. This is the closed loop control wave front for the Low Order DM
9. Science on-axis High Order Commands
10. Science on-axis Woofer Commands
11. All RTC Current Parameter Settings including the settings of the following: shutters, laser fiber In/Out actuators, laser On/Off state and intensity, filter wheel position, interlocks

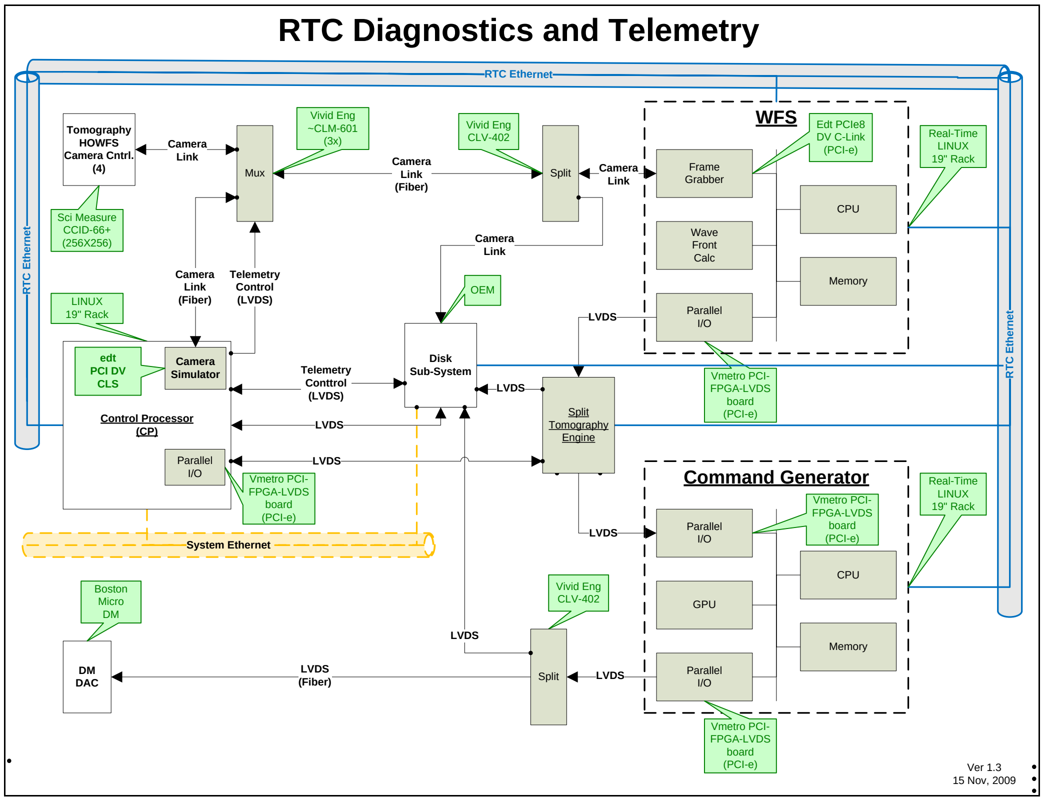


Figure 47 RTC Diagnostics and Telemetry

Diagnostic data is captured and sent to the AO Control at rates of approximately one sample per second, as determined by the AO Control.

A separate frame grabber associated with the disk sub-system captures camera data for diagnostics and telemetry. This data is a parallel stream supplied by a Camera Link splitter, shown in Figure 47.

10.1 *Architecture and Design*

The RTC processing elements must be able to send all data streams that can be monitored to the disk sub-system simultaneously. Because of this, we send these streams constantly to the Offload Processor. The Offload Processor selects the desired streams, from none to all, individually. This results in a reduction in the logic that the individual sources must support. The decision is made at one location.

The data is then routed to the disk sub-system and/or the Offload Processor will sample or filter the data and send it to the AO Control as appropriate.

10.1.1 **Offload Processor (OP)**

The offload processor is a multi-processor multi-core Linux box running a real time Linux kernel. Data is taken in from its raw source and sampled or temporally filtered (a simple 1 pole filter) as necessary. The processed data is sent to the AO Control for display or analysis at a predetermined rate of <3Hz. In the system diagrams the Offload Processor is shown as a part of the Disk Sub-System.

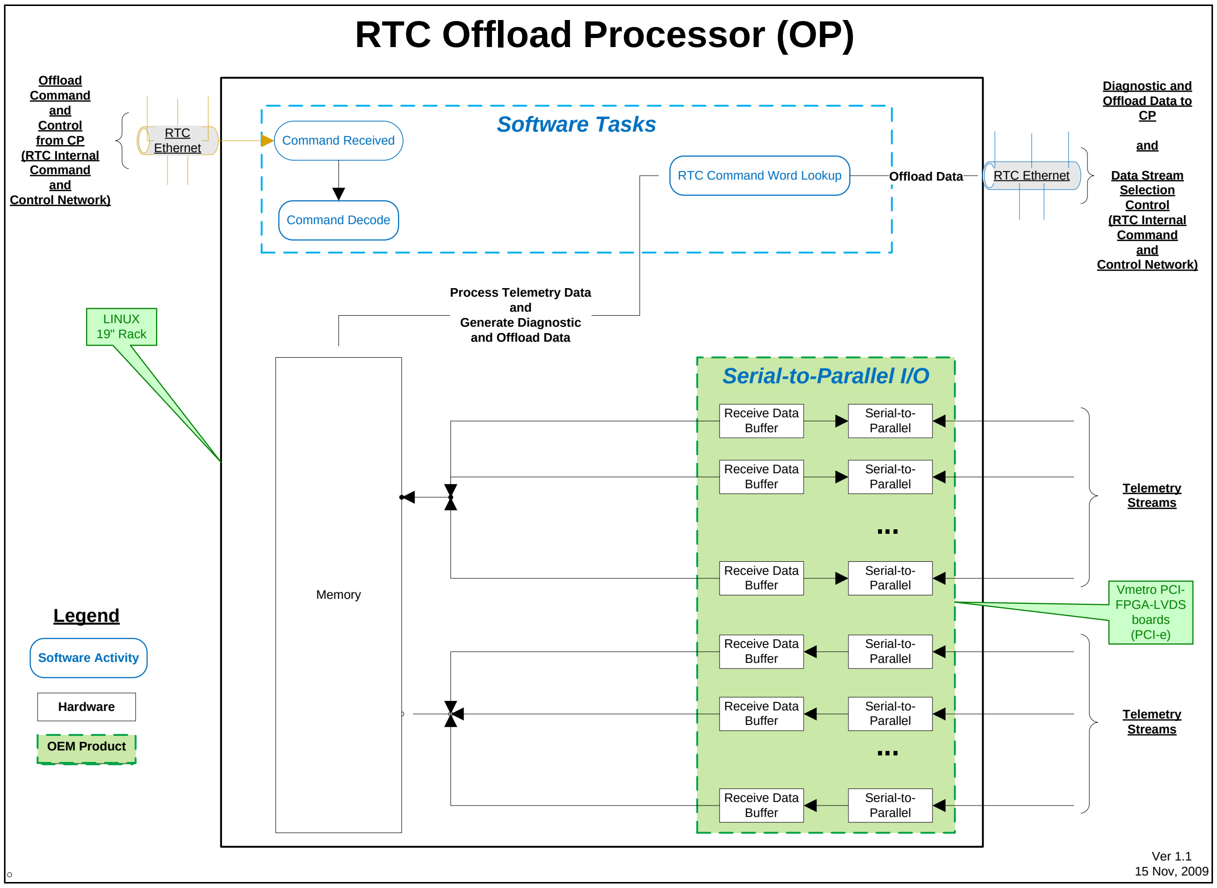

RTC Offload Processor (OP)
Offload Command and Control from CP (RTC Internal Command and Control Network)
RTC Ethernet
Software Tasks
Command Received
Command Decode
RTC Command Word Lookup
Offload Data
RTC Ethernet
Diagnostic and Offload Data to CP
and
Data Stream Selection Control (RTC Internal Command and Control Network)
Process Telemetry Data and Generate Diagnostic and Offload Data
LINUX 19" Rack
Memory
Serial-to-Parallel I/O
Receive Data Buffer
Serial-to-Parallel
...
Telemetry Streams
Vmetro PCI-FPGA-LVDS boards (PCI-e)
Telemetry Streams
Legend
Software Activity
Hardware
OEM Product
Ver 1.1
15 Nov, 2009

Figure 48 Offload Processor

Time averaged and filtered data will be provided to the AO control as specified in **FR-1405.**

10.2 *Interfaces and Protocols*

The interface between the Offload Processor and the rest of the RTC is through a combination of LVDS, fiber, and Ethernet.

10.3 *Data Flow and Rates*

The RTC has the ability to generate an enormous amount of data. Table 13 lists the major elements and their transfer rates.

Item	# per side	Total items	Bytes per Wd	Total Bytes	Telem Time (µsec)	Telem Rate (MB/sec)	Total Telem Rate (MB/sec)	Transfer Time (µsec)	Transfer Rate (MB/sec)	# of Instances	Total Xfer Rate (MB/sec)
Tomography WFS Camera	256	65,536	2	131,072	500	262	1,049	50	2,621	4	10,486
Point-and-shoot Camera	128	16,384	2	32,768	500	66	197	50	655	3	1,966
T Cent	60	7,200	2	14,400	500	29	115	50	288	4	1,152
P Cent	30	1,800	2	3,600	500	7	22	50	72	3	216
T WF	60	3,600	3	10,800	500	22	86	50	216	4	864
P WF	30	900	2	1,800	500	4	11	50	36	3	108
T Layer	88	7,744	3	23,232	500	46	232	50	465	5	2,323
Sci WF	88	7,744	3	23,232	500	46	46	50	465	1	465
Sci WF HO	88	7,744	3	23,232	500	46	46	50	465	1	465
Sci WF LO	88	7,744	3	23,232	500	46	46	50	465	1	465
Tweeter	64	4,096	2	8,192	500	16	16	50	164	1	164
Woofer	20	400	2	800	500	2	2	50	16	1	16
Telemetry Effect											
Total rate (MB/sec)							**1,869**				
Total data per 6 hours (TB)							**40**				

Table 13 Major Telemetry and Data Transfer Rates

Notes:

1. “Telem Time” is 500 µsec, since the system has an entire frame to transfer the data to the disk sub-system using the Telemetry pipes.
2. “Transfer Time” is 50 µsec, since we need to move data in a small portion of our frame.

10.4 *Synchronization of AO Data Time Stamps with the Science Image Time Stamps*

The RTC system will have a sub millisecond accurate clock, GPS based to time stamp the AO data for telemetry and diagnostics and provide to the SRT.

Currently, science image shutter open and close times at Keck are time stamped to a precision nominally of a small fraction of a second but may be as much as ±30 seconds in error (see Appendix B). New instruments should synchronize in hardware with the RTC time stamp for critical measurements.

10.5 *An Important Note on the Telemetry Rates*

Since the total storage rate can exceed 2 GB/sec, if all possible data sets are captured, care should be taken to insure there is enough space remaining in the disk sub-system to save the data. The disk sub-system can support multiple TB of data, but after a number of nights of heavy use, a great deal of the storage may be used.

The disk sub-system is not meant to archive data. Its purpose is merely to store it temporarily until the desired data can be archived to another location and the unneeded data can be deleted.

10.6 ***Handling System Diagnostic Functions***

The RTC does not need to know that a wave front was generated by a star or a laser diode (with or without simulated turbulence). Its job is the same; calculate the wave front and tomography. No special features are in place for system diagnostics beyond the normal diagnostics described here.

11. RTC Disk Sub-System

All major data signals can be sent to the Telemetry/Diagnostic data path at the request of the AO Control. Any data in this path can be sent to either-or-both the AO Control, for diagnostics, or the RTC disk sub-system, for storage.

Since the total storage rate can exceed 2 GB/sec if all possible data sets are captured, care should be taken to insure there is enough space remaining in the disk sub-system to save the data. The disk sub-system can support multiple TB of data, but after a number of nights of heavy use, a great deal of the storage may be used.

The disk sub-system is not meant to archive data, merely to store it temporarily until the desired data can be archived to another location and the undesired data can be deleted.

11.1 *Interfaces and Protocols*

The structure of the telemetry data will be FITS files.

11.1.1 Interface to the RTC

The interface to the telemetry of the RTC will be fiber link.

11.1.2 Interface to the System AO Control

The interface to the AO Control will be through Ethernet.

11.2 *Data Flow and Rates*

See Table 11.

11.2.1 RTC Telemetry Port

The telemetry input will be multiple streams of data with the rates shown. The aggregate rate will not exceed 3 GB/sec.

11.3 *Storage Capacity*

The storage capacity of the RTC Disk Sub-system will be 80 TB.

12. Timing Generation and Control

The Timing Generation and Control subsystem is responsible for generating and synchronizing all major clocks for the RTC.

- Keeping the cameras synchronized to the frame clock and insures that even if cameras are working at different frame rates, their data is synchronized to the Tomography engine
- Generating the correct transfer clocks for data in the system (600 MHz LVDS)
- Synchronizing the real time components of all real time systems in the RTC

12.1 *Camera Synchronization and Timing*

Camera frame times must be synchronized to within 500 ps.

Camera transfer clocks must be synchronized to within 500 ps.

12.2 *GPS Sub millisecond Time Stamp Source*

The sub millisecond timer must be accurate to within 10 µsec.

