Spring 5-15-2014

Real-time Image Processing on an FPGA for an Intraoperative Goggle Device

Yiyi Zhang
Washington University in St. Louis

Washington University in St. Louis

School of Engineering and Applied Science

Department of Electrical and Systems Engineering

Thesis Examination Committee:
Viktor Gruev, Chair
Humberto Gonzalez
Hiro Mukai

Real-time Image Processing on an FPGA for an Intraoperative Goggle Device

by

Yiyi Zhang, B.S.

A thesis presented to the School of Engineering and Applied Science
of Washington University in partial fulfillment of the
requirements for the degree of

Master of Science

May 2014
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

Special thanks to Dr. Gruev and to the students in the research group.

Yiyi Zhang

Washington University in Saint Louis
May 2014

Dedicated to my parents and my wife.

ABSTRACT OF THE THESIS

Real-time Image Processing on an FPGA for an Intraoperative Goggle Device

by

Yiyi Zhang

Master of Science in Electrical Engineering

Washington University in St. Louis, May 2014

Research Advisor: Professor Viktor Gruev

The emergence of near-infrared dyes for fluorescence imaging has had a tremendous impact in the medical field. In particular, indocyanin green (ICG) has been widely used for assessing tumor margins during intraoperative procedures. Typically, the dye is intravenously injected into the patient, and after 24 hours the dye is removed from the patients body, except where binding between the dye and tumor cells has occurred. This selective binding between ICG and cancerous tissue allows for easy and accurate detection of cancer margins as well as detection of metastasis throughout the patients body. In order to detect the binding sites, a near infrared light source at 780nm is used to excite the dye molecules, and emission is observed at 800nm. To take advantage of fluorescent imaging during intraoperative procedures, we have developed a goggle system equipped for real-time imaging in both visible and near infrared spectra. The goggle system has to be light, compact and able to perform real-time image processing to assist the physician for easy detection of tumor margins. To this end, I have developed Verilog code for a Spartan III FPGA that performs the following: 1) Real-time acquisition of both visible and near infrared spectrum images from a pair of CMOS

imaging detectors; 2) Real-time image processing for both near infrared and color images in order to enhance the captured information; and 3) Real-time display of fused visible-near infrared images in high definition (HDMI) format on a goggle device. The imaging device has been successfully used for intraoperative procedures during tumor resection in animal models.

Chapter 1

Motivation and System Overview

The chief motivation for the research project described in this thesis is the development of a compact, real-time, and dual spectrum (near infrared and visible) sensitive goggle system that will assist surgeons during an intraoperative procedure for cancer patients. The imaging system developed and described in this thesis combines standard off-the-shelf CMOS imaging devices that are sensitive to both the visible spectrum and the near infrared spectrum. The information from both imaging devices is combined by means of an FPGA device, where real-time data acquisition from the two imaging sensors and image processing is executed. The FPGA device is well suited for this implementation primarily due to two key factors: its small form which allows for compact implementation of the entire system, and real-time image processing due to parallel implementation of the image processing algorithm. The fused visible-NIR image is presented to the surgeon in real-time to assist in evaluating the tumor margins of the resected tissue as well as examining the spread of metastasis to other parts of the body. The entire imaging system has been tested in both lab and operating room settings, with great success.

1.1 Background and Motivation

Despite the numerous advances made in previous decades toward developing methods to treat cancer, surgery remains the primary treatment method for most solid tumors. To minimize lacerations at the surgical site and improve wound healing, minimally invasive techniques are gaining acceptance in most clinics. Yet, offline identification of tumors and tumor margins after preoperative computer tomography (CT), magnetic resonance imaging

(MRI), or positron emission tomography (PET), and the presence of positive tumor nodules not detectable by standard imaging methods, complicate tumor resection. For example, in breast-conserving reconstructive surgery (BCR), a lumpectomy is performed, where a small volume of breast tissue containing the tumor is surgically removed [2]. In comparison to the traditional surgical removal of one or both breasts (mastectomy), BCR features better cosmetic outcome, reduced wound infection risk, and less psychological impact [2] [3] [4]. However, studies have shown that 20-70% of patients have positive margins after BCR, where the presence of cancerous cells are found at or near the margins of surgically removed tissues [5] [6] [7]. These patients suffer from a higher cancer recurrence risk and need additional surgical excision [8] [9]. For invasive ductal carcinoma (IDC), the most common form of invasive breast cancer, the surgical outcome of BCR is even worse due to its poorly-defined tumor boundaries and its invasive nature, rendering mastectomy a more reliable procedure[10]. About 30% of IDC patients who receive BCR have cancer relapse. In addition, about 44% of patients with positive margins, 25% of patient with less than 2 mm margins, and 16% of patients with greater than 2 mm margins have recurrences[10]. A similar case can be made for other forms of cancer such as hepatocellular carcinoma, where positive nodules can be missed without methods that facilitate rapid scanning of the surgical site (see preliminary results section). These shortcomings attest to the need for reliable intraoperative imaging methods that can clearly differentiate tumors from surrounding tissues and offer surgical navigation[5] [8].

1.1.1 Image-guided intraoperative imaging systems

The primary goal of image guidance in the operating room is to provide a surgeon with accurate and real-time information about the location and boundaries of tumors. This will also allow the surgeon to explore alternative treatment plans based on the aggressiveness of the tumor. Some traditional imaging methods are currently used for surgical navigation, including fluoroscopy and intraoperative ultrasonography (IUS). However, X-ray-based modalities such as fluoroscopy expose patients and health professionals to radiation, and for soft tissue, a large amount of contrast agent is needed to visualize the tumors because the contrast agent detection sensitivity of the method is poor. IUS can be used for tumor detection, but it is primarily based on morphological differences between healthy and tumor tissues, leading to

significant false positive and negative rates[11] [12][13] [14]. More advanced instruments that mimic global positioning systems have been developed, where preoperative CT or MR images can be projected in 3D onto the appropriate anatomical structures. This system relies on priori images that are confined to the limitations of the imaging system, has unsatisfactory registration due to tissue deformation and motion artifacts, and does not provide additional information about lesions that are not detectable by the imaging modality. It also lacks the ability to interrogate surgical margins for the presence of tumors.

1.1.2 Optical imaging for intraoperative procedures

On the other hand, developments in biophotonics and semiconductor technologies within the last two decades have accelerated the emergence of optical imaging as a paradigm-shifting method to report the functional status of tumors. Optical imaging is able to detect biological events ranging from molecular and sub-cellular levels to the organ system. At the NIR region between 700 and 900 nm, absorption by intrinsic photoactive biomolecules is low, allowing light to penetrate several centimeters into tissues. [15] Unlike nuclear methods, optical imaging utilizes nonionizing radiation. This explains the recent surge in interest in applying the method in intraoperative procedures, where tumor boundaries, treatment response, and important physiological parameters can be monitored in real-time. Cost considerations also favor optical imaging technology as these systems are relatively cheap and their adoption does not require a steep learning curve. To date, various optical technologies such as fluorescent imaging have been applied intraoperatively to detect and assess tumor margins [16] [17] [18] [19] [20] [21] [22] [23] [24] [25]. However, these systems typically have complicated instrumentation and image analysis, and therefore need specialized or dedicated personnel to run the system. In addition, most of the systems require long times to process data and display results, which confines their use to offline applications. Finally, the current approach of displaying images on computer monitors may distract surgeons from focusing on the area of interest during surgery and compromise surgeons‘ coordination, thus affecting surgical outcomes.

To address the above mentioned challenges, there is a compelling need to develop an accurate, affordable, user-friendly, portable, and versatile intraoperative imaging system. Not only should it offer real-time imaging capability in a confined surgical suite, it should also be able

to bring remote collaborative efforts from geographically separated medical professionals to the bedside, if necessary.

To accomplish these goals, our research group has developed two generations of a prototype goggle-based device that is hands-free, affordable, battery-operated, compact, wireless, and wearable. The figure below illustrates the first generation of the goggle device.

Figure 1.1: First Generation Goggle System [1]

Preclinical and clinical testing of the first generation prototype revealed several shortcomings noted by surgeons and scientists that need to be addressed before the goggle system becomes a clinical tool. The major shortcoming is the weight of the device which is approximately 1 pound. Although the device is supported across the physician‘s skull, several physicians reported neck pains due to the weight. In addition, the device is not optimized for the correct near infrared wavelength leading to lower signal-to-noise ratio imaging.

The first generation goggle was used for imaging tumor resection in animal models. Examples are provided in Figure 1.2.

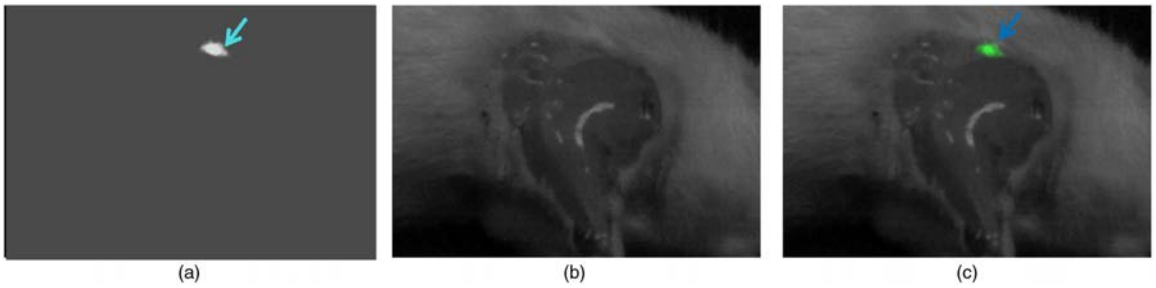


Figure 1.2: Results from Previous Generation Goggle System [1]

In Figure 1.2 (a), the raw data from the NIR sensor is provided. This image indicates the location of the tumor due to the fluorescence of the ICG molecules. In Figure 1.2 (b), the visible spectrum image is presented. This image does not contain information of the tumor location and provides a detailed visible spectrum image of the animal‘s extremities. Figure 1.2 (c) presents the final image, superimposing fluorescence and visible spectrum. This synthetic image provides visible clues for the surgeon of the tumor location as well as tumour margins, as they are superimposed on the visible spectrum image.

The NIR and visible spectrum image are taken at two different times by physically removing the optical NIR filter. During the recording of the two consecutive images, the camera cannot be moved with respect to the imaged tissue. If there is any movement between the two consecutive images, the superposition of the two images will be inaccurate and the location of the tumor will be compromised. This main drawback has motivated the design of the second generation of the goggle system.

