University of California

Santa Cruz

Systolic Array Implementation of the Fourier Transform

A thesis submitted in partial satisfaction

of the requirements for the degree of

MASTER OF SCIENCE

In

Computer Engineering

by

Christopher P. Hoseit

June 2013

The Thesis of Christopher P. Hoseit
is approved:

Professor Jose Renau

Marc Reinig

Professor Matthew Guthaus

Tyrus Miller

Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

Christopher P. Hoseit

Systolic Array Implementation of the Fourier Transform

Abstract

Adaptive optics is a relatively new technology originally designed to eliminate negative effects of the atmosphere on telescopic images. This technology allows ground based telescopes to achieve resolutions comparable to the Hubble space telescope. Adaptive optics is the "cheaper" solution to capturing better images than sending telescopes into space. Ground based telescopes can be built with larger mirrors providing more light collecting area and ultimately better pictures. Telescopes on the ground can also be more easily maintained and upgraded than telescopes in space. Using a combination of innovative technology such as Laser Guide stars and deformable mirrors, (MEMs) devices, along with significant software and hardware engineering to implement algorithms allow adaptive optics to improve imaging with telescopes.

This thesis work implements a subset of the hardware system required for the systolic array tomography algorithm on a Field Programmable Gate Array (FPGA). Previous technology analysis has concluded that a FPGA platform is suitable for implementing the tomography algorithm. The algorithm will be verified through simulation and comparing results with values generated in a Microsoft Excel script. The tomography engine is a subset of the real time control system described in [1]. Future work will involve implementing the full scale of the algorithm on all of the required hardware.

This thesis presents: 1) the scalable design of the Fourier transform, 2) implementation results and simulation verification.

Dedication/Acknowledgements

I would like to thank my parents for their support and funding through graduate school, my advisor Jose Renau, Marc Reinig for his help and support throughout this project along with Matthew Guthaus. Thank you to Don Gavel and the Laboratory for Adaptive optics for the equipment and laboratory space.

Chapter 1 Background

Adaptive optics has been prominent for over two decades now and is taking advantage of improvements in technology to engineer better images. Widely used in astronomy, adaptive optics is becoming more prevalent in the medical field as well. Why is adaptive optics necessary when there are space telescopes such as Hubble and Spitzer in space? Cost is a significant factor in any engineering enterprise and deploying, and maintaining space telescopes is very expensive. Designing for the space environment also puts constraints on telescopes, most importantly their size. The greatest benefit of a space telescope is the lack of atmosphere, which means crystal clear images. Adaptive optics enables ground based telescopes to eliminate atmospheric disturbances from earth.

Ground-based telescopes are advantageous to space telescopes in several ways. Telescopes designed for terrestrial implementation can be made much larger, and serviced much more easily. These larger telescopes coupled with adaptive optics technology allows for comparable images to those taken in space. "There is no question that with the advent of adaptive optics, ground-based telescopes of equivalent performance in the spectral bands accessible from Earth can be built and operated at lower cost, even including the laser beacons needed to obtain good sky coverage." [2]

Several systems have been in place at observatories for years. These include the Keck Observatory on Mauna Kea, Hawaii, the Lick Observatory on Mount

Hamilton and the Mount Wilson Observatory. Observatories are usually on tops of mountains or in deserts to take advantage of the favorable observing conditions. The main components of an adaptive optics system include: wavefront sensors, image processing and wavefront correction and a deformable mirror. Figure 2 illustrates the major components of an adaptive optics system. Reference stars in the region of sky near the science object are required to obtain an accurate atmospheric sampling. When natural stars are not available laser guide stars must be used to create an artificial star. Wavefront sensors measure "the direction of propagation of the optical wavefront rather than its optical phase" [2]. The goal of the tomography algorithm is to create an "image" of the atmosphere by calculating the indices of refraction at each layer of the atmosphere.

1.2 Problem

It is desirable to measure the atmosphere so that a deformable mirror can adjust to the distortions detected and recreate a perfect wave front. Turbulence and other disturbances in the atmosphere cause incident light to be distorted instead of planar. The atmosphere can be thought of as part of the optical system and no matter how big the telescope aperture becomes, it will always be limited by the light distorted through the atmosphere. Adaptive optics aims to correct for the atmosphere so that the resolving ability is only limited by the instruments in the telescope.

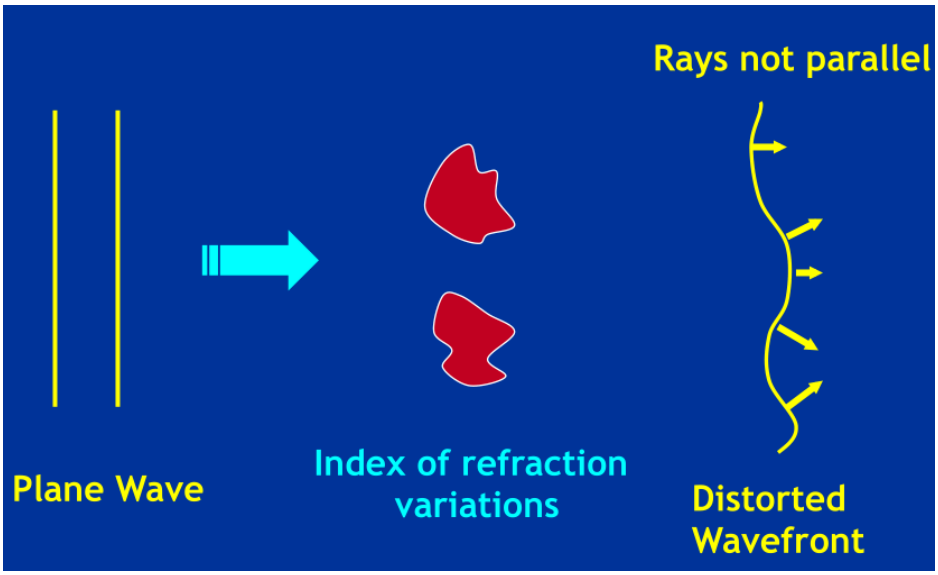


Figure 1 Atmospheric Disturbances [2]

1.3 Motivation

Observation time on telescopes is expensive, and it is desirable to be able to efficiently change observation targets without having to correct for disturbances around each object. Tomography of the atmosphere will enable multiple measurements to be performed within the same field of view of the telescope so that more rapid observations may be performed.

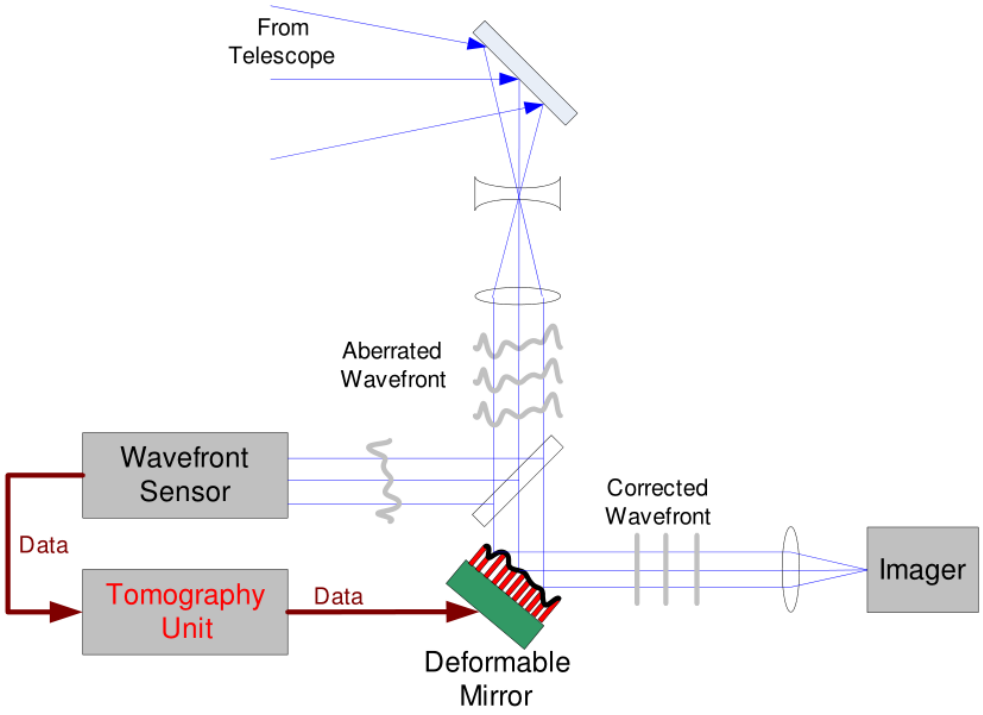


Figure 2 Adaptive optics System [3]

1.4 Components of an Adaptive optics System

1.4.1 Sensors

Wavefront sensors come in several different flavors but all have the goal of measuring the phase of incoming light. "The solution to the problem of wavefront sensing in astronomical adaptive optics is to measure the direction of propagation of the optical wavefront rather than its optical phase. This is done by measuring the wavefront gradients or curvature within an array of zones covering the telescope aperture. [2]" These wavefront sensors provide 2-dimensional data that represents the

sum of a line integral through the atmosphere. Backend processing techniques convert this information into a useful format for determining how to best correct for the distorted light.