12.3 *Global Tomography Synchronization*

The Global Synchronization signal must be synchronized at all locations to within 500 ps.

12.3.1 Global Reset

The Global Reset signal must be synchronized at all locations to within 500 ps.

12.3.2 Global Synch

The Global Synch signal must be synchronized at all locations to within 500 ps.

12.3.3 System Clock

The 100 MHz tomography clock must be synchronized at all locations to within 500 ps.

12.3.4 LVDS Clock (600 MHz)

The tomography LVDS clocks must be synchronized at all locations to within 500 ps.

More Description and diagram

13. RTC Physical Architecture

The RTC hardware is split between the Nasmyth and the computer room.

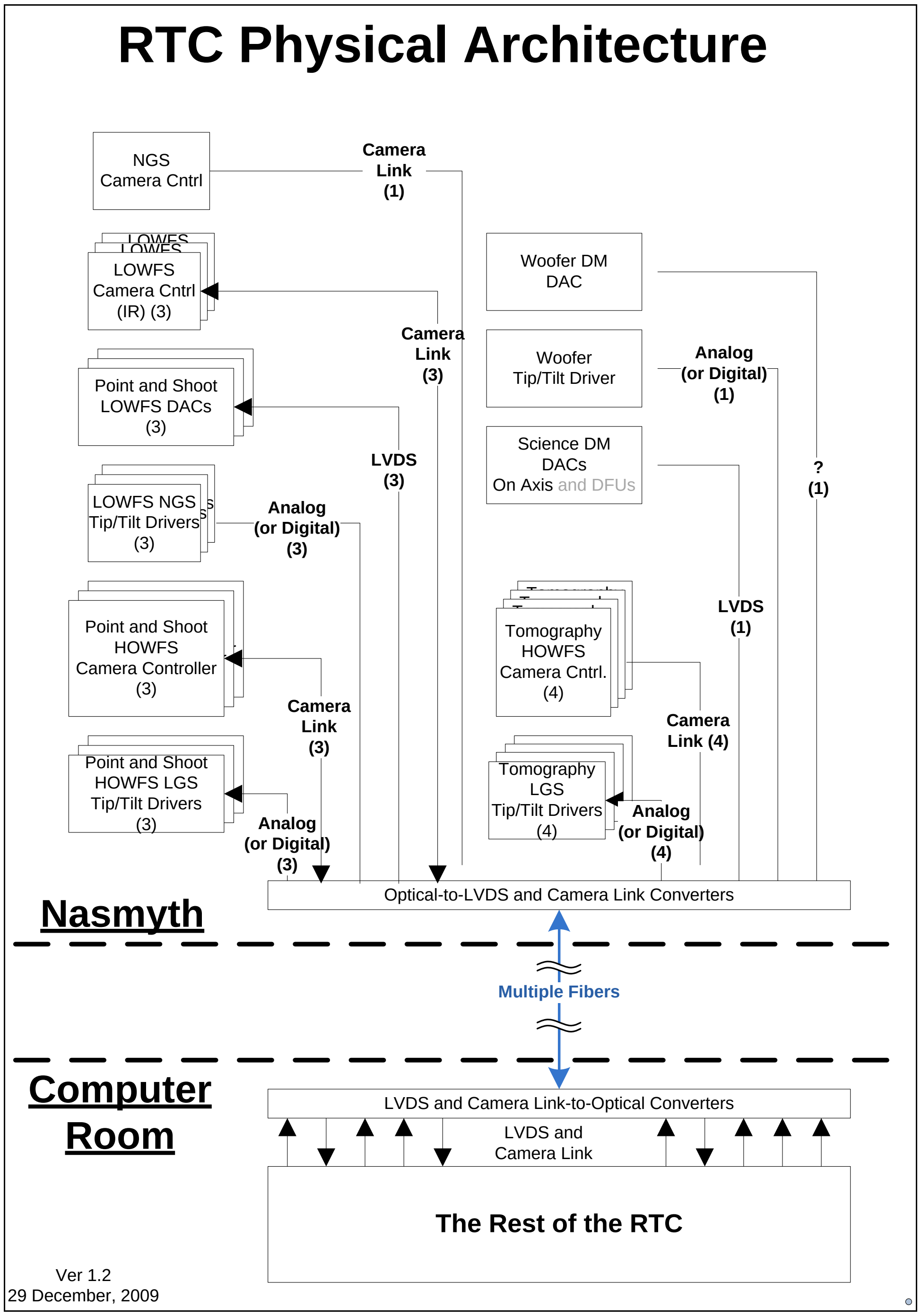

RTC Physical Architecture
NGS Camera Cntrl
Camera Link (1)
LOWFS Camera Cntrl (IR) (3)
Camera Link (3)
Point and Shoot LOWFS DACs (3)
LVDS (3)
LOWFS NGS Tip/Tilt Drivers (3)
Analog (or Digital) (3)
Point and Shoot HOWFS Camera Controller (3)
Camera Link (3)
Point and Shoot HOWFS LGS Tip/Tilt Drivers (3)
Analog (or Digital) (3)
Woofer DM DAC
Woofer Tip/Tilt Driver
Analog (or Digital) (1)
Science DM DACs On Axis and DFUs
? (1)
LVDS (1)
Tomography HOWFS Camera Cntrl. (4)
Camera Link (4)
Tomography LGS Tip/Tilt Drivers (4)
Analog (or Digital) (4)
Optical-to-LVDS and Camera Link Converters
Nasmyth
Multiple Fibers
Computer Room
LVDS and Camera Link-to-Optical Converters
LVDS and Camera Link
The Rest of the RTC
Ver 1.2
29 December, 2009

Figure 49 Split of RTC hardware between the computer room and the Naismith

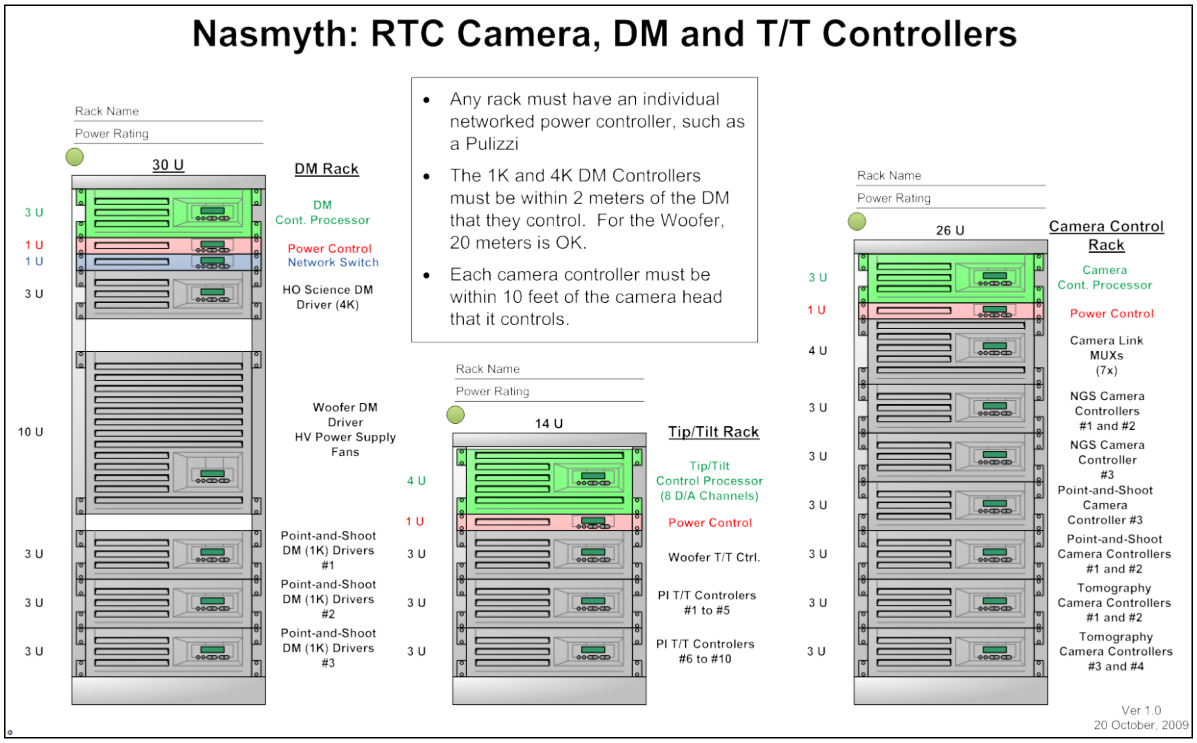

Nasmyth: RTC Camera, DM and T/T Controllers
Any rack must have an individual networked power controller, such as a Pulizzi
The 1K and 4K DM Controllers must be within 2 meters of the DM that they control. For the Woofer, 20 meters is OK.
Each camera controller must be within 10 feet of the camera head that it controls.
Rack Name
Power Rating
30 U
DM Rack
3 U
DM Cont. Processor
1 U
Power Control
1 U
Network Switch
3 U
HO Science DM Driver (4K)
10 U
Woofer DM Driver HV Power Supply Fans
3 U
Point-and-Shoot DM (1K) Drivers #1
3 U
Point-and-Shoot DM (1K) Drivers #2
3 U
Point-and-Shoot DM (1K) Drivers #3
14 U
Tip/Tilt Rack
4 U
Tip/Tilt Control Processor (8 D/A Channels)
1 U
Power Control
3 U
Woofer T/T Ctrl.
3 U
PI T/T Controlers #1 to #5
3 U
PI T/T Controlers #6 to #10
26 U
Camera Control Rack
3 U
Camera Cont. Processor
1 U
Power Control
4 U
Camera Link MUXs (7x)
3 U
NGS Camera Controllers #1 and #2
3 U
NGS Camera Controller #3
3 U
Point-and-Shoot Camera Controller #3
3 U
Point-and-Shoot Camera Controllers #1 and #2
3 U
Tomography Camera Controllers #1 and #2
3 U
Tomography Camera Controllers #3 and #4
Ver 1.0
20 October, 2009

Figure 50 Rack space required on the Nasmyth for the camera, DM and T/T controllers

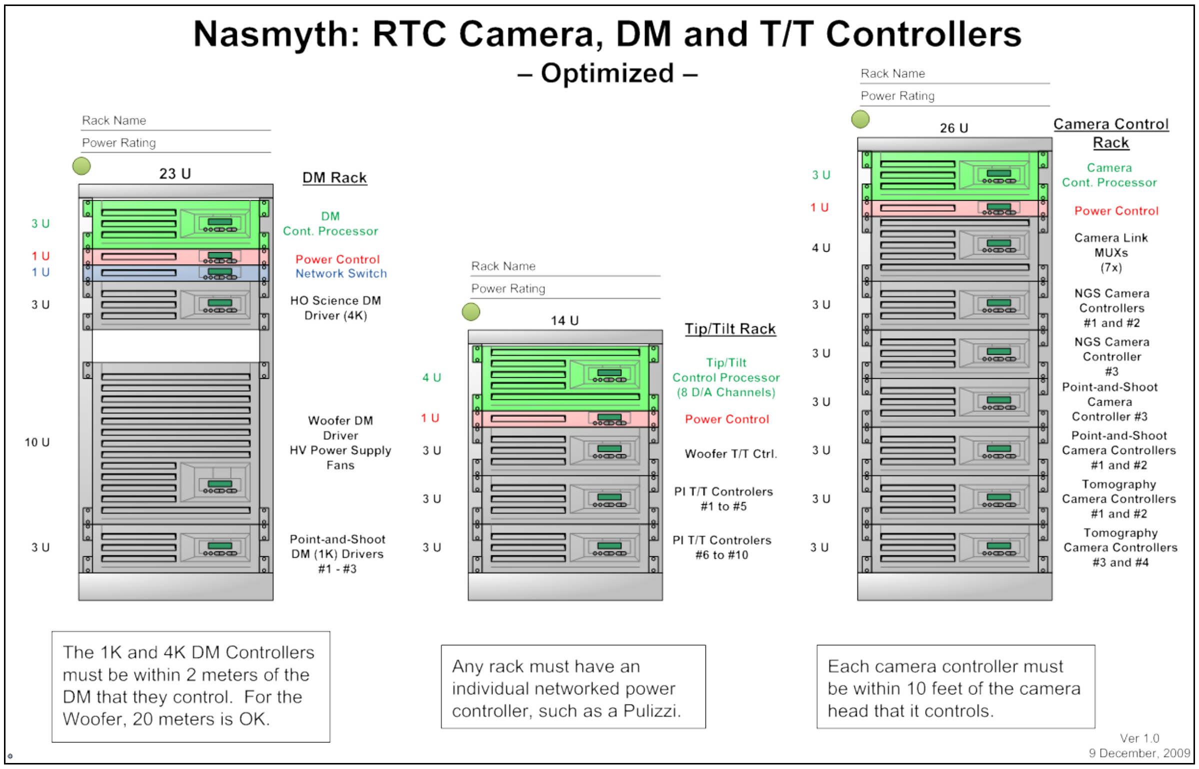

Nasmyth: RTC Camera, DM and T/T Controllers
– Optimized –
Rack Name
Power Rating
23 U
DM Rack
3 U
DM Cont. Processor
1 U
Power Control
1 U
Network Switch
3 U
HO Science DM Driver (4K)
10 U
Woofer DM Driver HV Power Supply Fans
3 U
Point-and-Shoot DM (1K) Drivers #1 - #3
14 U
Tip/Tilt Rack
4 U
Tip/Tilt Control Processor (8 D/A Channels)
1 U
Power Control
3 U
Woofer T/T Ctrl.
3 U
PI T/T Controlers #1 to #5
3 U
PI T/T Controlers #6 to #10
26 U
Camera Control Rack
3 U
Camera Cont. Processor
1 U
Power Control
4 U
Camera Link MUXs (7x)
3 U
NGS Camera Controllers #1 and #2
3 U
NGS Camera Controller #3
3 U
Point-and-Shoot Camera Controller #3
3 U
Point-and-Shoot Camera Controllers #1 and #2
3 U
Tomography Camera Controllers #1 and #2
3 U
Tomography Camera Controllers #3 and #4
The 1K and 4K DM Controllers must be within 2 meters of the DM that they control. For the Woofer, 20 meters is OK.
Any rack must have an individual networked power controller, such as a Pulizzi.
Each camera controller must be within 10 feet of the camera head that it controls.
Ver 1.0
9 December, 2009

Figure 51 Possible optimization of rack space required on the Nasmyth

Add diagram for the computer room too

14. RTC Test Bench

The Test Bench System is a part of the RTC system, but is not involved in the operations of the system. It is designed to perform bench tests on RTC major and sub-units: CP, WFSs, Tomography Engine boards, DM Command Generators …

Two permanent benches will be maintained: one at Keck and one in Santa Cruz.