1.2 Near Infrared and Visible Spectrum Goggle Imaging Device

In order to address the shortcomings of the first generation of goggle system described in the previous section, Prof. Gruev‘s lab has designed and fabricated a second generation of the goggle system which is composed of a pair of visible spectrum and near infrared optimized imaging sensors. Both sensors are based on an off-the-shelf imager provided by Aptina. One of the main reasons for choosing Aptina‘s sensors is the high quantum efficiency of these

sensors in the NIR band of 800nm. There are two ways to configure the two sensors in the system: 1) By using a beam splitter; 2) by using a single PCB board to hold both sensors. This section describes each configuration in detail, and then explains the advantages and disadvantages of different configurations. After that, the general background of the whole system will be presented.

1.2.1 Beam Splitter Configuration

The two imaging sensors are coupled together with a dichroic beam splitter optimized for NIR and visible spectrum imaging.

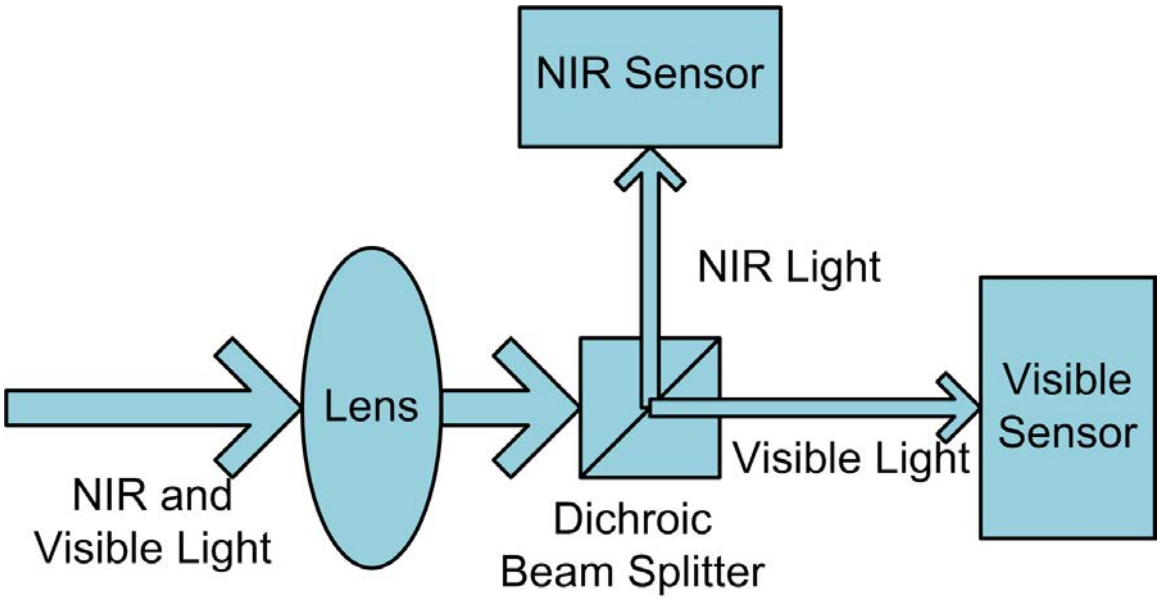
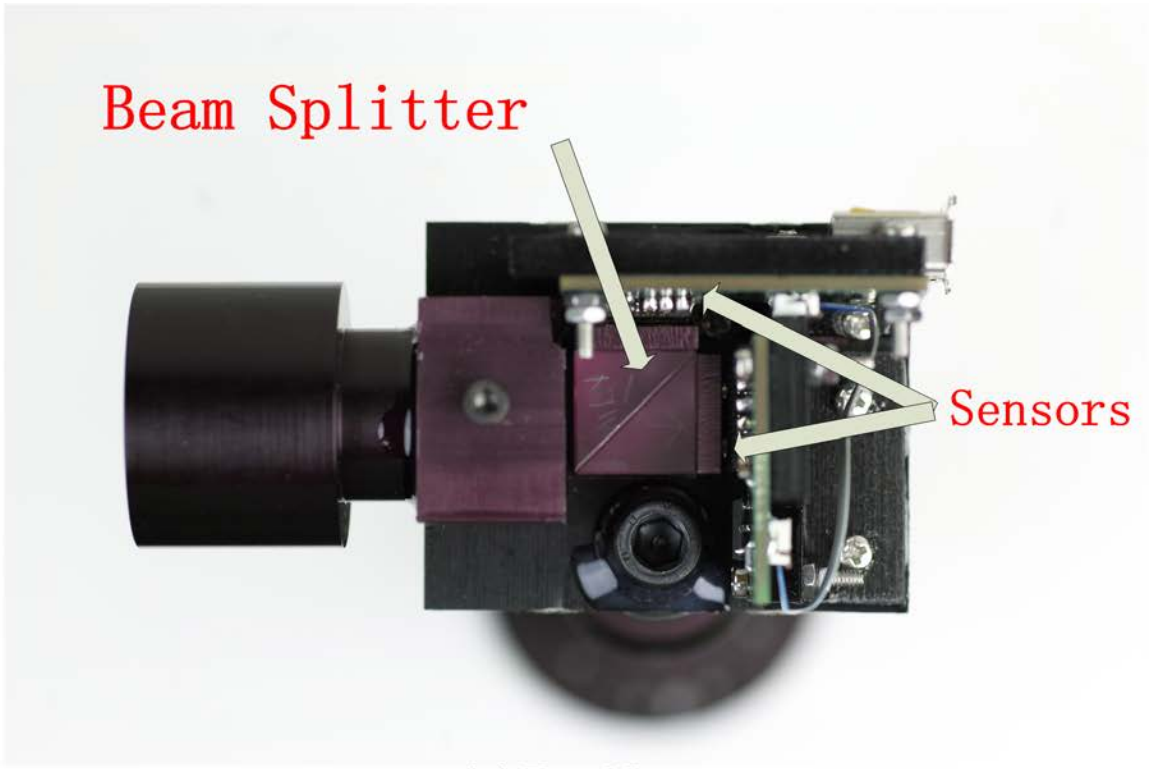
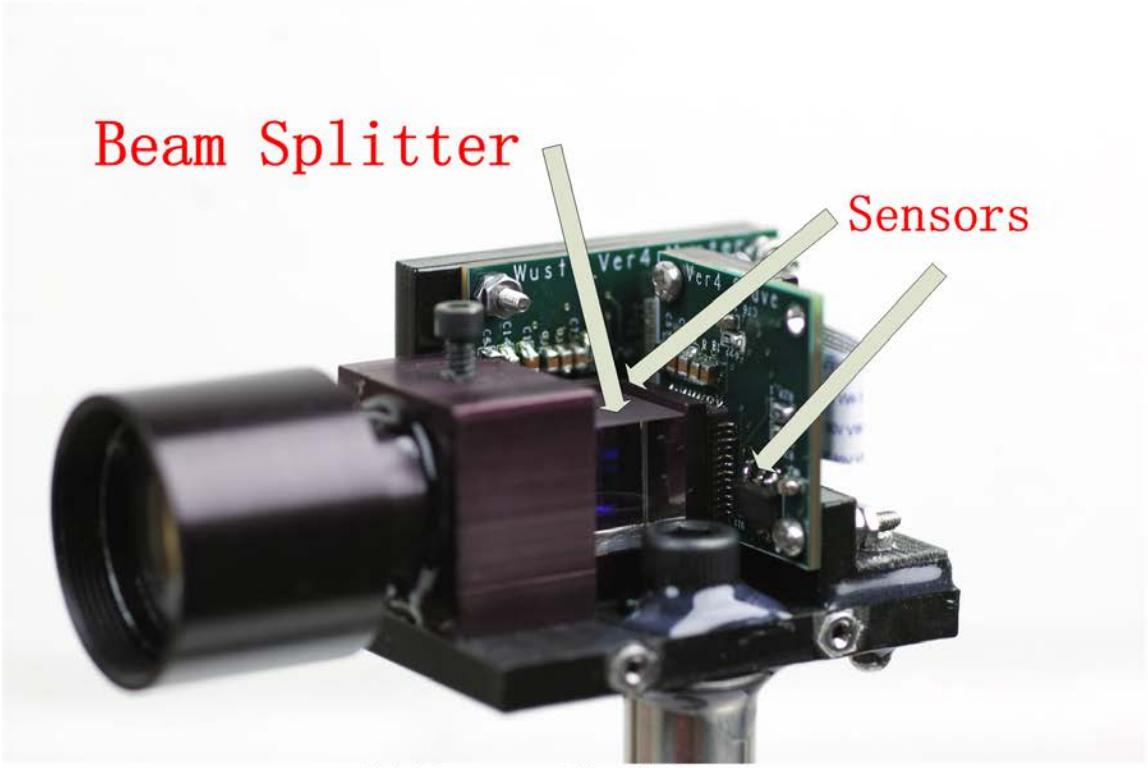


Figure 1.3: Block Diagram of the Dual Sensor System with Beam Splitter

As shown in Figure 1.3, the incoming light from a lens is projected onto the beam splitter where it is divided into two images. The first image is in the visible spectrum and passes light waves between 400nm and 700nm. The second image is optimized for the NIR band of 800nm and passes light waves between 800nm and 950nm. Figure 1.4 presents an image of the two imaging sensors coupled with a beam splitter and a lens.



(a) Top View



(b) Perspective View

Figure 1.4: The Dual Sensor System with Beam Splitter

1.2.2 Single PCB Board Configuration

Another way to configure the dual sensors is to mount them on a single PCB board, as shown in Figure 1.5 and Figure 1.6 below.

