Report Number: WUCSE-2012-84

2012

Implementation of Real-Time Calibration of Polarization Imaging Sensors

Authors: Collin Foster

Recent breakthroughs in nanofabrication techniques have led to development of sophisticated Division-of-Focal-Plane (DoFP) polarization imaging sensors. One such technique allows the fabrication of nanowire filters fabricated directly on the imaging sensor itself. This technique can be used to fabricate robust DoFP polarization imaging sensors. However, the polarization information captured by the imagers can be degraded due to imperfections in the fabrication of the nanowire filters on the imaging sensor. Polarization information can also be degraded from other sources including crosstalk between pixels. To compensate for these undesired effects, a calibration routine can be applied to each pixel after image capture. In this project, we implement a calibration routine as a step in the pipeline processing of captured polarization images. The polarization image processing and calibration are implemented on an FPGA development board to achieve a real-time response to image captures.
... **Read complete abstract on page 2.**

Implementation of Real-Time Calibration of Polarization Imaging Sensors

Complete Abstract:

Recent breakthroughs in nanofabrication techniques have led to development of sophisticated Division-of-Focal-Plane (DoFP) polarization imaging sensors. One such technique allows the fabrication of nanowire filters fabricated directly on the imaging sensor itself. This technique can be used to fabricate robust DoFP polarization imaging sensors. However, the polarization information captured by the imagers can be degraded due to imperfections in the fabrication of the nanowire filters on the imaging sensor. Polarization information can also be degraded from other sources including crosstalk between pixels. To compensate for these undesired effects, a calibration routine can be applied to each pixel after image capture. In this project, we implement a calibration routine as a step in the pipeline processing of captured polarization images. The polarization image processing and calibration are implemented on an FPGA development board to achieve a real-time response to image captures.

2012-84

Implementation of Real-Time Calibration of Polarization Imaging Sensors

Authors: Collin Foster

Abstract: Recent breakthroughs in nanofabrication techniques have led to development of sophisticated Division-of-Focal-Plane (DoFP) polarization imaging sensors. One such technique allows the fabrication of nanowire filters fabricated directly on the imaging sensor itself. This technique can be used to fabricate robust DoFP polarization imaging sensors. However, the polarization information captured by the imagers can be degraded due to imperfections in the fabrication of the nanowire filters on the imaging sensor. Polarization information can also be degraded from other sources including crosstalk between pixels. To compensate for these undesired effects, a calibration routine can be applied to each pixel after image capture. In this project, we implement a calibration routine as a step in the pipeline processing of captured polarization images. The polarization image processing and calibration are implemented on an FPGA development board to achieve a real-time response to image captures.

Type of Report: MS Project Report

Washington University in St. Louis
School of Engineering and Applied Science
Department of Computer Science and Engineering

Project Examination Committee:
Professor Viktor Gruev, Chair
Professor Roger Chamberlain
Professor William D. Richard

Implementation of Real-Time Calibration of Polarization Imaging Sensors

by

Collin Foster

A project presented to the School of Engineering
of Washington University in partial fulfillment of the
requirements for the degree of

Master of Science

December 2012
Saint Louis, Missouri

ABSTRACT OF THE PROJECT

Implementation of Real-Time Calibration

of Polarization Imaging Sensors

by

Collin Foster

Master of Science in Computer Engineering

Washington University in St. Louis, 2012

Research Advisor: Professor Viktor Gruev

Recent breakthroughs in nanofabrication techniques have led to development of sophisticated Division-of-Focal-Plane (DoFP) polarization imaging sensors. One such technique allows the fabrication of nanowire filters fabricated directly on the imaging sensor itself. This technique can be used to fabricate robust DoFP polarization imaging sensors. However, the polarization information captured by the imagers can be degraded due to imperfections in the fabrication of the nanowire filters on the imaging sensor. Polarization information can also be degraded from other sources including crosstalk between pixels. To compensate for these undesired effects, a calibration routine can be applied to each pixel after image capture. In this project, we implement a calibration routine as a step in the pipeline processing of captured polarization images. The polarization image processing and calibration are implemented on an FPGA development board to achieve a real-time response to image captures.

Acknowledgments

I would like to thank my project advisor, Professor Viktor Gruev, for his guidance and patience on my project. He has supported me every step of the way from forming my project goals to implementation.

I would also like to thank Sam Powell and the rest of Professor Gruev's research team in helping me with debugging the project.

I would also like to thank my project committee on providing me useful and insightful feedback and suggestions on my work.

Collin Foster

Washington University in St. Louis

December 2012

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Advancements in CMOS and CCD imaging sensor design have provided the electronic consumer market with cheap and reliable cameras for millions of electronic devices. Typically, these imaging sensors are designed to capture the properties of light the human eye receptors can process: wavelength and intensity. However, imaging sensors designed to capture the third property of light, polarization, have begun to gain popularity. These polarization imaging sensors are designed not only capture the polarization property of light but to allow processing of the image captured such that the polarization information can be presented in a visible format.

One type of polarization imaging sensor capable of capturing the necessary polarization information is the Division-of-Focal-Plane (DoFP) polarization imager. The DoFP polarization imaging sensor captures the polarization information in a single image frame but does so at the cost of decreased spatial resolution [1]. Polarization information is captured on a DoFP polarization imaging sensor by placing filters directly on the imager as shown in Figure 1.1. The filters are applied on the imager during fabrication by fabricating nanowire filters directly on the imager [2]. In Figure

1.1, the pattern consists of a 2x2 section of pixels where two pixels do not have filters applied, while the other two pixels have filters applied at different polarization angles. This pattern forms a set of pixels called the Super Pixel. The Super Pixels contain all the required polarization information to obtain a useful polarization image. This method is similar to the Bayer Filter pattern used on color imaging sensors by applying color filter patterns to obtain a full color image [3].

