LOW-NOISE IMAGE SENSOR DESIGNED FOR NEAR-INFRARED IMAGE-GUIDED SURGERY

BY

ERIC CHEN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Associate Professor Viktor Gruev

ABSTRACT

Various technologies can vastly augment the abilities of a physician. X-ray, magnetic resonance imaging (MRI), and near-infrared (NIR) fluorescence are a few categories of medical imaging that are capable of gathering information from below the skin surface.

NIR fluorescence imaging is very compatible with the needs of medical imaging. NIR imaging systems can be lightweight and portable. They are relatively safe for human exposure. These advantages make NIR imaging a great option for real-time image-guided surgery.

The project covered in this thesis produced a low-noise camera capable of seeing both visible and near-infrared light with a single image sensor. This information can be displayed to a physician in real time. The camera has 1024 x 1024 pixels, 22 fps, and 2 electron readout noise. It uses a pixelated filter array; it has a red, green, blue, or near-infrared filter over each pixel. The camera system is composed of an Opal Kelly XEM-7310 FPGA integration module, a low-noise image sensor chip, and a computer. A PCB was designed to hold the image sensor and auxiliary components. Verilog was written to communicate with the image sensor chip and retrieve real-time video data. USB 3.0 interface transfers the video data to the computer. The computer provides a real-time video display of the RGB and near-infrared video. Keypresses and a graphical user interface (GUI) are used for user inputs, such as video data saving and camera exposure control.

To my mother, father, and sister

ACKNOWLEDGMENTS

I would like to thank my adviser, Professor Viktor Gruev, for giving me the opportunity to be a part of his research group. It is Professor Gruev's guidance that helped me in my pursuit of becoming a better engineer. His encouragement and feedback were crucial factors in my personal growth during my time here at the University of Illinois

I would like to thank my senior lab mates Missael Garcia, Nan Cui, and Steven Blair for their help and support. I am grateful for how they shared their insight and experiences with me, which helped me learn and grow as a person.

I would like to thank the MNTL and ECEB staff for their assistance. Without their efforts, it would have been impossible to put together most of our experiments and projects.

Finally, I would like to thank my family. Their constant love and support helped me overcome many difficulties and get through graduate school. I am especially grateful for the amazing inspiration my sister has been for me.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION AND MOTIVATION 1

CHAPTER 2: CAMERA CAPABILITIES AND FEATURES 4

CHAPTER 3: CAMERA DESIGN 12

3.1: Camera System Overview 12

3.2: Image Sensor Chip Selection 14

3.3: FPGA Board Selection 15

3.4: PCB Design 16

3.5: FPGA: System Overview 19

3.6: FPGA: LVDS Training 21

3.7: FPGA: Modules for Pixel Deserializing and Control Signal Inputs 28

3.8: Computer Program: Camera Power-on and Startup Sequence 29

3.9: Computer Program: Functions for Continuous Operation Mode 32

CHAPTER 4: CONCLUSION AND FUTURE WORK 35

4.1: Conclusion 35

4.2: Future Work 35

REFERENCES 37

CHAPTER 1: INTRODUCTION AND MOTIVATION

In today's age of medicine, there is a wide variety of biomedical imaging tools that physicians can use to help them see. These tools help them see details that are not detectable using their own sense of touch or sight [1]. While the eyes of a surgeon can only see visible light, these specialized tools can see using sound waves, invisible light, magnetic fields, and other methods.

Near-infrared (NIR) light has properties that make it a good choice for medical imaging [2]. Humans can only see visible light, which is reflected and absorbed by the outer layer of skin. This means the physician's sight stops at the skin surface. Near-infrared light solves this issue because it is more capable of passing through human tissue. Near-infrared light has lower absorption and scattering coefficients for passing through human tissue. NIR imaging systems are capable of producing images a few centimeters below the skin surface. Using NIR imaging, the physician can see below the skin surface without needing to make a surgical opening.

Other imaging modalities are also capable of seeing underneath the skin surface, but near-infrared also holds the advantage of being relatively safe and flexible [3]. NIR imaging does not require lead-aprons like x-ray imaging and does not require machines that occupy an entire room like magnetic resonance imaging (MRI). NIR imaging can be used for real-time imaging. X-ray and MRI are typically used for preoperative and postoperative imaging.

NIR fluorescence imaging can be used alongside normal surgical operations, providing feedback to the physician in real time. NIR light is invisible to the human eye, so it does not obstruct the view of the physician. Near-infrared fluorescence (NIRF) imaging takes advantage of the fluorescence properties of certain particles.

Indocyanine green (ICG) is an FDA approved dye used for NIRF imaging. ICG particles are excited by NIR light: when ICG particles absorb the light, their valence electrons transition to

a higher energy state. After this occurs, the ICG particles will emit NIR light of a different shade (longer wavelength).

ICG has useful properties for NIRF imaging systems. ICG binds to albumin [4], a form of protein found in blood plasma. The enhanced permeability and retention effect [5] means that molecules of certain sizes tend to accumulate in tumor tissue in greater quantities than in normal tissue. This is a result of tumor angiogenesis, which involves increased blood vessel formation which is also of abnormal form. These additional blood vessels bring additional resources to the tumor site, while their abnormal form makes them less effective at draining molecules such as ICG.

ICG is used in the sentinel lymph node biopsy procedure [6]. The ICG dye marks the sentinel lymph nodes, and the NIR light emissions allow specialized imaging systems to display their location to the physician with high sensitivity and detail.

A typical NIRF IGS setup is shown in figure 1. After injecting ICG into a patient, the fluorophore will have a temporary accumulation in features such as tumors. Without turning off the surgical lighting (visible light) that the surgeon uses to see, an NIR laser is pointed at the region of interest. The fluorophores in that region will be excited by the NIR laser, and then emit NIR light. This allows an imaging system to make tumors and other features visible.

For technology to be relevant in real-time clinical imaging, it must solve a specific problem while not impeding the normal clinical workflow. There are already many FDA approved NIRF imaging systems available for clinical use [7]. Most of these have the disadvantage of being bulky and expensive.

This project aims to produce a lightweight system, capable of simultaneously detecting NIR fluorescence features and an RGB scene. This project uses a single image sensor with

pixelated filters directly above the imaging array. This concept has been explored before [8], and this project aims to push this technology further with high sensitivity capabilities. The result is a camera that is enclosed in a 2.5 in × 3.5 in × 3 in box, with a USB 3.0 interface that enables it to run on a laptop, with real-time video display. The image sensor used has a median readout noise of 2 electrons per pixel. This camera has 1024 ×1024 pixel array capable of running at 22 frames per second (fps).

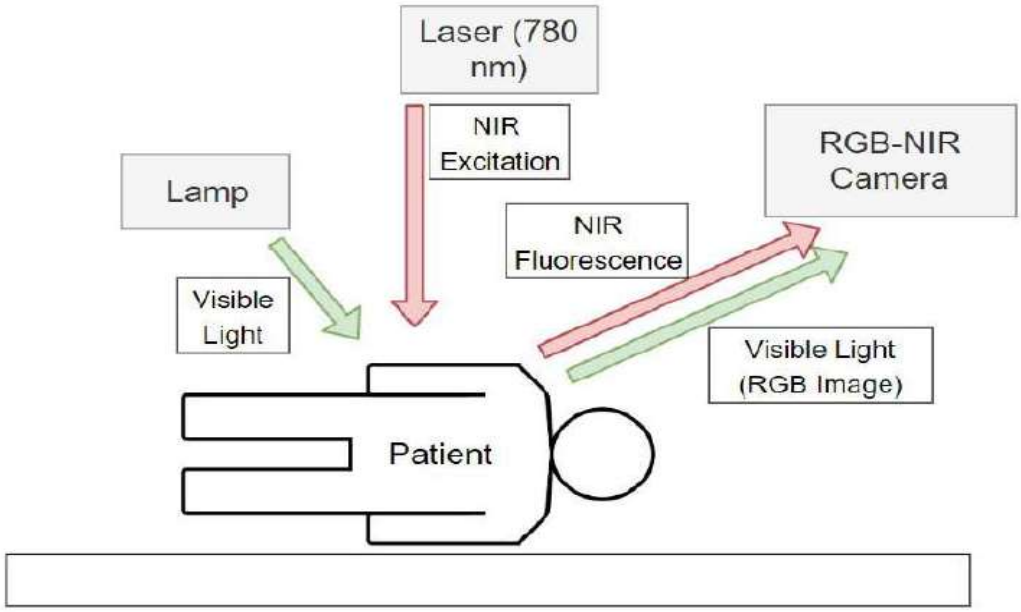


Figure 1: Diagram of near-infrared fluorescence image-guided surgery. Visible light illuminates the patient so the surgeon can see with their own eyes, and so an RGB image can be seen with the camera. NIR laser light excites the fluorophores inside the patient, emitting a NIR light signal which the camera can also see.

CHAPTER 2: CAMERA CAPABILITIES AND FEATURES

This project produced a camera capable of simultaneously seeing RGB and NIR light, with low-noise specifications. Figure 2 shows the prototype camera with a cover on top. Standard Canon EF 50 mm lenses can be mounted on this camera. Figure 3 shows the camera prototype with the PCB exposed.



Figure 2: Image of low-noise camera designed for near-infrared fluorescence image-guided surgery. Top cover is in place. A standard Canon EF lens can be mounted.

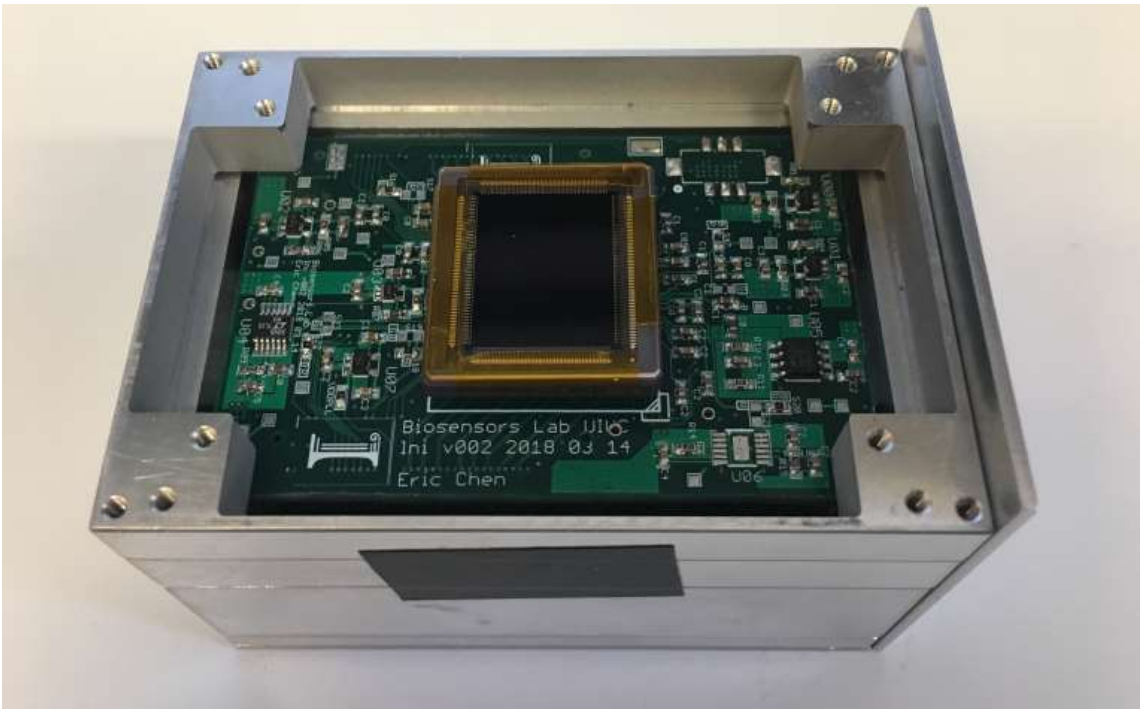


Figure 3: Image of low-noise camera designed for near-infrared fluorescence image-guided surgery. Top cover is removed, exposing the image sensor and the top PCB.