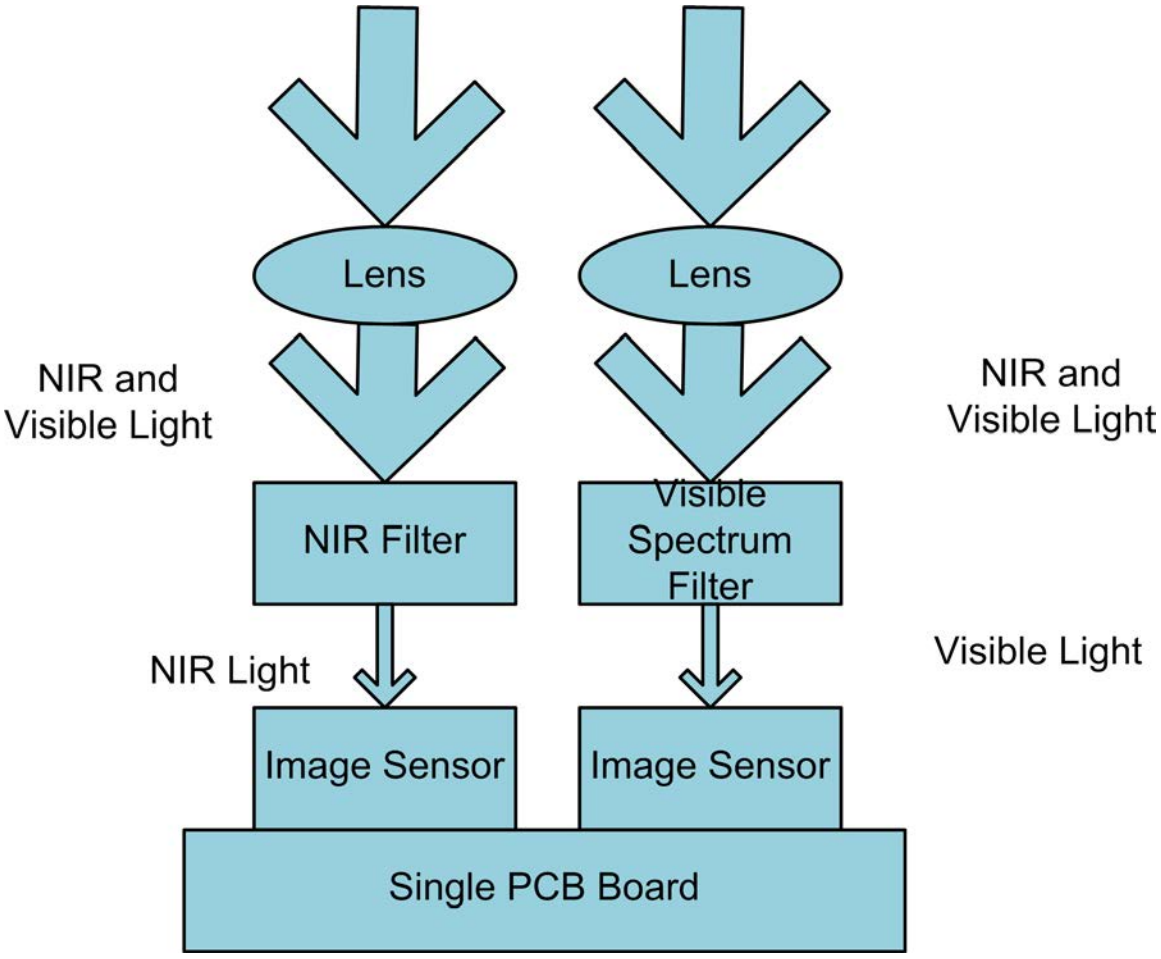


Figure 1.5: Block Diagram of Single PCB Board Configuration

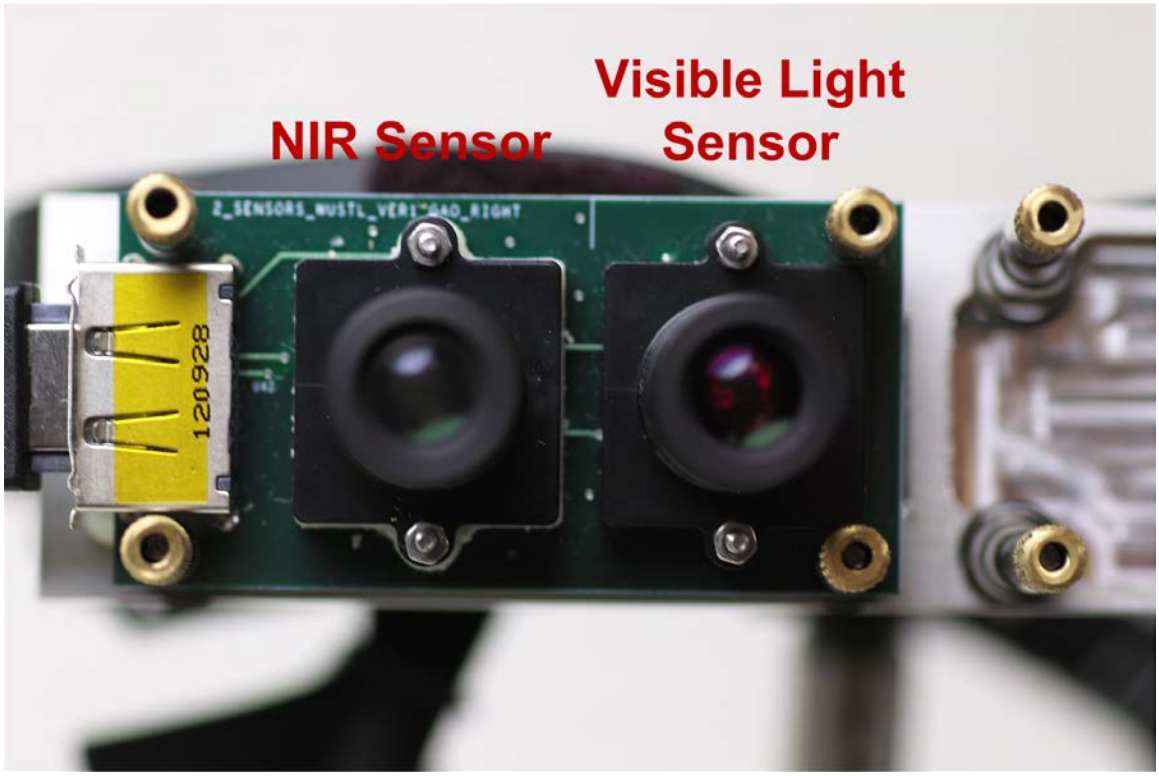


Figure 1.6: Single PCB Board Configuration

The two imaging sensors are housed on a custom PCB board next to each other. Comparing with beam-splitter version, signal to noise ratio (SNR) is better, because beam splitter configuration will decrease the intensity of light coming towards each sensor.

However, the disparity between the two sensors varies depending on the depth of the object. Changes in disparity will make precise superimposition more difficult. Since the disparity in beam splitter design remains constant across all depths, the superimposition will be easier. The details of disparity of this configuration will be presented in Chapter 3.

1.2.3 Goggle System

The data from the two imaging sensors is transmitted to a custom PCB board which is coupled with an FPGA board. The custom PCB board coupled with the off-the-shelf FPGA board is presented in Figure 1.7. The FPGA board is provided by OpalKelly and houses a

Spartan III FPGA with 32 MB of DDR2 memory. The two imaging sensors are configured in a master/slave configuration. This mode of operation allows the data from one of the sensors (slave imager) to be sent to the other sensor (master sensor). The data from both sensors is then combined in a one 16 bit long word and sent via a low differential signal line (LVDS) to the custom board. By using the master/slave configuration, only a minimal interface is required, with a total number of five wires required. Using these five wires, power is provided to the sensors, and a bidirectional LVDS line is provided for programming the sensors and receiving image data.

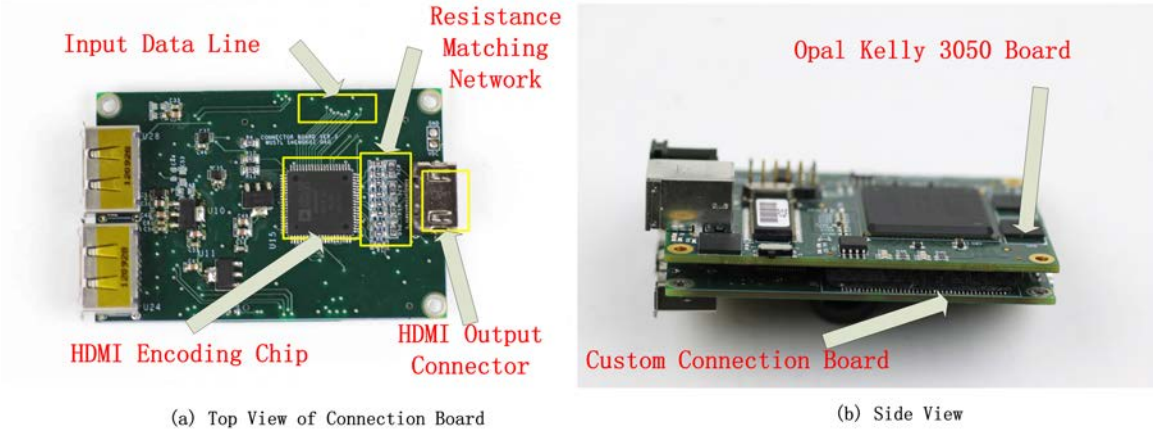


Figure 1.7: FPGA Board and Connection Board

The data from the imaging sensors is transmitted to the PC for image processing. The PC merges the visible spectrum and NIR images into a synthetic image, which is displayed to the goggle via the PC‘s HDMI interface. One of the main challenges and drawbacks of this implementation is that the surgeon is tethered to a PC via a USB cable. This is necessary in order to send the information to the PC from the imaging sensors. To alleviate this drawback, I have developed all the necessary image processing algorithms on an FPGA and removed the need for a PC for image processing. The development of the image acquisition and image processing in the FPGA allows for a compact, easily portable and low weight imaging system for intraoperative goggle system. A summary of the three different generations of the goggle system is shown in the Figure 1.8.