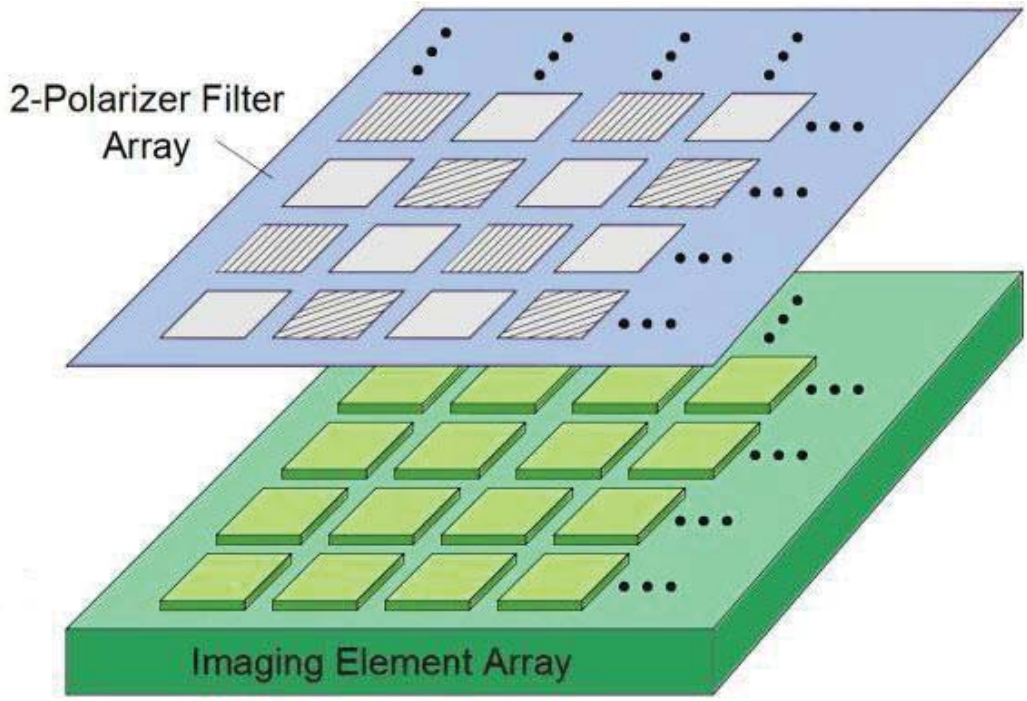

Figure 1.1: Diagram of polarization filter array placed over standard imaging array [2].

The filter patterns applied to obtain a linearly polarized image capture are designed to capture the intensity of the light filtered at various polarization filter angles. Capturing the light at these different angles provides the information necessary to determine the first three Stokes Parameters (S_0, S_1, S_2). The first three Stokes Parameters can be used

to determine the Degree of Polarization (DoP) and the Angle of Polarization (AoP). The Degree of Polarization is a measure describing how polarized the light is, while the Angle of Polarization is a measure describing the major axis of propagation [1].

With DoFP polarization imagers, post processing of the image must be performed to obtain the three Stokes Parameters, DoP, and AoP for the entire image. The image processing steps utilized for the DoFP polarization imaging sensors is displayed in Figure 1.2.

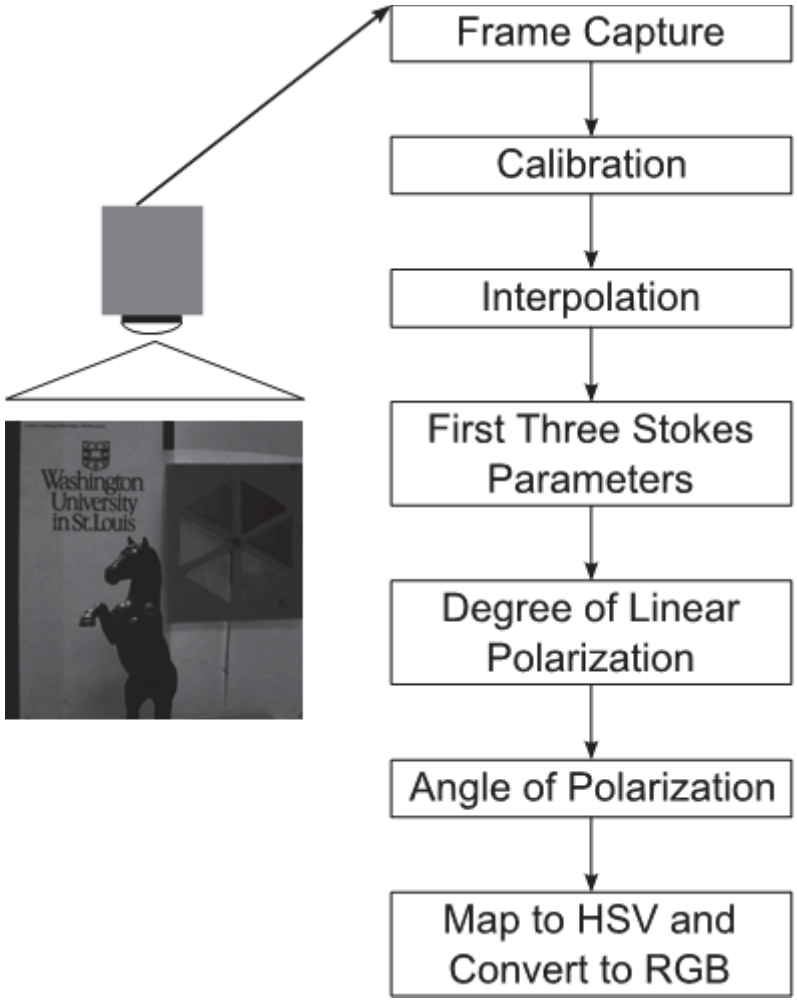

Figure 1.2: Diagram depicting the polarization image processing steps required to obtain Stokes parameters, DoP, AoP, and visual representations of these polarization characteristics.

To insure the polarization imager's response is optimal, a calibration routine is applied before other post processing operations. Calibration is necessary to correct imperfections that can degrade the imager's performance. Some sources of degradation include imperfections in the manufacture of the nanowire polarization filters, possible errors in alignment when the filters are deposited on the sensor, dark currents, and

possible noise in readout circuitry of the imaging sensor [1]. Applying a calibration routine before other post processing routines compensates for these imperfections and achieves optimal performance from the imager. An example of a calibration implementation is presented in [1]. In this implementation, calibration is performed to keep the filter responses across the local neighborhood of a super pixel consistent. This is achieved using a gain and offset value for each pixel resulting in Equation 1.1.

$$I_{cal}(x,y) = I_{orig}(x,y) * gain(x,y) + offset(x,y) \qquad (1.1)$$

As shown in Equation 1.1, the raw pixel value is multiplied by the gain value, and the offset is added to arrive at the calibrated pixel value. Each value in Equation 1.1 is a function of its x and y position in the image. This is so because calibration values across the image array are not uniform and are also dependent on the location in the image array. The derivation of the gain and offset values are described more in [1].

As seen in Figure 1.2, interpolation techniques are applied after the calibration image processing step. While the Super Pixel sets contain all the information needed, spatial resolution is lost. This is due to the fact that each individual pixel only contains information for the one polarization filter applied directly on it. Therefore, interpolation algorithms are needed to obtain all necessary polarization information for each individual pixel. The interpolation algorithms use intensities of neighboring pixels to recover all required intensities at different polarization angles for every pixel. For example, bilinear interpolation averages neighboring pixel values with the desired

filtered angle of polarization to recover that filtered angle's pixel intensity value for a given pixel. Figure 1.3 displays a 4x4 section of filtered pixels for the DoFP polarization imager.