1.4.2 Processing

The processing system is illustrated in Figure 2, between the wavefront sensors and deformable mirror. Image processing, tomography computation and deformable mirror control sub-systems require a diverse computation system. Modern systems require multiple implementation technologies to support the complex instructions. "This includes the use of field programmable gate arrays (FPGAs), graphics processing units (GPUs) and multi-core CPUs." [5]

1.4.3 Actuators

Micro electro mechanical (MEMs) devices are used to correct for light's perturbed wavefront by deforming itself in the "opposite" structure at which the wavefront arrives Figure 3, thereby flattening the wavefront. There are many different types of deformable mirrors; many are made using MEMS devices. These devices have many small moveable segments with varying degrees of freedom.

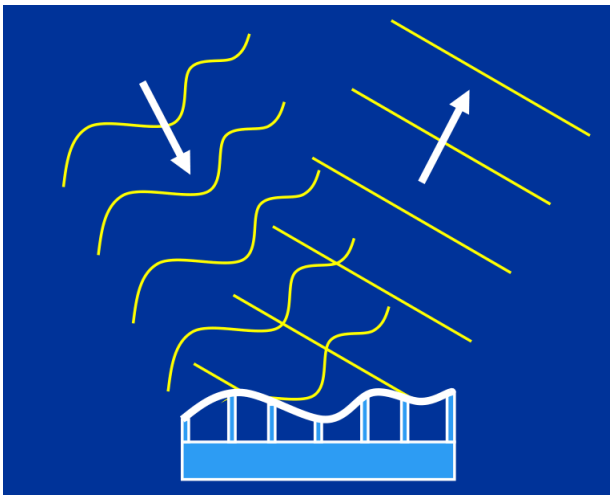


Figure 3 Wavefront Correction

1.5 Current Needs of Observatories

Adaptive optics measurements require the use of a guide star, real or artificial, to take atmospheric measurements. Because of the scarcity of appropriate magnitude stars, artificial guide stars generated by lasers, are often used. Artificial guide stars however, are not at infinity like real guide stars but instead are high in the upper atmosphere. A function of not being at infinity is that measurements of these artificial stars miss information from parts of the atmosphere. A multiple guide star system uses several guide stars to gain a better picture of what the atmosphere looks like, along with minimizing the blind spots, Figure 4.

Given the high cost of operating telescopes it is desirable to observe multiple science objects quickly. Traditional AO systems can obtain 2-D images of the atmosphere by using a single guidestar to look at the atmosphere in a region of the sky. Multi-guide star tomography will enable multiple measurements to be

performed within the same field of view of the telescope so that observations may be performed more rapidly. The data obtained on the volume of atmosphere corresponds to indices of refraction in that particular section of sky.

Figure 4 illustrates sky coverage of a single guide star compared to multiple guide stars. Using only a single guide star allows only partial atmospheric sampling of the object and therefore valuable data is not incorporated into the system's compensation. With multiple guide stars a much more representative sampling of the atmosphere can be obtained.

1.6 Tomography

The goal of the tomography engine is to measure a volume of atmosphere above a telescope and determine the indices of refraction in that volume. Up till now, telescopes have only been able to measure a plane of the atmosphere using a single reference star and a single wavefront sensor. Using only one reference star limits the ability to obtain a good picture of what the atmosphere looks like around a star of interest. Using multiple reference stars along with multiple wave front sensors improves the "picture" of the atmosphere around the science object. Tomography is the process of measuring a volume by taking cross sections of it. There are several methods for performing tomography of the atmosphere and the one being implemented will be discussed.

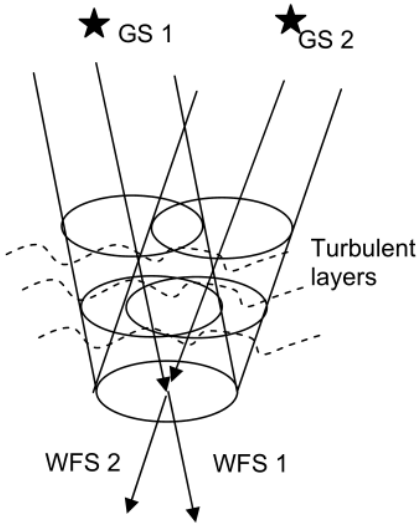


Figure 4 Multiple Guide Star AO [4]

The tomography algorithm is a common problem spanning multiple fields. In the medical industry tomography is used in imaging inside the body such as with CAT (Computed Axial tomography) scans, MRI (Magnetic Resonance Imaging) or X-ray imaging. This process is useful in imaging the atmosphere and determining wavefront distortion. Unperturbed light as seen in space Figure 1 would appear planar and would yield a uniform tomographic depiction.

However because each photon passes through a different part of the atmosphere with a different index of refraction, the light does not all arrive to the telescope at the same time, meaning the wavefront is deformed. The tomographic image in this case shows the variation in light refraction indices.

The atmosphere is a turbulent place, with higher and lower pressures from different temperatures which cause different air densities. The lower, denser layers in the atmosphere carry much more "weight" in that they affect images much more than

the higher and thinner layers of the atmosphere. Upper layers of the atmosphere change much more rapidly but the light refraction index differences and absorption are lower in magnitude than the lower atmosphere. The turbulent atmosphere causes stars to "twinkle", which is undesirable for astronomers because it means the sky is turbulent. Only on particularly still nights will the starlight appear as a steady source. The motivation of adaptive optics is to remove this "twinkle" which blurs and distorts images. A.O. technology enables ground based telescopes to achieve images of equal or greater quality than that of Hubble.

The atmosphere changes at varying rates depending on each site and each geographic location. It is this rate of change that determines the sampling requirements and overall system processing requirements. "System bandwidth requirements demand that we sample the atmospheric turbulence 2,000 times a second, produce a tomographic estimate of the atmosphere from them, and use this estimate to control deformable mirrors to correct the effect of the turbulence." [1] The entire processing loop of the real time control system must be completed within several milliseconds of a measurement because the atmosphere is constantly changing. Implementation of the tomography algorithm on a standard processing system is not feasible because of the high bandwidth real time requirements of the adaptive optics system. The high rate change of adaptive correction and degree of parallel computation of the tomography algorithm lends itself to implementation on an FPGA.

Chapter 2 Tomography Algorithm

2.1 Spatial Domain

Computing the tomography of a volume of atmosphere can be performed in either the spatial domain or spatial frequency domain. In the spatial domain the algorithm operates as follows.

The problem begins with measuring the light received from each reference star. The atmosphere is represented as a grid of volumetric boxes called voxels. Each of these voxels is provided an initial guess as to what the index of refraction is at that point. Light travels through the atmosphere and lands somewhere on the wavefront sensor array. The exact path the light traveled through the atmosphere is not known; all that is known is the summation of all the voxel values, which is incident upon the sub aperture array. The problem then is how to find the components of the sum i.e. the parts of the atmosphere the light traveled through.

The goal is to build a volumetric estimation of the atmosphere's refraction indices. The tomography engine iteratively processes wavefront data to form a current estimate of the atmosphere. A solution to the problem is to know the index of refraction at any subvolume. The available data is 2-D wavefront information from the wave-front sensor. This data represents a line integral through the atmosphere. The problem can be arranged as a large linear algebra problem of the form $Ax = y$. A brute force method is to invert the matrix and perform matrix multiplication however, this is much too inefficient to meet the processing time requirements of the tomography engine.

Below is the equation of the brute force method for performing tomography.

$$Ax = y$$

Where: y –Wavefront Sensor Measurements

A – given by geometry of location of guidestar of interest

x – Volume of atmosphere being observed, measured as the time delay through each voxel.

Currently we have performed forward propagation of the initial voxel values. Next, as real measurements arrive on the wave front sensors, the current estimate of the voxels' values are added and compared to the real measurements. This provides the error between the propagated value and measured value. If the error is less than or equal to a predefined limit value then the process of estimating the voxel values is complete. Otherwise we continue on to the next step of the algorithm.

The error is distributed back through the voxels from the sensor array using back propagation. At the same time, the error is weighted with the strength of the atmospheric layer in which the voxel resides.

The final step of the algorithm is to average the errors from each of its neighbors along with adding this to the current estimated value for the voxel. This step forms the new estimate which is then forward propagated, compared to the measured value and the process continues.

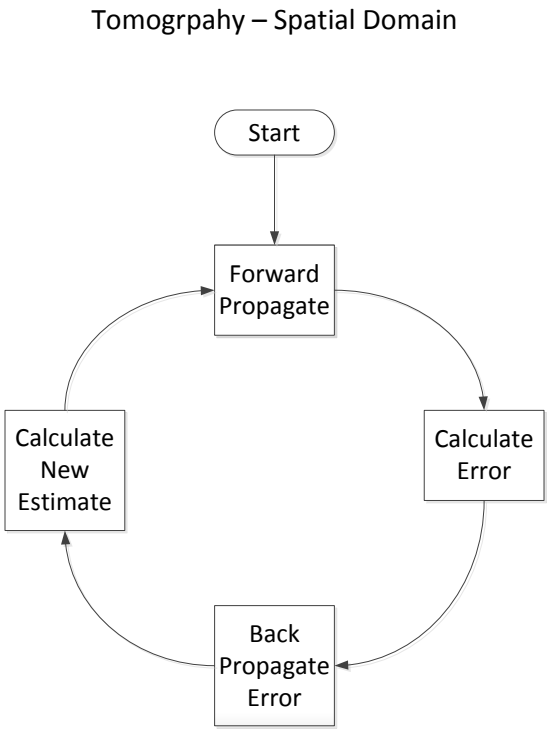


Figure 5 Spatial tomography algorithm

Next the tomography algorithm will be described, incorporating the Frequency domain to illustrate the computational benefits.