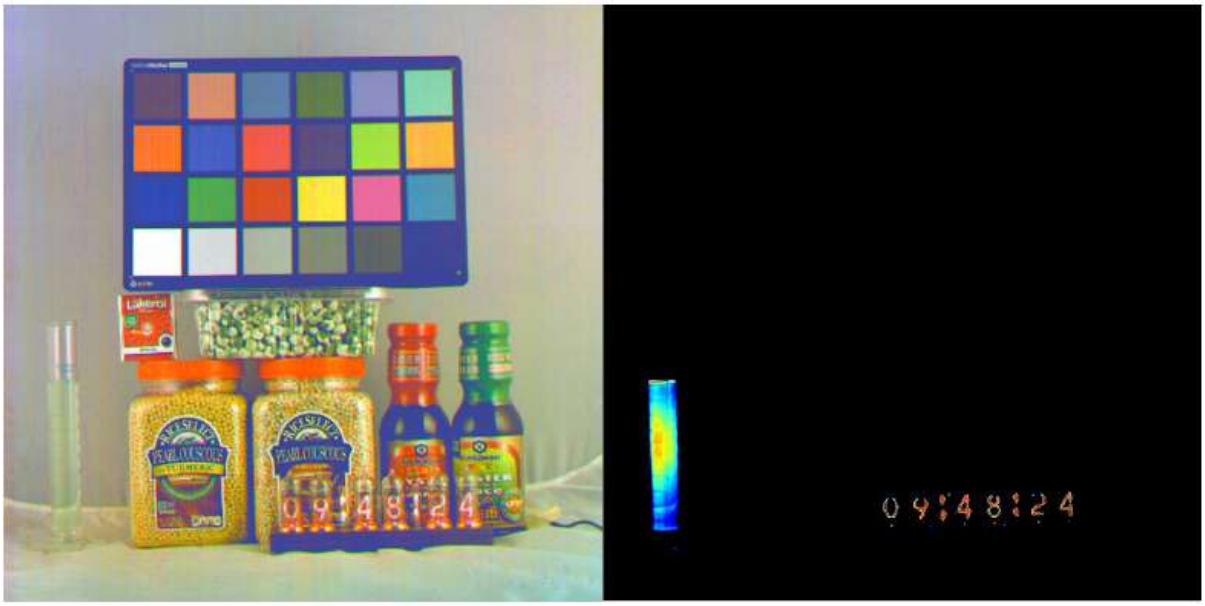


Figure 4: Images of a scene with NIR and RGB light. These images were taken simultaneously by the same camera, as seen by the clock's time. The left side demonstrates the camera's ability to take RGB images. The right side demonstrates its ability to see NIR light.

The camera operates at 22 fps. The resolution is 1024 x 1024 pixels. The readout noise is 2 electrons. This statistic was measured by calculating the noise for each pixel, then taking a median across the entire frame. A median was used to avoid outliers such as hot pixels and dead pixels. The camera has two 12-bit channels operating simultaneously. One channel has a gain of 30× the other channel. These can be combined into a single high-dynamic-range image of 90 dB.

The noise and fps could be improved as the hardware and software are optimized. The expected limit for the framerate is 88 fps; the current limitation to the fps is timing constraints in the FPGA. Alternative implementations of LVDS training will allow the framerate to improve significantly. If active cooling systems are used, the readout noise could be pushed to 1 electron.

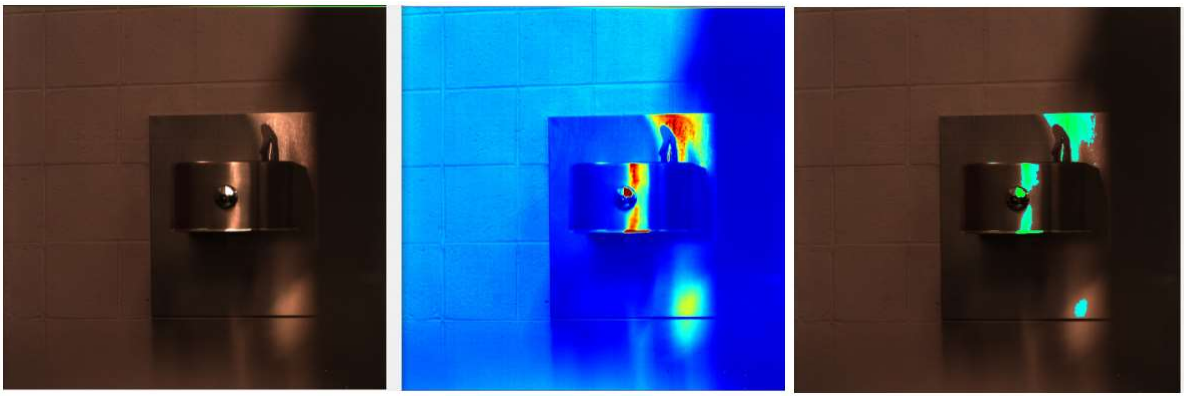


Figure 5: Captures from the real-time operation of this camera. Camera was pointed at a metal water fountain. Left: RGB image. Middle: NIR image, converted to a jet color map. Red represents high values, blue represents low values. Right: threshold mode. The camera checks if the NIR value is above a customizable threshold. If it is, the NIR image at those pixels is overlaid across the RGB image.

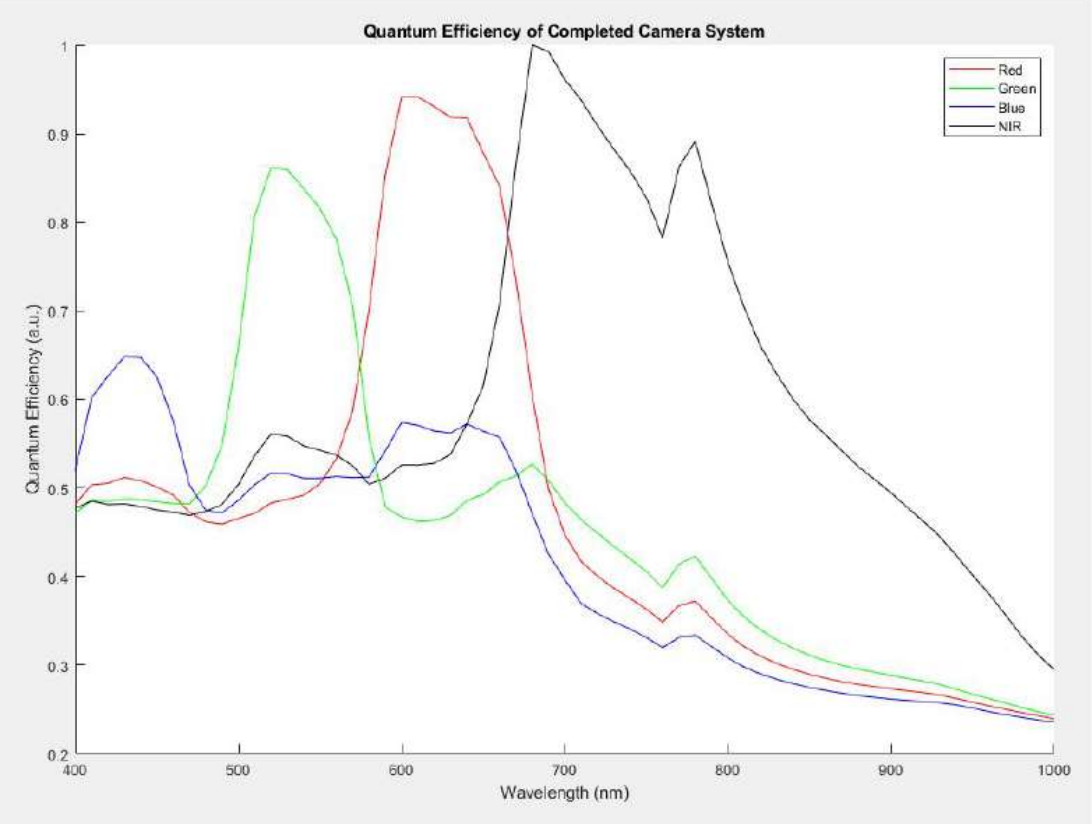


Figure 6: Quantum efficiency of the low-noise RGB-NIR camera. The values are normalized to the highest point. The four curves represent the blue, green, red, and NIR channels.

The real-time video display is capable of displaying both the RGB and NIR image. This is shown in figures 4 and 5. Figure 5 shows a side-by-side screenshot of the real-time video display. The left side shows an RGB image of a sink in room lighting, and the middle shows the NIR light displayed in a jet-color scale. The right image of figure 5 shows the overlay mode, which checks if the NIR value is above a certain threshold. If the NIR value is high enough, those pixels in the RGB image are replaced with the NIR value. The overlay uses a modified jet-color map, which favors green and does not use red. This decision was made because of the target application of image-guided surgery. Colors found in human patients typically favor reds and browns, so green provides a good contrast. In the application of NIRF IGS, this overlay function highlights the

sections that correspond to high values of NIR light, which corresponds to fluorescence emission. This lets the physician more easily locate features such as lymph nodes.

This camera uses a scientific CMOS image sensor chip sCMOS that was purchased from an external company. PCBs were designed and fabricated for interfacing this image sensor chip with the FPGA and computer. The video data is sent to the computer so that it can be processed, displayed in real time, and saved into h5 files.

This camera uses specialized pixelated filters. Pixelated filters are placed above image sensor arrays so that each pixel is only able to see a certain type of light.

This camera uses an RGB-NIR variant of the Bayer pixelated filter array. In every 2 × 2 pattern, it has a pixel dedicated to NIR light, instead of two pixels dedicated to green light. This allows images of NIR light to be seen simultaneously with the RGB image, while only using one image sensor chip.