Its purpose is to provide rapid and accurate acceptance, functional, development, diagnostic and debug test of all major RTC components

It can plug into the LVDS or Camera Link output from a Camera, a WFS, a tomography engine board, or a DM Command Generator and capture and analyze their outputs.

It can supply simulated input information to WFS, Tomography Engine boards, or DM Command Generators and simultaneously capture their outputs to validate unit functionality.

It is meant to be a bench tool in the lab rather than as an operational piece of the RTC.

It has all facilities necessary to provide the correct control to the devices under test through either LVDS, camera link, or Ethernet.

Two types of tests will be provided:

1. Push button (Go/No Go) confidence tests to validate the basic functionality of the unit.
2. Manually run tests to accomplish detailed analysis of the functionality or performance of the unit.

Additionally, to test the Tomography Engine boards or any custom boards, special jigs will be supplied that includes power supplies, cables, clock generators, monitoring, cooling and any other items needed to simulate the boards environment.

Figure 52 shows a diagram of the components of the test bench.

Outputs are capable of DMA'ing from memory to simulate complex data streams.

Inputs are capable of capturing the data streams from the device under test.

Programs will be written to control the input and output and display results.

14.1 *Interfaces and Protocols*

The user interface will be through either the console or the Ethernet.

The interface to the device under test (DUT)

14.2 *Data Flow and Rates*

See Table 11.

14.3 *Architecture*

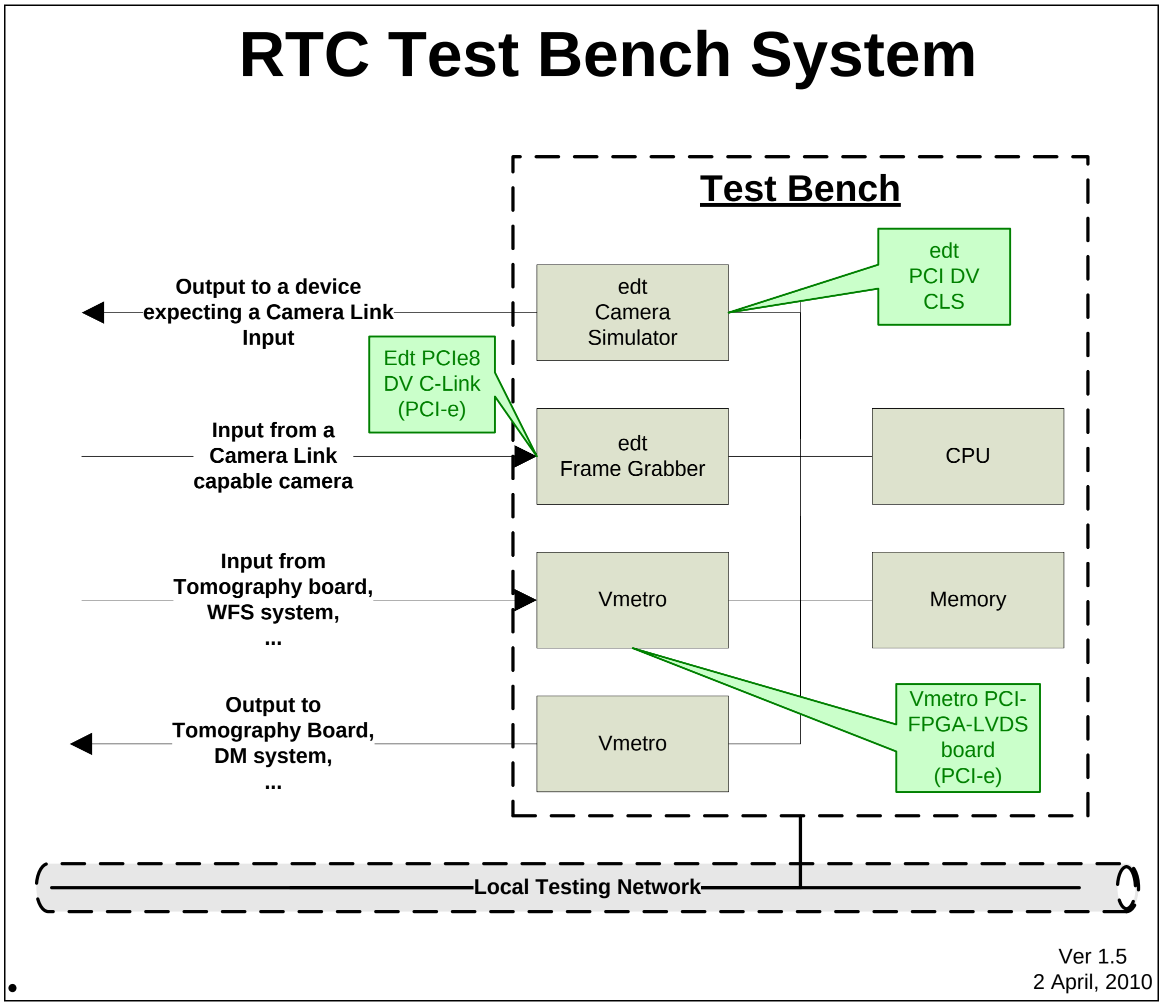

RTC Test Bench System
Test Bench
Output to a device expecting a Camera Link Input
edt Camera Simulator
edt PCI DV CLS
Edt PCIe8 DV C-Link (PCI-e)
Input from a Camera Link capable camera
edt Frame Grabber
CPU
Input from Tomography board, WFS system, ...
Vmetro
Memory
Output to Tomography Board, DM system, ...
Vmetro
Vmetro PCI-FPGA-LVDS board (PCI-e)
Local Testing Network
Ver 1.5
2 April, 2010

Figure 52 RTC Test Bench

15. Test Plan

The test plan details the requirements and strategies for testing the NGAO RTC and specifies the hardware and software to accomplish it. It covers the following:

- Trouble shooting and validation of individual modules during development
- Acceptance testing of individual modules subsequent to development
- Trouble shooting individual modules after commissioning: while they are in the system and on the bench
- Performing system end-to-end testing during integration-and-test, and after commissioning

Note: Tests for the cameras, deformable or Tip/Tilt mirrors themselves are beyond the scope of this document and are not covered.

Testability comes from obervability and controllability. The RTC system is designed so that critical inputs to the RTC, its sub-systems, and outputs can be forced to arbitrary values and the results of applying these inputs can be observed at key points down stream.

Files of known good stimuli, their results, and a written procedure for applying capturing them form a test set. The results for these test sets should be constant over time for a working system, can be used to easily validate the system or sub-systems, and are necessary for unit and acceptance tests.

They can also be used to help diagnose the locations of faults during the life of the system.

Further, the ability to use dummy camera data and analyze the resultant centroids and wave allows not only for testing and diagnosis of problems, but for easy analysis of new algorithms.

Similarly, the ability to modify the system and unit clocking and supply voltage parameters allows us to well characterize the healthy system's parameter envelopes. Periodic monitoring of these envelopes will allow us to see deviations that might lead to for failure during operations if not diagnosed early.

In some cases, a sequence of test files will need to be applied to test pattern sensitivity and signal interactions.

Key signals or voltage levels in today's dense high-speed circuits need to be easily monitored.

The design goal of the system is to provide these capabilities by designing them in from the start.

The RTC consists of many hardware components. These units will need to be tested in a variety of ways, at a variety of times: this includes testing during acceptance, integration and test, and after commissioning, including calibration and diagnosis of errors both in the integrated system and as separate units removed from the system.

The test plan is covered in a separate document.

16. General Requirements for System Components (HW and SW)

16.1 *All Components*

16.1.1 Acceptance Testing

All components must have an established acceptance test requirement as part of their individual design package.

16.1.2 Long Term Maintenance Plan

The cost of ownership of a complex system such as the NGAO RTC is significant and may exceed the acquisition cost over time. A clear plan must be in place for any custom hardware or software that will describe how the product will be maintained over time. This must include:

1. Types of personnel required for various levels of maintenance including upgrades, feature additions, or bug fixes; i.e., what tasks would it be expected for astronomers, telescope technicians, software or hardware staff
2. A maintenance agreement must be included by the vendor that covers the delivered product for bug fixes for a minimum of 1 year after acceptance.

16.1.3 Document Control

All elements of the system are revision controlled with CVS or a functional equivalent.

16.1.4 Hardware

16.1.4.1 Panel Retention

All panels that can be removed during normal maintenance or operation shall be held in place with self-retaining screws or Velcro if the attachment to the system is satisfactory for EMI/RFI, light tightness, and ventilation concerns.

16.2 *Custom Components*

Custom components include systems that are built from an aggregate of standard components, but combined into a custom configuration.

16.2.1 Documentation and Training

These products will need a variety of levels of documentation and training targeted at different audiences:

16.2.1.1 Diagnostics Capability

Each board must have the ability to have a minimum set of diagnostics performed on it while in the system and operating normally. The following are required for all boards. Requirements for specific board types are specified in their section.

16.2.2 Software

All custom software will be written in compliance with the existing Keck software standards. In addition to those standards, the following will be added if not already covered.

16.2.2.1 Software Module Identification

All software modules must contain a revision number and build date that can be queried during diagnostics to track currently installed module revisions. This applies to each module in any compiled unit and any .DLL or .so, which is a custom piece of code.

Further, any executable must have a command that can be given during diagnostics that will list the information recursively for **all** such modules with which it is built or to which it is linked.

16.2.2.2 ROM Code

The revision number of the code, a unique functional ID number, and the build date will be included in the ROM code or data.

16.2.2.3 Table Generation Algorithms

Algorithms for generating any data stored in tables.

16.2.3 Hardware

16.2.3.1 Board ID, Revision, S/N, and location reporting

The following must be present in a computer readable form to assist in diagnostics and health monitoring. Additionally, as noted, some must also be visible in a human readable form.

16.2.3.1.1 Board ID

The board ID is a simple text description in a very few words of the boards function accompanied by a part number that corresponds to the description, i.e., NGAO Tomography Processor, K100.

16.2.3.1.2 Revision

The board major revision number must be updated with any change to any layer. This must be placed in a location that is easily visible and must be in copper. A sub-revision location immediately adjacent to the major revision must be available for manual entry (indelible ink) and rework that changes the board such as the addition of wires or the addition or substitution of components.

16.2.3.1.3 S/N

Each board must have a space adjacent to the revision location where the serial number of the board can be written.

16.2.3.1.4 Location

If there are multiple identical board of identical types installed (i.e., Tomography boards or WFS boards) there must be a means of identifying in which position the board sits in the arrangement of identical boards. To insure that it this cannot be incorrectly set, this should be not be set manually, but automatically by its position in the installation: a coded position socket into which each board plugs that is coded during manufacture.

16.2.3.2 Trouble shooting and monitoring capability

Basic trouble shooting procedures will need to be performed occasionally: voltage or waveform viewing with a meter or scope are typical examples. All custom boards must have easily accessible connections to measure its voltages and critical waveforms such as clocks. For high speed clocks or differential signals where a scope probe could significantly alter the signal in question, a buffer should be provided between the signal and the connection.

If an extender is to be used to perform these operations, the operation of the board in a system during normal operations must be demonstrated using such an extender.

All computer systems should include a general purpose digital I/O board that can be used by software components to signal important events, such as start of module, error conditions, completion of task, etc.

16.2.3.3 Cable Attachment Verification

Cables and connectors should be designed in such a manner that a device at one end can determine whether the cable is plugged in to the receptacle at the other end.

Circuitry will be included in each board for each cable to perform this analysis and report its result during diagnostics.