2.2 Tomography Algorithm – Fourier Domain

Computing tomography in the spatial domain is quite a time consuming process. Each convolution of voxels with the Kolmogorov spectrum requires large matrix multiplications. Another approach to applying the Kolmogorov spectrum is to perform the filtering in the Fourier domain. The Fourier domain has some nice properties that allow more efficient computation. A drawback to this approach is that

both the forward and inverse Fourier transforms must be performed, however these operations remain less costly than large matrix multiplications.

First perform the Discrete Fourier Transform of the initial voxel values and forward propagate them by adding the values vertically through each frequency bin. Complex multiplication incorporates the spatial shift for a certain path. After forward propagation perform the Inverse Discrete Fourier Transform to return the voxel values to the spatial domain. The next step is to filter out the values outside of the telescope aperture and eliminate any higher order frequencies from persisting through the rest of the algorithm. These filtered values are subtracted from the measured values at each sub aperture and the Discrete Fourier Transform is again taken. The last step is to back propagate and add this value to the previous estimate. Creation of the new estimate takes several steps in the Fourier domain. First, the calculated error must be adjusted for the weighting values for each layer of the atmosphere by multiplying the error by the corresponding weight. Second, this weighted error must then be adjusted by atmospheric statistics given by the Kolmogorov spectrum by multiplying the weighted error by the Kolmogorov filter. Lastly, this value is added to the previous estimate, and the new estimate is generated.

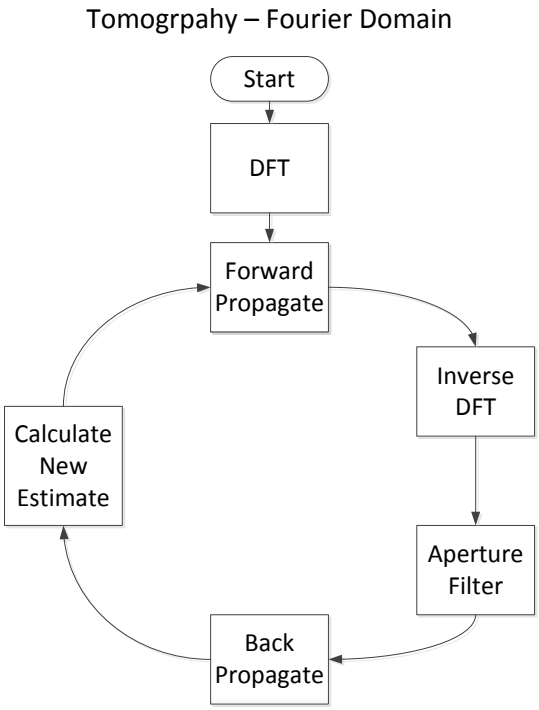


Figure 6 Fourier tomography algorithm

A flow chart of the tomography algorithm is displayed in Figure 7.

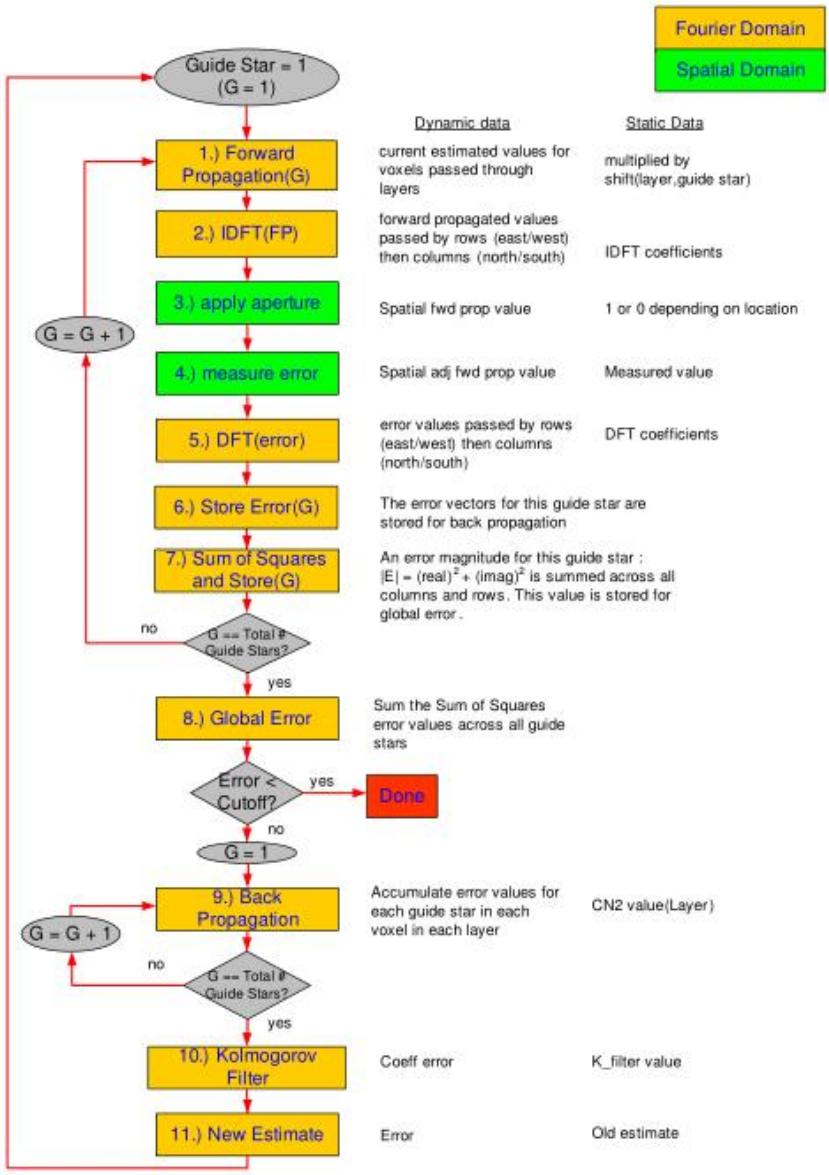


Figure 7 Tomography Algorithm [3]

Chapter 3 Problem Analysis

The majority of the tomography algorithm is performed in the Fourier domain because it is most efficient to do so. If possible the entire tomography algorithm could be performed in the frequency domain, saving valuable time from performing the DFT. Unfortunately, there are several steps of the tomography algorithm that are most easily performed in the spatial domain. These steps include: applying the telescope aperture to filter out spatial values lying outside the field of view, along with measuring the error from forward propagation. A significant amount of time must be spent on the DFT operation as illustrated in the figure below.

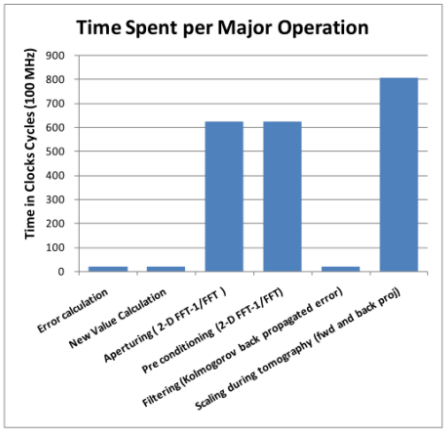


Figure 8 Tomography Algorithm Operation Times [1]

3.1 Choice of Implementation Technology

Performing tomography for a multiple guide star system creates large system requirements that render systems with standard architectures insufficient. "A key challenge of implementing a multi-guidestar adaptive optic system on an

astronomical telescope is the extraordinary amount of computation needed to perform this volumetric tomography hundreds of times per second in order to keep up with the changing atmosphere." [5]

The systolic architecture departs from standard von Neumann architectures and implements many simple arithmetic units. Each unit operates on a different portion of the data simultaneously. FPGA technology provides a reconfigurable platform good for the development phases along with being easily updatable for different systems and algorithms.

3.2 Implementation Technologies

Back of the envelope research analyzed several different implementation technologies before determining digital hardware most closely met tomography engine requirements. "Real-time control for Keck Observatory next-generation adaptive optics" [1] analyzes three different implementation technologies: CPUs, GPUs and FPGAs.

Multi-Core CPUs: Multiple Core CPUs can achieve a degree of parallelism through the use of parallel programming techniques and distributing the workload across the cores.

GPUs: GPUs are quite fast along, with offering an extremely parallel platform for solutions. The most significant drawback is the high latency of memory accesses.

FPGAs: FPGAs offer a reconfigurable, massively parallel solution at less power than CPUs or GPUs. This means they are better at scaling out a parallel algorithm into the hardware. They offer a fixed point representation, which although it generates rounding error still meets the requirements of the system.

3.3 Previous Work

This work is meant as a continuation of thesis work by Matthew Fischler. The Keck Next Generation Adaptive optics proposed system (NGAO) imposes real time constraints with very large throughput requirements. The basis for the processing constraints of the tomography engine is the atmospheric sampling rate, which is the minimum sampling rate in order to accurately sample all frequency content of the atmosphere within the field of view of the telescope. Therefore the tomography engine must be able to complete its processing loop under around 1ms [4] to be ready for the next set of data.

3.4 Hardware

The previous design by Matthew Fischler was synthesized for the Xilinx Virtex 5 FPGA family [4]. Another generation has since been released and the Virtex 6 is the specified hardware for the tomography algorithm. The Virtex 6 has a large I/O capability, which supports inter-FPGAwiring for the future full scale implementation of the algorithm. Although the Virtex 7 is now available from

Xilinx, the Virtex 6 was chosen because of its existing market presence. The Virtex 6 also provides full backward compatibility with the Virtex 5 dsp48e block, which is the core processing unit for the tomography algorithm. Xilinx provides the ML605 development board as an environment for testing custom applications on the Virtex 6 FPGA.