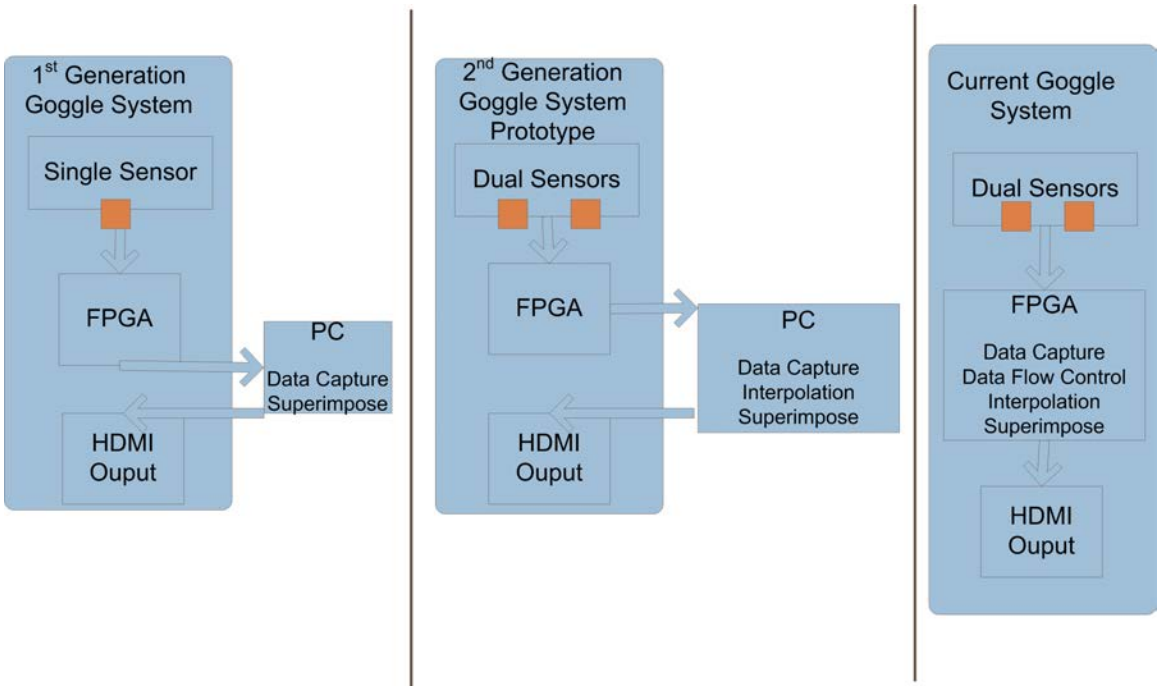


Figure 1.8: The Improvements between Different Generations

In both second and third generations of the system, two sensors are used rather than a single sensor in the first generation. Thus NIR and visible images will be taken at the same time at the same location, without changing the filter as in the 1st generation.

As shown in Figure 1.8, the PC is eliminated from the cycle in the current system, and the goggle system can process all the data on board. With this design, the system is compact and convenient enough to be carried to remote areas in developing countries, where expensive and bulky systems are not feasible.

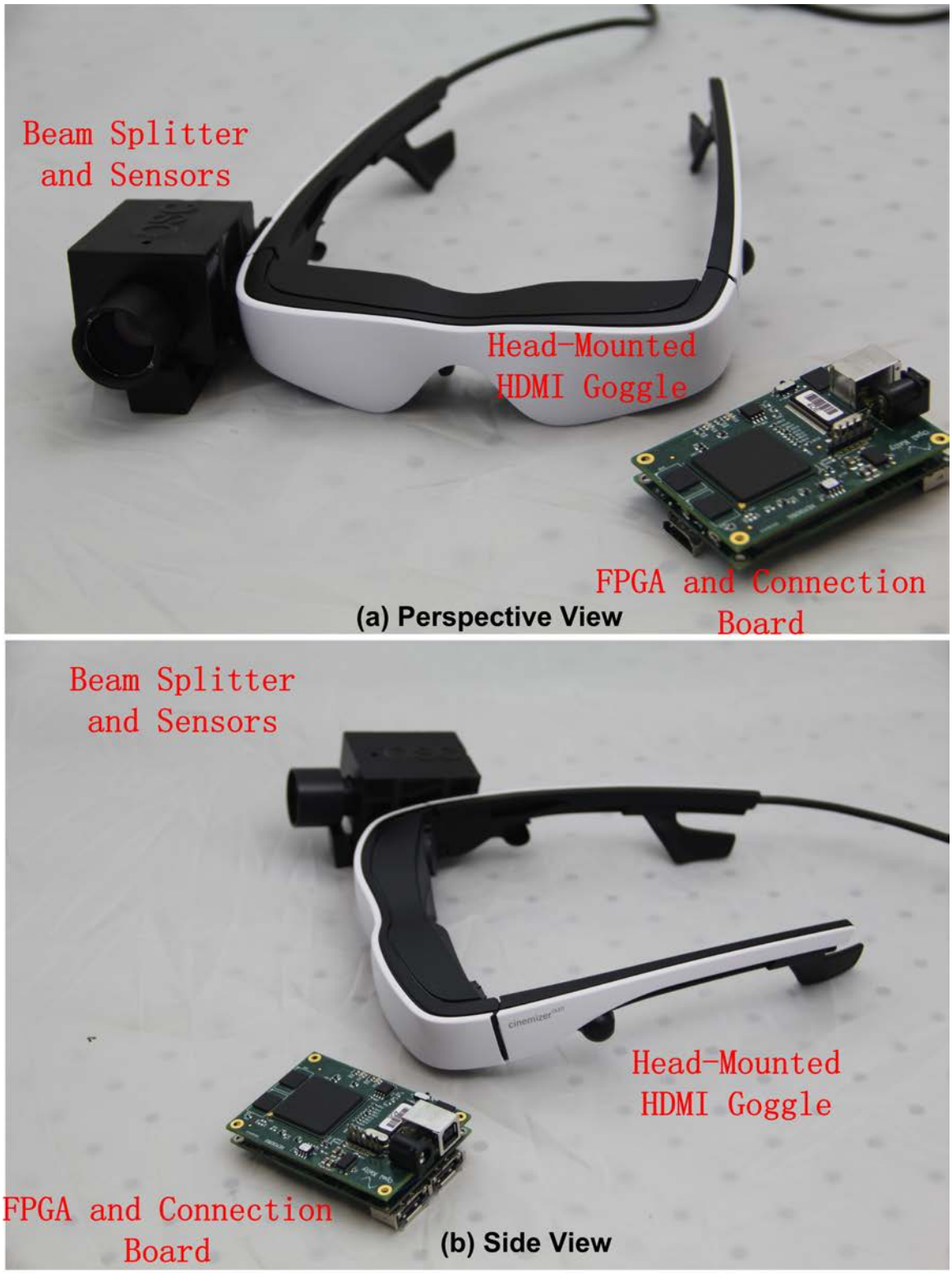


Figure 1.9: Major System Components of Beam Splitter Configuration

Figure 1.9 shows the major components of the system. The beam splitter and sensors will be mounted on the head-mounted goggle to capture the view. Raw data coming from sensors will be transmitted to the FPGA and connection board for image processing. The fused final images will be displayed through an HDMI cable on the head-mounted goggle.

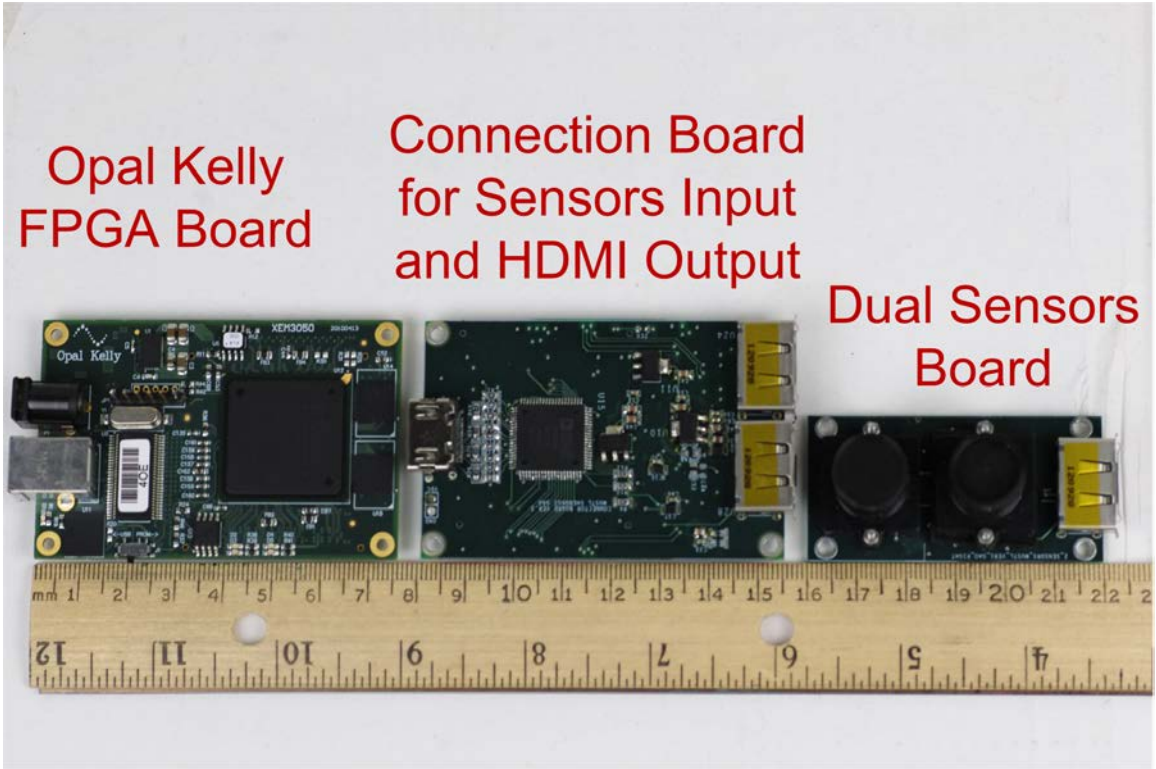


Figure 1.10: Major System Components of Single PCB Configuration

As shown in Figure 1.10, the size of the system with the single PCB configuration is also very compact for surgical purposes. The size of the FPGA and connection board is 7.5cm x 5cm, and the size of the sensor board is 6cm x 3cm. Because the two sensors are adjacent to each other in a row, soldered on a single board, the disparity of y direction is less than 10 pixels for experimental purposes.

All the image processing is performed on the image processing block implemented on the FPGA. The raw data of the visible light sensor will be processed to generate an interpolated color images. Then the processed color image and the threshold NIR image are than superimposed to produce a final image. The final image contains all the information from the visible sensor and highlights the NIR information.