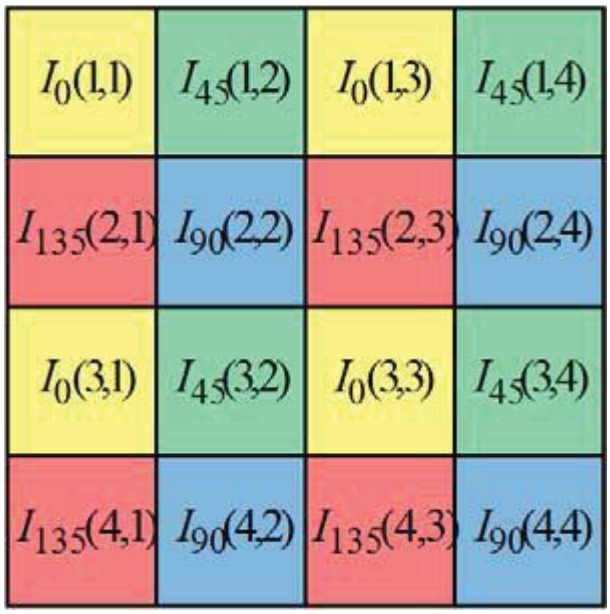

Figure 1.3: Diagram depicting a 4x4 section of pixels with polarization angles filtered at 0, 45, 90, and 135 degrees on a DoFP polarization imager [7].

Figure 1.3 shows each pixel in the DoFP is filtered with one angle of polarization. I_0 implies the pixels is filtered at a polarization angle of 0, I_{45} is a pixel filtered at a polarization angle of 45, etc. The pixel at location (2,2) is filtered at an angle of 90 degrees. To obtain the other three angles of polarization, bilinear interpolation can be applied as shown in Equations 1.2, 1.3, and 1.4.

$$I_0(2,2) = \frac{1}{4}\left(I_0(1,1) + I_0(1,3) + I_0(3,1) + I_0(3,3)\right) \quad (1.2)$$

$$I_{45}(2,2) = \frac{1}{2}\left(I_{45}(1,2) + I_{45}(3,2)\right) \quad (1.3)$$

$$I_{135}(2,2) = \frac{1}{2}\left(I_{135}(2,1) + I_{135}(2,3)\right) \quad (1.4)$$

Bilinear interpolation is demonstrated in Equations 1.2, 1.3, and 1.4 for the pixel at location (2,2) from Figure 1.3 [7]. As seen in these equations, the intensity pixel values for the three missing interpolation angles of 0, 45, and 135 degrees are obtained by averaging the neighboring pixel values containing filters for these angles. Through averaging the neighboring values, all pixel values for each polarization angle can be recovered.

After applying interpolation algorithms to obtain all intensity values at each polarization angle for every pixel, the first three Stokes parameters can be calculated for every pixel as well. Equations 1.5, 1.6, and 1.7 display the calculation for S_0, S_1, and S_2 respectively.

$$S_0 = I_0 + I_{90} \quad (1.5)$$

$$S_1 = I_0 - I_{90} \quad (1.6)$$

$$S_2 = I_{45} - I_{135} \quad (1.7)$$

As is seen in Equations 1.5, 1.6, and 1.7, the intensity values of a pixel at each polarization angle is used to obtain the first three Stokes parameters from that pixel.

After obtaining the Stokes parameters, the DoP and AoP can for a pixel can be calculated as follows in Equations 1.8 and 1.9.

$$DoP = \frac{\sqrt{S_1^2 + S_2^2}}{S_0} \tag{1.8}$$

$$AoP = \frac{1}{2}\arctan\left(\frac{S_2}{S_1}\right) \tag{1.9}$$

After obtaining the DoP and AoP for each pixel in an image, visual representations of these two parameters can be created for observation.

1.2 Project Goals and Contribution

In this project, our goal is to implement a portable real-time response to images captured on a DoFP polarization imaging sensor. In terms of our project, real-time is defined as performing all necessary post image capture processing at the same rate of image captures. By achieving a real-time response, we can immediately display a visual representation of a polarization image. For the purposes of this project, portable is defined as implementing the polarization image processing with minimal to no use of a traditional personal computer or other large digital signal processor (DSP) devices. To achieve portability, the image processing was chosen to be implemented on an FPGA integration board. An FPGA implementation has many advantages over a PC or DSP including portability, lower power consumption, and dedicated hardware configuration

for image processing. An application specific integrated circuit (ASIC) design is another possible solution to implement dedicated hardware for polarization image processing. However, an FPGA implementation also has some advantages over an ASIC design. One obvious advantage is the ability to quickly change the hardware configuration by simply reprogramming the FPGA with a new configuration file. An ASIC cannot be altered once the design is implemented and the chip is fabricated. With this advantage in mind, an FPGA design provides faster turnaround time in changes to the polarization image processing techniques applied.

The contribution of this project was to expand the capability of the original project, specifically to implement the usage of DDR2 SDRAM present on the chosen FPGA integration module and utilize the DDR2 SDRAM for the image processing routines, specifically the calibration routine. The original project was developed by PhD candidate Sam Powell in our research lab. The original project implements a polarization image processing pipeline on an FPGA. The pipeline processes raw image data and obtains the DoP and AoP as the final steps of processing. These are used to generate a visual representation of the polarization information as the image output of the pipeline. The original project includes a calibration routine that uses gain data to calibrate incoming pixels where each pixel is calibrated with its associated gain data. With addition of the DDR2 SDRAM to the project, the calibration routine is changed to use gain and offset data to calibrate incoming pixels with a gain and offset value associated with each pixel. The addition of the DDR2 SDRAM also allows enhancement in data resolution of the gain data. Originally, each gain data point had 8-

bit resolution with 4 fractional bits. The gain data points are enhanced to have 16-bit resolution with 8 fractional bits. The offset data also has 16-bit resolution. By increasing the resolution of the gain and introducing offset, the calibration routine is able to more accurately calibrate incoming pixels which, in turn, allows more accurate polarization image processing in later steps of the image processing pipeline.

Chapter 2

FPGA Implementation

As mentioned before, the full polarization image processing is implemented on an FPGA integration module. The chosen module will be discussed in this chapter. The chapter will then present a high level overview of the FPGA implementation describing each step in the polarization image processing pipeline. In the later chapters the DDR2 SDRAM implementation and updated calibration routine will be discussed in detail.

2.1 FPGA Integration Module

The FPGA board chosen to implement the real-time image processing is the Opal Kelly XEM5010 FPGA integration module. An image of the XEM5010 is displayed in Figure 2.1, while a block diagram depicting the components of the board is displayed in Figure 2.2.