3.4 Hardware Description

1) State-of the art FPGA
2) Large I/O capability – high bandwidth for future interconnections between chips
3) Reliable platform – development boards have been around for several years
4) Ml605 development board – Virtex 6 XC6VLX240T – 1FFG1156
    1. Fully compatibility with the DSP48e from the Virtex 5
5) Industrial Simulation Tools provided by Xilinx – Isim

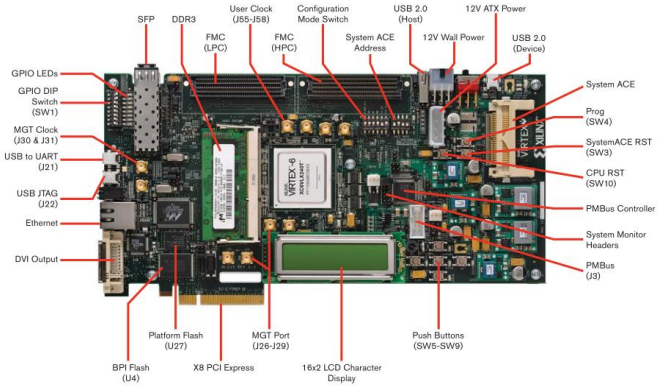


Figure 9 ML605 Development Board [5]

Chapter 4 Implementation Approach

A trivial implementation of the Discrete Fourier transform operates on data of size N in polynomial time (O (N^2)). To perform a standard Discrete Fourier Transform using FPGA technology a trivial approach is using matrix multiplication. A matrix of Fourier coefficients of dimension NxN where N is determined by how many point transform it is to be - which is determined by the number of samples in the data to be transformed. Standard matrix multiplication provides the standard 1-D transform.

The above approach can significantly be improved. Cooley and Tukey discovered many redundancies with the standard transform implementation and were able to reduce the computation time to N*log(N).

4.1 Systolic Architecture

A systolic architecture derives its name from the systolic system of the heart. Each beat of the heart sends blood through the arteries passing oxygen to all organs and limbs. Following the analogy to a computer system, "A systolic system consists of a set of interconnected cells, each capable of performing some simple operation." [7] Each of these cells by themselves is simple and insignificant. However these cells can be connected into trees or grids (Figure 10) which allow much higher throughput of data. Systolic arrays "pulse data instead of blood through the array of

cells." [8] Systolic architectures also allow for the use of simple and uniform cells, elimination of global broadcasting, along with modular expansibility.

Matrix multiplication is a compute bound task because all elements of one matrix are multiplied by all entries in a row or column of another matrix. A systolic array is beneficial for these compute bound tasks.

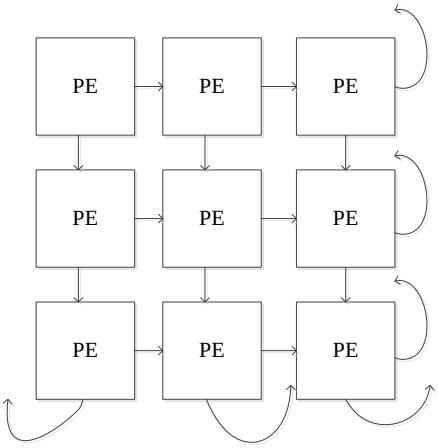


Figure 10 Systolic Array Structure

There are several variations of systolic arrays: special-purpose, general-purpose and programmable. [8] Special purpose arrays are hardwired for a specific application, general purpose are able to be reprogrammed or reconfigured and reconfigurable are implemented in FPGA technology operate in a SIMD or MIMD architecture. Systolic arrays are particularly good at intense parallel problems. Systolic arrays present several implementation issues such as: system integration, and reliability. These systems have large input/output requirements which require consideration. This tomography engine falls under the category of programmable and reconfigurable. FPGA technology allows for quick reconfiguration of the chip and

the SIMD architecture allows for changing out the sequence of control operations are performed.

4.2 Implementation of Fourier Transform

Figure 11 and Figure 12 describe the 1-dimensional forward and inverse Fourier transforms. These equations are decomposed into the systolic array architecture for implementation.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N}$$

Figure 11 Discrete Fourier Transform

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k \cdot e^{i2\pi kn/N}$$

Figure 12 Inverse Discrete Fourier Transform

Figure 13 demonstrates the one dimensional transform performed in the Systolic array architecture.

In hardware, the transform is implemented using a circular shift register of N-measurements. This structure is called a systolic array because each element is performing the same operation at the same time. The processing element contains an accumulator along with a table of values that contains the Fourier coefficient values. Each value in the shift register is multiplied by the Fourier coefficients and added to

the accumulator value. Then the values are shifted and the computation repeats for the next set of values. After n-computations the accumulators hold the transformed results.

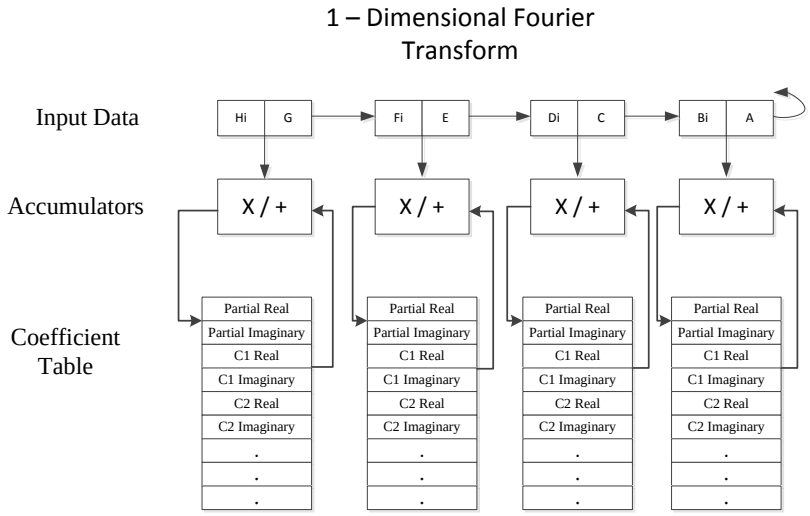


Figure 13 One-D Transform

To compute the two-dimensional transform perform the 1-D transform followed by another 1-D transform in the orthogonal direction.

The systolic array provides the framework for a generic matrix computation engine. Data is piped through the NxN array in one direction until it has been shifted through each element; next the data is shifted in the orthogonal direction through each element. In this way even if the data size is large, the computation still performs in linear time. This approach was designed specifically with the real time control system constraints in mind.

4.3 Processing Element

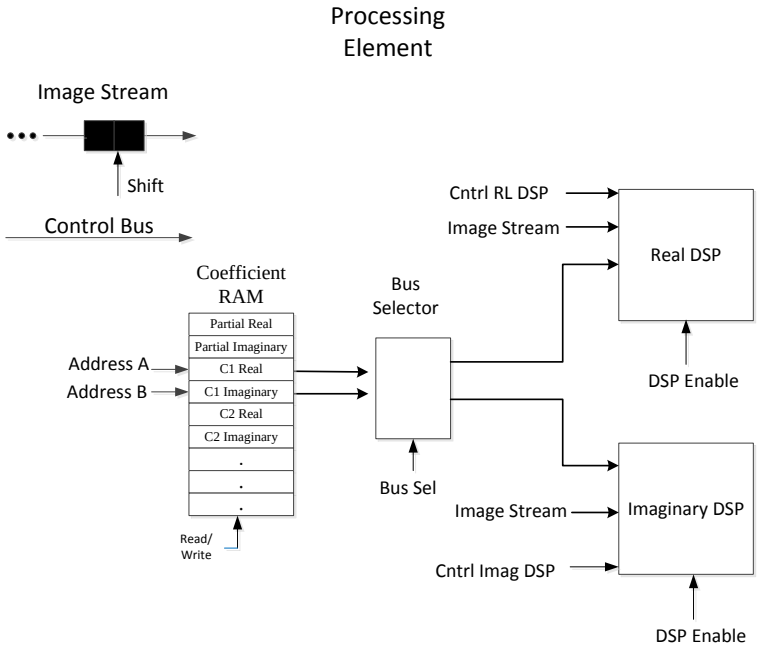


Figure 14 Processing Element

The systolic array is composed of many processing elements, each of which performs the same operation. Because each element is identical, once a single element is developed it is simple to arrange a matrix of them. The architecture of elements for performing the DFT is an NxN matrix. The major components of the processing element module are the following: two DSP48s, a Coefficient RAM, a bus selector, and control bus decode logic.

Each element incorporates two DSP48E logic slices, one to process real data and the other for imaginary data. Surrounding logic controls the movement of data through each element.

Single instruction Multiple Data (SIMD) architecture directs control signals and data in the processing element. A ROM filled with bit code is indexed to drive

the DSP operations along with pass data through the system. The DSP48 logic slice is illustrated in the image below.