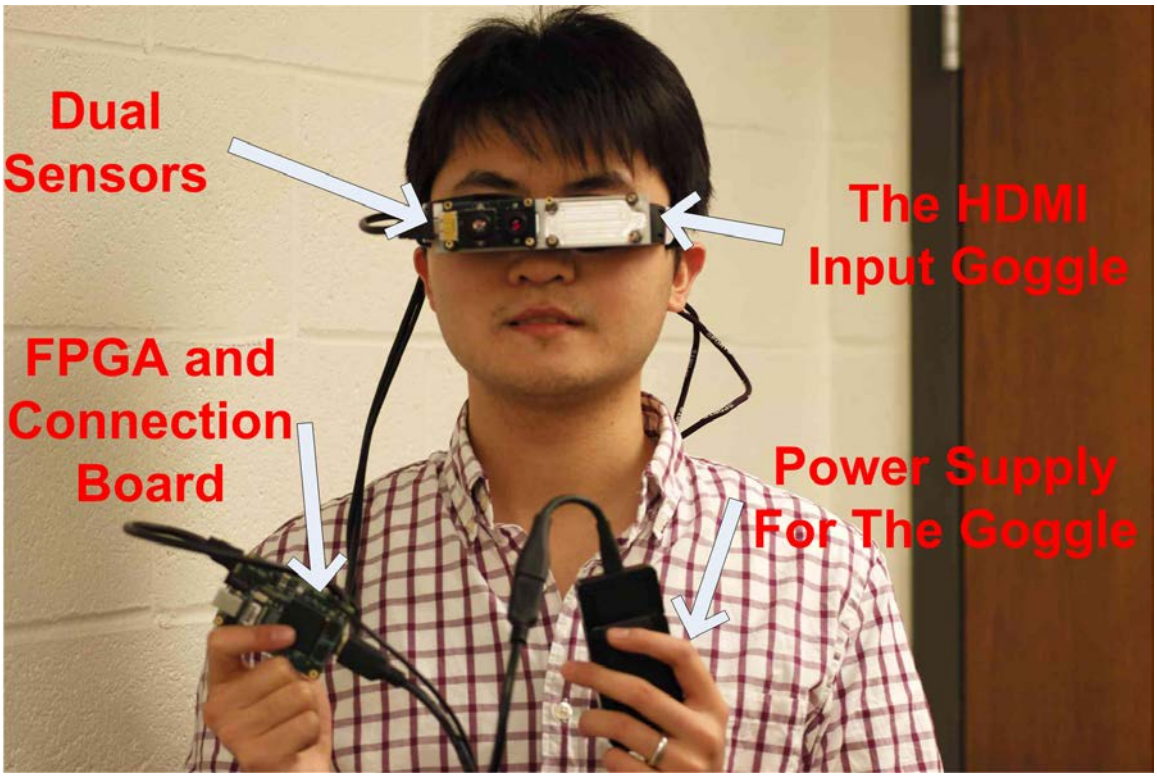


Figure 1.11: The Wearable Goggle System of Single PCB Configuration

From Figure 1.11, it is very clear that our goggle system is wearable and compact. The beam splitter version will be similar in size to the single board configuration. Processed data from sensor will be presented in real time on the goggle screen through an HDMI cable.

Chapter 2

HDMI Port and Driver Control Block

One of the essential components of the goggle system is real-time display of combined color and near infrared images to the physician during an intraoperative procedure. The data presented to the physician should contain both visible spectrum and NIR enhanced imagess superimposed into a single image. This image will be provided directly to the goggle display unit via an HDMI interface. This chapter describes the hardware implemented on a custom PCB board that houses an HDMI chip and the necessary hardware description code to initialize the display and stream real-time video.

2.1 The System Background

One of the drawbacks of the second generation goggle system developed by Prof. Gruev‘s lab is that the physician wearing the goggle is tethered to a computer. This serious drawback significantly restricts the movement of the physician and can impede the implementation and usability of such a system in the operating room. Furthermore, the goggle system should be portable and usable in developing countries as well. For this purpose, the PC will need to be eliminated from the loop and all signal processing algorithms will need to be moved to the FPGA. The PC performs two sets of computations: image fusion between visible and NIR images, and real-time display of images on the goggle via an HDMI interface. In this chapter, the intricacies of the HDMI hardware and firmware are described in detail.

2.2 The HDMI Port Hardware Overview

The HDMI related hardware components are mainly on the custom daughter board which connects to an FPGA main board. The connection board receives simultaneously data from both color and NIR sensors at 30 frames per second. The data from both sensors is transferred to an FPGA, where real-time image processing is implemented. The final image produced from the image processing pipeline is transferred to an HDMI decoder chip (Analog Devices AD9889B) before being displayed on the goggle screen. The HDMI encoder chip is housed on the custom daughter board.

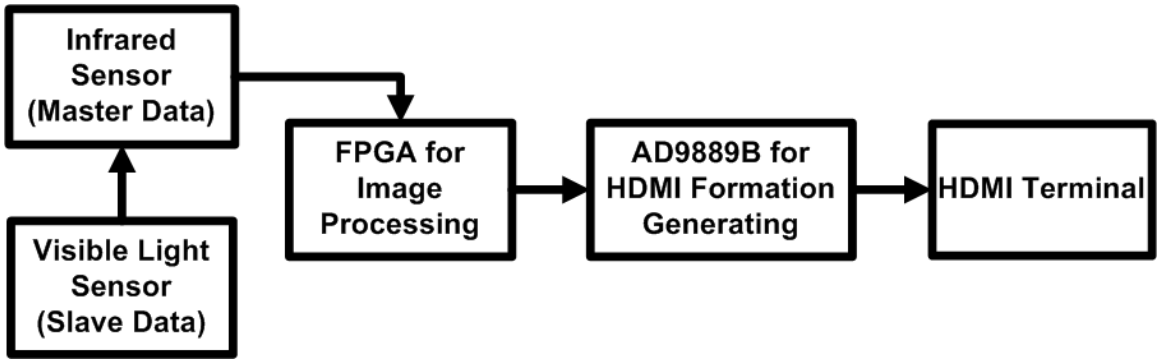


Figure 2.1: General System Flow Chart

The infrared sensor and visible light sensor are in serial connection and configured in a master-slave mode. In this mode, the data from the color sensor is serially passed to the NIR imager where the data from both sensors is combined into a single 16 bit word. The output from the color sensor is 752 x 480 x 8bits. The output from the NIR sensor is 752 x 480 x 16bits, where the lower 8 bit contain information from the color camera and the upper 8 bits contain information from the NIR imager. The data is sent via an LVDS channel to the FPGA. Once the data from both cameras is recived in the FPGA, a series of image processing algorithms are performed on the images. A final image, which contains information from both color and NIR sensors, is provided as an output from the FPGA. The output image from the FPGA is provided to an HGMI encoder chip before the image is displayed on a monitor or a goggle device.

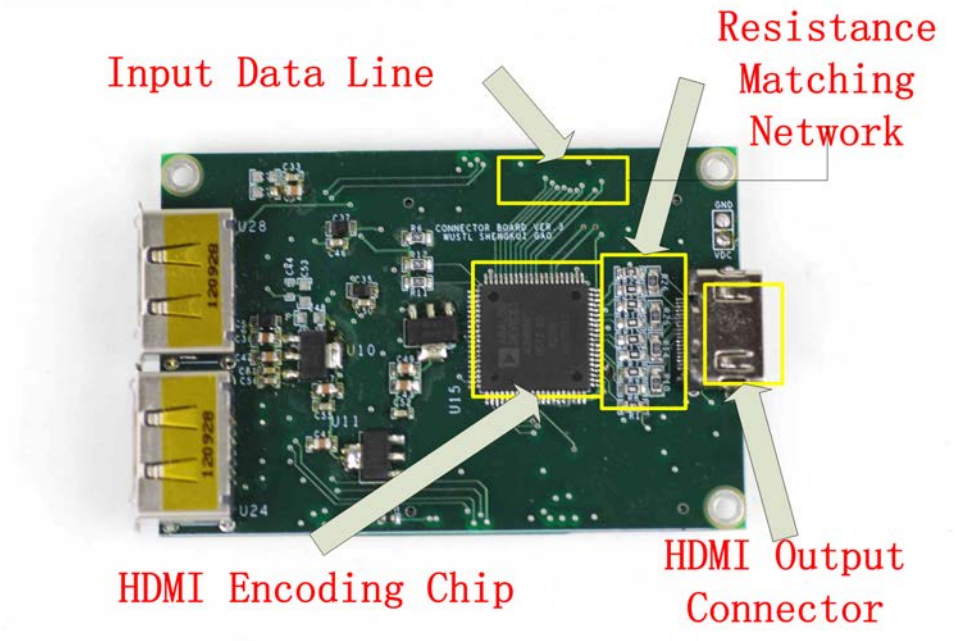


Figure 2.2: Connection Board Top View

The printed circuit board that houses the HDMI integrated circuit (IC) was designed by Shenkui Gao, and I developed the necessary firmware for this chip in an FPGA. The board reads 24 input data from FPGA via the input data lines as shown below.

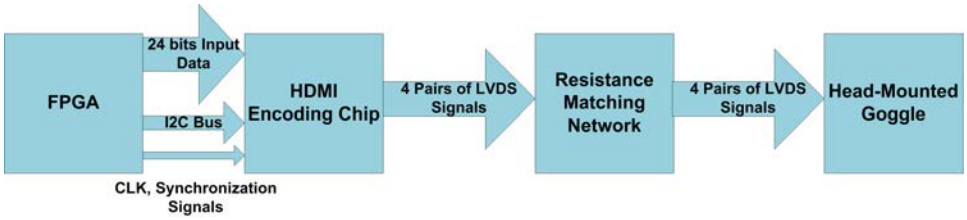


Figure 2.3: Connection Board Data Flow Chart

After encoding the receiving image, HDMI format output will be generated to go through resistance matching network and then output to HDMI output connector. Any HDMI terminals, including the goggle in our system, could be connected to this output connector to display HDMI images.

A standard RGB 444 is used for the input data. The format is shown in the table below. This standard specifies that the lower 8 bits contain information for the blue channel, the middle

8 bits contain information for the green channel and the higher 8 bits contain information for the red channel.

Table 2.1: Input Data Format

Input Format	Data[23:16]	Data[15:8]	Data[7:0]
RGB 444	R[7:0]	G[7:0]	B[7:0]

2.3 I2C Bus Specification for AD9889B

An I2C bus is used to communicate with and control the HDMI encoding chip, AD9889B. All the commands need to be transmitted to the chip by an I2C Bus Protocol. This section examines the I2C bus communication protocol. Some general I2C bus protocols will be discussed first. The detailed procedure to access registers on AD9889B will be presented at the end. The I2C bus consists of a data line and a clock line, SDA and SCL line respectively. The data is transmitted serially on the bus. Generally, each data bit is valid for every rising edge of the clock line. The transmission is controlled by a master device, which is the FPGA in our case. The master device will provide both clock signal and control signals. The default logic for both SDA and SCL lines will be high when the lines are released by both master and slave devices.

2.3.1 START and STOP Condition

In the I2C bus, a START condition is used to initialize a transmission while a STOP condition is used when the transmission is finished. A START condition is established when logic high to logic low transition happens on the SDA line while the SCL remains high during the transition. The STOP condition is established when logic low to logic high transition happens on the SDA line while the SCL remains high during the transition.