Figure 6 shows the camera's quantum efficiency for each color channel. For an image sensor, the quantum efficiency is the ratio of detected photons to the incident photons. This data was collected using a monochromator (Acton SpectraPro 2150) and an optical power meter (Thorlabs PM100D).

Figure 7 shows a transmission microscope image of this RGB-NIR filter array. In this image, the red, green, and blue filters can be seen, but the NIR pixel appears as a black square. This is because the microscope is not sensitive to NIR light. Figure 11 shows a block diagram for the image sensor system, with a 2 × 2 pattern representing the RGB-NIR pixelated filter array. Figure 8 shows the optical properties of this camera's filter array. Figure 9 shows a cross section of the pixel array. There is an NIR filter placed above one pixel, and a red filter placed above the adjacent pixel.

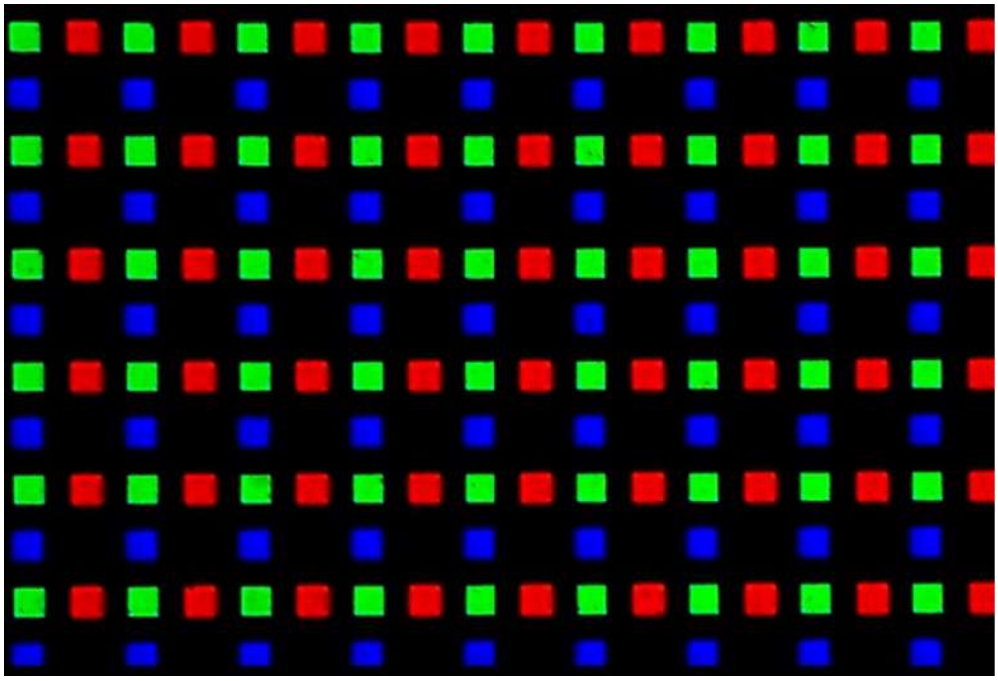


Figure 7: Image of pixelated filter array used on this camera. This image was taken with the transmission mode of a microscope. The red, green, and blue filters are visible. The NIR filter is black because it blocks red, green, and blue light.

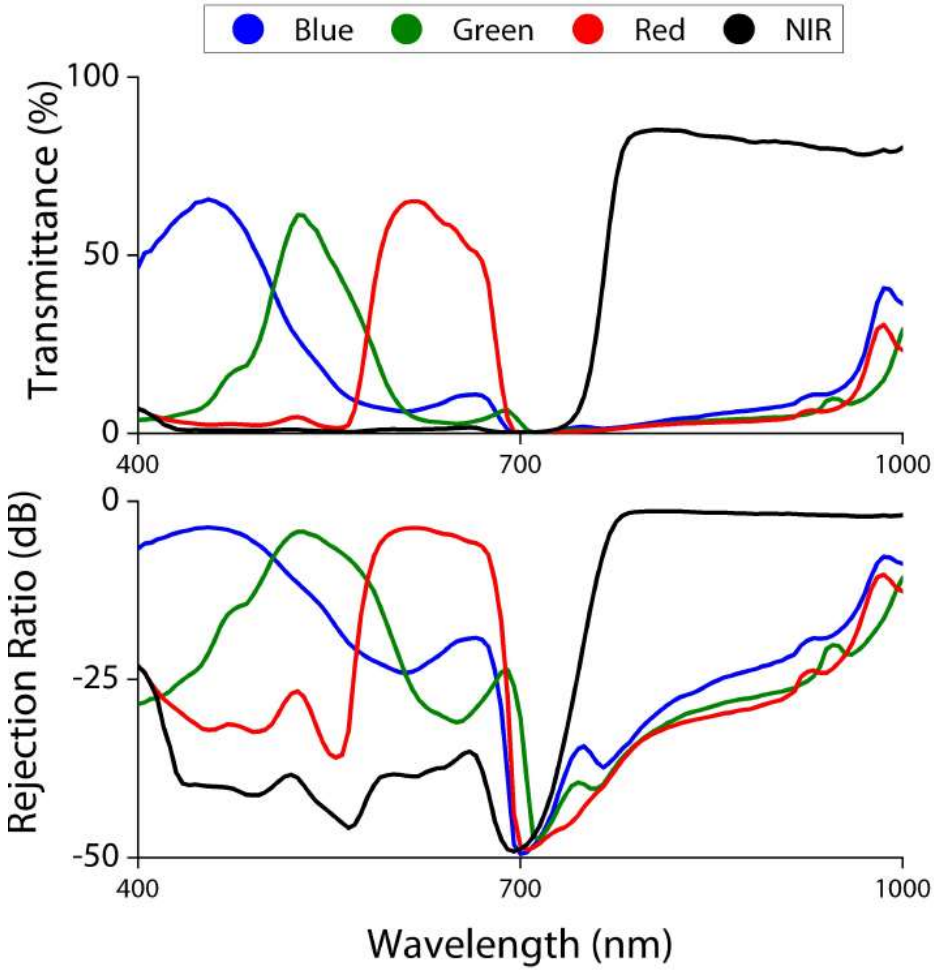


Figure 8: Transmission characteristics of the pixelated filters. The curves for the blue, green, red, and NIR filter are shown.

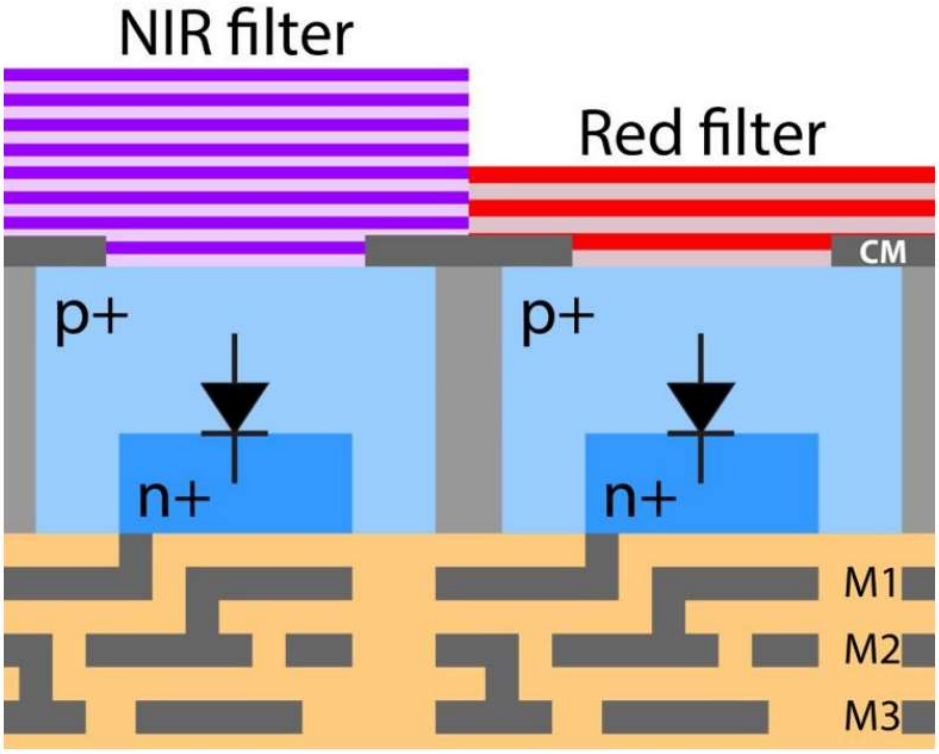


Figure 9: Cross section diagram of the camera. Pixelated filters are shown to be placed above each pixel. A NIR filter and a red filter are shown above adjacent pixels.

CHAPTER 3: CAMERA DESIGN

3.1: Camera System Overview

This project involved developing a low-noise image sensor for RGB-NIR real-time video. It uses a scientific CMOS (sCMOS) image sensor chip to acquire images. An RGB-NIR pixelated filter is placed above the pixels so that each pixel primarily only sees one type of light (red, green, blue, NIR).

This camera system is composed of a computer, an FPGA, and a sCMOS image sensor chip. This is shown in figure 10. The computer handles image processing, real-time video display, and saving video data for later processing. The FPGA is part of an Opal Kelly XEM7310 FPGA integration module. This includes an Artix-7 FPGA, power regulators, and pin connectors to allow for connection to other PCB boards. Verilog code was written for the FPGA system to handle the image sensor's power-on sequence, input control signal timing, SPI register programming, and reception of the raw video data. This FPGA puts this video data into a FIFO module. Another module involves a USB 3.0 interface, which sends the information to the computer.

The sCMOS image sensor is a low-noise image sensor with an RGB-NIR pixel filter array. This pattern is similar in concept to the Bayer RGB filter 2 × 2 pattern. In every 2 × 2 pixel set, the Bayer filter has one red, two green, and one blue pixels. This RGB-NIR camera has one red, one green, one blue, and one NIR filter. This is shown in figure 7, which is a transmission image of the pixelated filter array, illuminated by a white light source. The microscope used for taking this image is not sensitive to NIR light, so the NIR pixels appear as black.

The details and specifications are covered in the following section. This image sensor was selected for its impressive noise performance and sensitivity to both RGB and NIR light.