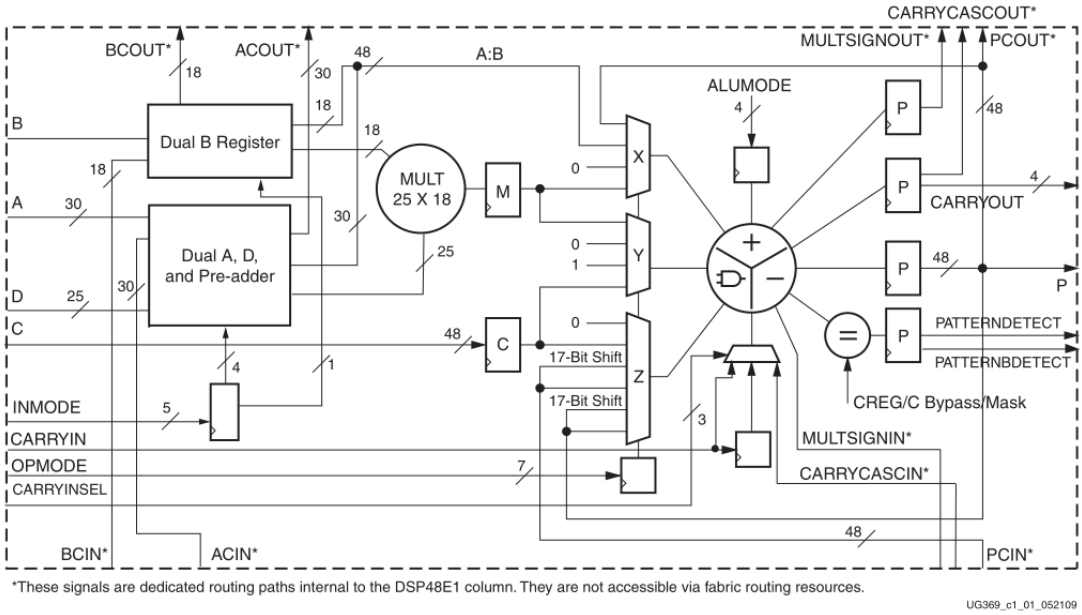


Figure 15 DSP48 Processing Block [6]

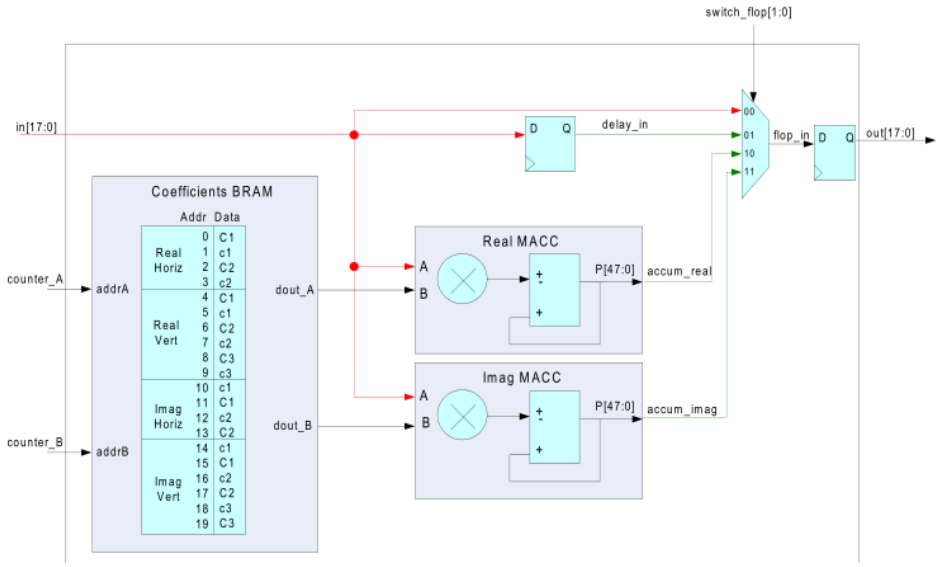


Figure 16 High Level Processing Element

4.4 Complex Multiplication

The majority of operations being performed during both of these transform operations (Forward & Inverse) are multiply accumulates. The Virtex 6 has specially built logic units to optimize these operations. Using the macro instantiation for each unit proved easiest and most relevant for the controlling the necessary signals. Below is the breakdown of a single complex multiplication cycle.

Step By Step Multiplication:

Input: C + Di

Coefficient: A + Bi

$(A * C) + (A * Di) + (Bi * C) - (Bi * Di)$

$AC - BD + ADi + BCi$

Real Result: $\boldsymbol{AC - BD}$

Imaginary Result: $\boldsymbol{ADi + BCi}$

Reduced Multiplication:

See Table 1

This operation is more quickly implemented with two DSP48 blocks. One block operates on real data and the other block processes imaginary data.

Real/Imaginary DSP

The DSP units are built with a multiplier fed by an adder/subtractor input to enable rapid multiply accumulate operations. The Real DSP toggles between adding to the accumulator on the first cycle and subtracting on the second. The Imaginary DSP continually adds to the accumulator.

	Real	Imaginary
First Cycle	$\boldsymbol{A * C}$	$\boldsymbol{C * Bi}$
Second Cycle	$\boldsymbol{A * C - B * D}$	$\boldsymbol{C * Bi + Di * A}$

Table 1 Real & Imaginary Elements of Complex Multiplication Operation

4.5 Single Instruction Multiple Data (SIMD) Control System

The entire system is controlled by a series of instructions, which are 16-bits wide. These instructions are stored in a ROM and indexed by a counter every clock cycle. There seven different instruction types for running the system: load shift, shift, Real Coef Mult Accum, Shift Real Coef Mult Sub, Pre shift DSP RM EN, Write Partials , and Clear accumulator. *Table 2* below lists these instructions along with their resulting operations.

Instruction	**Resulting Operation**
Load Shift	Serially shifts data into shift register
Shift	Circularly shifts data through shift register
Real Coef Mult Accum	Performs multiply accumulate in DSP48
Shift Real Coef Mult Sub	Performs mult accum and subtracts in real dsp

Pre shift DSP RM EN	Shifts input stream for next instruction
Write Partials	Writes Partial Values to RAM
Clear Accumulator	Clears DSP48 accumulator

Table 2 Control ROM Instructions

4.5.1 Load Shift Register

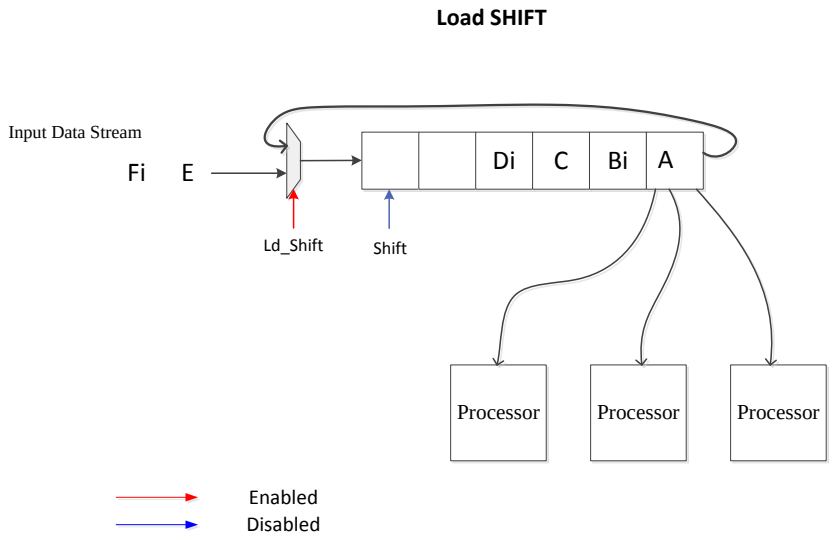


Figure 17 Load Shift Register

Load shift selects external input data and serially shifts it through the shift register until the register is filled. The serial shift takes n-clock cycles and is dependent on the size of array that is synthesized. To determine the number of necessary elements in the shift register multiply the number of Processing Elements by two. A factor of two is required because each number requires two registers, one for the real component and another for the imaginary component.

After the shift register is filled a Data Ready signal is asserted stating that data can begin being shifted through the system. Ld_shift is de-asserted and the next instruction begins.

4.5.2 Shift Data

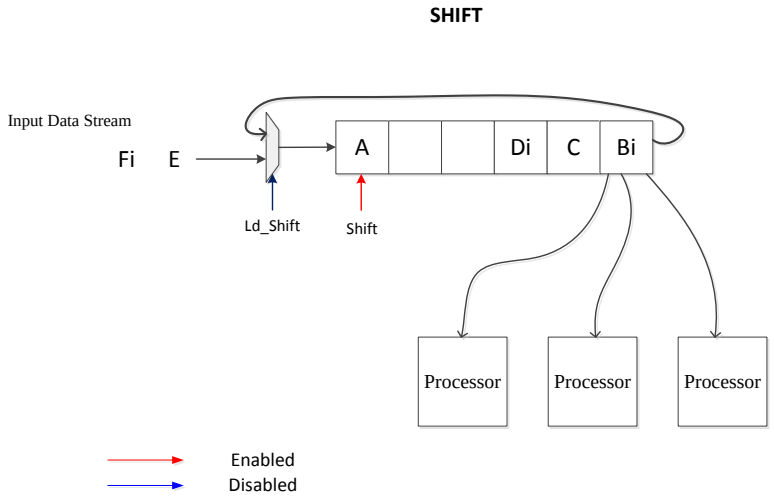


Figure 18 Shift Data

Figure 18 is really a sub-operation of the multiply accumulate instruction. In this manner the instructions are designed in the CISC style so that multiply functions may be performed simultaneously. Shift moves each data element through the shift register and providing each processor with different input data.

After all data in the shift register has completed computation, new data is shifted into the shift register by once again enabling the load shift operation.