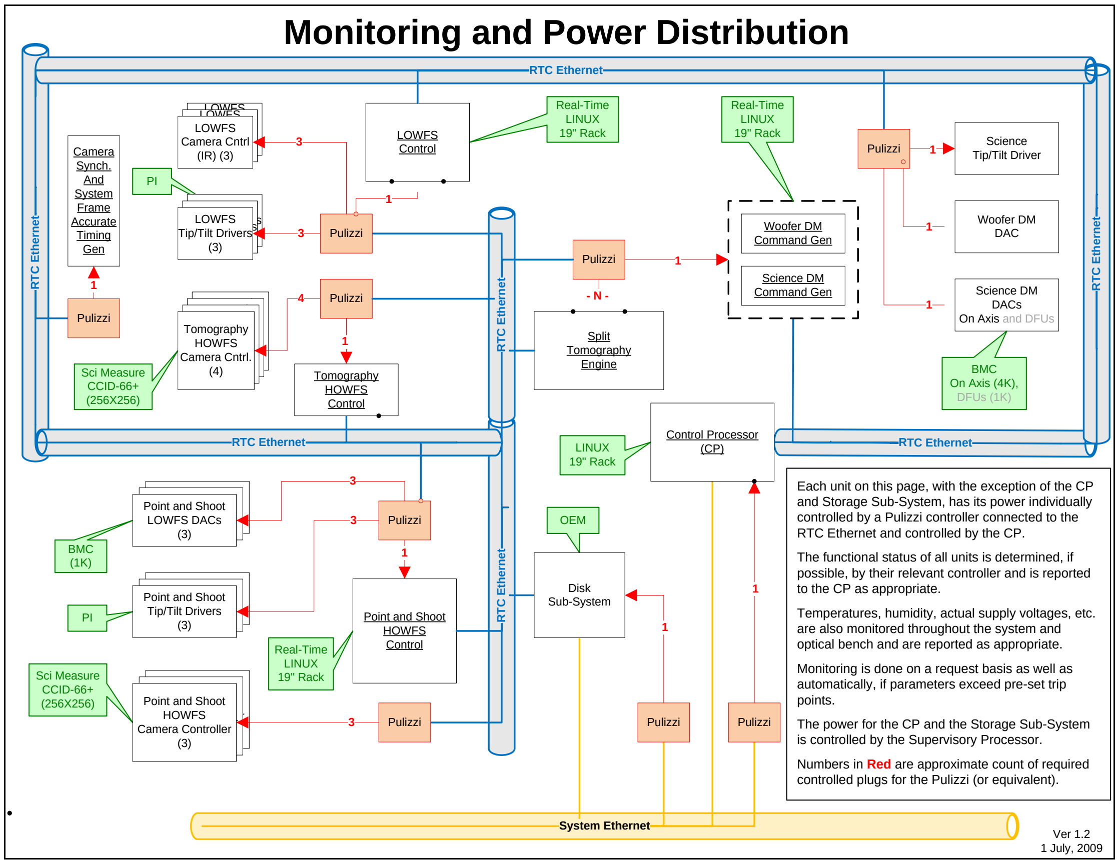
Note: If this feature cannot be implemented, then this requirement must be specifically waived for the specific board type and connector in question.

16.2.3.4 Temperature

Any custom board will have a temperature monitor which will report local temperature over an SPI link.

17. Power Distribution, Environment and Cooling

Power to the various sub-systems will be controlled by a networked power control system such as a Pulizzi. These are controlled by commands through the CP by the AO Control. The power to the CP and the Disk Sub-System will be controlled directly by the AO Control. Figure 53 shows the power control for the RTC.



Each unit on this page, with the exception of the CP and Storage Sub-System, has its power individually controlled by a Pulizzi controller connected to the RTC Ethernet and controlled by the CP.

The functional status of all units is determined, if possible, by their relevant controller and is reported to the CP as appropriate.

Temperatures, humidity, actual supply voltages, etc. are also monitored throughout the system and optical bench and are reported as appropriate.

Monitoring is done on a request basis as well as automatically, if parameters exceed pre-set trip points.

The power for the CP and the Storage Sub-System is controlled by the Supervisory Processor.

Numbers in Red are approximate count of required controlled plugs for the Pulizzi (or equivalent).

Figure 53 Power Distribution

17.1 *Power Sequencing*

Each sub unit in the RTC will be power sequenced during start up and shut down. This will be done in an orderly manner to avoid power spikes in the telescope system. The CP will perform the startup and shut down sequence on command from the AO Control. All power control is accessible by the CP.

17.1.1 FPGA Power Dissipation

The majority of the power dissipated by the tomography engine is in the FPGA's. There will be less than 150 FPGAs, each dissipating a maximum of approximately 15 Watts for a total of 2.25 KW.

17.1.1.1 Sample Chip Power Analysis

The initial evaluation of the FPGA design was based on the following estimated parameters in Table 14.

Item	Value	Comment
Chip:	VSX95T	
System Clock:	100 MHz	
LVDS Clock:	600 MHz	
Sub apertures per chip:	100 (4 sub apertures per PE)	
Process Parameters	Nominal	
Ambient Air Temp	30° C	
Air Flow	250 LFPM (moderate air flow)	
Heat Sink	High Profile (Cool junctions)	
Junction Temp (θ_J)	52° C	
I/O utilization:	100%	
DSP-48 Mult/Acc utilization	40%	
Logic utilization	56%	

Table 14 Preliminary FPGA Power Dissipation Analysis

The results of the preliminary power estimate per chip was a worst case of < 15 Watts.

17.1.2 Tomography Engine Cooling

The initial results of the pre-design power estimator tool look good. We are not pushing any limits yet. No exotic cooling requirements seem indicated. A more detailed examination will be done later with a sample design actually synthesized and routed into a chip.

17.2 *DM command Generation*

The DM Command Generators reside in Linux PCs and require no special cooling beyond that provided by the manufacturer.

17.3 *Disk Sub-System*

The Disk Sub-System requires no special cooling beyond that provided by the manufacturer.

17.4 ***Control Processor (CP)***

The CP resides in a Linux PC and requires no special cooling beyond that provided by the manufacturer.

17.5 ***Environment***

The environmental conditions will be monitored to insure they fall within the operating bounds of the various sub-systems: this includes monitoring humidity, temperature, and air flow. These are all monitored by the RTC.

18. Other Hardware and Software not Otherwise Covered

18.1 *Non RTC System State Information*

In order to maintain the state of the system for accurate and easy use of the telemetry data, several key pieces of non-RTC information must be maintained. These include the following.

- Shutters
- Laser fiber In/Out actuators
- Laser fiber On/Off and intensity
- Filter wheel positions

18.2 *Internal Networking*

Networking between RTC components will be over Gbit Ethernet.

18.3 *Interlocks*

Air flow, temperature, humidity, system voltages will all be monitored to insure they stay within predetermined limits. Operation outside those limits will result in an error being generated and sent to the AO Control and the trip of an interlock, shutting down the sub system.

Warnings will also be sent to the AO Control when the values approach the critical levels. These levels will be set by the AO Control.

18.4 *Inter-Unit Cables*

Cables between units must provide the ability to verify the physical connection has been made between both units from a remote location. An example would be a loopback pair in the cable, with a permanent jumper between the pins on the connector. Other methods can be used, but the requirement is a direct method to insure connectivity from a remote location.

19. Appendices

Appendix A Glossary

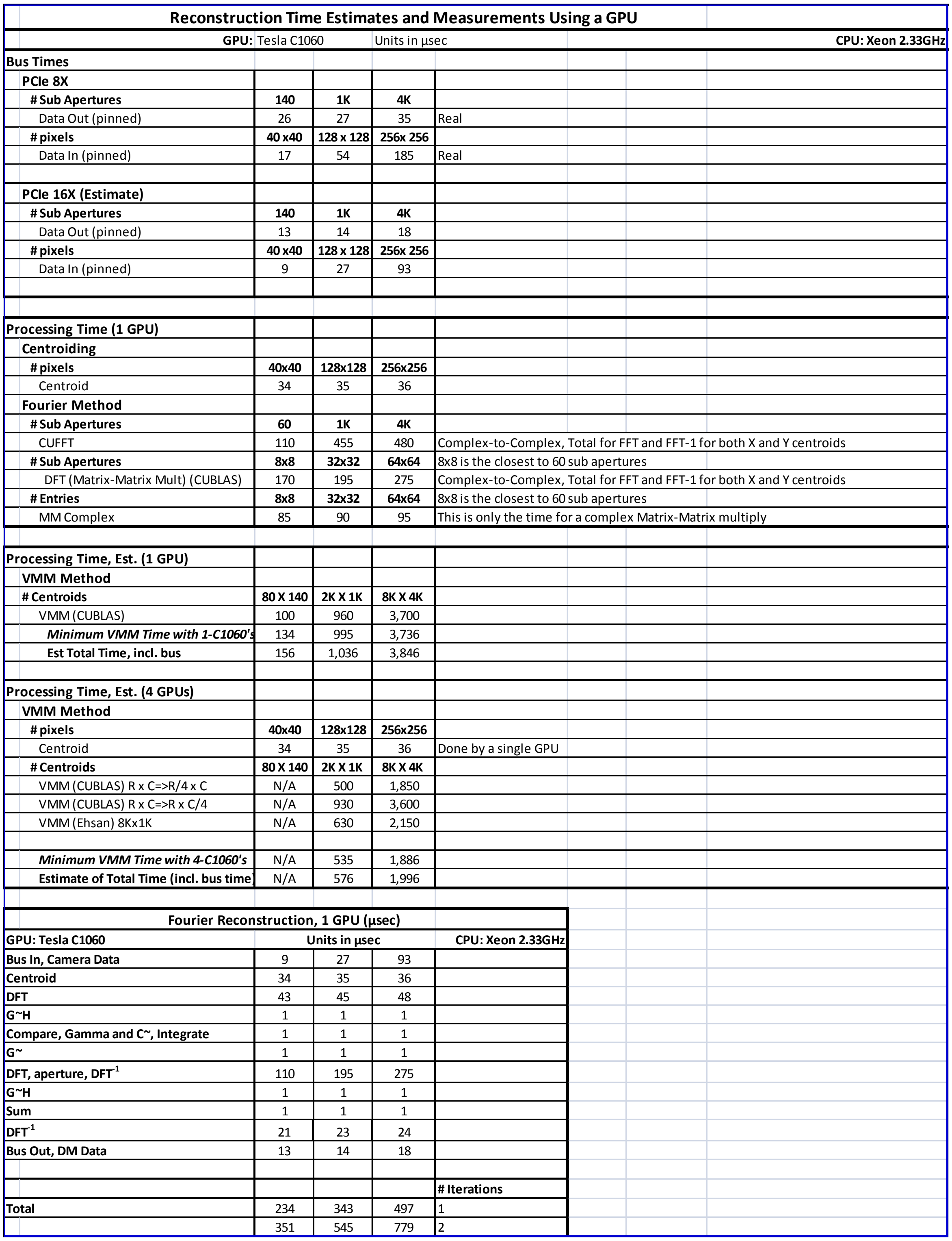
Item	**Description**
AO Control	Supervisory controller for the AO system
BER	Bit Error Rate
Camera Link™	
Centroid	
Closed Loop Control	
Container/component	
COTS	Commercial Off the Shelf
CP	Control Processor for the RTC
CP	Control Processor
DAC	Digital to Analog Converter
DM	Deformable Mirror
DUT	Device under test
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
Harvard Architecture	Data and instruction storage are separate for faster operation
HOWFS	High Order Wave Front Sensor
Junction Temperature	
Latency	
LGS	Laser Guide Star
LOWFS	Low Order Wave Front Sensor
LVDS	Low Voltage Differential Signaling
MTBF	Mean Time Between Failures
MTBR	Mean Time Before Repair
NGS	Natural Guide Star
OEM	Original Equipment Manufacturer
OP	Offload Processor
Open Loop Control	
PE	Processing Element
Pipelined	
Point-and-Shoot	
RTC	Real Time Computer
SEU	Single Event Upset

Item	**Description**
SIMD	Single Instruction Multiple Data
Tip/Tilt Mirror	
Tip/Tilt stage	A tip/tilt actuator that has no mirror, but can move one.
Tomography	
Tweeter	High order DM
WFS	Wave Front Sensor
Woofer	Low order DM
OPD	Optical Path Delay

Appendix B GPU RTC Benchmarking

GPU Computation Speeds

The GPU has speeds capable of implementing the WFS algorithms for the Point-and-shoot HOWFS with some modest optimization. To implement the Tomography WFS, we would use multiple GPUs.