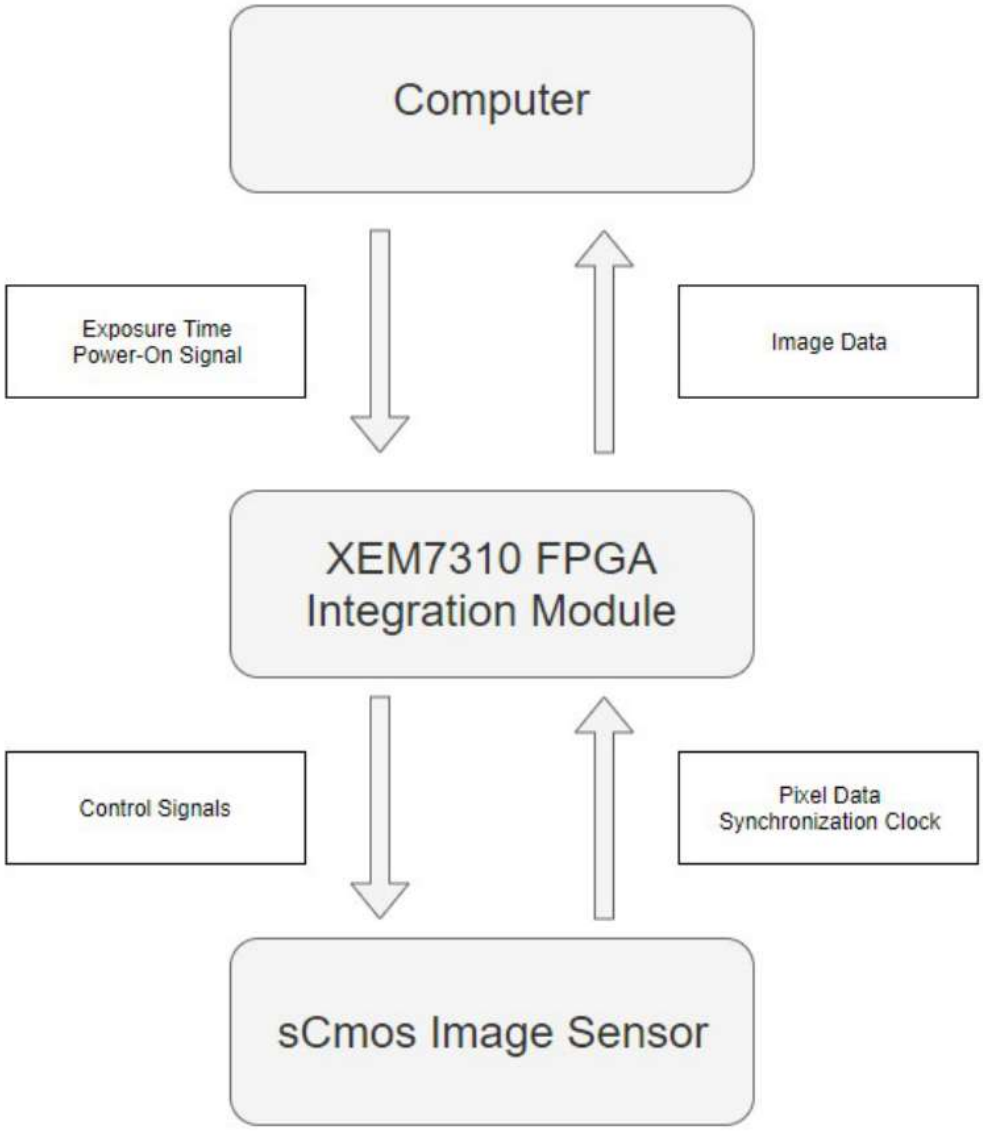


Figure 10: Block diagram of the camera subsystems. The camera is composed of a computer, an FPGA integration module, and a sCMOS image sensor.

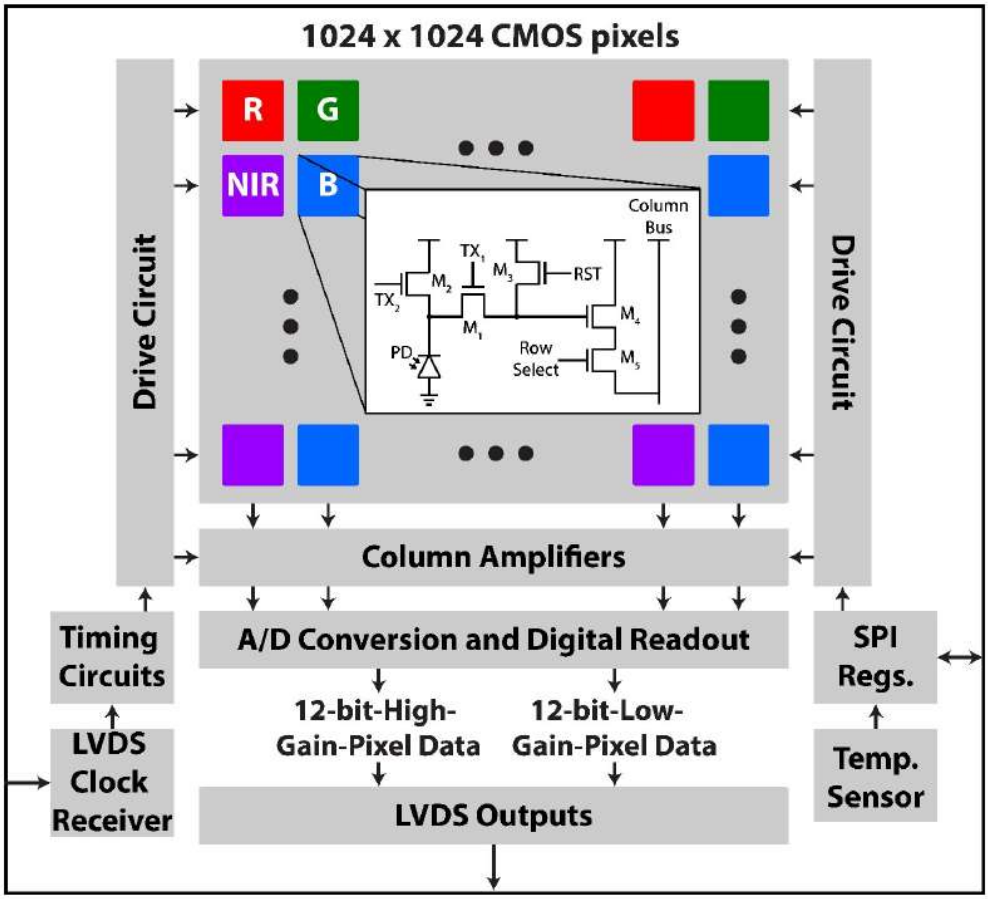


Figure 11: Block diagram of the sCMOS image sensor chip. It uses LVDS communication to transfer data. The outputs are two 12 bit channels, which can be combined into a single image of 90 dB dynamic range. A pixelated filter array with red, green, blue, and NIR pixels is placed above the imaging array.

3.2 Image Sensor Chip Selection

A low noise scientific CMOS image sensor is the substrate for the imaging system. The sensor was modified to have a pixelated RGB-NIR filter array, giving it the ability to differentiate red, green, blue, and NIR light.

Figure 11 shows a block diagram of this camera. It shows some of the subsystems within the camera, and labels the pixelated filter array. This project's sCMOS image sensor has a dark current of 0.2 electrons/second/pixel when cooled to -20 °C. It operates with dual gain channels,

each with 12 bit resolution. One channel has 30× higher gain than the other, allowing for a combined dynamic range of 90 dB.

Pixel data is transferred using eight LVDS channels, with a clock signal for synchronization running on another LVDS channel. There are various single-sided signals used for exposure control, SPI register programming, and other control signal inputs.

3.3 FPGA Board Selection

An FPGA integration module was required for USB 3.0 communication between a computer and the PCB system with the sCMOS image sensor. High data throughput, high number of I/O pins, and differential signal capability were required. The Opal Kelly XEM7310 was selected because it satisfied these constraints. It has a measured data throughput of 340 MiB/s, 124 I/O pins routed to Samtec connectors, and includes its own power regulators and clock generator. The FPGA housed on this integration module is an Artix-7 FPGA (XC7A200T-1FBG484).

In figure 10, it is labeled as “XEM7310 FPGA Integration Module.” In general, the FPGA serves as an intermediate device between the computer and the image sensor. It also handles control signals that have microsecond resolution timing requirements. The image sensor sends image data to the FPGA, which is then sent to the computer over the USB 3.0 interface. The computer sends general instructions to the FPGA, which are routed to the task-specific modules. These modules in the FPGA handle image acquisition through LVDS communication, control signal timing, exposure control, enable of voltage regulators, enable and settings of the clock synthesizer, and timing of the image sensor SPI register writing/reading. The selected Opal Kelly FPGA integration module has plenty of resources for all of these operations.

3.4 PCB Design

There are three PCBs in the camera. A diagram of the PCB layers is shown in figure 12, and a screenshot from Eagle PCB Layout is shown in figure 15. This diagram shows what is on each board and each layer. The bottom PCB is the Opal Kelly XEM7310 FPGA integration module. The middle layer holds some auxiliary components and provides redundant electromagnetic shielding between the image sensor and the Opal Kelly board. One major concern of this design was the large number of switching components on the Opal Kelly board, and whether the electromagnetic waves would affect the image sensor's performance.

Decoupling capacitors were placed adjacent to the voltage regulator outputs and adjacent to the sCMOS image sensor input pins. Each power input pin of the image sensor was given a pair of ceramic surface-mount 0603 capacitors. Each pair includes a 10 μF capacitor for larger volume of current from noise, and a 10 nF capacitor for handling the higher frequency spikes of current from noise.

The voltage regulators were selected to satisfy the constraints of voltage level, noise, and current. All regulators were given a few additional requirements. All selected regulators have an enable pin, so that they can be powered on/off according to the image sensor's startup sequence. This also allows for an additional method of emergency shutdown. Most of the regulators are low-dropout to avoid fluctuations in voltage level.

Figure 13 is a PCB-level block diagram. It shows the relations between the FPGA, the sCMOS image sensor, and the auxiliary components.



Figure 12: Diagram of the PCBs in this camera. There are three boards in total. The top board holds the image sensor. The middle board holds auxiliary components. The bottom board holds the FPGA integration module, which has the USB 3.0 interface for sending data to the computer.