4.5.3 Multiply Accumulate

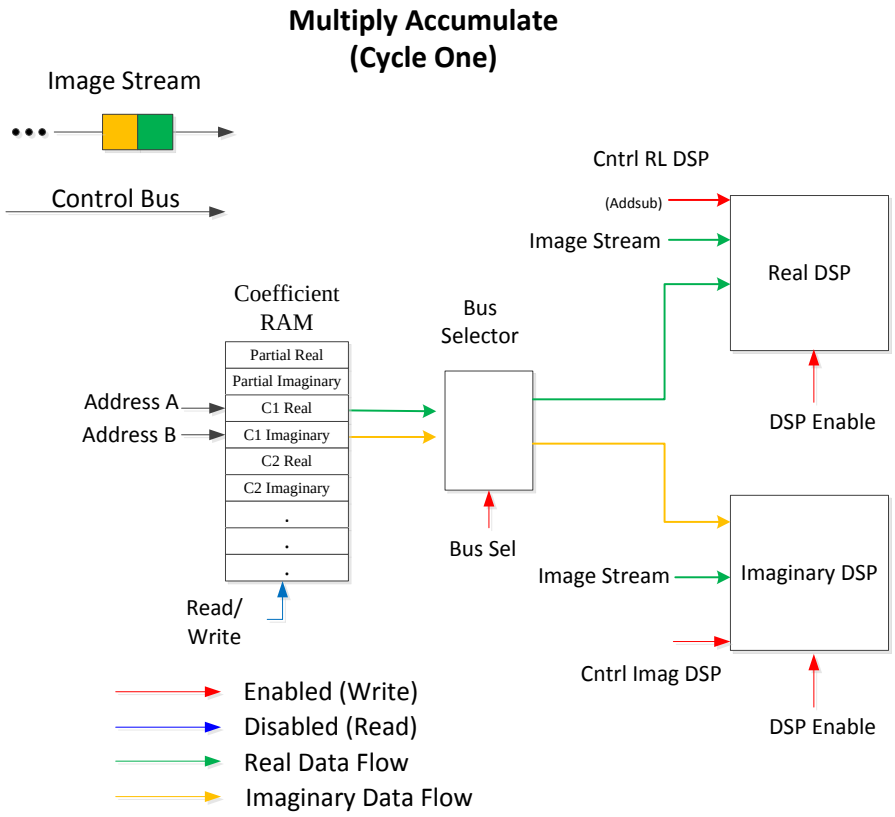


Figure 19 Real Coef Mult Accum

The main operation of the processing element is multiply accumulate. The fundamental computation of the Fourier transform is complex multiplication and is implemented in each element. There are three cycles of multiply accumulate. The first of which selects the real coefficient value from RAM to the real DSP and the imaginary coefficient to the imaginary DSP. The real DSP switches between adding and subtracting the multiplied values with its accumulator.

Using two DSP48 blocks simplifies processing real and imaginary values because the real DSP exclusively performs the two multiplications that result in real data and likewise with the imaginary DSP.

4.5.4 Shift and Multiply Accumulate

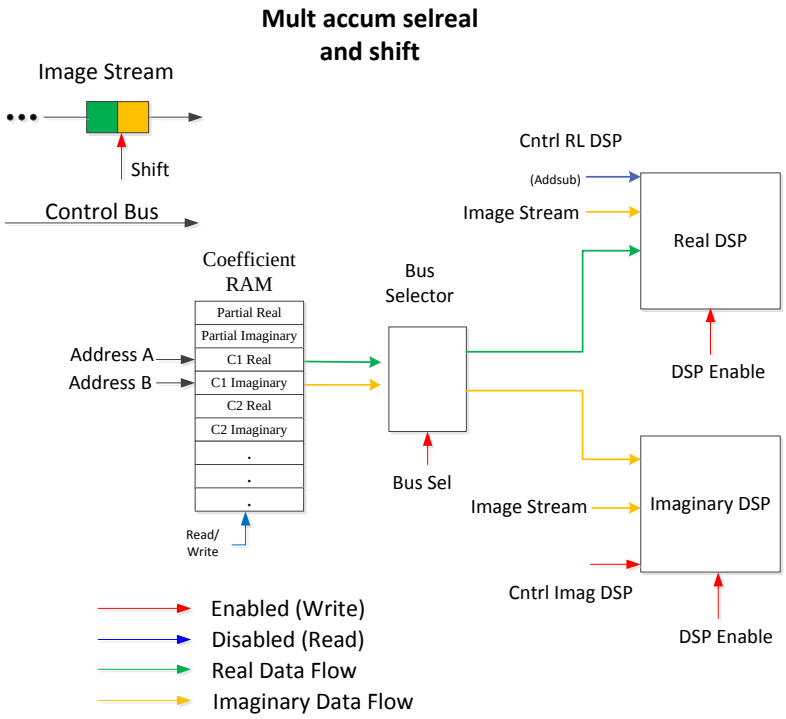


Figure 20 Shift Real Coef Mult Sub

In the second multiply accumulate operation the input data is shifted and the imaginary input element is sent to the real DSP for computation with the Imaginary Coefficient from memory. The converse applies for the Imaginary DSP, the real coefficient is multiplied with the imaginary coefficient. Because the imaginary DSP

is always adding, it does not require toggling on its add/sub line but rather always enables addition.

4.5.5 Multiply Accumulate and Pre Shift

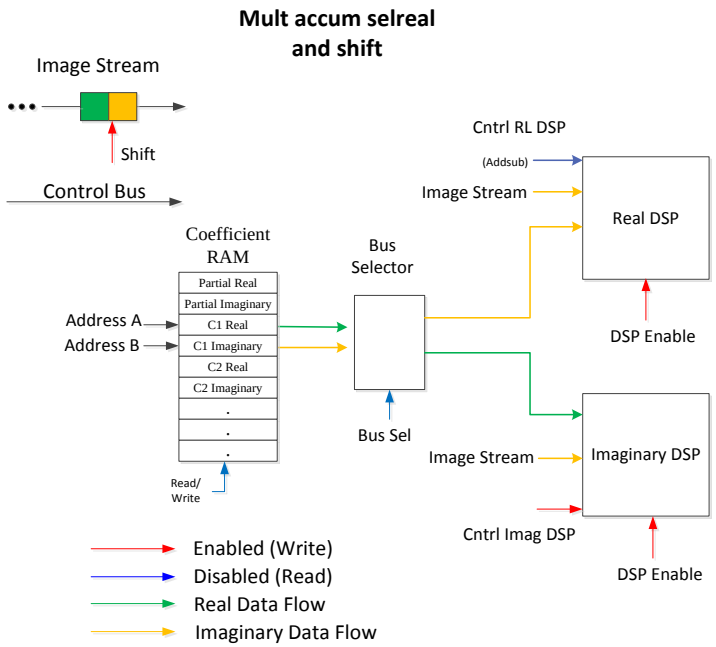


Figure 21 Pre shift DSP RM EN

In the third cycle, the coefficient data is swapped and data is shifted through the shift register to prepare for the next cycle of instructions.

4.5.6 Write Partials to Memory

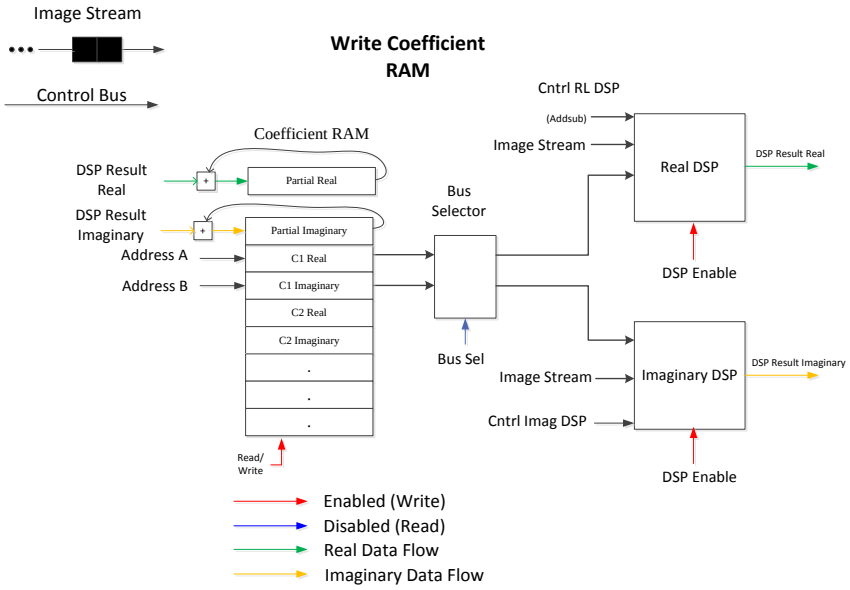


Figure 22 Write Partials

Once a complex coefficient has been multiplied with a complex input value the result must be saved to RAM to retain the partial real and partial imaginary results. This process is illustrated in *Figure 22*. By the time an entire shift through a row or column of processing elements the values stored in the "Partial Real" and "Partial Imaginary" will be the transform results. During Write RAM the flow of data through the processor is paused. Only the Write is enabled on the RAM all other control signals are disabled.