Figure 2.1: Picture of the Opal Kelly XEM5010 FPGA integration module [4].

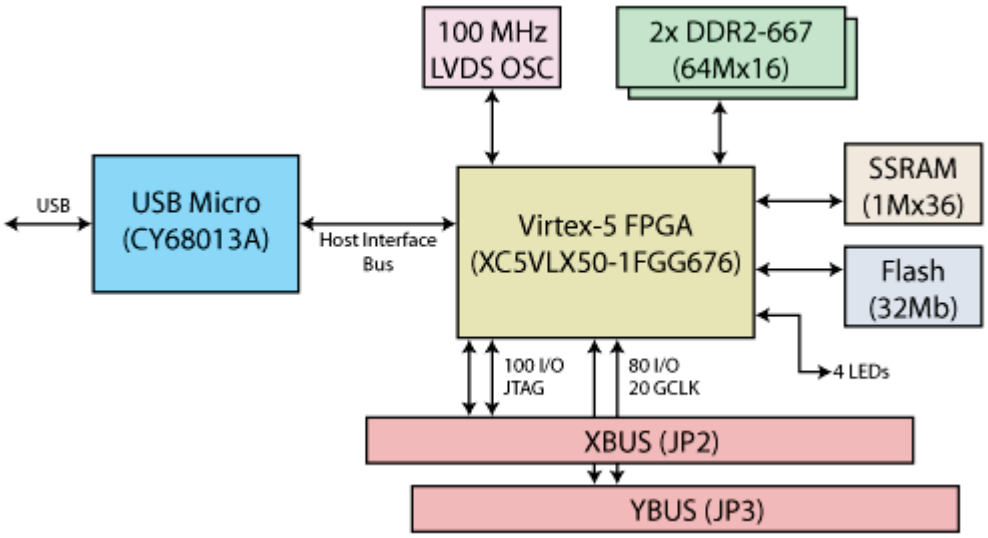

Figure 2.2: Block diagram of components present on XEM5010 FPGA integration module [4].

As can be seen in Figure 2.2, the XEM5010 includes many components for use in our polarization image processing application:

- Xilinx Virtex-5 FPGA
- Two 128-MByte DDR2 SDRAMs
- 32-MBit Serial Flash
- 36-MBit SSRAM (1Mx36)
- Very low jitter 100 MHz LVDS clock oscillator
- USB 2.0 for PC interface

The original polarization image processing project utilizes most of the components on the board except for the 100 MHz LVDS clock oscillator and the two DDR2 SDRAMs. The XEM5010 also contains expansion buses for IO. These are utilized to input pixels from the camera and to output the processed images after pipeline processing. The USB 2.0 interface provides communication to a PC. Opal Kelly provides an API for developing C++ programs to easily communicate with the FPGA. For this project, software is written to control the FPGA and to feed the calibration data to the DDR2 SDRAM for use in the calibration routine. The C++ software is detailed in Appendix B.

2.2 Top-Level Overview

As described before, the polarization image processing is performed in real-time with the implementation of an image processing pipeline on the Virtex-5 FPGA. The pipeline design allows each image processing routine to be fully utilized at all times,

since new data is fed into each routine as it finishes processing the data previously input. The implementation is developed in Verilog HDL using the Xilinx ISE Design Suite for synthesis and programming file generation. The Xilinx ISE Design Suite also provides synthesized Xilinx Core modules that allow the designer to quickly implement common controls on an FPGA. The designer need only interface with the Core without designing the control itself. Various Xilinx Cores are utilized in the design for this project. A block diagram representation of the original FPGA implementation is displayed in Figure 2.3.

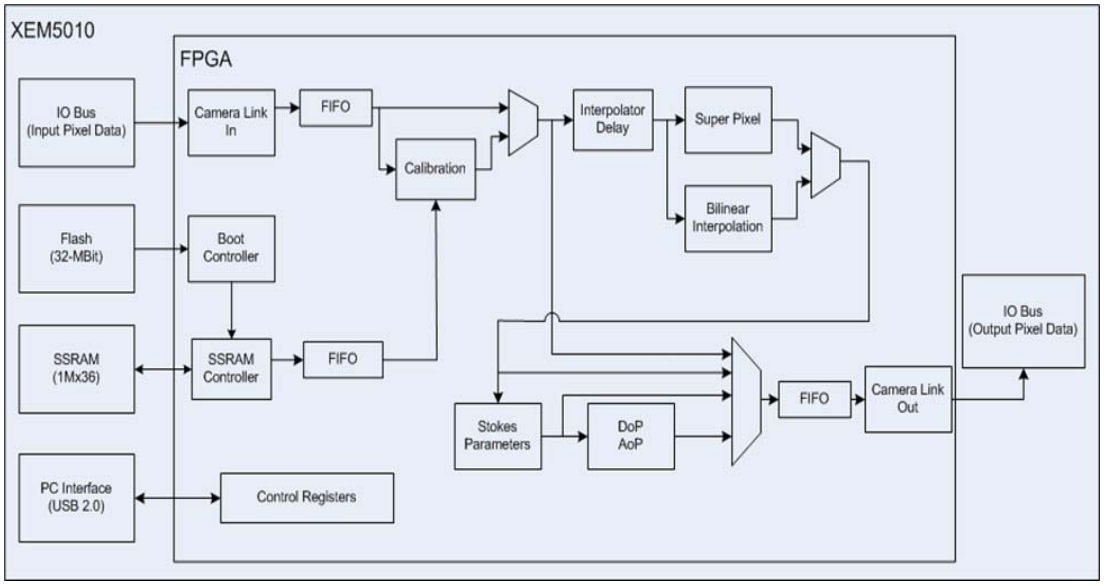

Figure 2.3: Block diagram depiction of the original project for processing polarization image data.

Referring to Figure 2.3, the image processing pipeline implemented on the FPGA utilizes many components of the XEM5010. The IO buses are used to feed camera pixel data into the FPGA and also feed the processed pixel data as outputs from the FPGA. Also in the original implementation, the Flash memory is used to load the

FPGA programming when the board is powered. The Flash also holds the calibration gain data used in the original implementation of the calibration routine. The SSRAM is also used in the original implementation to store the calibration gain data for fast access during polarization image processing. There are also multiple control registers implemented on the FPGA. These registers are used to control various options that can be read or written during operation of the image processing pipeline. The registers are observed through the Opal Kelly USB interface. Through this interface, a user can change some register values during operation or read current states of the registers. The default values for each register are stored in Flash memory as well.

After the board is powered and the FPGA programmed, the calibration gain data is read from the Flash and written to the SSRAM. The default register values are also read from Flash and stored in each register. After this is complete, the image processing pipeline is enabled.