Reconstruction Time Estimates and Measurements Using a GPU

GPU:	Tesla C1060	Units in µsec		CPU: Xeon 2.33GHz
Bus Times				
PCIe 8X				
# Sub Apertures	**140**	**1K**	**4K**	
Data Out (pinned)	26	27	35	Real
# pixels	**40 x40**	**128 x 128**	**256x 256**	
Data In (pinned)	17	54	185	Real
PCIe 16X (Estimate)				
# Sub Apertures	**140**	**1K**	**4K**	
Data Out (pinned)	13	14	18	
# pixels	**40 x40**	**128 x 128**	**256x 256**	
Data In (pinned)	9	27	93	
Processing Time (1 GPU)				
Centroiding				
# pixels	**40x40**	**128x128**	**256x256**	
Centroid	34	35	36	
Fourier Method				
# Sub Apertures	**60**	**1K**	**4K**	
CUFFT	110	455	480	Complex-to-Complex, Total for FFT and FFT-1 for both X and Y centroids
# Sub Apertures	**8x8**	**32x32**	**64x64**	8x8 is the closest to 60 sub apertures
DFT (Matrix-Matrix Mult) (CUBLAS)	170	195	275	Complex-to-Complex, Total for FFT and FFT-1 for both X and Y centroids
# Entries	**8x8**	**32x32**	**64x64**	8x8 is the closest to 60 sub apertures
MM Complex	85	90	95	This is only the time for a complex Matrix-Matrix multiply
Processing Time, Est. (1 GPU)				
VMM Method				
# Centroids	**80 X 140**	**2K X 1K**	**8K X 4K**	
VMM (CUBLAS)	100	960	3,700	
Minimum VMM Time with 1-C1060's	134	995	3,736	
Est Total Time, incl. bus	156	1,036	3,846	
Processing Time, Est. (4 GPUs)				
VMM Method				
# pixels	**40x40**	**128x128**	**256x256**	
Centroid	34	35	36	Done by a single GPU
# Centroids	**80 X 140**	**2K X 1K**	**8K X 4K**	
VMM (CUBLAS) R x C=>R/4 x C	N/A	500	1,850	
VMM (CUBLAS) R x C=>R x C/4	N/A	930	3,600	
VMM (Ehsan) 8Kx1K	N/A	630	2,150	
Minimum VMM Time with 4-C1060's	N/A	535	1,886	
Estimate of Total Time (incl. bus time)	N/A	576	1,996	

Fourier Reconstruction, 1 GPU (µsec)

GPU: Tesla C1060		**Units in µsec**		**CPU: Xeon 2.33GHz**
Bus In, Camera Data	9	27	93	
Centroid	34	35	36	
DFT	43	45	48	
G~H	1	1	1	
Compare, Gamma and C~, Integrate	1	1	1	
G~	1	1	1	
DFT, aperture, DFT^{-1}	110	195	275	
G~H	1	1	1	
Sum	1	1	1	
DFT^{-1}	21	23	24	
Bus Out, DM Data	13	14	18	
				# Iterations
Total	234	343	497	1
	351	545	779	2

Table 15 Reconstruction Time Estimates for GPU

GPU Bus Transfer Speeds

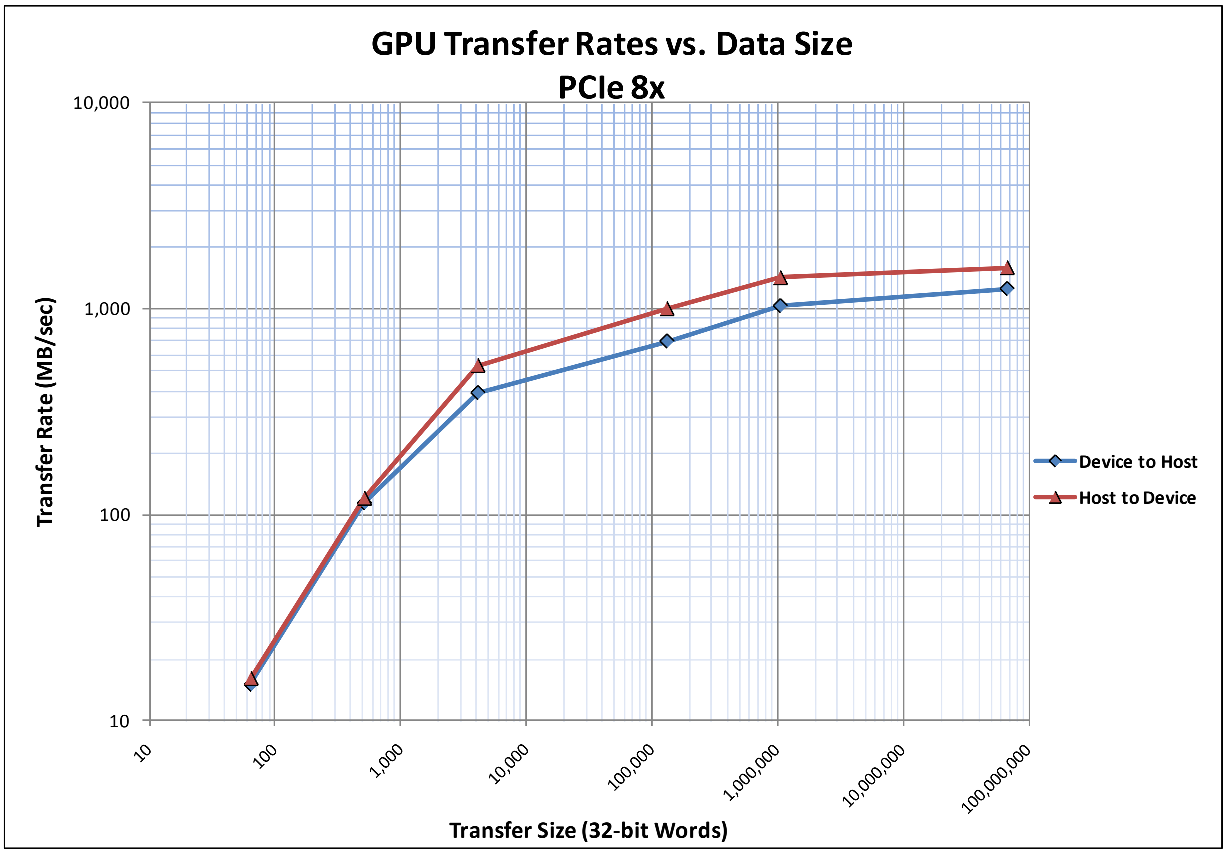


Figure 54 GPU Transfer Rates

It is apparent from the chart above that for small data sizes (<3 KWords: 12 KBytes) the GPU currently does not make efficient use of the potential bus bandwidth available. This can impact latency but can be mitigated by overlapping transfer with computation.

Appendix C Time-related keywords in Keck Observatory FITS files

[15]

As an overview of the time-related keywords found in FITS files from the optical instruments at Keck Observatory please, see this computer-generated documentary list.

The two meanings of "keyword" at Keck

In the subsequent discussion, it is important to distinguish between KTL keywords and FITS keywords.

KTL keywords are obtained by interprocess (and often inter-machine) communication when a client process indicates interest. The values of the KTL keywords communicate information about the state of the various subsystems of the Keck telescopes and instruments. Depending on the nature of the quantity described by the keyword it may be continuously varying during an observation or it may never change. Each KTL keyword is associated with a "service" that corresponds to some subsystem of the operations at Keck. Some KTL services are the responsibility of the Keck staff and describe observatory-wide systems (e.g., DCS describes the telescope pointing and dome, ACS describes the primary mirror). Other KTL services are associated with the particular instrument (e.g., CCD subsystems, motor control subsystems, etc.).

FITS keywords are recorded in the headers of the files, which are produced by the instrument systems when they obtain an image. In many cases, there is a one-to-one correspondence between KTL keywords and FITS keywords. There are, however, a number of FITS keywords inserted into every image that do not correspond to any KTL keyword. There can also be FITS keywords, which correspond to KTL keywords, but where the FITS keyword name does not match the KTL keyword.

Finally, it is relevant to note that the FITS DATE keyword indicates a time stamp related to the construction of the FITS file itself. There should be no expectation that the value of the FITS DATE keyword is related to the time at which the data in the FITS file were acquired.

Historical review of Keck timing keyword systems

During the development of the keyword handling systems at Keck Observatory, there was no requirement for a method of obtaining a precise time stamp. None of the specifications for the initial optical instruments at Keck (HIRES, LRIS, ESI, DEIMOS) included any requirements for precise timing. As a result, the ability of these instruments at Keck to indicate the time for any exposure-related event is haphazard.

The Keck telescope pointing control system (DCS) uses commercial GPS receivers as the time base. The time provided by these receivers has experienced "interesting" events during the GPS W1K rollover in 1999 and during theater-level GPS jamming experiments conducted by DoD. It is not clear what the DCS system reports during a leap second. Other than in these exceptional conditions, the time base used by DCS is relatively reliable.

Any of the Keck instruments is able to ask DCS for its current values of its timing keywords. Most notable among these KTL keywords are DATE-OBS, UTC, and MJD. Reading the KTL DATE-

OBS keyword from DCS provides a FITS Y2K-agreement-compliant value of the calendar date according to Coordinated Universal Time. Reading the KTL UTC keyword from DCS provides a sexagesimal character string representation of the value of Coordinated Universal Time. It is important to note that these are two separate keywords. It is not possible to request both keywords simultaneously, nor to receive values, which are guaranteed to refer to the same instant. As a result, it is possible that around 0 hours UTC the values of the KTL DATE-OBS and UTC keywords from DCS can refer to different calendar days -- thus resulting in a one-day discrepancy in the available notion of the time. (Given that 0 hours UTC occurs around mid-day in Hawaii this will probably never be an issue for observational data.)

The manner by which DCS can be queried for date and time is a KTL keyword request. In the current scheme, the KTL keyword read generates a EPICS request that is sent over the network to the DCS systems. After that request is received and processed in DCS the resulting value is sent back over the network to the KTL requestor. The design of the system provides no guarantees about how long this round trip takes. Under normal circumstances the response to the KTL read is received within a small fraction of a second, but under adverse circumstances the DCS system has been observed to take as long as 30 seconds. In these adverse circumstances, it is not clear how to ascertain what point in the transaction is represented by the time value.

Gathering the time-related keywords at Keck

The initial optical instruments at Keck (HIRES, LRIS, ESI, DEIMOS) monitor the sequence of events of an exposure using a process named watch_ccd. The watch_ccd process is responsible for gathering most of the KTL keywords, which will be written, into the FITS file along with the image data of an exposure. The watch_ccd process has a list of KTL keywords, which indicates when each keyword should be gathered during the exposure sequence. The three points during the exposure are erase, shutter open, and shutter close.

In these instruments, the computer, which has command of the CCD, is called the CCD crate. The CCD crate issues signals to the CCD controller to initiate all operations of the CCD electronics and the shutter. When the CCD crate issues exposure-related signals to the CCD controller, it also transmits a MUSIC broadcast message to the traffic process. The traffic process then re-transmits that MUSIC broadcast message to every other process, one of which is watch_ccd. Each of these hops between processes on the network takes time.

When watch_ccd receives an exposure-related event it proceeds to issue KTL read requests for each of the keywords whose value is desired at that event. These KTL read requests are issued sequentially. Each request must complete before the next KTL read is done. A successful KTL read involves a round trip between watch_ccd and the system(s), which know(s) the value(s) of each keyword. Usually each KTL read succeeds in a small fraction of a second, but under adverse circumstances, the servers for the keywords may respond slowly, or not respond at all.

Each KTL keyword read has a timeout. If the server does not respond within that timeout a KTL read will fail, and watch_ccd will proceed to read the next keyword in its list. There are often dozens, even hundreds of keywords in the lists that watch_ccd has to acquire. The total time to gather the keywords, even under ideal circumstances can be several seconds, and under adverse circumstances, it has been monitored to be as many as 30 seconds or more. If the delay is long then the values of the KTL DATE-OBS and (more importantly) UTC keywords from DCS may not bear much relation to the instant at which the shutter events happened.

Strategies employed to get the best possible time stamps

These issues regarding the validity of the time-related keywords came to the attention of the UCO/Lick Scientific Programming Group during the retrofit, which added the exposure meter to HIRES. Starting with that deployment the CCD readout software has been tailored to provide the best possible indication of event times given the constraints of systems, which were not specified with that as a design requirement. These principles are now in use with HIRES and DEIMOS. They will be applied to LRIS during the Red Mosaic upgrade.

The first remedy was to carefully arrange the order of the keywords in the lists provided to watch_ccd. In the more recent deployments, the list has the KTL UTC keyword as the first in the list. There can be no quicker way of getting that keyword from DCS.

Another remedy was to enhance watch_ccd to distribute the keyword collection over three events rather than two. Initially the keywords were only gathered in response to shutter open and shutter close. Those lists were rather large, and not optimally arranged. In the more recent versions of watch_ccd, it is possible to gather keywords in response to the initiation of the erase of the CCD prior to opening the shutter. The objective is to gather keywords whose value tends not to change during erase, and thus reduce the length of the lists of keywords desired at shutter open and shutter close.

Another remedy was to enhance watch_ccd to be able to rename a KTL keyword when writing it to the FITS file. This renaming is often done when it is desired to sample the value of a KTL keyword more than once during an observation. In the case of HIRES the KTL UTC and DATE-OBS keywords are read in response to both shutter open and shutter close. For shutter open, they are written to the FITS file with the same name, and for shutter close, they are written to the FITS file with the names UTC-END and DATE-END. This has been in use for all instruments deployed or upgraded since the HIRES exposure meter. (Note that in the unusual case of an exposure very near to 0 h UTC the combination of all four of these keywords can probably be used to ascertain whether the pair of keywords for one of the shutter events may have straddled the day boundary.)

The most definitive remedy was to create an independent method for watch_ccd to give some indication of the times of the shutter events. In recent versions when watch_ccd receives the shutter events, it immediately queries the system to get the UNIX system time. The values of these queries are inserted into the KTL keywords DATE_BEG (for shutter open) and DATE_END (for shutter close). These keywords are only as precise as one second. Furthermore, these keywords rely on the UNIX system clock being set correctly. In normal operation, the Keck systems should be using NTP to keep their clocks on time, but this is not guaranteed.

Interpretation of time keywords in Keck FITS files

In any instrument, which supplies the FITS DATE_BEG and DATE_END keywords along with the UTC and UTC-END keywords, it is prudent to inspect their values. If the various subsystems were operating nominally then the keywords for shutter open will be in reasonable agreement, as will the keywords for shutter close. Furthermore, the difference between the times of shutter close and shutter open should be in reasonable agreement with the various keywords that indicate the duration of the exposure. If all of these agree, then their values are probably reliable. If they do not agree, then their discrepancies give some indication of how unreliable the time stamps may be.