4.5.7 DSP Reset

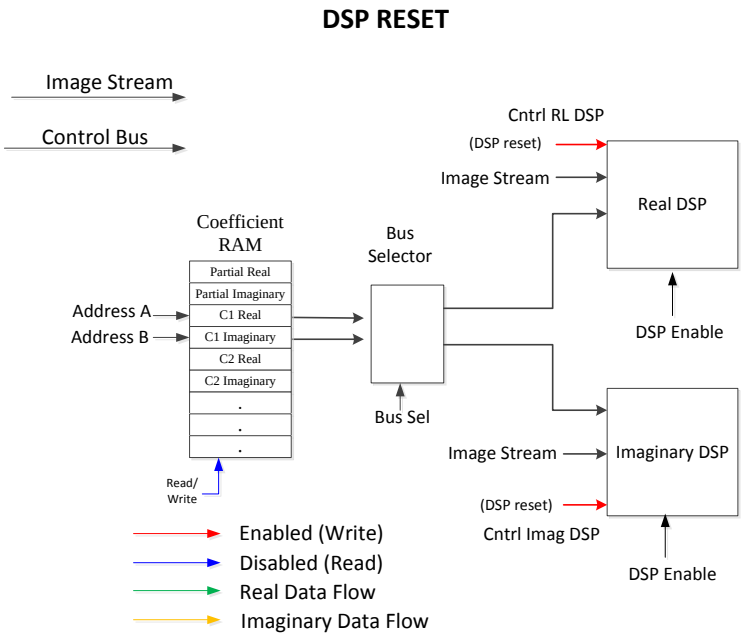


Figure 23 Clear Accumulator

The last instruction in the instruction sequence is DSP Reset. Both DSP blocks are disabled, data is paused from shifting and the read/write of the RAM is disabled. The only signal enabled is the reset line of the DSP blocks. This clears out the accumulator so the next instruction can be performed.

Chapter 5 System Architecture

The processing element is the main computational engine of the system with surrounding data movement logic. The main logic blocks in the system include a Shift register to pass the input data through the system, a control ROM to drive all control signals, the processing element and simple counter to index the control ROM.

Below is a System Level Block diagram.

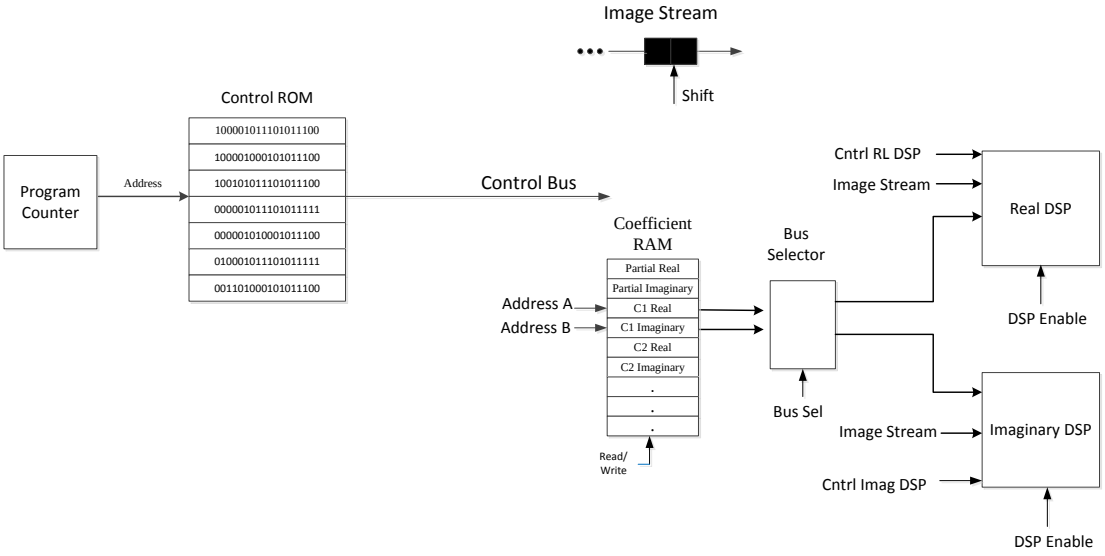


Figure 24 System Block Diagram

5.1 Control ROM

A single PE controller specifies the control signals for each processing element along with the Data shift register. The PE controller consists of a variable width counter driven by the system clock that indexes the Control ROM, illustrated in

the below figure. Depending on the size of system created the Control ROM varies its size, so the counter width varies to account for this.

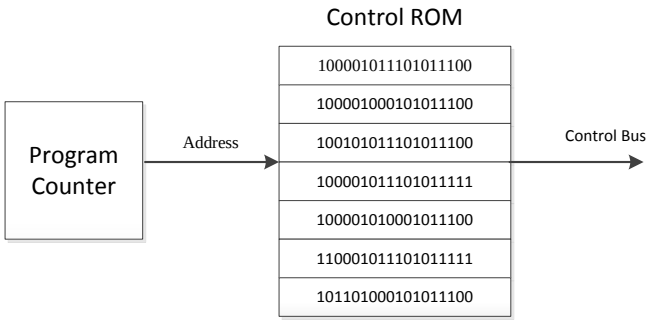


Figure 25 Processing Element Controller

The control ROM is an 18-bit wide structure filled with bitcode that drives all control signals. Below are all of the fields that the Control ROM controls. A set of instructions performing an operation can simply be duplicated in memory for repetition.

Shift	Ld_Shift	Sel_Real	Read_write	DSP_reset	DSP_addsub	DSP_EN	RAM_EN	RAM_ADDR B	RAM_ADDR A

Multaccum_add	0010011100110010
Multaccum_sub	0010011100110010
multaccum	0000001100110010
writeram	0001000100010000
dspreset	0000100000000000

Figure 26 Control ROM Bitcode

The Control ROM is the brain of the entire algorithm.

5.2 Coefficient RAM

The Coefficient RAM stores each set of DFT and inverse DFT coefficients along with holding the real and imaginary partial results of the transforms. Because two DSP48 units are in each processing unit, a dual port RAM is necessary feed coefficients to each DSP. A figure of the Coefficient RAM is illustrated below.

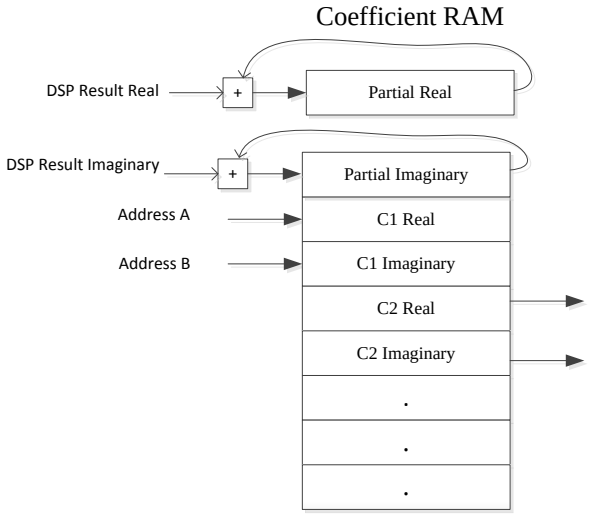


Figure 27 Coefficient RAM

Two memory locations are used to accumulate the current partial real and imaginary results. These addresses are written to after each multiply accumulate operation finishes in each DSP48 block.

5.3 Bus Selector

The bus selector is a multiplexor that selects either a real or imaginary coefficient depending on what cycle the system is executing. Each multiply accumulate instruction toggles between selecting real and imaginary coefficient data.

5.4 Shift Register

The shift register is the mechanism of passing data through the system at the desired time. Data is serially loaded in for the first n clock cycles and then shifted based on the shift control line. When data is finished being processed the shift signal is re-enabled and data flows through the system to the next processing element.

Shift Register

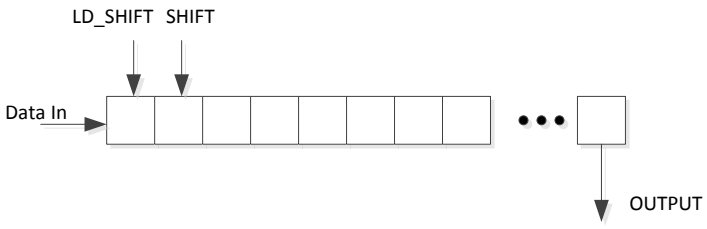


Figure 28 Shift Register

5.5 Interconnection

These processing elements are arranged in a topology such that a single row in the matrix performs a 1-D transform. Data is first shifted “left” to “right” in the array followed by “up” to “down”. To perform the 2-D transform, the data is shifted in orthogonal directions.

Chapter 6 Verilog Architecture

The entire project is written in synthesizable Verilog 2000.

top_level.v - Highest level design file. Contains shift register, processing element controller along with all processing elements.

pe_controller.v - Single Control unit with program counter and Control ROM with all instructions

program_counter.v - Indexes Control ROM for each instruction

control_rom.v – Holds binary instructions that drive all control lines

processing_element.v – Major logic unit of system.