The new implementation requires more control from the PC interface, so the Flash memory is not utilized. The PC now acts as the main controller for the FPGA with the elimination of the Flash memory. Using the C++ API from Opal Kelly, a software program is written to function as the main controller. The new implementation is displayed in Figure 2.4.

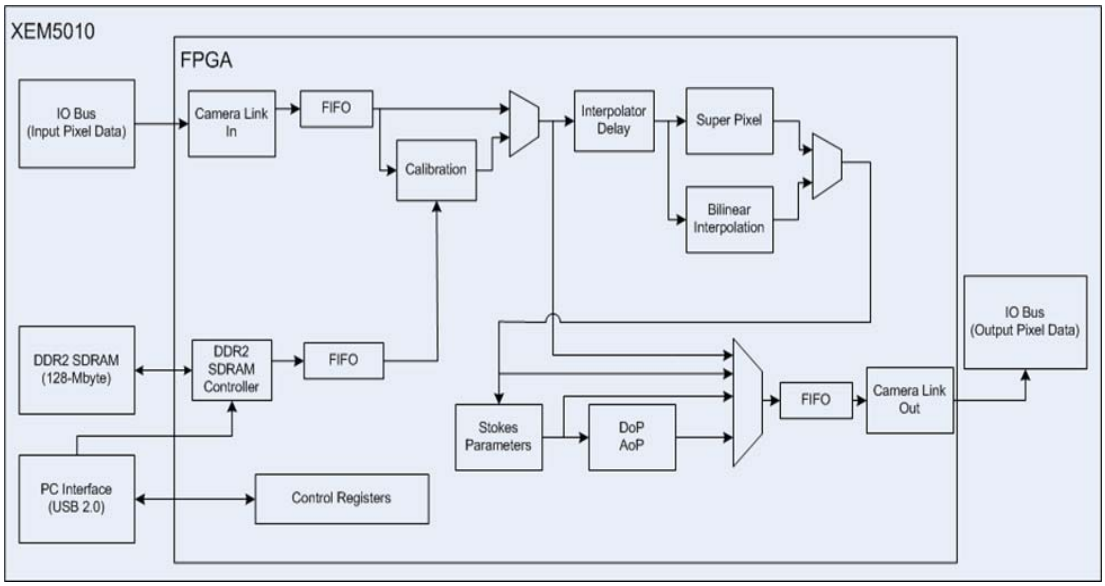

Figure 2.4: Updated implementation of the polarization image processing pipeline project.

As seen in Figure 2.4, the Flash and SSRAM are not utilized. Instead, the board interacts with a PC through the USB interface, and the calibration data is stored in DDR2 SDRAM. The image processing pipeline functions the same as the original design. The only change required is to the calibration routine. The calibration routine is changed to apply gain and offset data instead of gain data only. The implementation of the DDR2 SDRAM interface and the updated calibration routine are discussed later in this chapter.

For the new design utilizing the DDR2 SDRAM for storing and reading calibration data, the PC acts as the main controller. When the XEM5010 is powered up, the PC writes the calibration data into the DDR2 SDRAM. The calibration data is only written into memory once at startup. After writing the calibration data, the PC writes the

required values into the various controller registers implemented throughout the design. Once this is accomplished, the PC issues a boot done flag which allows the pipeline to start processing pixel data. The pipeline continues to function until power is removed from the board.

2.3 Polarization Image Processing Pipeline

The pixel data from the camera is input to the FPGA using the Camera Link standard. The Camera Link standard is a communication standard for transferring camera video data to a frame grabber. Using this standard, the pixel data is input into the FPGA with image enable signals. Frame valid, line valid, and data valid enable signals are input to indicate valid pixel data for an image. Each image is transmitted line by line until the full image is sent. The enable signals are sent as well to indicate when a frame is valid, lines in that frame are valid, and when the pixel data is valid. The Camera Link standard is explained in detail in [5]. The pixel data and Camera Link enable signals are sent through the Camera Link In routine before being sent through the pipeline. The purpose of this module is to convert the enable signals into signals that are easier to implement in logic performed in the image processing pipeline. The frame enable and line enable signals are converted into frame end and line end signals. The frame enable and line enable signals are active during the entire frame and line respectively. The Camera Link In module converts these to be active for one clock cycle at the end of a valid frame or valid line of image pixel data. After the image data passes through the

pipeline, the Camera Link Out routine converts the data and control signals back to the real Camera Link standard to be transmitted to a frame grabber.

The polarization image processing pipeline implements different routines to implement the image processing. Multiplexers are utilized to allow a user to bypass certain stages of the pipeline or display the processed pixel data at various stages of the pipeline. The selections for these multiplexers are controlled using the controller registers. The first stage of the image processing pipeline is the calibration routine. This routine is described in detail later in this chapter. This stage can be bypassed such that the image processed does not receive calibration. The next stage is the interpolator delay. This module is used to delay the pixel data until multiple lines of image data are fed into the pipeline. This is necessary for the bilinear interpolation routine implement the algorithm. The bilinear interpolation routine performs nearest neighbor averages to recover all intensity values at every filtered polarization angle. In this algorithm, a pixel missing an intensity value at a polarization angle obtains its value by averaging the pixel values of its nearest neighbor pixels that were filtered at that polarization angle. Doing this requires multiple lines of image data for pixels to properly average all nearest neighboring pixels since a neighboring pixel could be a line above or below the current pixel being applied. After performing the interpolation, enough polarization data is recovered to obtain the Stokes parameters and then the DoP and AoP. These represent the last stages of the polarization image processing pipeline.

2.4 DDR2 SDRAM Control and Implementation

As discussed in this chapter, the XEM5010 board is equipped with two Micron 128-MByte DDR2 SDRAM chips. The greatest advantage of using the DDR2 SDRAM over the SSRAM in the implementation of the polarization image processing pipeline is the amount of storage space available for use. The greater amount of storage space allows for more accurate calibration data to be stored and read into the calibration routine. By utilizing the DDR2 SDRAM, the calibration routine is enhanced to apply gain and offset calibration data for every pixel in an image.

The design for utilizing the DDR2 SDRAM in the polarization image processing pipeline is as follows. A DDR2 SDRAM controller is implemented on the FPGA to issue memory reads and writes to a DDR2 SDRAM chip. The controller is implemented as a state machine. It remains in an idle state until a read or write command is issued. The controller changes states to perform the commanded action and returns to idle state until a new command is issued. When the XEM5010 board is powered, the calibration data is written into the DDR2 SDRAM memory from the software on the PC. Once the calibration is stored in memory, the DDR2 SDRAM controller remains in an idle state until read commands are issued from the calibration routine.