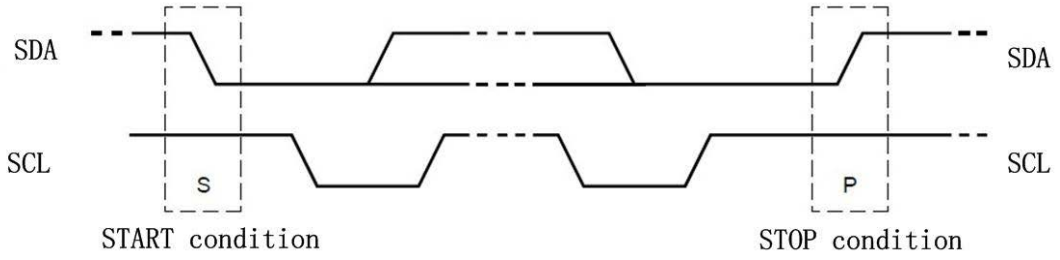


Figure 2.4: I2C Bus START and STOP Conditions

START and STOP conditions are provided by the master device. The bus state will remain in the busy state after the initiation of the START condition. A certain time after the STOP condition is initiated, the bus state will change back to free. If a repeated START condition is generated after a START condition, rather than STOP condition, the bus will remain busy.

2.3.2 Byte Format

The byte transmitted on the SDA line is 8-bits long. Every time 1 byte is received, a single acknowledge bit has to be generated. Hence, the SCL changes 9 times for each byte transmission. There is no limitation for the number of bytes to be transmitted per transfer. The register address will increase by 1 automatically each time whenever one byte is received. The most significant bit (MSB) is always transferred first.

The acknowledge bit is obligatory for each byte transfer. The clock for the acknowledge bit is also generated by master device, regardless of whether the receiver is master or slave device. The SDA line will be released during the acknowledge bit pulse. The receiver is supposed to pull down the SDA line during the acknowledge bit pulse to indicate that the data has been received. The logic of the SDA line should be stably low when the clock pulse is high.

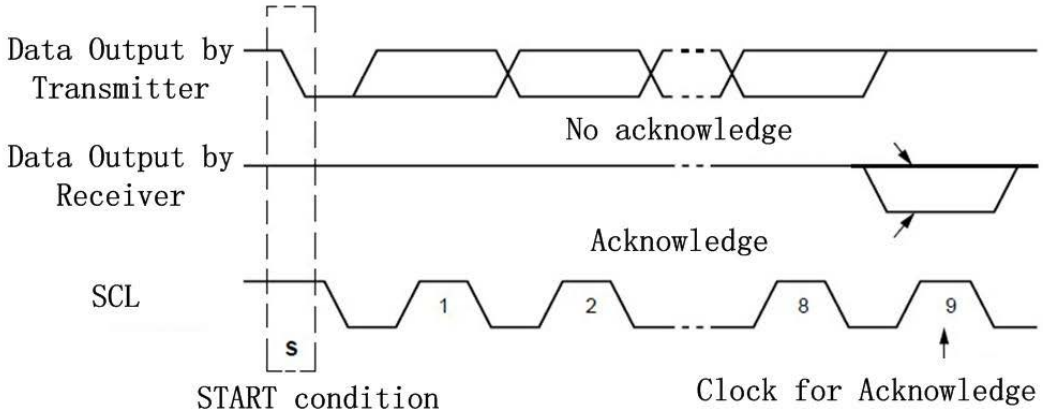


Figure 2.5: I2C Bus Byte Format and Acknowledge Signal

2.3.3 7-Bit Address Format

To initialize a transmission, a START condition has to be generated by the master device first. Then, a 7-bits long slave address is followed. The eighth bit of this byte is a data direction bit (R/W). Logic low of the data direction bit indicates to transfer data to the slave device, while logic high indicates to receive data from the slave device.

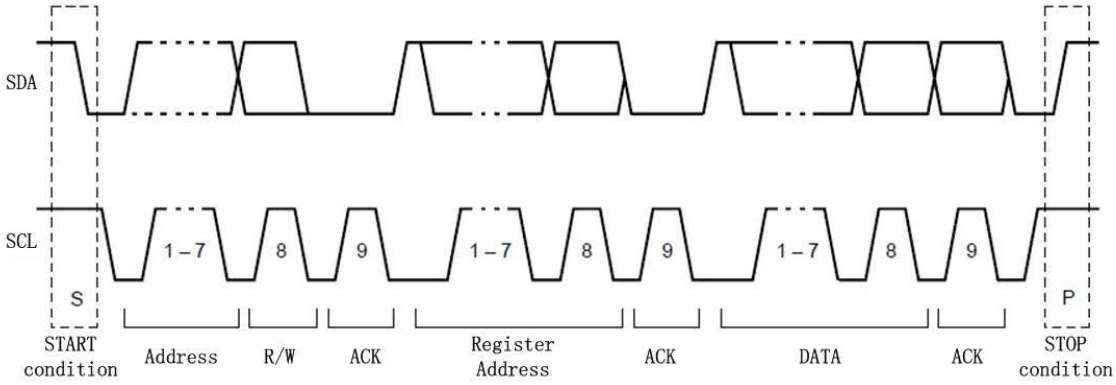


Figure 2.6: The Byte Format of 1-Byte Writing

To transfer 1 byte of data to a certain register of the slave device, the procedure is listed below:

- Master device initializes the transmission by sending the START condition.

- Slave address, with the 8th bit indicating transmission direction, will then be transmitted. This bit is low in this case.
- The device with the same address just received will acknowledge by pulling down the SDA while the SCL remains high.
- After receiving the acknowledge bit, the master device will then transmit a certain register address of the slave registers.
- Next byte transmitted on SDA line will be the actual data.
- After receiving the acknowledge bit, the master device will then terminate the transmission by sending the STOP condition.

To read 1 byte of data from a certain register of the slave device, the procedure is listed below:

- Master device initializes the transmission by START condition.
- Slave address, with the 8th bit indicating transmission direction, will then be transmitted. In this cause, the RW bit is also low.
- The device with the address just received will acknowledge by pulling down the SDA while the SCL remains high.
- After receiving the acknowledge bit, the master device will then transmit a certain register address of the slave registers.
- Then the slave address, with the 8th bit indicating transmission direction, will be transmitted again. In this cause, the RW bit is high to indicate the reading.
- Then the data will be presented on the SDA line. After receiving the data, the master should acknowledge by pulling down the SDA line.
- Then master device will then terminate the transmission by sending the STOP condition.

2.4 I2C Bus Control Block on FPGA

An open source master core was used as the basic layer of I2C bus communication. Bit level implementation was performed by the open source core. On the core interface, 3 layers were created to form the control block to handle command state machine, transmission error and initialization. TThe initialization block is designed specifically for the HDMI integrated circuit (AD9889B chip) to operate correctly with an output format of 1920 x 1080P @ 24Hz. The rest block could be reused as a general I2C control block. If the code needs to be reused in the future, configurations concerning device address, register address and data value will need to be slightly changed. The HDMI_TOP module will map each command, writing or reading from a certain register, to several commands. Those commands are used to control the I2C_MASTER_TOP module by writing to several command registers. The timing and control signals for I2C_MASTER_TOP module are provided by the WB_MASTER_TOP module.

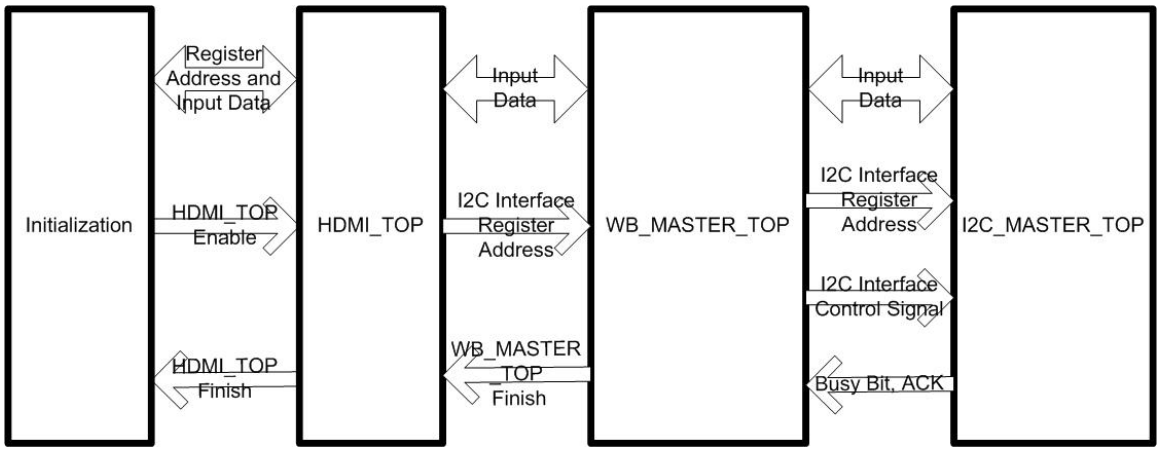


Figure 2.7: The Block Diagram of I2C Bus Control Block

In this section, the registers of the HDMI chip that are relevant to the project will be discussed first, to give a general picture about control flow of the chip. After that, the initialization block will be presented to show the detailed implementation of the control flow. The interfaces and functions of HDMI_TOP, WB_MASTER_TOP and I2C_MASTER_TOP will then be discussed to give detailed steps for communicating with the chip.

2.4.1 AD9889B Register Overview

Before going into detail about the control blocks, detailed description of the registers functionality in the HDMI chip will be presented. The registers we need to set are power control related registers, fixed value registers, interrupt related registers, and video format registers. All the registers used to control the chip to perform basic HDMI format transform and output are shown below.