coeff_ram.v – Dual port ram that holds forward and inverse coefficients along with storing partial results

bus_arbiter.v- Controls movement of input data and coefficient data to processing elements

macc_unit.v – Instantiated DSP48 unit

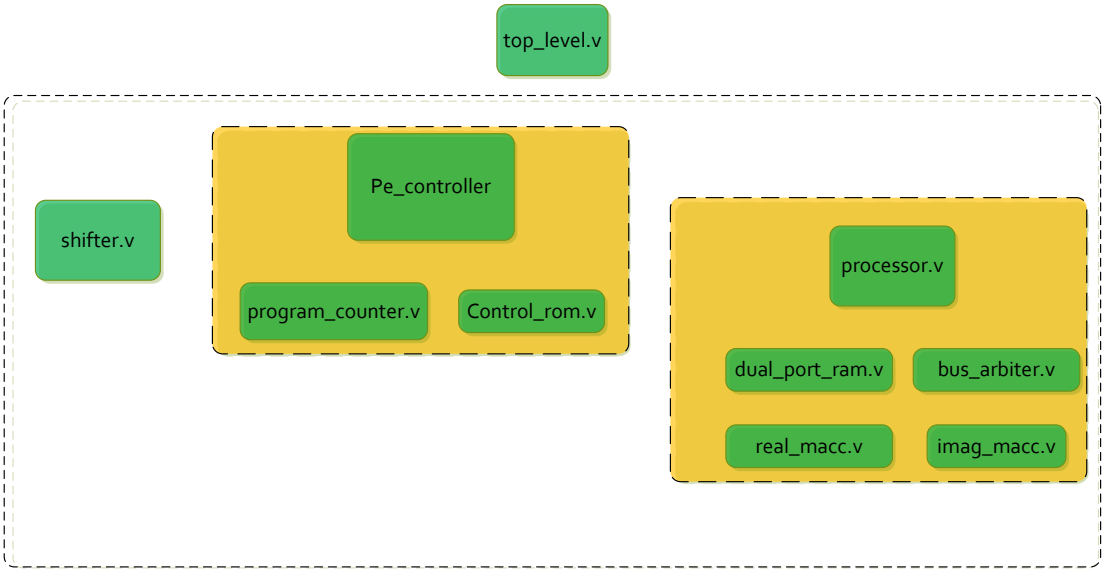


Figure 29 Verilog Architecture

Chapter 7 Verification

The algorithm has been verified in software using simulation tool ISim 14.5 © Xilinx, to simulate the waveforms. Timing requirements are not a significant concern for this stage of development and the entire system can be tuned depending on what frequency the operation fails computation. Test-benches drive the input ramp data through the processing blocks allowing the outputs to be monitored easily using Isim.

An 8 – element array was generated to illustrate a one dimensional transform. Microsoft Excel was used to generate the Fourier Transform Coefficients and the reference results.

7.1 Generation of Reference Data

Input data generated was an 8 sample complex valued ramp function with zeros for the imaginary components. Fourier coefficients were generated in Microsoft excel using the trigonometric form of the fundamental equation of the Fourier series. These values were then normalized to 18-bits and converted to Hexadecimal representation for storage in each processing element's RAM. Below are the equations for generating the real and imaginary components of the coefficients.

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos\frac{2\pi tn}{T}dt$$

$$b_n = \frac{-2}{T}\int_{-T/2}^{T/2} x(t)\sin\frac{2\pi tn}{T}dt$$

The Fourier Transform results were computed using the Excel Data Analysis package. This package provides a library of functions for computing different mathematical formulas.

7.2 Implementation Results

A 1x8 processing element system was synthesized to prove functionality of the eight point transform. Simulation results match with Microsoft Excel generated reference values. Each intermediary partial result was verified along with each of the results from the 8-point transform. See Figure 13 for a diagram how data is shifted through the processing elements and each result is stored. Data from the shift register is fed to each processing element, each of which multiplies the input data by the appropriate coefficient and then accumulates the data into memory. By the time the entire input is shifted through the system, the data has been multiplied by each coefficient and the final accumulated results are the transform of the original input.

The latency of each computation cycle is 5 clock cycles, and given an 8-point transform this operation is performed in 40 clock cycles. Initializing the shift register with data requires 16 clock cycles because a complex 8-element transform is being performed.

See Appendix A for synthesis results generated using Xilinx's ISE tool. The maximum frequency to achieve timing closure is 207 Mhz. For the 1x8 array of

Processing Elements synthesized 9 RAMs are used, 8 for the 8 processing elements and 1 for the system shift register. 16 adders are used 2 for each processing element. 1 Counter in the design is used to address the Program Counter. 32 multiplexers are used, 4 for each processing element.

Based on the slice Logic Utilization report very few, 0% (2,990/301,440) slice registers are used, and 3% (5,066/150,720) Slice LUTs are used. As for the DSP units, 2% (16/768) are used. Therefore the array can be scaled to a matrix on the order of 18x18 Processing Elements.

The first resources to run out with the current implementation structure are LUT-FF pairs, which after synthesizing a 1x8 array are at 57% utilized. With further code optimization this can be brought down to allow for further scaling of the array.

Chapter 8 Conclusion

The Systolic Array implementation of the Fourier Transform performs the forward and inverse Fourier transforms in linear time. This custom architecture was designed to meet the overall system requirements of the Real Time Control System. The Fourier transform is the most computationally intensive operation in the entire tomography algorithm. Now that it has been implemented it is possible to use the existing architectural foundation to implement the remaining steps of the tomography algorithm. These include forward propagation, error calculation, back propagation and calculation of a new atmospheric estimate.

8.1 Future Additions

The next step is validating the 2-Dimensional transform which is directly achievable by generating more coefficients along with changing the sequencing of the control ROM. Also, performing the inverse transform is a small step because all that is required is the generation of Inverse Fourier Coefficients.

In the current implementation the systolic array suffices with a single Control ROM that generates the control bit streams for each processing element in the array. When the array scale becomes very large, delay from the control ROM to the PEs could become significant and then we will consider using optimal placement of multiple control ROMs throughout the array to reach all PE's within timing

constraints. Another consideration will be optimizing placement of the Coefficient RAMs closer to the DSP48 units to minimize wiring distances.

Appendix A

Synthesis Results

Device utilization summary:

Selected Device : 6vlx240tff1156-1

Slice Logic Utilization:
Number of Slice Registers: 2990 out of 301440 0%
Number of Slice LUTs: 5066 out of 150720 3%
Number used as Logic: 5048 out of 150720 3%
Number used as Memory: 18 out of 58400 0%
Number used as SRL: 18

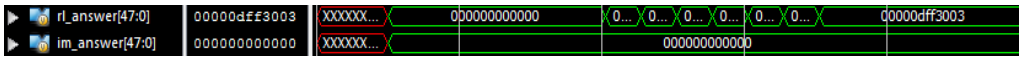
Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 5113
Number with an unused Flip Flop: 2123 out of 5113 41%
Number with an unused LUT: 47 out of 5113 0%
Number of fully used LUT-FF pairs: 2943 out of 5113 57%
Number of unique control sets: 23

IO Utilization:
Number of IOs: 68
Number of bonded IOBs: 20 out of 600 3%

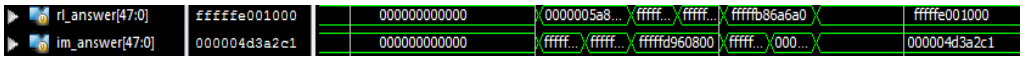
Specific Feature Utilization:
Number of BUFG/BUFGCTRL/BUFHCEs: 1 out of 176 0%
Number of DSP48E1s: 16 out of 768 2%

Appendix B Simulation Results of 8-point Transform

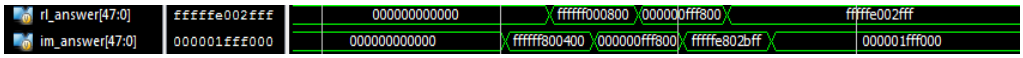
PE 1



PE 2



PE 3

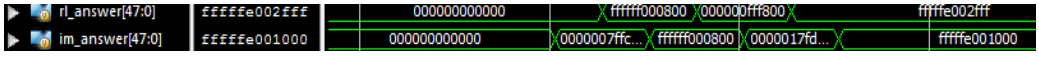


PE 4

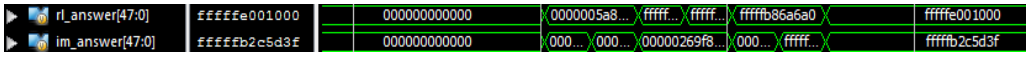
PE 5

PE 6

PE 7



PE 8



Appendix C Microsoft Excel Generated Results

	Real	Imaginary
PE 1	**DFF3003**	**0**
PE 2	**FFFE001000**	**4D3A2C1**
PE 3	**FFFE002FFF**	**1FFF000**
PE 4	**FFFE001000**	**D402BF**
PE 5	**FFFE002FFF**	**0**
PE 6	**FFFE001000**	**FFFF2BFD41**
PE 7	**FFFE002FFF**	**FFFE001000**
PE 8	**FFFE001000**	**FFFB2C5D3F**

Bibliography

[1] Marc Reining, Donald Gavel, Ehsan Ardestani and Jose Renau. “Real-time control for Keck Observatory next-generation adaptive optics.” Proposal No. AST-0335461

[2] Max, Claire. Astronomy 289. Class Lecture, Topic: “Adaptive Optics and its Applications.” U.C. Santa Cruz, January 8, 2013

[3] Fischler, Matthew. “The LAO Systolic Array Tomography Engine.” M.S. thesis, University of California Santa Cruz, U.S., 2007.

[4] Donald Gavel, Marc Reinig and Carlos Cabrera. “Fast Hardware Implementation of Tomography for Multi-guidestar Adaptive Optics.” SPIE Optics and Photonics, v5903, 2005.

[5] Xilinx, Inc. Getting Started with the Virtex – 6 FPGA ML605 Evaluation Kit, 2011

[6] Virtex 6 FPGA DSP48E1 Slice User Guide, (2011)

[7] Hardy, John W. Adaptive Optics for Astronomical Telescopes. New York: Oxford University Press, 1998, pp. 40,61,62

[8] Kung, H.T. “Why Systolic Architectures?” IEEE, 1982, pp. 37-46

[9] Kurtis T. Johnson, A.R. Hurson and Behrooz Shirazi. “General-Purpose Systolic Arrays.” IEEE, 1993, pp. 20-31

[10] “Adaptive Optics Wavefront Control and Performance” Don Gavel – Lawrence Livermore National Labs