The DDR2 SDRAM controller with the state machine described is the interface for other modules on the FPGA to control the DDR2 SDRAM. The low level physical control implementation for DDR2 SDRAM is much more involved and complicated. To successfully interact with the DDR2 SDRAM memory chips, the low level and physical layer controls and interfaces are implemented using the Xilinx Memory Interface Generator (MIG) Core. The MIG implements a top level interface layer, a control layer, and a physical layer to the DDR2 SDRAM. The DDR2 Controller state machine interfaces with the top level interface layer of the MIG. The top level interface layer of the MIG contains multiple FIFOs. The read commands, write commands, and corresponding addresses are sent through one FIFO. The write data is sent through its own FIFO, and the data read is fed out through a FIFO as well. The MIG uses the FIFOs to issue the commands in the order received to the DDR2 SDRAM through its control and physical interface layers. Opal Kelly provides a sample project implementing the MIG Core to read and write data to the DDR2 SDRAM. Using this sample project, much of the setup to use the MIG and DDR2 SDRAM is ready for use to tailor on a designer's own implementation. The sample is used as a base to implement the MIG Core. The setup provided in this sample includes parameters required for the MIG to interface with the specific DDR2 SDRAM chips on the XEM5010 board, user constraints for defining the timing constraints and pin outs from the FPGA to the memory chips, and necessary clocks generated using the 100 MHz LVDS clock oscillator as a source. Extensive details of the MIG Core are provided in the Memory Interface Solutions User's Guide [6].

As stated before, the DDR2 SDRAM controller interfaces with the MIG to issue read and write commands to the memory. The state machine in the controller and the top layer interface of the MIG both run at 100 MHz. The current design implements an 80 MHz clock for the image processing pipeline. When writing data to the memory from the PC interface, the Opal Kelly interface runs off a 40 MHz clock. To cross clock domains for both reads and writes, the DDR2 Controller implements Xilinx FIFO Cores. The FIFOs implemented have different clock domains for their read and write interfaces. This allows an easy transition between the different clock domains. The state machine in the controller handles sending the correct commands to the MIG, while the FIFOs handle the data being read from or written to the MIG. There are two distinct FIFOs implemented in the controller, the write FIFO and read FIFO. The write FIFO is generated to allow data writes from the PC interface to cross the clock domain for the MIG. In this FIFO data is written in from the PC interface. The controller state machine issues a read enable to this FIFO when it is given a memory write command. The data read from the FIFO is input to the MIG when read from the FIFO. The read FIFO has the opposite implementation. When the controller state machine receives a read memory command, it issues the read command to the MIG. The MIG output data is written into the read FIFO. The data can then be read from this FIFO by the calibrator routine. The FIFOs also function to allow different data widths to be written to and read from the FIFOs. A diagram of the data flow is displayed in Figure 2.5.

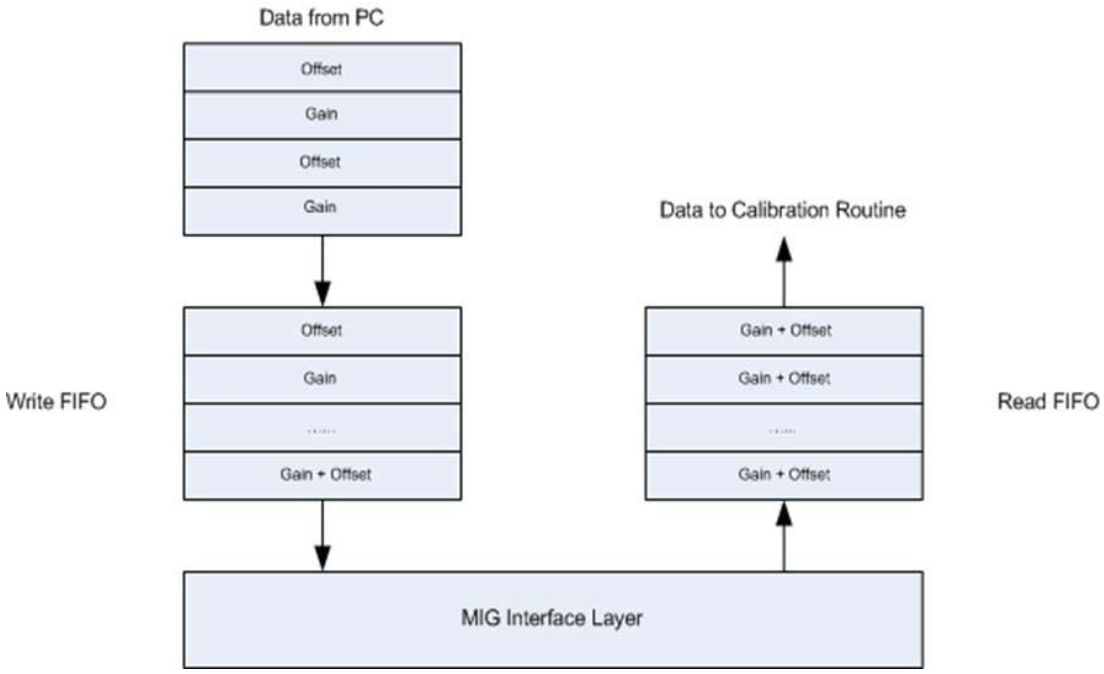

Figure 2.5: Diagram depicting data flow into and out of the DDR2 SDRAM interface.

Referring to Figure 2.5, each 16-bit gain and offset are written separately into the Write FIFO from the PC interface. When a write command is issued to the MIG, it reads one calibration set in one read from the Write FIFO. This is done because DDR2 SDRAM performs data reads and writes on the positive and negative edge of a clock cycle. The Read FIFO keeps the same data width for its read and write interfaces. When a read command is issued to the MIG, both the gain and offset for a pixel are read back as a pair into the Read FIFO. This is advantageous because it allows the calibration routine to read back the gain and offset for a pixel in one clock cycle. By doing this, additional delays in processing are not required in the calibration routine due to extra data reads from memory to apply calibration to an incoming pixel.

2.5 Calibration Routine

The original design of the polarization image processing pipeline implemented on the FPGA applied gain calibration values for each pixel from SSRAM. In the new implementation, the calibration routine is updated to apply a gain and offset calibration to each incoming pixel. The calibration routine is required to apply the correct gain and offset for each incoming pixel value. The updates to the calibration routine include implementing the calculation of the offset in addition to the gain calculation and controlling the calibration data reads from the DDR2 SDRAM controller described in the previous section.