Table 2.2: Registers Used on AD9889B

Register Address	**Type**	**Register Name**	**Function**	**Value to Set**
0x41[6]	R/W	Power Down	Main Power Down	0x10
0xA1	R/W	Rx Sense Power Down	Rx Sense Monitoring and Level II Power Down	0x7C
0x0A[6:5]	R/W	Fixed	Must be set to 0b00 for proper operation	0x00
0x98	R/W	Fixed	Must be set to 0x07 for proper operation	0x07
0x9C	R/W	Fixed	Must be set to 0x38 for proper operation	0x38
0x9D[1:0]	R/W	Fixed	Must be set to 0b01 for proper operation	0x61
0xA2	R/W	Fixed	Must be set to 0x87 for high speed operation	0x87
0xA3	R/W	Fixed	Must be set to 0x87 for high speed operation	0x87
0xAB	R/W	Fixed	Must be set to 0x00 for proper operation	0x00
0xBB	R/W	Fixed	Must be set to 0xFF for proper operation	0xFF
0xBA	R/W	Clock Delay	Delay adjust for Input Video CLK capture	0x60
0x44	R/W	Audio InfoFrame Enable	Audio InfoFrame Enable	0x00
0x95	R/W	HDCP Controller Error Interrupt Enable	0 = interrupt disabled 1 = interrupt enabled	0x00
0xAF	R/W	HDMI DVI Select	1 = HDMI Mode	0x06
0x15	R/W	Low Refresh Rate Video	0 = Normal Mode	0x01
0x16[7:6]	R/W	Output Format	00 = RGB 4:4:4	0x00
0x17	R/W	Aspect Ratio	0 = 4:3 1 = 16:9	0x00
0x3B[4:3]	R/W	PR PLL Manual	The clock multiplication of the input clock used in manual pixel repetition.	0x00
0x3C	R/W	VIC to Rx	User programmed VIC to send to Rx.	0x00

2.4.2 Initialization Block

The initialization block will be executed only once after power up. After receiving the finished signal from this block, the top module will disable this block. In the first iteration of the implementation, these registers were set through the PC via a USB interface. In the final implementation of the code, the register values are set via the initialization block in the FPGA and the PC is removed from the loop.

The initialization state machine in the block will send and verify all the necessary commands to setup the HDMI chip correctly. All the registers listed in Section 2.4.2 will be set up before the termination of the state machine.

Other state machines could be added to this block very easily by adding the finish signal for each state machine. A global finish flag is used and the appropriate logic could be used to quit the initialization block after all the state machines is completed.

Table 2.3: Ports of Initialization Block

Port	**Width**	**Direction**	**Description**
clk	1	Input	Clock signal
rst	1	Input	Reset signal
en	1	Input	Block enable signal
finish	1	Output	Finish indicator
commandHdmiTop	16	Output	Register address and data
rstnHdmi	1	Output	hdmiTop finish signal
enHdmiTop	1	Output	enHdmiTop
finishHdmiTop	1	Input	hdmiTop finish signal

To add another state machine into this block, only a few input signals, like ENABLE, FINISH and RESET need to be added to the block. The finish signal of the block will depend on all the finish signals from each state machine. If all the state machines have received the finish signals from their lower level block, the initialization will be terminated.

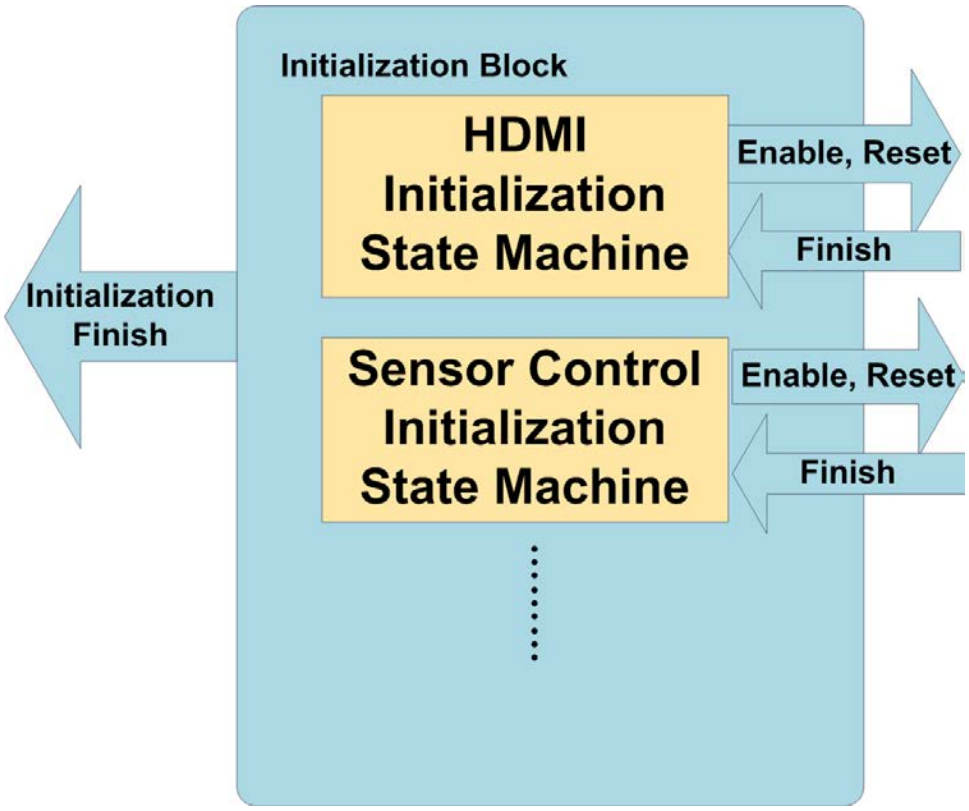


Figure 2.8: The Block Diagram of Initialization Block

2.4.3 I2C_MASTER_TOP Block

This block belongs to the open source I2C master package. It provides basic interface and functionality for a more complicated design. The registers and signals necessary for the HDMI control block will be described briefly.

Table 2.4: Ports of I2C_MASTER_TOP Block

Port	Width	Direction	Description
wbClkI	1	Input	Master clock
wbRstI	1	Input	Synchronous reset, active high
arstI	1	Input	Asynchronous reset
wbAdrI	3	Input	Lower address bits
wbDatI	8	Input	Data towards the core
wbDatO	8	Output	Data from the core
wbWeI	1	Input	Write enable input
wbStbI	1	Input	Strobe signal Core select input
wbCycI	1	Input	Valid bus cycle input
wbAckO	1	Output	Bus cycle acknowledge output
wbIntaO	1	Output	Interrupt signal output

All the inputs fore this block are provided by the WB_MASTER_TOP block. Asynchronous reset is used for convenience. The wbAdrI signal is used to write value to a register in the block. The registers used in the design are listed below.

Table 2.5: I2C_MASTER_TOP Registers

Register Address	Type	Register Name	Function
0x00	RW	PREPlo	Clock Prescale register lo-byte
0x01	RW	PREPhi	Clock Prescale register hi-byte
0x02	RW	CTR	Control register
0x03	W	TXR	Transmit register
0x03	R	RXR	Receive register
0x04	W	CR	Command register
0x04	R	SR	Status register

Pre-scale registers are used to set the operation frequency of the SCL clock line. The open core uses a 5*SCL internal clock for timing. The scale coefficients could be computed using the following equation:

$$prescale = wbClkI/(5 * i2cclk) - 1 \tag{2.1}$$

In our case, the input clock is 148.5MHz while the desired I2C bus clock frequency is 100KHz, according to the I2C specification. So the pre-scale coefficient will be 0x0095.

The most significant bit of the control register is used as enable signal of the block. This register should be set after all other registers are configured correctly.

Transmitter register and receiver register are registers to buffer the data on the I2C line. The data stored in the transmitter register will be the next byte on the I2C bus. The receiver register will buffer the last byte received by the I2C bus.

Table 2.6: I2C_MASTER_TOP Command Register

Bit Position	**Type**	**Description**
7	W	STA, generate (repeated) start condition
6	W	STO, generate stop condition
5	W	RD, read from slave
4	W	WR, write to slave
3	W	ACK, when a receiver, sent ACK(ACK = 0) or NACK (ACK = 1)
21	W	Reserved
0	W	IACK, Interrupt acknowledges. When set, clears a pending interrupt

Command register is used to control the behavior of the I2C bus. For example, the STA bit needs to be set before we enable the core so that the start signal can be generated after the enable signal.

Table 2.7: I2C_MASTER_TOP Status Register

Bit Position	**Type**	**Description**
7	R	RxACK, Receuved acknowledge from slave
6	R	Busy, I2C Bus busy
5	R	Arbitration lost
42	R	Reserved
1	R	TIP, Transfer in progress
0	R	IF, Interrupt Flag

The status register has to be checked by a higher level block every time a bus related operation is requested. RxACK should be checked when a bus transfer is initialized and slave is set on the bus. After receiving RxACK, the state machine of the higher level could move to the next state. Otherwise, the slave address has to be set on the bus again. TIP is mainly used to check whether the last transmission has finished or not. After each address or data is presented on the bus by request of a higher level control block, TIP will go to 1

when the bus is still working on the request. Only when TIP goes back to 0, can the next request be processed.

2.4.4 HDMI_TOP Block

This block contains the actual logic of sending and receiving a command to and from a certain register address of the device. By doing that, all registers in I2C_MASTER_TOP block have to be set and checked to generate a bus transmission and then a bus read to verify the write. There are two types of state machine inside this block to control the behaviour of I2C_MASTER_TOP block. One is a register setting related state machine while the other one is a bus related state machine.

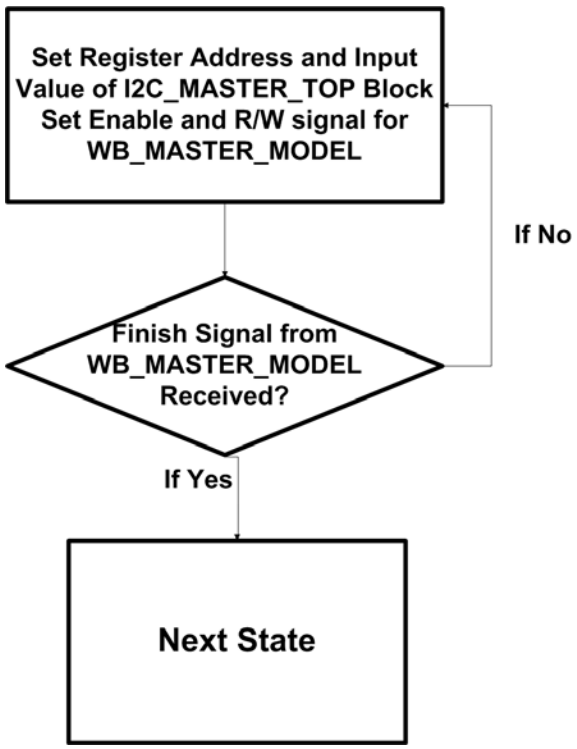


Figure 2.9: Register Setting Related State Machine

In this state machine, the communication between I2C_MASTER_TOP and HDMI_TOP has been taken care of by the WB_MASTER_MODEL. So the only signal which should be checked is the finish signal from WB_MASTER_MODEL.