Steve Allen <sla@ucolick.org>

2008/12/08

Appendix D WFS Detailed Timing Calculations

WFS Performance		Fixed Subap Size	Units	Comments
Ver 5.0	15 January, 2010			
Assumed Parameters				
Layers		1		L
Science Objects		1		Sci_Obj
No. of fwd prop. cycles needed to propagate all science objects		1		Sci_Obj_Iter = ABS(1 + INT((Sci_Obj-1)/ MAX(GS, L)))
Atmosphere Height (height of highest layer)		15	Km	Atmos_Ht
Guide Star Contselation Angle (full width, edge to edge)		40.0	Arc Sec	FOV
Max Angle from Zenith		54	Degrees	Angle_from_Zenith
Tomography Guide Stars		4		GS
Subapertures Across Primary Aperture		60		SA
Primary Aperture Size		10	meters	Dia
Subaperture Size		16.7	cm	SA_Size = Dia * 100 / SA
Number of Extended Subapertures at Max Height		91		ESA, Extended sub apertures at max height **[1] [8]**
Clock Speed		100	MHz	CLK
Transfer In/Out time		182	Cycles	Xfer_Cycles = ESA * 2 **[2]**
Number of iteration cycles to converge		3		Cnvg_Iter
Time for Each Elements of the Algorithm				
Error calculation		20	Cycles	Error Calculation
New Value Calculation		20		New Est Value
Aperturing (2-D DFT^{-1}/DFT)		647	Cycles	Aperture = 7 * ESA + 10 **[2]**
Pre conditioning (matrix multiply + 2-D DFT^{-1}/DFT)		647	Cycles	Pre_Cond = 7 * ESA + 10 **[2]**: **Not used in WFS**
Filtering (Kolmogorov applied to back propagated error)		20	Cycles	Filter
Scaling during tomography (includes fwd and back proj)		736	Cycles	Scale = 2 * (4 * ESA + L^2 + 2); Fwd/Back: X/Y: complex **[3]**: **Not used in WFS**
Scaling of Science Object(s) with Variable sub aperture size		N/A	Cycles	= Sci_Obj_Iter * Scale_V / 2
Use Fast Scaling		no		Potential for a scaling speed-up to be investigated **[7]**
Propagating science objects		659	Cycles	Prop_Sci = L * 2 + 2 + Aperture
Total Iteration Time				
Total Clock Cycles per Iteration		1,445	Cycles	It_Cycles = Error_Calc + New_Value + (2 * Aperture) + Filter + ESA
Total Time per Iteration		**14.5**	µSec	It_Time = It_Cycles / CLK
Summary				
Total Cycles/Time to Converge				
Clock Cycles to Converge		4,335	Cycles	Cnvg_Cycles = Cnvg_Iter * It_Cycles **[4]**
Time-to-Converge		43.4	µSec	Cnvg_Time = Cnvg_Cycles / Clk
Total Tomography Time, including load, unload and science propagation				
All-Layers-on-a-chip				
Min Time (1 Voxel per Processing-Element)		**56.9**	µSec	= ((Xfer_Cycles + Prop_Sci) / CLK) + Cnvg_Time + (Xfer_Cycles + Prop_Sci/2) / CLK
Max Frame Rate (1 Voxel per Processing-Element)		**16.62**	KHz	Max_Fr_Rate = 1000 / (((Xfer_Cycles + Prop_Sci) / CLK) + Cnvg_Time + (Xfer_Cycles + Prop_Sci) / CLK)
4-voxels-per-Processing-Element Time		**228**	µSec	4X
4-voxels-per-Processing-Element Frame Rate		**4.40**	KHz	

Notes:

1	ESA = INT(((100 * Dia / SA_Size) + 2 * ((1 / COS(RADIANS(Angle_from_Zenith)) * (Atmos_Ht * 1000 * 100) * SIN(RADIANS((FOV / 2) / 60))) / (SA_Size)) + 1)
2	Assumes one clock to transfer a real and one for an imaginary part => 3 clocks per element for real 2-D DFT and 4 for Imaginary 2-D IDFT (ie. 7N clocks per Real DFT/IDFT pair)
3	Scaling
	Shift a complex number for L-1 layers (ground layer doesn't shift)
	Each shift (using bi-directional shifting) is a max of 17% (total shifts ~ 50%) for each direction => ~100% for a single scaling
	Scaling is done for forward and back prop (and science obj for variable sized subaps)
4	For variable sized subaps, we need to also scale the forward projection of the science objects + V_scale
5	Includes time to propagate the science objects, load data from the WFS and send data to the DM Command Processor
6	Includes overhead of board-per-layer plus time to load data from the WFS and send data to the DM Command Processor
7	Possib le reduction in scaling time from 2 * (4 * ESA + 2 * L^2 + 2) to 2 * (ESA + 2 * L * GS + 2)
8	For variable sized subaps, we use: IF((SA + 20) > ESA, ESA, SA+20)

Table 16 Detailed WFS Timing Calculations

Appendix E Tomography Engine Detailed Timing Calculations

Tomography Performance			
Ver 5.0 15 January, 2010	**Fixed Subap Size**	**Units**	**Comments**
Assumed Parameters			
Layers	5		L
Science Objects	1		Sci_Obj
No. of fwd prop. cycles needed to propagate all science objects	1		Sci_Obj_Iter = ABS(1 + INT((Sci_Obj-1)/ MAX(GS, L)))
Atmosphere Height (height of highest layer)	15	Km	Atmos_Ht
Guide Star Contselation Angle (full width, edge to edge)	40.0	Arc Sec	FOV
Max Angle from Zenith	54	Degrees	Angle_from_Zenith
Tomography Guide Stars	4		GS
Subapertures Across Primary Aperture	60		SA
Primary Aperture Size	10	meters	Dia
Subaperture Size	16.7	cm	SA_Size = Dia * 100 / SA
Number of Extended Subapertures at Max Height	91		ESA, Extended sub apertures at max height **[1] [8]**
Clock Speed	100	MHz	CLK
Transfer In/Out time	182	Cycles	Xfer_Cycles = ESA * 2 **[2]**
Number of iteration cycles to converge	3		Cnvg_Iter
Time for Each Elements of the Algorithm			
Error calculation	20	Cycles	Error Calculation
New Value Calculation	20		New Est Value
Aperturing (2-D DFT^{-1}/DFT)	647	Cycles	Aperture = 7 * ESA + 10 **[2]**
Pre conditioning (matrix multiply + 2-D DFT^{-1}/DFT)	647	Cycles	Pre_Cond = 7 * ESA + 10 **[2]**
Filtering (Kolmogorov applied to back propagated error)	20	Cycles	Filter
Scaling during tomography (includes fwd and back proj)	832	Cycles	Scale = 2 * (4 * ESA + L^2 + 2); Fwd/Back: X/Y: complex **[3]**
Scaling of Science Object(s) with Variable sub aperture size	N/A	Cycles	= Sci_Obj_Iter * Scale_V / 2
Use Fast Scaling	no		Potential for a scaling speed-up to be investigated **[7]**
Propagating science objects	20	Cycles	Prop_Sci = L * 2 + 2
Total Iteration Time			
Total Clock Cycles per Iteration	2,277	Cycles	It_Cycles = Error_Calc + New_Value + Aperture + Pre_Cond + Filter + Scale + ESA
Total Time per Iteration	**22.8**	µSec	It_Time = It_Cycles / CLK
Summary			
Total Cycles/Time to Converge			
Clock Cycles to Converge	6,831	Cycles	Cnvg_Cycles = Cnvg_Iter * It_Cycles **[4]**
Time-to-Converge	68.3	µSec	Cnvg_Time = Cnvg_Cycles / Clk
Total Tomography Time, including load, unload and science propagation			
All-Layers-on-a-chip			
Min Time (1 Voxel per Processing-Element)	**70.3**	µSec	= Cnvg_Time + Sci_Obj_Iter * (Xfer_Cycles + Prop_Sci) / CLK
Max Frame Rate (1 Voxel per Processing-Element)	**14.22**	KHz	Max_Fr_Rate = 1000 / (Cnvg_Time + Sci_Obj_Iter * (Xfer_Cycles + Prop_Sci) / CLK)
4-voxels-per-Processing-Element Time	**275**	µSec	= Cnvg_Time * 4 + Sci_Obj_Iter * (Prop_Sci + Xfer_Cycles) / CLK
4-voxels-per-Processing-Element Frame Rate	**3.63**	KHz	= 1000 / (Cnvg_Time * 4 + Sci_Obj_Iter * (Prop_Sci + Xfer_Cycles) / CLK)

Notes:

1	ESA = INT(((100 * Dia / SA_Size) + 2 * ((1 / COS(RADIANS(Angle_from_Zenith)) * (Atmos_Ht * 1000 * 100) * SIN(RADIANS((FOV / 2) / 60))) / (SA_Size)) + 1)
2	Assumes one clock to transfer a real and one for an imaginary part => 3 clocks per element for real 2-D DFT and 4 for Imaginary 2-D IDFT (ie. 7N clocks per Real DFT/IDFT pair)
3	Scaling
	Shift a complex number for L-1 layers (ground layer doesn't shift)
	Each shift (using bi-directional shifting) is a max of 17% (total shifts ~ 50%) for each direction => ~100% for a single scaling
	Scaling is done for forward and back prop (and science obj for variable sized subaps)
4	For variable sized subaps, we need to also scale the forward projection of the science objects + V_scale
5	Includes time to propagate the science objects, load data from the WFS and send data to the DM Command Processor
6	Includes overhead of board-per-layer plus time to load data from the WFS and send data to the DM Command Processor
7	Possib le reduction in scaling time from 2 * (4 * ESA + 2 * L^2 + 2) to 2 * (ESA + 2 * L * GS + 2)
8	For variable sized subaps, we use: IF((SA + 20) > ESA, ESA, SA+20)

Table 17 Detailed Tomography Timing Calculations

Appendix F PE Detailed Design

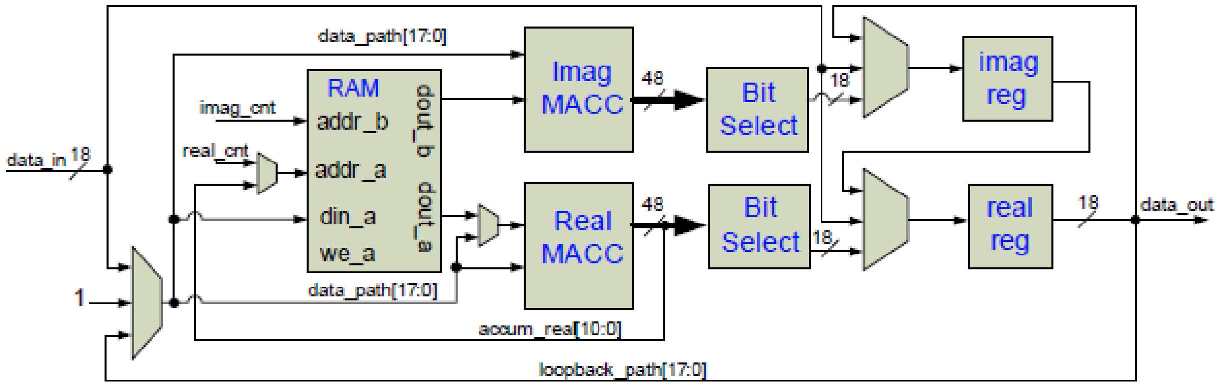
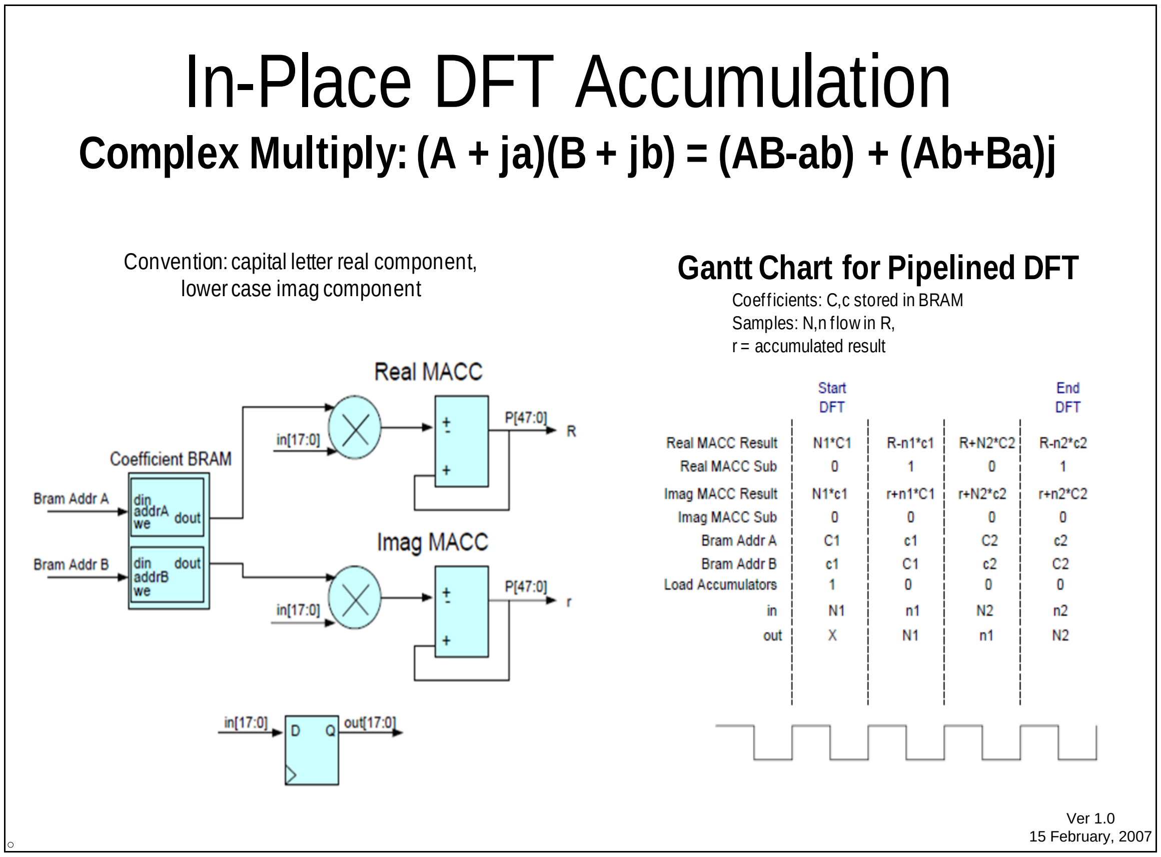