The calibration routine controls the data reads by interfacing with the DDR2 SDRAM controller. The routine utilizes the frame end, line end, and data valid signals input with the pixel data in its logic to determine when to read data from the DDR2 SDRAM controller. The data valid signal is used to determine when to read the next calibration data set from the DDR2 SDRAM controller. The frame end signal is used by the calibration routine to determine when to issue a reset to the DDR2 SDRAM controller. This design is implemented such that the first pixel received for an image frame corresponds to the first calibration data set stored in the DDR2 SDRAM. Each subsequent pixel data point read in thereafter receives the next calibration point in memory. When the end of the image frame is signaled by the frame end signal being set, the calibration routine sets a reset to the DDR2 SDRAM controller. With the reset

issued, the controller begins reading data from the first memory address when read commands are issued and increments the address itself every time it enters the read state due to a read command. The calibration routine is then responsible for reading the calibration data from the Read FIFO in the controller when it is available. The calibration routine does not control the memory addressing itself due to the different clock sources controlling it and the DDR2 SDRAM controller. Instead, the calibration routine interfaces with the Read FIFO which has its read interface synchronized with the calibration clock source. The only signals issued to the DDR2 SDRAM controller state machine are the reset signal when an image frame end is reached and the read enable signal. The read enable signal is set when the pipeline is enabled and stays set. This allows the DDR2 SDRAM controller to continually request the data from DDR2 SDRAM and allows the calibration routine to simply interface with the controller's Read FIFO for the data.

To apply gain and offset calibration values, the calibration routine requires two clock cycles of the 80 MHz pipeline clock. This is done to apply the gain in one clock cycle and the offset in another. The pixel data is never delayed to wait on data reads from the DDR2 SDRAM for calibration points. Since the calibration routine continually receives new pixel data when the pipeline is enabled, delaying the input pixels to wait on available calibration data from memory can cause the images to lose alignment. Instead, an alignment flag is tracked in the calibration routine. This flag is set when the end of frame for the image data is reached and the number of gains read for that image matches the number of pixels in an image frame. If these are not the same, then the

image is out of alignment and calibration is not applied until alignment is achieved. By implementing this logic, the calibration routine never throttles the incoming pixel data.

Chapter 3

Analysis and Results

When using the DDR2 SDRAM to feed calibration data into the calibration routine, the correct calibration data must be applied to its corresponding pixel. To test this, the XEM5010 is connected to a camera containing a regular imaging sensor and a frame grabber. The camera used for testing is an Imperx camera. The Imperx FrameLink Express application is utilized on a PC to present data grabbed from the frame grabber as a visual representation on the PC. The camera contains test images that can be fed into the FPGA for testing. Using the FrameLink Express application, commands can be sent to the camera to display different test images.

To verify each calibration data set is consistently applied to its corresponding pixel value, random calibration values are generated in the C++ software and written into DDR2 SDRAM. By using random data for calibration, any variations in calibrations applied over time to the pixels will be easily seen on the visual display in the FrameLink Express application. This is so because the calibration data will differ randomly from pixel to pixel. In the tests performed, portions of the image will contain uniform calibrations as well. A clear boundary is set between the random calibrations and uniform calibrations applied. Test images are displayed below in Figures 3.1 and 3.2

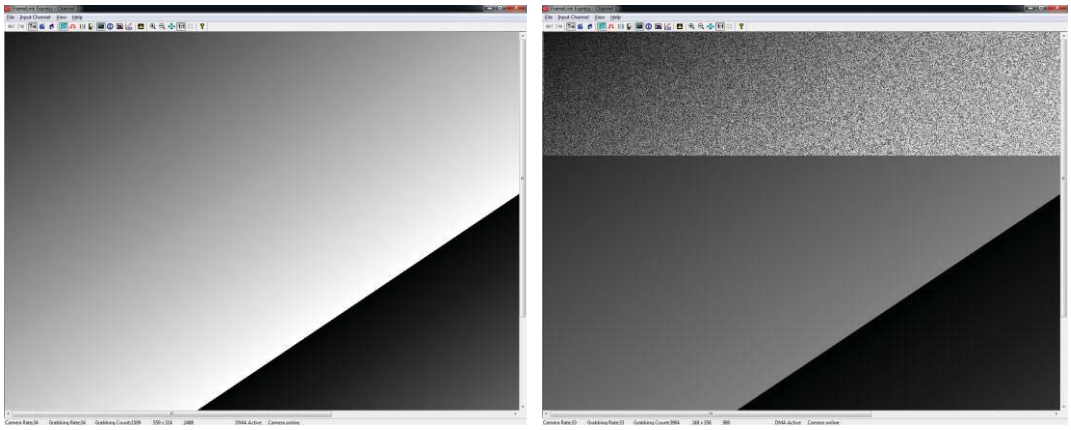

Figure 3.1: First test image to apply random and uniform calibration. The image is not calibrated on the left, while the calibration is applied on the right. Notice, the random calibration is distinct from the uniform.

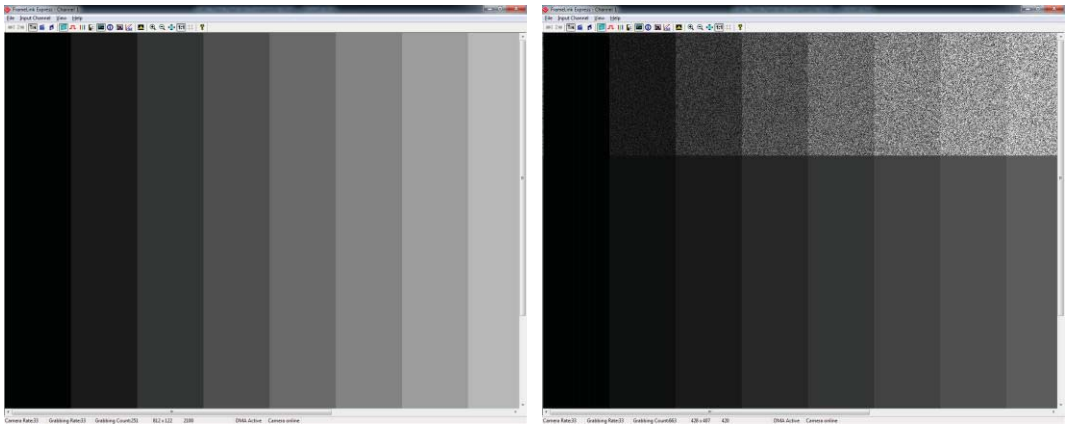

Figure 3.2: Second test image to apply random and uniform calibration.

Figures 3.1 and 3.2 contain a test image on the left without the random calibration data applied. On the right side of each, the same test image is shown with the random calibration data applied. As can be seen in both figures, there is a distinction between the random calibration regions and uniform calibrations. When observing these images in real-time using FrameLink Express, there is no visual change in the random

calibration region over time. From this, it can be concluded that the same calibration data is being applied to a corresponding pixel consistently.