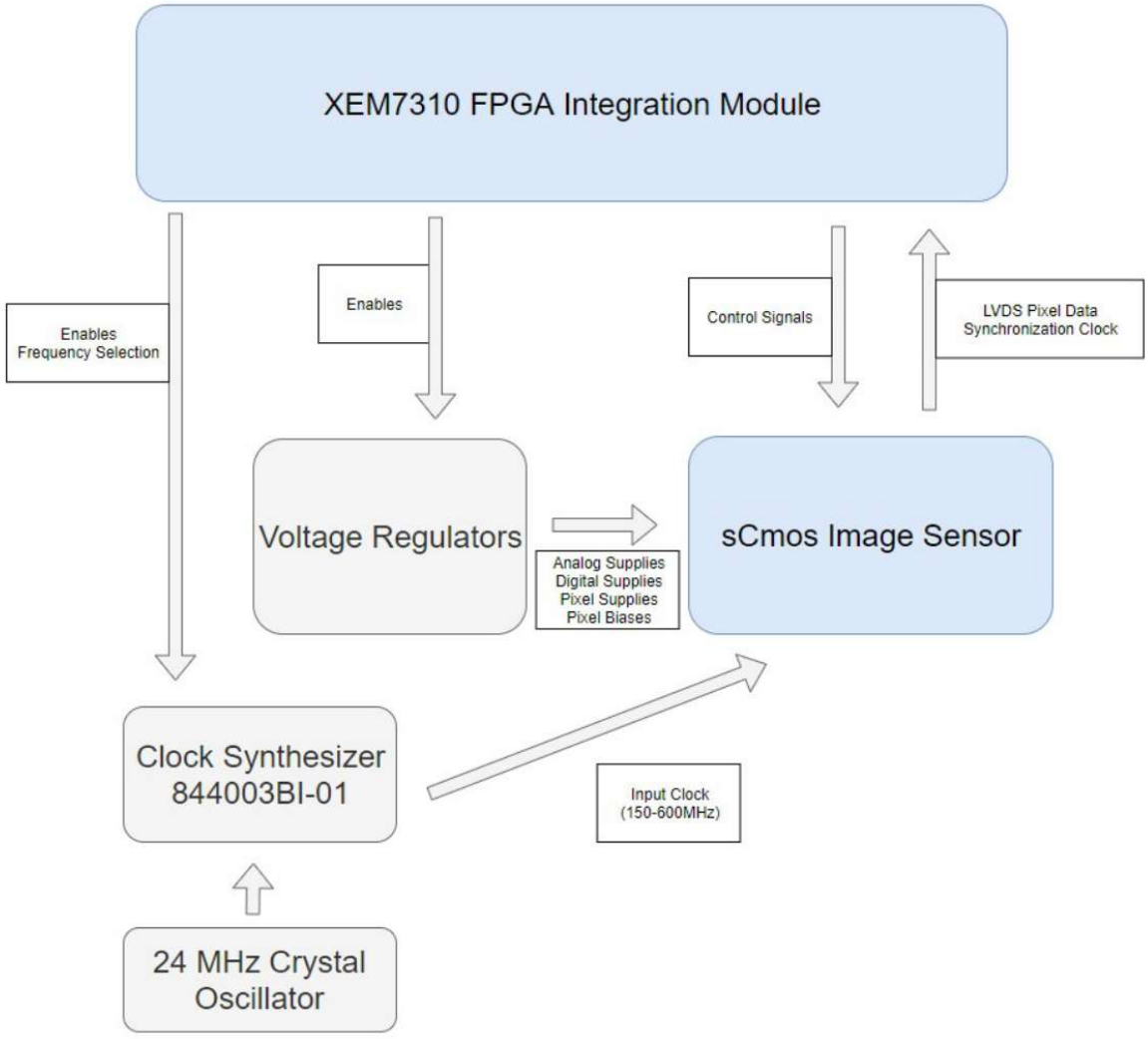


Figure 13: PCB-level block diagram of the camera. The parts shown in this diagram are the ones directly related to the image sensor data acquisition.

3.5 FPGA: System Overview

The FPGA handles three overall functions: the power-on sequence (figure 14), the image sensor SPI communication, and the image acquisition. Verilog code was written in Vivado for programming the FPGA.

The FPGA controls the enables of the voltage regulators and the enables/inputs of the clock generator chip. The inputs to the clock generator chip allow the FPGA to specify the image sensor's clock frequency in the range of 150 MHz to 600 MHz. A higher frequency allows for greater resolution in exposure time and a higher frame rate. Using a lower clock frequency allows the camera to consume less power. These signals are received from the computer program. By default, all the regulators are off because of pull-down resistors. This is to prevent damage to the parts before the FPGA bit file is loaded.

The FPGA controls the image sensor SPI communication. SPI is a serial communication protocol that typically involves a master-to-slave data line, a slave-to-master data line, a clock line, and enable signal(s). The typical rule for SPI is: Data lines are written on the falling edge of the clock, and the data should be read on the rising edge of the clock because devices will hold a steady value through the rising edge. The image sensor's SPI registers are composed of a single 256-bit sequence. These 256 bits represent various values, with their positions specified by the datasheet. The 256-bit sequence is sent from the computer program to the FPGA. When the computer program sends the start signal, the FPGA executes the SPI communication sequence. It writes all 256 bits to the image sensor, reads them back for verification, and reads out the image sensor's internal temperature sensor. The SPI communication for this image sensor allows the FPGA to transfer 256 bits of information using 4 PCB traces. The image sensor's SPI registers control

important values such as ADC resolution (12-bit or 11-bit), LVDS training pattern, ADC offset, and other operating mode setting.

The FPGA handles movement of images from the image sensor to the FPGA, and from the FPGA to the computer. Eight (8) LVDS channels are used to transfer a stream of pixels from the image sensor chip to the FPGA, where it is stored in a FIFO. The FIFO is a module used for storing data, and can be written and read at the same time. While this occurs, an Opal Kelly module moves the image data from the FPGA to the computer, using the USB 3.0 interface.

Before the pixel data can be put into the FIFO, the FPGA also must handle LVDS training, control signal timing, and data deserialization.

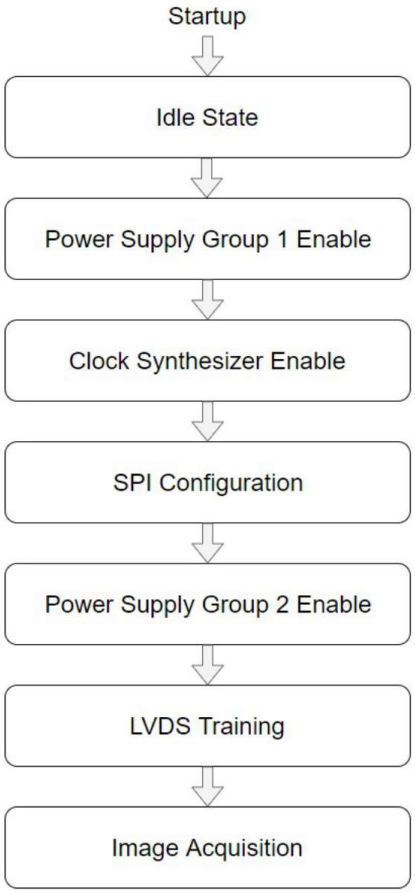


Figure 14: State diagram of the image sensor power-on sequence.

3.6 FPGA: LVDS Training

LVDS training is necessary because the time delays between the synchronization clock and the data lines are unknown. The delay is created by (1) exiting the image sensor, (2) differing trace routing lengths on the PCB, shown in figure 15, (3) differing parasitic capacitances and resistances for each LVDS trace on the PCB (clock included), (4) differing routing lengths on the Opal Kelly FPGA integration module, and (5) differences in routing through the FPGA itself.

The image sensor has a LVDS training mode to make obtaining valid data possible. Figure 16 shows the image sensor outputs when in LVDS training mode. Instead of pixel data, the image sensor sends a training pattern. This training pattern is set during the SPI programming.

LVDS training for image acquisition has two important requirements. Requirement A: it is necessary to avoid bit-error when reading from the LVDS pairs. The sampling of the LVDS data lines must occur at stable points of the signal, to prevent bit error. LVDS training mode A (figure 16, Top) is used here. Requirement B: it is necessary for the locations of pixel-zero and bit-zero to be known. If this is not achieved, the pixels will potentially be bit-shifted and/or shifted in location. This would result in inconsistent and distorted images. LVDS training mode B (figure 16, Bottom) is used here. Figure 17 is a state diagram of the LVDS training process.

The following paragraphs are associated with requirement A of the LVDS training. The LVDS data lines have phase offsets from the synchronization clock. For each data line, this phase offset is important to make sure the data line is read at the stable points, and not at the unstable points. Reading the data line at a stable point means a zero will be read as a zero, and a one will be read as a one. Reading the data line at a metastable point means a zero has a chance of being interpreted correctly as a zero, but also a chance of being interpreted incorrectly as a one. The same issue exists for reading a one correctly as a one, or incorrectly as a zero.

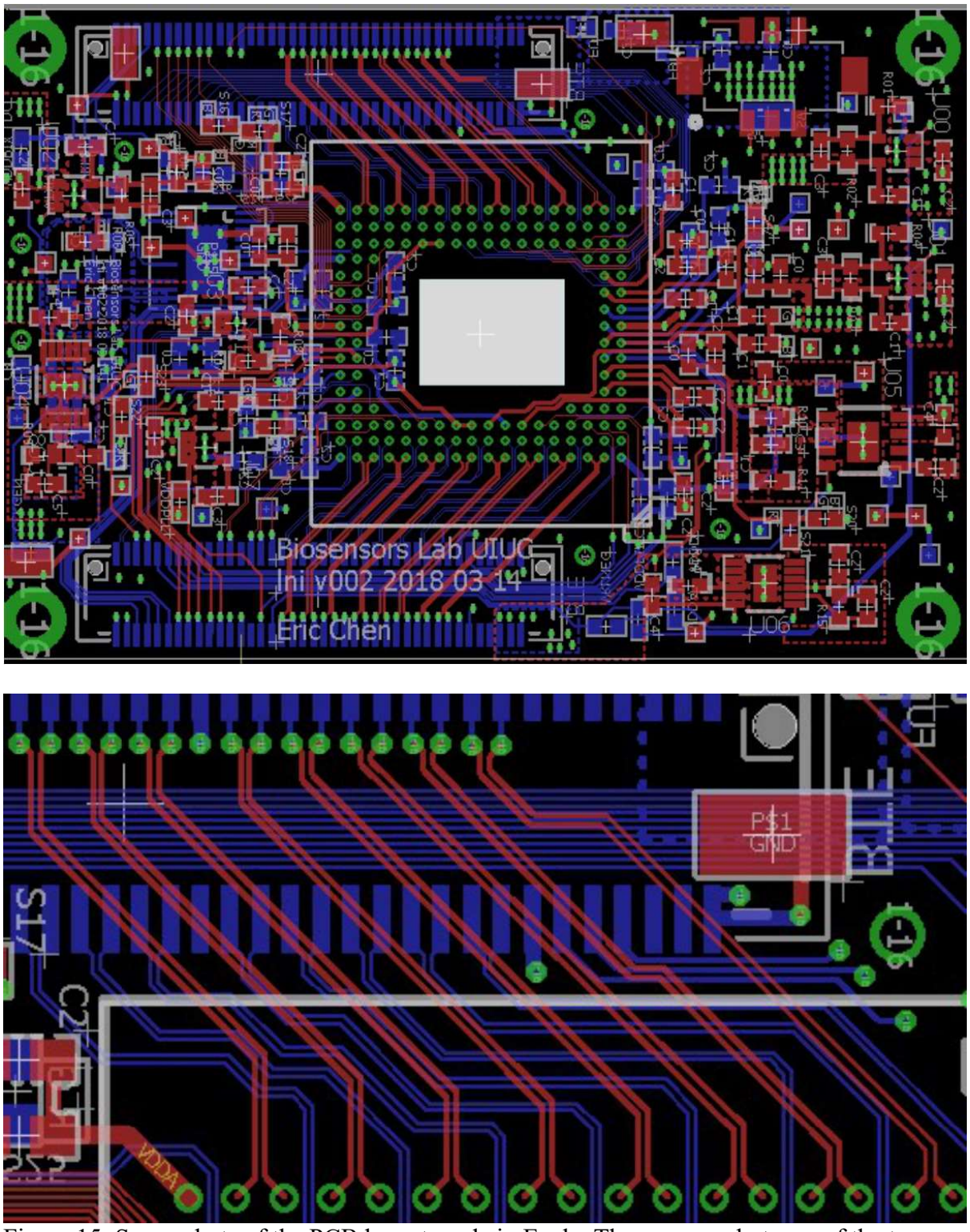


Figure 15: Screenshots of the PCB layout made in Eagle. These screenshots are of the top PCB, which holds the image sensor. Top: Entire PCB. Bottom: Zoomed in view of the LVDS traces. For each pair, the N and P traces match each other. The individual channels vary in length, which is part of the reason that LVDS training is required.