Figure 55 PE Detailed Design



In-Place DFT Accumulation

Complex Multiply: (A + ja)(B + jb) = (AB-ab) + (Ab+Ba)j

Convention: capital letter real component, lower case imag component

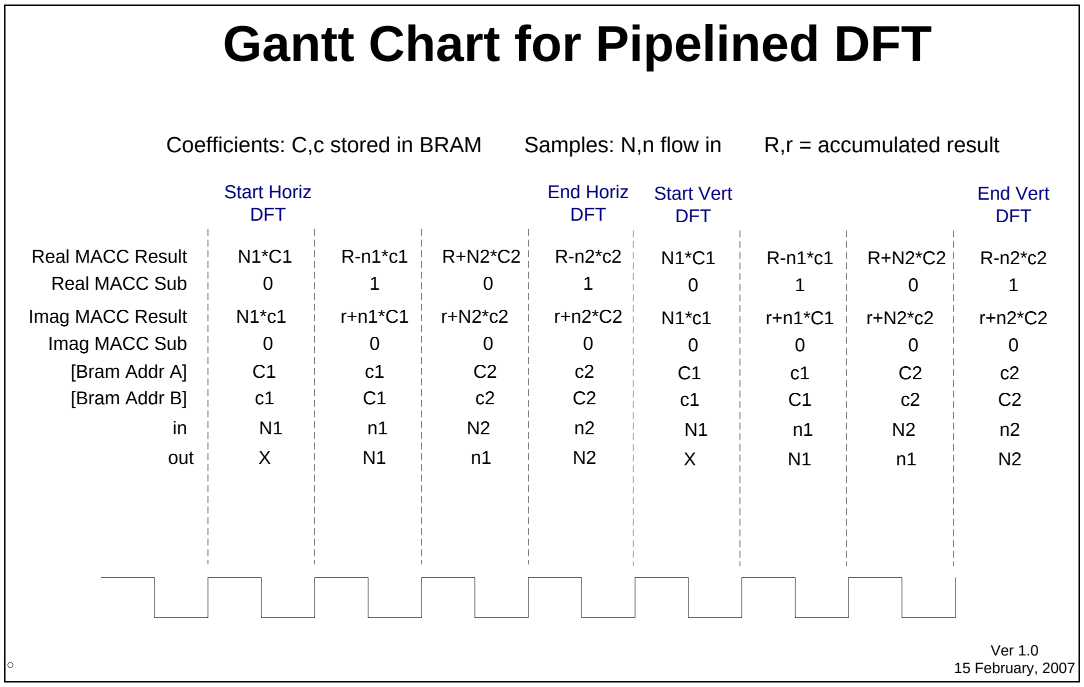
Gantt Chart for Pipelined DFT

Coefficients: C,c stored in BRAM
Samples: N,n flow in R,
r = accumulated result

	Start DFT			End DFT
Real MACC Result	N1*C1	R-n1*c1	R+N2*C2	R-n2*c2
Real MACC Sub	0	1	0	1
Imag MACC Result	N1*c1	r+n1*C1	r+N2*c2	r+n2*C2
Imag MACC Sub	0	0	0	0
Bram Addr A	C1	c1	C2	c2
Bram Addr B	c1	C1	c2	C2
Load Accumulators	1	0	0	0
in	N1	n1	N2	n2
out	X	N1	n1	N2

Ver 1.0
15 February, 2007

Figure 56 In-place DFT accumulation



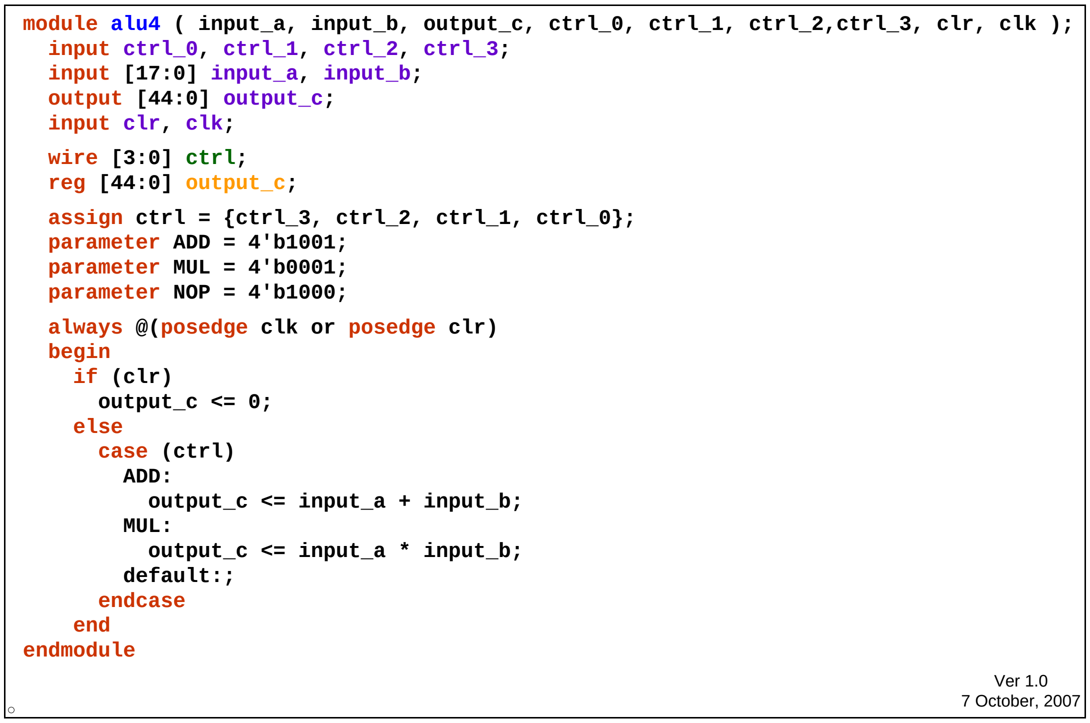
Gantt Chart for Pipelined DFT

Coefficients: C,c stored in BRAM   Samples: N,n flow in   R,r = accumulated result

	Start Horiz DFT			End Horiz DFT	Start Vert DFT			End Vert DFT
Real MACC Result	N1*C1	R-n1*c1	R+N2*C2	R-n2*c2	N1*C1	R-n1*c1	R+N2*C2	R-n2*c2
Real MACC Sub	0	1	0	1	0	1	0	1
Imag MACC Result	N1*c1	r+n1*C1	r+N2*c2	r+n2*C2	N1*c1	r+n1*C1	r+N2*c2	r+n2*C2
Imag MACC Sub	0	0	0	0	0	0	0	0
[Bram Addr A]	C1	c1	C2	c2	C1	c1	C2	c2
[Bram Addr B]	c1	C1	c2	C2	c1	C1	c2	C2
in	N1	n1	N2	n2	N1	n1	N2	n2
out	X	N1	n1	N2	X	N1	n1	N2

Ver 1.0
15 February, 2007

Figure 57 Gantt chart for pipelined DFT



```
module alu4 ( input_a, input_b, output_c, ctrl_0, ctrl_1, ctrl_2,ctrl_3, clr, clk );
  input ctrl_0, ctrl_1, ctrl_2, ctrl_3;
  input [17:0] input_a, input_b;
  output [44:0] output_c;
  input clr, clk;

  wire [3:0] ctrl;
  reg [44:0] output_c;

  assign ctrl = {ctrl_3, ctrl_2, ctrl_1, ctrl_0};
  parameter ADD = 4'b1001;
  parameter MUL = 4'b0001;
  parameter NOP = 4'b1000;

  always @(posedge clk or posedge clr)
  begin
    if (clr)
      output_c <= 0;
    else
      case (ctrl)
        ADD:
          output_c <= input_a + input_b;
        MUL:
          output_c <= input_a * input_b;
        default:;
      endcase
    end
endmodule
```

Ver 1.0
7 October, 2007

Figure 58 Example Verilog code for ALU

Appendix G Itemized List of Cameras, T/T Mirrors, DMs, and Actuators

Ver 1.3	8 December, 2009	NGAO Current Target Design				
	WFS	DM	LOWFS	Tip/Tilt	Tip/Tilt	
Item	Cameras	Size (K)	Cameras	Mirrors	Actuators	Comments
LOWFS 1 *	N/A	1,024	1	1	N/A	LOWFS DMs get their wavefront correction from the POS WFS
LOWFS 2 *	N/A	1,024	1	1	N/A	LOWFS stars are at 2am to preserve sky coverage.
LOWFS 3 *	N/A	1,024	1	1	N/A	LOWFS get their T/T from the LOWFS, not a LGS
PAS 1 **	128x128	N/A	N/A	1	N/A	Point and Shoot LGSs are used only for correcting the
PAS 2 **	128x128	N/A	N/A	1	N/A	wavefront of their LOWFS star, not for tomography
PAS 3 **	128x128	N/A	N/A	1	N/A	
Tomog 1	256x256	N/A	N/A	1	N/A	
Tomog 2	256x256	N/A	N/A	1	N/A	
Tomog 3	256x256	N/A	N/A	1	N/A	
Tomog 4	256x256	N/A	N/A	1	N/A	
Woofer	N/A	400	N/A	0	1	Woofer T/T value is calculated from the on-axis tomography
Tweeter	N/A	4,096	N/A	0	N/A	Woofer will serve as T/T for the central science object
MOAO 1	N/A	(1K)	N/A	(1)	N/A	MOAO T/T values are calculated from off-axis tomography
MOAO 2	N/A	(1K)	N/A	(1)	N/A	
MOAO 3	N/A	(1K)	N/A	(1)	N/A	
MOAO 4	N/A	(1K)	N/A	(1)	N/A	
MOAO 5	N/A	(1K)	N/A	(1)	N/A	
MOAO 6	N/A	(1K)	N/A	(1)	N/A	
Totals	7	5	3	10	1	

() Indicates possible future expansion

MOAO 2 - 7 may not be installed in initial configuration, but the RTC must be able to handle them, if and when they are installed, while still meeting the PRD requirements: 2KHz, etc.

* What size are the T/T and Truth cameras?

** T/T mirrors for the LOWFS LGS get their value from their LOWFS LGS, not their LOWFS star

*** T/T mirrors for the HOWFS get their values from their HOWFS, not from tomography

Table 18 List of sensors and actuators connected to the RTC

Appendix H Real-Time Control Numerical Analysis

Introduction

Any hardware that executes a numeric algorithm has an inherent loss of accuracy in its representation of real numbers given it has a fixed word length. This applies to floating point as well as integer based systems and is independent of implementation, whether it is on FPGAs, Graphic Processing Units (GPUs), or DSPs.

This report analyzes the system requirements associated with producing the required tomographic accuracy for the NGAO RTC.

Requirements

From Hardy [16], for 99.4% probability, the peak excursion for which we need to compensate (in meters), assuming overall tilt removal, is given by:

$$\pm 0.149\lambda_0 \left(\frac{D}{r_0}\right)^{5/6} \text{ meters}$$

where **λ₀** is the wavelength at which **r₀** is measured, and **D** is the telescope aperture.

For λ_0 = 0.5 µm, r_0 i= 16 cm, and **D** = 10 m, this range is ±2.34 µm or a total required range of ~4.7 µm.

We represent the numbers in this range by a 12-bit integer giving a minimum resolution of 4.7/2^^{12} or 1.15 nm.

However, using the total LODM Stroke referenced in FR-493 we have ±4 µm. Thus we have 8 µm total. This requires a 13-bit integer for a resolution of 8/2^^{13} or 0.98 µm.

We have chosen to use the latter.

We need to insure that from camera input to DM output that we can guarantee to retain this accuracy.

General Analysis

Loss of accuracy occurs from several sources: the input data itself, the algorithm used, the arithmetic capabilities of the system, and the coefficient data used by the algorithm.

A rounded quantized number carries an inherent ±½-bit potential for error, since we cannot know if it was slightly more or less than the current lowest order bit. All computer systems, floating point or fixed point have these issues. In large computations, these errors can build up and result in a loss of accuracy in the result.

For the purpose of this analysis, we assume the camera data is accurate to 12-bits.