Conclusion

Polarization imaging sensors have gained popularity in recent years, and the application of observing polarization images in real-time is also finding advantages. By implementing calibration to imagers in real-time, the real-time polarization image processing allow more polarization information to be consistently retained. Utilizing DDR2 SDRAM on the chosen integration module provides a proven method for applying the calibration in real-time. By applying the calibrations, the polarization imager retains the optimum amount of polarization information for processing.

Appendix A

Verilog HDL Documentation

The polarization image processing pipeline project is implemented on the FPGA using Verilog HDL. Each module is documented in this appendix. Access to the Verilog HDL code and the project files for Xilinx ISE are available on the VLSI lab share at Washington University in St. Louis, Missouri. In all cases of the Verilog modules described in this appendix, the module name is the same as the file name of the Verilog source file containing the module.

TestContainer

This module is the top-level module of the entire project. This module contains the implementation of the Opal Kelly USB interface signals utilized. It also implements the IO for the FPGA itself and wires those IO pins to the contained modules. Clock generation is also performed in this module for the image processing pipeline. When using Xilinx ISE, this should always be set as the top module for the project.

PipelineContainer

The container connects the polarization image processing pipeline to all of modules that interface with it. The main pipeline module is contained as well as the interface to the

DDR2 SDRAM Controller, Camera Link In, and Camera Link Out. This module wires up those interfacing modules with the pipeline.

CameraLinkIn

This module converts the Camera Link standard pixel data and control signals into easier formats for implementation in the pipeline.

CameraLinkOut

This module changes the data processed in the pipeline back into the Camera Link standard signals before being sent out of the FPGA.

DDR2Controller

This is the top-level interface to the DDR2 SDRAM. The module wires up the Memory Interface Generator (MIG) modules with the state machine controlling access to the DDR2 SDRAM.

ddr2_idelay_ctrl, infrastructure, ddr2_top

These three modules are MIG modules and should not be hand edited. They are wired to the correct modules in the DDR2Controller module.

DDR2_Sequencer

This module contains the state machine for controlling read and writes requests to the MIG layer to access data in DDR2 SDRAM. It also implements FIFOs to allow clock

domain crossing between the DDR2 controller modules and other modules implemented on the FPGA.

ImagePipeline

This module contains all modules for the image processing pipeline itself. The pipeline modules are wired together in this module to implement pipeline behavior when processing the incoming image data.

Calibrator

This module implements the calibration routine described in detail in this project. Refer to Chapter 2 for more details on the calibration routine.

InterpolatorDelay

This implements the delay to obtain three lines of image data to pass onto the next stage of the pipeline. The three lines are needed to apply the bilinear interpolation algorithm to recover polarization data for all pixels.

SuperPixel

This module takes the pixel data and converts the pixels into 2x2 super pixels.

Bilinear

This module applies the bilinear interpolation algorithm to recover polarization information.

StokesParameters

This module calculates the Stokes Parameters.

DegreeAndAngle

This module calculates the DoP and AoP.

Appendix B

C++ Software Documentation

As described in this paper, software is written in the C++ programming language utilizing the Opal Kelly API. This API allows easy communication with the XEM5010 FPGA integration module. Using this interface, the software is designed as the main controller of the polarization image processing pipeline. Through the software, calibration data is written into DDR2 SDRAM, control registers can be accessed and changed, and the image processing pipeline is enabled. The software runs as a Windows console application. When first starting up the application, messages are printed to display progress of startup which includes writing data to DDR2 SDRAM, initializing control registers, and enabling the pipeline. A menu then appears that allows a user to pick a control register and alter its contents.

The software is written in an object oriented design. Each class and their member functions are detailed in this appendix.

CameraLinkControl

This is the main function of the program. The main connects to the FPGA and programs it with the specified FPGA programming file. It also instantiates the DDR2 and ControlRegisters objects. Once initialization is done, the function prints the menu

for the user. The program keeps this menu until a user chooses the option to exit the program.

SDRAM Class

The SDRAM class contains the functionality to interact with the DDR2 SDRAM controller on the FPGA.

SDRAM::TestSDRAM

This member of SDRAM runs a full write and read test on a selected DDR2 SDRAM chip on the XEM5010. The test writes a set of data to the memory and reads it back. It then compares the data written to the data read back. A comparison is performed to verify the data matches exactly. A successful test shows the DDR2 SDRAM and its controller are functioning correctly.

SDRAM::WriteToSDRAM

This function receives a byte array and a number of bytes arguments to indicate how many bytes of data are written into DDR2 SDRAM. This function does not handle the addressing of the data in DDR2 SDRAM.

SDRAM::ReadFromSDRAM

This function receives and argument for number of bytes to read back from DDR2 SDRAM. Using this amount, the data is read back into a byte array from memory. This function does not control DDR2 SDRAM addressing.

SDRAM::ResetSDRAM

This function issues a reset command to the DDR2 SDRAM controller on the FPGA. When the reset is executed, the controller starts at address zero for the next read or write to the DDR2 SDRAM.

ControlRegisters Class

This object contains the implementation for initializing and issuing commands to the control registers on the FPGA.

ControlRegisters::InitializeAllRegisters

This function initializes all writable control register to their default startup values.

ControlRegisters::WriteDataToRegister

This function writes data to a particular control register. The function receives the data and address of the control register as arguments.

References

[1] T. York and V. Gruev, Calibration Method for Division of Focal Plane Polarimeters in the Optical and Near Infrared Regime, Proc. SPIE 8012, Infrared Technology and Applications XXXVII, 80120H (May 20, 2011).

[2] V. Gruev, Fabrication of a Dual-Layer Aluminum Nanowires Polarization Filter Array, Optics Express Vol. 19 Issue 24 Pages 24361-24369 (November 21, 2011).

[3] B. E. Bayer, Color Imaging Array, U.S. Patent 3,971,065, (July 20, 1976).

[4] Opal Kelly XEM5010 Product Webpage. Available: http://www.opalkelly.com/products/xem5010/.

[5] Camera Link Specification, Automated Imaging Association, January 2007.

[6] Memory Interface Solutions User Guide, Xilinx, September 2010.

[7] Shengkui Gao and Viktor Gruev, Image Interpolation Methods Evaluation for Division of Focal Plane Polarimeters, Optics Express Vol. 19 Issue 27 Pages 26161-26173 (December 19, 2011).

Vita

Collin Foster

Degrees	B.S. Cum Laude, Computer Engineering, May 2008 M.S. Computer Engineering, December 2012
Professional Societies	Institute of Electrical and Electronic Engineers
Experience	Software Engineer, The Boeing Company, June 2008 – Present

December 2012

Calibration, Foster, M.S. 2012