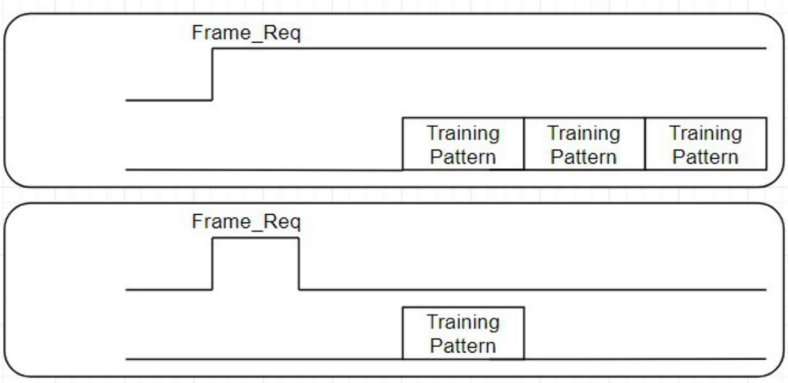


Figure 16: LVDS training modes of the image sensor. Top: LVDS Training mode A, continuous mode. When the frame request is held high, the image sensor will repeatedly send the training pattern. Bottom: LVDS Training mode B, single pulse mode. When one clock cycle of frame request is sent, the image sensor will send back one repetition of the training pattern.

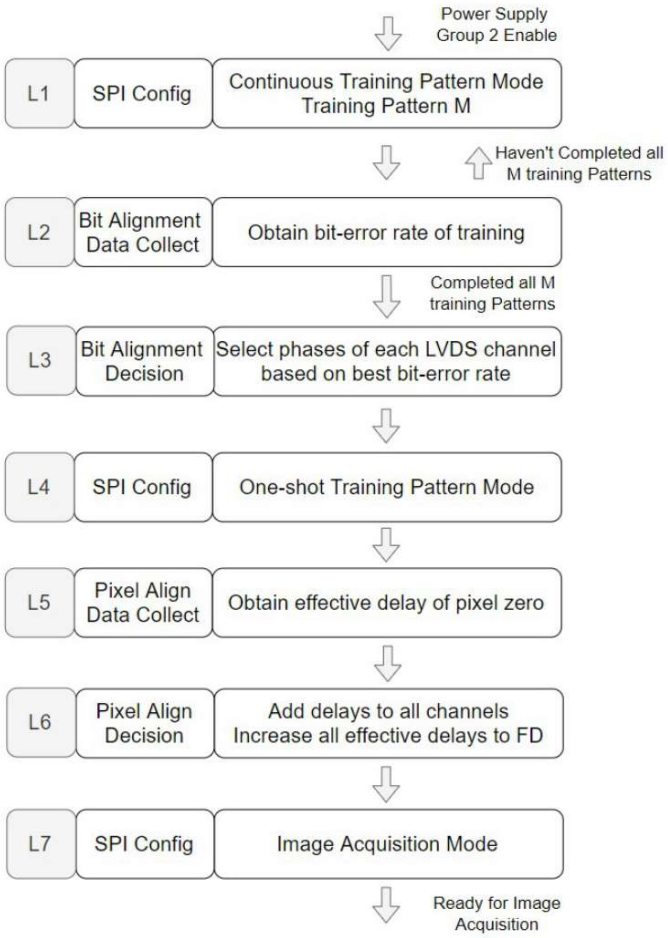


Figure 17: State diagram of LVDS training procedure.

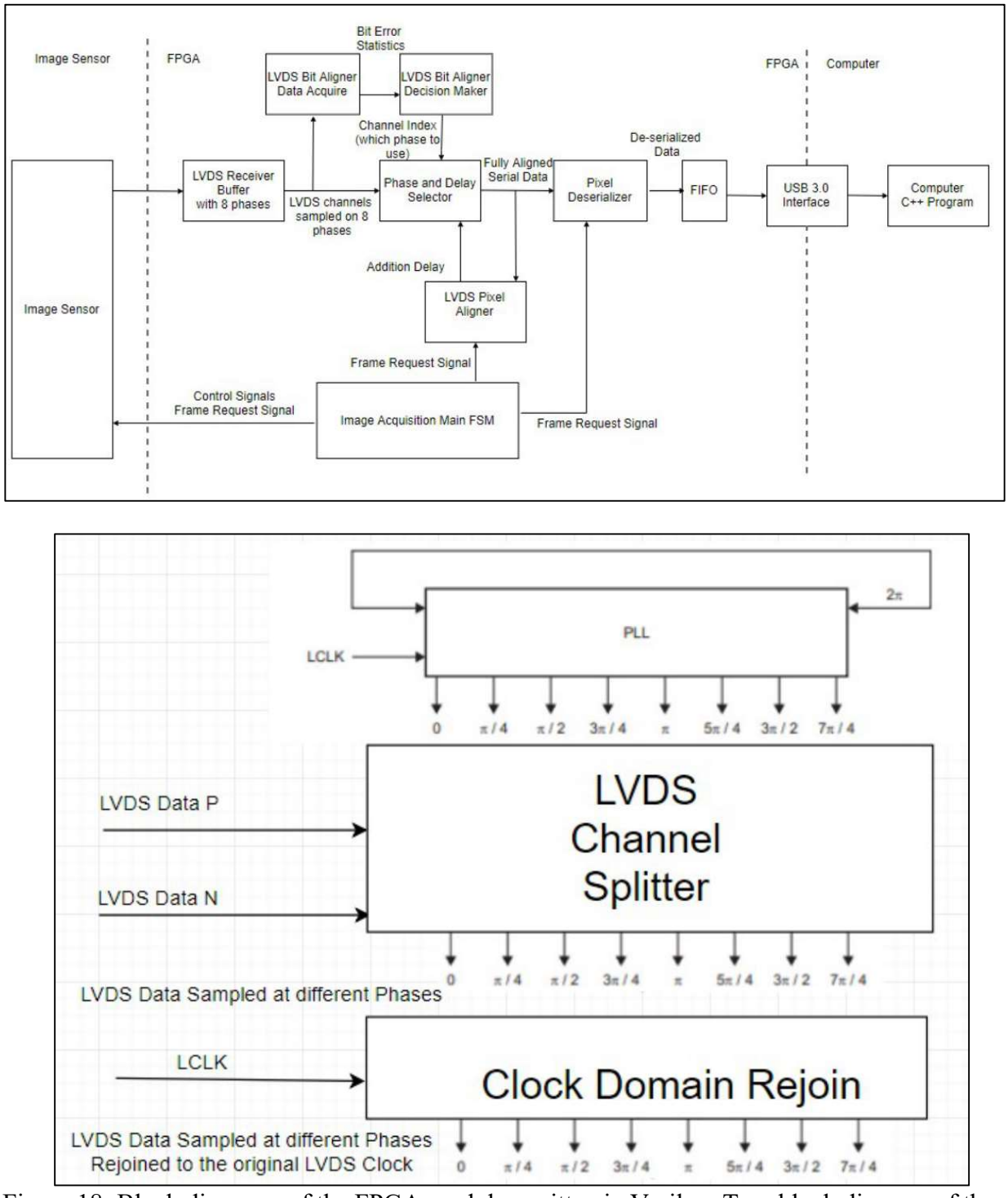


Figure 18: Block diagrams of the FPGA modules written in Verilog. Top: block diagram of the modules used for acquiring LVDS pixel data. Shows relations to the adjacent systems (computer and image sensor). Bottom: contents of LVDS Receiver Buffer with 8 phases.

The solution to requirement A is to sample the data line at various phase offsets with respect to the synchronization clock, and then test the bit error rate of each. The one with the lowest bit error rate is selected. Ideally, no bit error rate exists in the selected phase delay. Figure 18 shows the module being used for this purpose. Each data line is sampled at phase shifts of 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315° offset from the rising edge of the synchronization clock. In figure 18, this is labeled as the "LVDS Receiver Buffer with 8 phases".

The following module (figure 18, labeled as LVDS Bit Aligner Data Acquire) receives these 8 channels. Bit aligning means taking the unaligned clock, and making sure the data lines are sampled on the stable parts. The LVDS Bit Aligner Data Acquire module is composed of shift register and a counter. The image sensor is programmed into this mode and the frame request signal is held high. The LVDS channels repeatedly send the training pattern. In figure 19, a repeating 6-bit training pattern is shown. The LVDS Bit Aligner Data Acquire module checks every rising edge of the synchronization clock if the shift register matches the training pattern. If there is no bit error, the shift register will match every sixth clock cycle. To obtain decent statistics, this module (figure 18, LVDS Bit Aligner Data Acquire) runs through 1020 clock cycles. After 1020 clock cycles, exactly 170 (1020 ÷ 6) times the shift register should have matched the training pattern. Otherwise, the number of matches can be used to calculate the bit-error rate.

The output of the LVDS Bit Aligner Data Acquire module is stored for several unique training patterns. In this system, eight different training patterns were used. More training patterns could have been used to obtain even better statistics. After finishing all bit alignment data, the following is stored in the LVDS Bit Align Decision Maker module: for each LVDS channel, for each phase delay, and for each training pattern, the number of training pattern matches out of 170

is known. The decision maker will decide which phase delay should be used for each LVDS channel. The phase delay used for each LVDS channel is decided by which one provides either no bit-error or the lowest bit-error. Better statistics could be obtained by increasing the sample size from 1020 to a larger number and using more training patterns. The choice of 1020 clock cycles was to limit the counter to a 10-bit number, but this could easily be increased by many orders of magnitude without much additional resource consumption.

In the current implementation of this camera, the training process is run on every camera startup, taking 2 seconds. The phase delay should be relatively consistent between camera startups. To reduce the startup time, the phases for each channel could be recorded from a single run and loaded from a configuration file.

After the correct phase of each LVDS channel is determined, the index number of the phase is sent to the Phase and Delay Selector module (shown in figure 18). This module acts as a multiplexor, passing the correct phase channel to the output. This output moves on to the pixel deserializer.