We start by normalizing the data to the basic word length at the start and end of each stage, so that at the input to the tomography calculation, our wavefronts use the full range of values to represent 4.7 µm.

We maintain this scaling through to the output to the DMs.

Possible sources of loss of accuracy

Each item below has the potential to introduce errors into the result. In this analysis, we use the standard deviation of the error for each error source to calculate the total standard deviation of the result. This standard deviation must be less than or equal to the accuracy required by the tomography error specified above, 1.15 nm.

Supplied data

For the purpose of this analysis, we assume the input data is exact and has a range of 12 bits.

Coefficient data

During our processing of the input data, we use multiple constant coefficients for processing: Fourier coefficients, C_n^2, Kolmogorov coefficients … The values used here are rounded and truncated versions of their 'real' values. Thus, the method, under our control, used to generate these values will determine their accuracy and the resulting accuracy of the operation. It is necessary to minimize this error to have a standard deviation of <0.98 µm.

Algorithms

The algorithm used can also introduce errors by the type and number of operations performed and by the inherent nature of the algorithm in providing an approximate result.

Overflow

When two signed numbers of length A and B are added, they produce a result that can require N + 1 bits to represent the sum, where N = A + B. When these numbers are multiplied, they can require 2N-1 bits to represent the product. Either of these issues can lead to overflow in long calculations.

The longest sequence of complex multiply-accumulates (MACs) we have is for DFTs or IDFTs. For the real or imaginary part of a complex multiply, there are two multiply accumulates. For a given coefficient calculation in a DFT or IDFT L of these are summed, where L is the number of elements. In the NGAO system, L = 88. Therefore the total number of multiplies is 2L and the total number of additions is 3L. At the end of the calculation, the data is normalized back to the starting precision, in our case 13-bits of accuracy.

The maximum length needed to represent the result of a sequence of additions without loss of precision is $(N - 1) + \log_2(A)$ bits, where N is the maximum size of the operands and M is the number of operations, in bits.

Assuming a data size of 16 bits, a multiply would produce a result of at most 32 bits. If we assume a 32-bit result for each multiply then N=32 and A=3L additions. This gives

$$(32 - 1) + \log_2(264) < 40$$

Thus, the length of the accumulator must be at least 40 bits

In the FPGA implementation, our processors use 18-bit data for storage and processing and have 48-bit accumulators. We therefore have an 8 bit margin and have no overflow potential.

If we increase the precision of our data to the full 18 bits of our word we have a requirement of at least 44 bits, a margin of 4 bits. So, we are still within our capabilities to avoid overflow for the FPGA implementation.

Also, saturation due to the atmosphere or cold starts occurs occasionally. We protect the integrators in the reconstruction and tomography algorithms from winding up and exceeding the ±4 µm.

Arithmetic Errors [17]

A quantized, rounded, signed N-bit number has a maximum quantization error of $\varepsilon = \pm 1/2^{N+1}$. However, using maximum errors leads to overly conservative designs. Instead, in the following we calculate the RMS error to evaluate accuracy.

Addition

If the error is uncorrelated and uniform over this range, the RMS error for additions will be $\frac{\varepsilon}{\sqrt{3}}$.

Thus the RMS error after M additions would be $\frac{\varepsilon M}{\sqrt{3}}$.

Multiplication

If the error is uncorrelated and uniform over this range, its RMS error for multiplications will be $\frac{\varepsilon}{\sqrt{12}}$. Thus the RMS error after M multiplications would be $\frac{\varepsilon M}{\sqrt{12}}$.

Coefficients

Since we control the generation of the coefficients used in our operations, we can keep their RMS errors to $\frac{\varepsilon}{\sqrt{12}}$. Thus the RMS error after M multiplications would be $\frac{\varepsilon M}{\sqrt{12}}$.

Algorithms

The longest string of operations in the tomography algorithm consists of two 2D DFT/IDFT pairs of size MXM. Each element of a 2D DFT/IDFT pair has N = ~20M operations. The RMS associated with this is [17]:

$$E_{RMS} = \sqrt{\left(N-1\right)\left(\frac{1}{12}+\frac{1}{3}\right)\varepsilon^2 + (N-1)\sigma_C{}^2}$$

$$= \frac{\varepsilon}{\sqrt{2}}\sqrt{(N-1)}$$

With M = 144, this represents an RMS error of ~27 which is a loss of < 5 bits of accuracy. Our basic word length is 18 bits, so the resulting accuracy is equal to 13 bits, equal to the 13 bits required.

Conclusions

As a note, while the current hardware implementation utilizes FPGAs using scaled fixed-point arithmetic an alternative approach using Graphic Processing Units (GPUs) has also been investigated (although less thoroughly). Initial analysis shows that the current code and architecture for the FPGA approach can be moved almost intact to a GPU approach. However, the same is not true of attempting to move to a DSP implementation.

Appendix I Tomography Array Size and Cost Estimate

Estimate of the Tomography Engine Array Size and Cost

Virtex 6

(Cost does not include power supplies, fans, rack, etc.)

Wd Size (bits)	PE Clock Speed (MHz)	Req. bit Rate per Voxel (Gb)	LVDS Xfer Rate (Mb)		LVDS Ports Needed per Voxel	Array Size (per side)	Layers	Number of Chips per Board
18	100	1.80	600		3	91	5	9

22 July, 2009
ver 1.5

Chip	DSP-48s Avail.	DSP-48s Req.	I/O Pins Avail per Chip	I/O Not Used per Chip	I/O Pins Avail. per Face	LVDS Ports Avail. per Face	GTH (3Mb)	GTX (5Mb)	Max. Voxels per Face	Data Rate Needed Per Face (Gb)	Sub Aps per chip (5 layers, 1 Voxel per PE)	Sub Aps per chip (5 layers, 4 Voxels per PE)	Total Chips	Number of Boards	Est. Cost per Board ($)	Dist Cost for 1 chip ($)	Total Chip Cost ($K)	Total Array Cost (K$)	Chips	Boards	Cost	Approx. Incremental Cost of 4X
6VSX315T	1,344	360	720	0	180	90	0	24	30	54	36	144	57	6	$5,000	$4,057	$231	**$261**	**228**	**24**	**1,045**	784
6VSX475T	2,016	490	840	0	210	105	0	36	35	63	49	196	42	5	$5,000	$12,000	$504	**$529**	**168**	**20**	**2,116**	1,587
6VLX75T	288	250	640	40	160	80	0	12	26	46.8	25	100	82	9	$5,000	$1,000	$82	**$127**	**328**	**36**	**508**	381
6VLX130T	480	250	600	0	150	75	0	20	25	45	25	100	82	9	$5,000	$1,800	$148	**$193**	**328**	**36**	**770**	578
6VLX195T	640	250	600	0	150	75	0	20	25	45	25	100	82	9	$5,000	$2,500	$205	**$250**	**328**	**36**	**1,000**	750
6VLX240T	768	360	720	0	180	90	0	24	30	54	36	144	57	6	$5,000	$2,885	$164	**$194**	**228**	**24**	**778**	583
6VLX365T	576	360	720	0	180	90	0	24	30	54	36	144	57	6	$5,000	$4,600	$262	**$292**	**228**	**24**	**1,169**	877
6VLX550T	864	810	1,200	120	300	150	0	36	50	90	81	324	25	3	$5,000	N/A	N/A	**N/A**	**N/A**	**N/A**	**N/A**	N/A
6VLX760T	864	810	1,200	120	300	150	0	0	50	90	81	324	25	3	$5,000	N/A	N/A	**N/A**	**N/A**	**N/A**	**N/A**	N/A
6VHX250T	576	40	320	80	80	40	0	48	13	23.4	4	16	517	57	$5,000	N/A	N/A	**N/A**	**N/A**	**N/A**	**N/A**	N/A
6VHX255T	576	160	480	0	120	60	24	24	20	36	16	64	129	14	$5,000	N/A	N/A	**N/A**	**N/A**	**N/A**	**N/A**	N/A
6VHX380T	864	360	720	0	180	90	24	48	30	54	36	144	57	6	$5,000	N/A	N/A	**N/A**	**N/A**	**N/A**	**N/A**	N/A
6VHX565T	864	360	720	0	180	90	24	48	30	54	36	144	57	6	$5,000	N/A	N/A	**N/A**	**N/A**	**N/A**	**N/A**	N/A
																					651	

Wd Size	We use 18 bits per word and we want to transfer 1 word per clock
PE Clock Speed	Clock Speed of each processing element (PE or CPU). Each PE can process one or more voxels
Req. bit Rate per Voxel	Each voxel needs to move in one new word (18 bits) on each clock
LVDS Xfer Rate	Each LVDS differential pair (Port) can operate at this bit rate
LVDS Ports Needed per Voxel	Given the required rate per voxel and the capabilities of the LVDS Ports, this is the number of LVDS ports each voxel needs
Array Size	The size of the tomography array in sub apertures, given the number of sub apertures across the primary, angle from the zenith and FOV
	88 is based on 1 am FOV, 46 Degrees off azimuth, and 54 sub apertures, 10m primary, 15 Km highest layer
Layers	Number of layers in tomography
Number of Chips per Board	Number of chips we put on one board
Chip	Specific chip under consideration
DSP-48s Avail.	The number of DSP-48's (Multiplier/Accumulators) a given chip has available
DSP-48s Req.	The number of DSP-48's (Multiplier/Accumulators) we need inthis chip for this configuration, Each PE requires 2 DSP-48s
I/Os	Number of I/O pins available on a given chip (LVDS ports take two pins)
I/Os per Face	Each face can use 1/4 of the pins available the chip
LVDS Ports Avail. per Face	Each LVDS Port uses two I/O pins
GTP/GTX	Ignore
Voxels per Face	How many voxels will fit, given the "Ports Avail. per Face" and "PortsNeeded per Voxel"
	Each chip is a cube that is "Sub Apertures" x "Sub Apertures" at the base and "Layers" high.
	So, each x and y face of the cube (chip) has Sub Apertures x Layers number of voxels needing to connect to the next chip
	Each face on the cube needs "Ports Needed per Voxel" x Layers x "Sub Apertures" of LVDS ports to talk to the next chip
Data Rate Needed per Face	Given the "Voxels per Face" and the "Req. Rate per Voxel" what is the required data rate needed per face?
Sub Aps per chip (5 layers, 1 Voxel)	Given "Voxels per Face" and "Layers" how many sub apertures would fit in this chip at 1 processing element per voxel
Sub Aps per chip (5 Layers, 4 Voxels)	If we shared one processing element among 4 voxels, how many sub apertures would fit in this chip?
Total Chips	Given the "Array Size" and the "Sub Aps (2KHz)", how many chips will we need?
Number of Boards	Number of boards needed given the "Number of Chips per Board" and "Total Chips"
Est. Cost per board	Estimated cost of one board not including the above chips.
Dist Cost for 1 chip	Catalog price for 1 chip from a distributor
Total Chip Cost	Cost of the chips in the array based on the "Dist Cost"
Total Array Cost	Total array cost with chips and boards

Table 19 Estimate of the Tomography Engine Array Size and Cost

20. References

1 Gavel, D., NGAO Real Time Control Algorithms Document, KAON xxx, 2009.

2 Poyneer, L., Gavel, D., and Brase, J., Fast wave-front reconstruction in large adaptive optics systems with use of the Fourier transform, Journal of the Optical Society of America A, vol. 19, Issue 10, pp.2100-2111, (2002).

3 Tokovinin, A., and Viard, E., Limiting precision of tomographic phase estimation, Journal of the Optical Society of America A: Optics, Image Science, and Vision, Volume 18, Issue 4, April 2001, pp.873-882, (2001).

4 GPI Adaptive Optics Computer (AOC) SDD, 5/3/08, v0.10, Genimi Planet Imager (GPI) Instrument, Critical Software Design Description

5 SPIE 2005 spatial paper

6 Kak, Avinash C. "Principles of Computerized Tomographic Imaging", Ch 7, IEEE Press, 1988

7 "General Purpose Systolic Arrays", Johnson, Hursom, Shirazi, 1993, Computer, IEEE, (p24)

8 H. T. Kung and C. E. Leiserson. Systolic Arrays (forVLSI). Proceedings of Sparse Matrix Symposiums, pages 256–282, 1978.

9 R. Hughey and D.P. Lopresti. Architecture of a programmable systolic array. Systolic Arrays, Proceedings of the International Conference on, pages 41–49, 1988.

10 M. Gokhale. Splash: A Reconfigurable Linear Logic Array. Parallel Processing, International Conference on, pages 1526–1531, 1990.

11 K.T. Johnson, A.R. Hurson, and Shirazi. General-purpose systolic arrays. Computer, 26(11):20–31, 1993.

12 A. Krikelis and R. M. Lea. Architectural constructs for cost-effective parallel computers. pages 287–300, 1989.

13 H.T. Kung. Why Systolic Architectures? IEEE Computer, 15(1):37–46, 1982.

14 Xilinx, Inc. Virtex-5 Xtreme DSP Design Considerations User Guide,7 January 2007.

15 Steve Allen, http://www.ucolick.org/~sla/fits/kecktime.html, Jan 28, 2009

16 Hardy, Adaptive Optics for Astronomical Telescopes, Oxford, 1998

17 Schatzman, James C., Accuracy of the discrete Fourier transform and the fast Fourier transform, SIAM J. Sci. Comput. Vol17, No 5, pp 1150-1166, September, 1996

PAGEREF