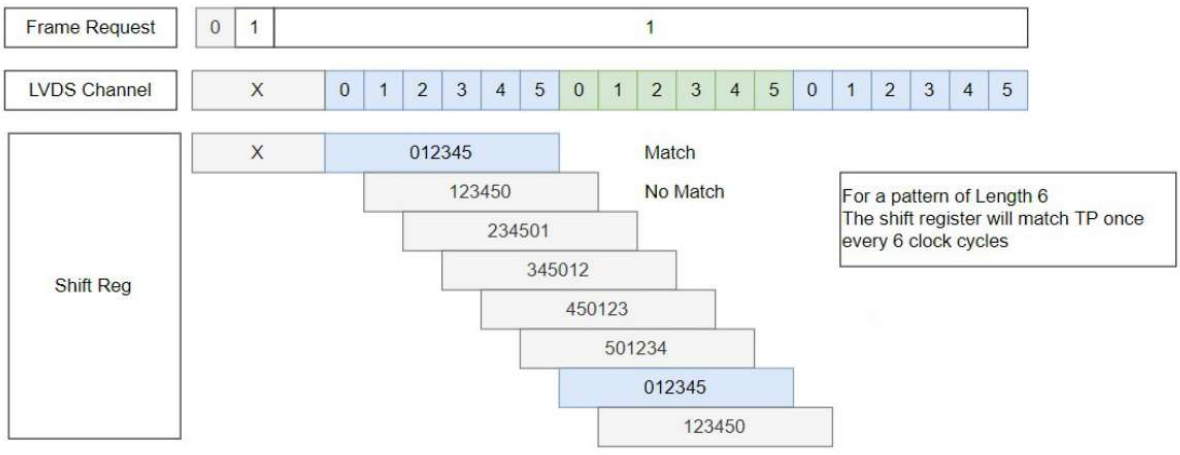


Figure 19: Timing diagram of LVDS bit alignment, acquiring the bit-error information using continuous training mode.

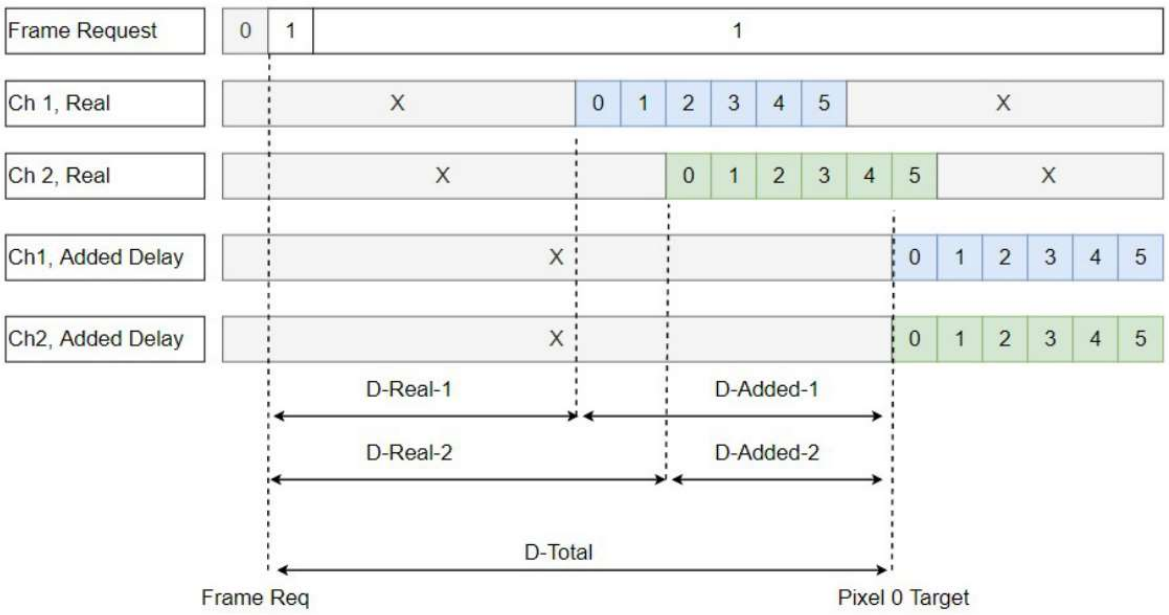


Figure 20: Timing diagram of the LVDS pixel alignment. Two channels are shown, with differing delays. “Added Delay” shows their timings after the compensation delay is added, to move their pixel zero’s to the pixel zero target.

Requirement B of the LVDS training is the pixel alignment. The goal of this step is to make all LVDS channels have their bit-zero pixel-zero occur at the same clock cycle. This is done by adding an additional delay to each LVDS channel, so their bit-zero pixel-zero all occur at the same clock edge. This process is shown in figure 20. Initially, pixel 0 of channel 1 and channel 2 are not aligned. The goal is to make pixel 0 of both channels occur at “Pixel 0 target”.

In figure 17, Phase and Delay Selector is capable of adding an arbitrary delay to each LVDS channel. Ch 1 Real and Ch 2 Real in figure 20 would be the input to the Phase and Delay Selector of figure 17. The output would be Ch 1, Added Delay and Ch2, Added Delay. As shown in the diagram, these channels are both aligned to the same point in time.

In this pixel alignment process, the image sensor is used in a one-shot training pattern mode. The frame request signal is pulsed high for a single clock cycle. After a fixed delay, each channel sends the training pattern once, which is adjacent to where the pixel 0 would be. To

determine the required delay of each channel, the outputs of Phase and Delay Selector go to the LVDS Pixel Aligner module. This module (figure 18, LVDS Pixel Aligner) receives the frame request signal, and starts a counter. The LVDS channels are stored into shift registers. When this shift register matches the training pattern, the count is held. For each channel, the delay before the training pattern appears is known. In figure 20, this is labeled as "D-Real-1". For example, the training pattern appears 15 clock cycles after the frame request, for channel 1. Meanwhile the training pattern appears 16 clock cycles after the frame request. The module then calculates how much additional delay should be given to each channel, for all channels to have pixel 0 appear at the pixel 0 target. For example, pixel 0 target is 32. Channel 1 is given an additional delay of 17, for a total of 32 clock cycles. Channel 2 is given an additional delay of 16, for a total delay of 32 clock cycles.

After this process has completed, all pixel-0's of each channel are aligned and the pixel-0 coordinate is known. The following module knows all channels will have pixel-0 on clock cycle 32.

3.7 FPGA: Modules for Pixel Deserializing and Control Signal Inputs

In figure 17, the pixel deserializer accumulates a full pixel from each channel and pushes it into the FIFO. There are eight LVDS channels with 12-bit pixels, for a total of 96 bits. This is written into a 128-bit wide input FIFO. There is some inefficiency in this FIFO, since every write only uses 96 out of 128 bits. The output of this FIFO is 32 bits wide to be compatible with the USB 3.0 interface provided by Opal Kelly. This USB 3.0 interface reads data from the FIFO and sends it to the computer.

The image acquisition involves various control signals, sent from the FPGA to the image sensor. This is handled by the Image Acquisition Main FSM module in figure 17. These signals run on a 3360 clock cycle loop. A state machine counts from 0 to 3359, resetting to 0 when it reaches 3360. When this counter passes certain thresholds, the control inputs are changed. Verilog code defining these thresholds were generated from the specified signal timing diagram. A lookup table defines when each signal should transition from high to low, or low to high. For example, the frame request signal will be held low until the last count. At the last count of the loop, it will be held high for one clock cycle. Afterwards, the frame request is returned to low.

Included in these control signals is the read and reset address. In every loop of 3360, one row is read and one row is reset. This is the method for controlling the exposure time of each row. As an example, each loop takes 44 microseconds. In this example, row 0 is reset in the loop 0 and read in loop 55. Therefore, the exposure time will be 44 microseconds times 55 loops, which is 2.42 milliseconds. The exact loop for resetting and reading each row is decided by the Image acquisition Main FSM in figure 17, calculated by what exposure time is desired by the user. The state machine staggers the resets and reads of the rows, for operation in rolling shutter mode. Rolling shutter mode means the state machine will sequentially run through resetting each row, and sequentially read out each row when its exposure time is reached.

3.8 Computer Program: Camera Power-on and Startup Sequence

The computer program acts as the master of the camera system. Figure 10 shows the relation between each subsystem. The computer program sends settings and variables to the FPGA. The FPGA sends the image data to the computer program. The FPGA sends control signals to the image sensor. The image sensor sends the pixel data to the FPGA, which then sends the data to the

computer over a USB 3.0 interface. This program is written in C++, in Microsoft Visual Studios. Standard arrays are used for image processing. Libraries from OpenCV are used for real-time video display.

The computer program handles user inputs, processing data, saving data, and displaying real-time video. The computer decides important camera operation settings, such as the exposure time and the operating clock frequency.

The raw image data also must be processed before being displayed as an image or video. The FPGA sends the pixels to the computer in the same order that the image sensor sends them to the FPGA. As a result, the pixels are slightly scrambled. The computer program sorts the pixels into the proper positions. For each row sent, for a row size of 1024, the pixels arrive in the following order (in terms of horizontal position): 0, 256, 512, 768, 1, 257, 513, 769, 2 and so on. This sorting could have been implemented on the FPGA, with pixels being sent to the computer being in the correct order. The main trade-off considered here is the usage of either FPGA resources or computer resources. This project has the computer handle this processing, because it is simply an additional operation to be handled by a thread. There is no additional latency created by having this sorting operation occur in the computer.

The startup sequence of the camera is handled by the C++ program. The C++ program leads the FPGA through the sequence shown in figure 14. The C++ program is used to load the FPGA bit file to the XEM7310 Opal Kelly FPGA integration module. This is done over a USB 3.0 interface. Error messages are used to determine if the computer has properly connected to the FPGA.

After proper loading of the FPGA bit file, the computer program proceeds to execute the power-on sequence for the sCMOS image sensor. This is done by setting registers which are routed

to FPGA output pins, which are routed to the enables of the voltage regulators. This task is done using the computer's sleep function, which allows it to sleep for a specified number of milliseconds. Although this method is not precise, there is no issue created because the startup sequence is not highly constrained with its timing. The sCMOS image sensor datasheet specifies wait times of "at least X milli/micro seconds". During this power-on sequence, the clock synthesizer is also enabled.

The computer sends a SPI start signal to the FPGA, ordering it to start the SPI sequence. SPI is a serial communication protocol. In this camera system, it is used for programming the settings of the sCMOS image sensor. There are registers in this sCMOS image sensor, totaling to 256 bits of data. These include information such as the LVDS training pattern, LVDS training mode select, ADC bit resolution (11 bit or 12 bit), and other operating mode settings. To allow for the most flexibility possible in camera operation, the 256 bit sequence sent from the FPGA to the image sensor is set by the computer program. The 256 bit sequence is sent from the computer to the FPGA, which is then transferred to the image sensor after the start signal is also sent.

The computer program is involved in the FPGA LVDS training sequence. The LVDS training sequence involves gathering data on the bit error rate for each LVDS channel. Each LVDS channel is tested for 8 different phase offsets from the original clock. Each of these is tested for 8 unique training patterns. The computer program decides which phase delay each channel should use. The computer program will select the phase delay with either no bit error, or the least bit error. In the case where multiple phase delays all have no bit error, the program picks which phase delay should be used.

3.9 Computer Program: Functions for Continuous Operation Mode

The previous section described the startup sequence. This section will describe the continuous operation part of the program. These threads split the various tasks to allow them to run in parallel. The computer program threads are shown in figure 21. Following are the general descriptions of the threads. Thread I (tI) handles user inputs, such as button presses and inputs from the graphical user interface (GUI). Thread A (tA) handles all Opal Kelly related operations, meaning the USB 3.0 interface. Thread A includes setting the exposure, and retrieving raw image data from the FPGA. Thread B (tB) handles processing of the data. Thread C (tC) handles saving the processed data into h5 files, for later processing and creation of video files. Thread D (tD) handles real-time video display.

Thread A (tA) handles all Opal Kelly related operations. Thread I sends the exposure value to thread A. Thread A sends this exposure value to the FPGA in between frames. Thread A also uses functions from Opal Kelly's API to retrieve image data from the FPGA. The image data streams into a FIFO on the FPGA. After a full frame is retrieved, Thread A updates an index variable which is shared between thread A and thread B. When thread B sees this index variable has changed, it knows there is a new frame to be processed.

Thread B handles processing of image data. The image data arrives to the computer program in a scrambled format. The pixels of each row are ordered: 0, 256, 512, 768, 1, 257, and so on. The first part of thread B sorts each row into the proper order of: 0, 1, 2, 3, 4 and so on. Thread B must output two different images. The first image has datatype unsigned short (16-bit size), and is sent to thread C to be saved. This allows the full 12-bit raw data to be saved for more complicated processing. The second image is type unsigned char (8 bit size), which is sent to thread D to be displayed. The real-time video only needs an 8 bit resolution. The actual process is as

follows. Thread B sorts the data and places it into an array of unsigned shorts. Another image is created by scaling this image down to an unsigned char, additionally scaling it by another factor that is decided by the GUI. The GUI sets a max value and a minimum value, so the image can be displayed using the full range. The formula for the image sent to thread D is the following:

pixelVal_uchar = (pixelVal_ushort - guiMin) *256 / (guiMax-guiMin)

Thread D displays the data in real-time video. Before being displayed, the image data is interpolated. The simplest algorithm for interpolation is used here. In the typical RGB Bayer filter camera, the pixels use the average of their nearest neighbors. A blue pixel already has the blue value. To get the red value, it takes the average of the closest red pixels. To get the green value, it takes the average of the closest green pixels. This camera is similar, except for being RGB-NIR instead of RGB. In every 2 × 2 superpixel, this camera has one red pixel, one green pixel, one blue pixel, and one NIR pixel. A typical RGB Bayer filter has one red pixel, two green pixels, and one blue pixel. This does not change the main interpolation principle of average of the nearest neighbor pixels.

Thread C does not interpolate the data before saving it into h5 files. This is to preserve the raw numbers of each physical pixel. The raw numbers must be preserved to calculate characteristics of the camera. This includes quantum efficiency, temporal noise, fixed pattern noise, and other pixel-by-pixel statistics. Thread C is actually composed of two threads. Thread tC accumulates sets of frames, and tHC handles h5 file creation. This is to avoid stalling and data loss when saving. The computer program accumulates sets of 64 frames. Whenever a set of 64 frames is reached, it is saved to an h5 file. However, saving an h5 file takes more than one frame of time. To allow the next set to begin immediately, the old 64 frame set is passed to another thread for the h5 saving. As a result, the saved data does not skip frames.

The user interface has two general specifications: minimal user attention and maximum intuitiveness. These are important because the camera is designed to be used alongside clinical studies. The ease of use is important because the camera must disturb the typical workflow of the clinical study. The user interface has sliders in a GUI, but also involves single-button controls. There are key presses for increasing/decreasing exposure, toggling the saving/recording of video data, and switching between display modes.

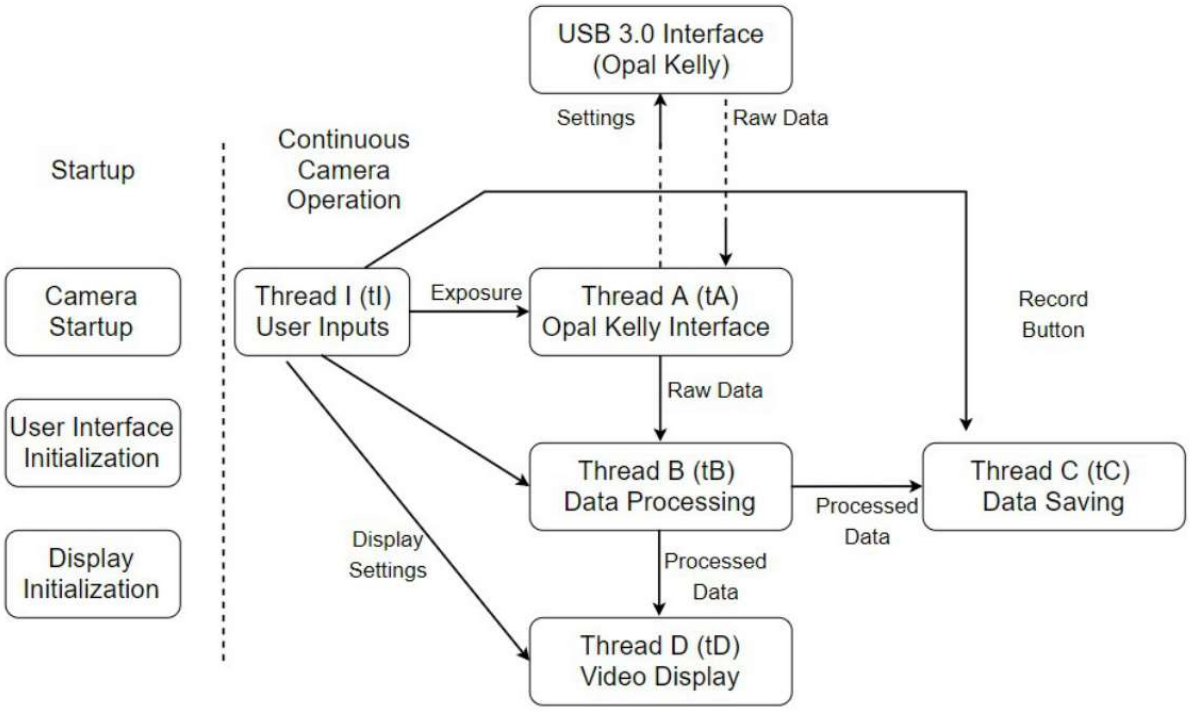


Figure 21: Block diagram of the computer program written in C++.

CHAPTER 4: CONCLUSION AND FUTURE WORK

4.1 Conclusion

NIR fluorescence imaging involves exciting fluorophores with NIR light and collecting images/videos of the NIR emissions. The properties of NIR light allow imaging of features that are below the outer layer of skin. NIR light is absorbed and scattered less than visible light [1]. NIR light is also safe for human exposure. Indocyanine green (ICG) is an FDA approved dye; its ability to bind to albumin provides high contrast in imaging tumors and sentinel lymph nodes.

This project's camera was designed with the following goals: low readout noise, low weight and high portability, and capability of simultaneously imaging NIR and RGB light. The system involves a low noise sCMOS image sensor chip, an Opal Kelly XEM-7310 FPGA integration module, and a computer. The sCMOS image sensor was purchased from an external company for its low noise and high sensitivity. The FPGA integration module is used for image sensor control input timing, LVDS communication for retrieving pixel data, and transfer of video data to the computer by USB 3.0 interface. Verilog code was written to handle these. A PCB was designed, manufactured, and tested for connecting the FPGA to the sCMOS image sensor. The computer processes and sorts the raw image sensor data, displays real-time video, handles user inputs, and saves video data to h5 files.

4.2 Future Work

While the goals of the project were achieved, they could be further improved in the following ways.

1. The sCMOS image sensor purchased has specifications of 1 electron readout noise. This could be achieved with better design of the PCB and auxiliary components, related

to noise in pixel biases and power supplies. Active cooling could be added to the camera to reduce dark current. The camera currently operates at room temperature.

2. Optimization could be made to the mechanical case design. The camera case could be made more compact.

3. The camera can see both NIR and RGB light. However, there is significant crosstalk between color channels. The NIR channel is primarily sensitive to NIR light, but it has significant sensitivity to red, green, and blue light as well. Normal surgical lighting (RGB) creates a large background signal in the NIR channel. This creates a problem with shot noise, which is proportional to the square root of the total signal. Reducing this crosstalk would reduce the shot noise in the context of in-vivo imaging.

REFERENCES

[1] R. Acharya, R. Wasserman, J. Stevens and C. Hinojosa, "Biomedical imaging modalities: A tutorial," *Computerized Medical Imaging and Graphics,* 1995.

[2] G. Hong, A. L. Antaris and H. Dai, "Near-infrered fluorophores for biomedical imaging," *Nature Biomedical Engineering,* 2017.

[3] D. P. Schaap, G. A. Nieuwenhuijzen and M. D. Luyer, "The use of near-infrared fluorescence imaging in the surgical treatment of esophageal cancer," *Journal of Thoracic Disease,* 2017.

[4] S. Yoneya, T. Saito, Y. Komatsu, I. Koyama, K. Takahashi and J. Duvoll-Young, "Binding properties of indocyanine green in human blood," *Investigative Ophthalmology & Visual Science,* 1998.

[5] A. K. Lyer, G. Khaled, J. Fang and H. Maeda, "Exploiting the enhanced permeability and retention effect for tumor targeting," *Drug Discovery Today,* 2006.

[6] C. Hirche, D. Murawa, Z. Mohr, S. Kneif and M. Hunerbein, "ICG fluorescence-guided sentinel node biopsy for axillary nodal staging in breast cancer," *Breast Cancer Research and Treatment,* 2010.

[7] A. V. D'Souza, H. Lin, E. R. Henderson, K. S. Samkoe and B. W. Pogue, "Review of fluorescence guided surgery systems: Identification of key performance capabilities beyond indocyanine green imaging," *Jounal of Biomedical Optics,* 2016.

[8] M. Garcia, C. Edmiston, T. York, R. Marinov, S. Mondal, N. Zhu, G. P. Sudlow, W. J. Akers, J. Gargenthaler, S. Achilefu, R. Liang, M. A. Zayed, M. Y. Pepino and V. Gruev, "Bio-inspired imager improves sensitivity in near-infrared fluorescence image-guided surgery," *Optica,* 2018.