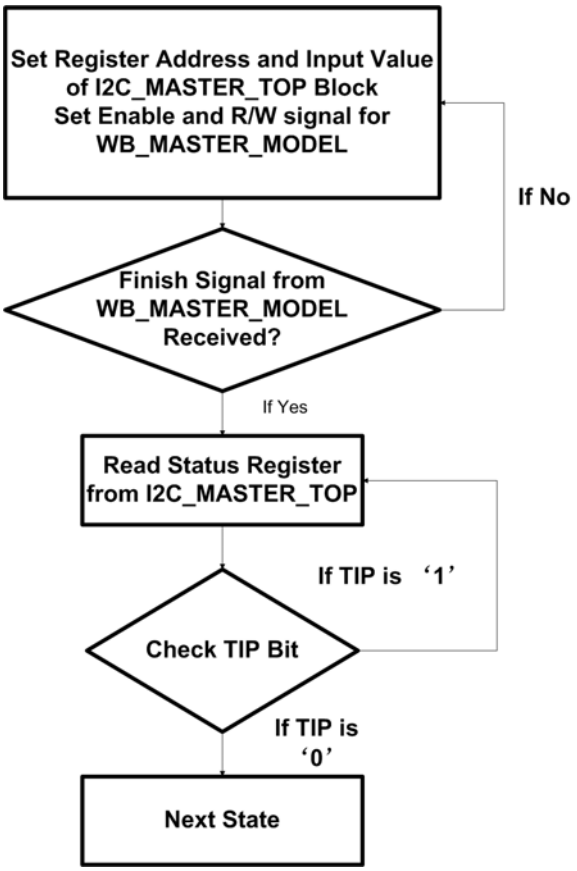


Figure 2.10: Bus Action Related State Machine

In this state machine, the TIP bits also need to be checked, because the next command has to wait until the I2C_MASTER_TOP finishes the current bus operation. The clock of HDMI_TOP block is 75 MHz while I2C bus operates at 100 KHz, so the block has to wait for a few thousand cycles.

Table 2.8: Commands Sent From HDMI_TOP Block

Order	Type	Address	Data	Description
1	RS	PRERlo	0x95	Set lower part of PRER register
2	RS	PRERhi	0x00	Set higher part of PRER register
3	RS	CTR	0x80	Enable I2C_MASTER_TOP block
4	RS	TXR	SADR,WR	Write command, Present slave address on the I2C bus
5	BA	CR	0x90	Set command: Start, Write
6	RS	TXR	DATA[15:8]	Present register address of slave device
7	BA	CR	0x10	Set command: Write
8	RS	TXR	DATA[7:0]	Present data for slave device
9	BA	CR	0x50	Set command: Stop, Write
10	RS	TXR	SADR,WR	Present slave address on the I2C bus
11	BA	CR	0x90	Set command: Start, Write
12	RS	TXR	DATA[15:8]	Present register address of slave device
13	BA	CR	0x10	Set command: Write
14	RS	TXR	SADR,RD	Read command, Present slave address on the I2C bus
15	BA	CR	0x90	Set command: Start, Write
16	BA	CR	0x28	Set command: Read, NACK
17	None	None	None	Check data just received

The steps 1 to 9 are to write data to a certain register of the device. To verify the writing, steps 10 to 17 will read the data from the register previously written. The received data will be transferred to a higher level to decide if an error has occurred and if a re-transmission is needed.

For read-only registers in the AD9889B, these steps are also valid. However, the data written to the register will not necessarily be the data received. The write command on the bus will be valid while the data will not be written into the register on the device.

If the values of some registers need to be read out, steps 10 to 17 could be executed separately without executing 1 to 9 first, because step 10 will regenerate a start condition on the bus and call the device address again. Thus the code could be reused in this block.

2.4.5 WB_MASTER_MODEL Block

This block is a middle interface between I2C_MASTER_TOP and HDMI_TOP. WB_MASTER_MODEL will receive commands from HDMI_TOP and output the control signals for I2C_MASTER_TOP to execute them.

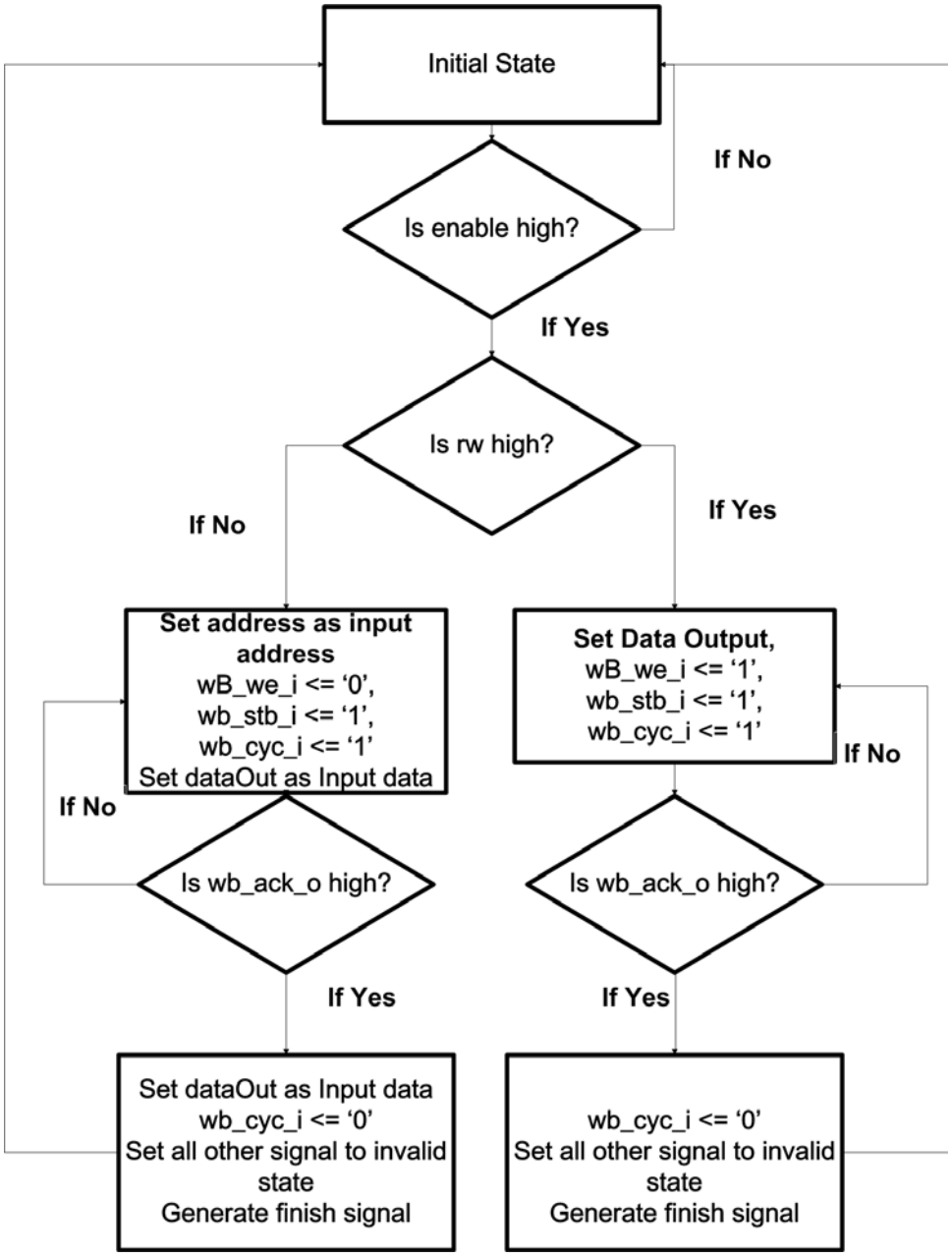


Figure 2.11: Flow Chart of WB_MASTER_MODEL Block

There are only two state machines inside this block. One is to write values to the I2C_MASTER_TOP to perform bus operations, and the other one is to read values from I2C_MASTER_TOP to check the value or wait for data. After receiving an acknowledge signal from I2C_MASTER_TOP, a finish signal will be generated to tell the HDMI_TOP that one command, read or write, has finished and that next command is ready to be executed. If the current command is to read from I2C_MASTER_TOP, the data received will be transmitted to HDMI_TOP.

2.5 Simulation Result and Test Results

The Verilog state machine for programming the HDMI encoder chip was tested in detail using ModelSim software. Once the state machine was fully operational and verified in the simulator, the Verilog code was executed on an FPGA. Several test patterns were generated on the FPGA and displayed directly on the goggle via the HDMI interface. Simulation results as well as real HDMI images obtained from the system are presented in this section.

2.5.1 I2C Bus Control Block Results

To ensure that the I2C Bus control related blocks have worked fine, simulation of I2C Bus behavior was carried out. The simulation includes transferring a certain address via I2C Bus according to the I2C Bus specification. The simulation result of I2C control block is shown below.

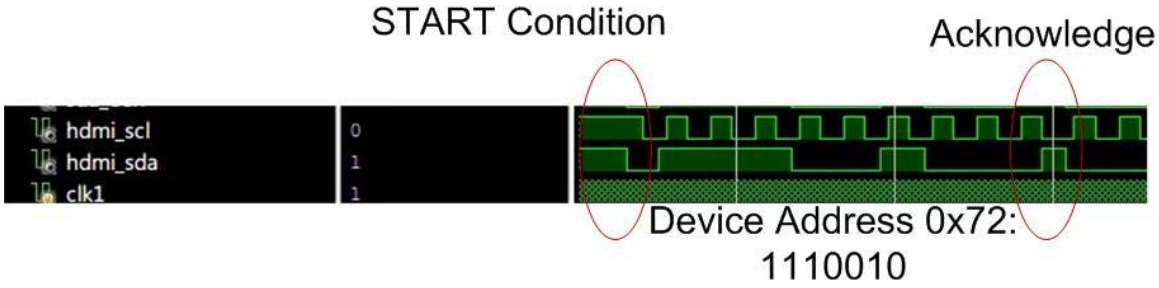


Figure 2.12: Simulation Result of I2C Bus Control Block

The ISim developed by Xilinx was used to simulate the whole HDMI I2C Control block. The clk1 signal is the input clock for all the I2C bus related blocks. As shown in Figure 2.12, the 7-bit address of AD9889B is sent through the hdmi_sda port. The positive edge of hdmi_clk is generated at the stable stage of every bit of the address. At the ninth clock pulse on hdmi_clk, the bus is released by the master device, which is FPGA in our case, and then pulled down to logic low by the slave device.

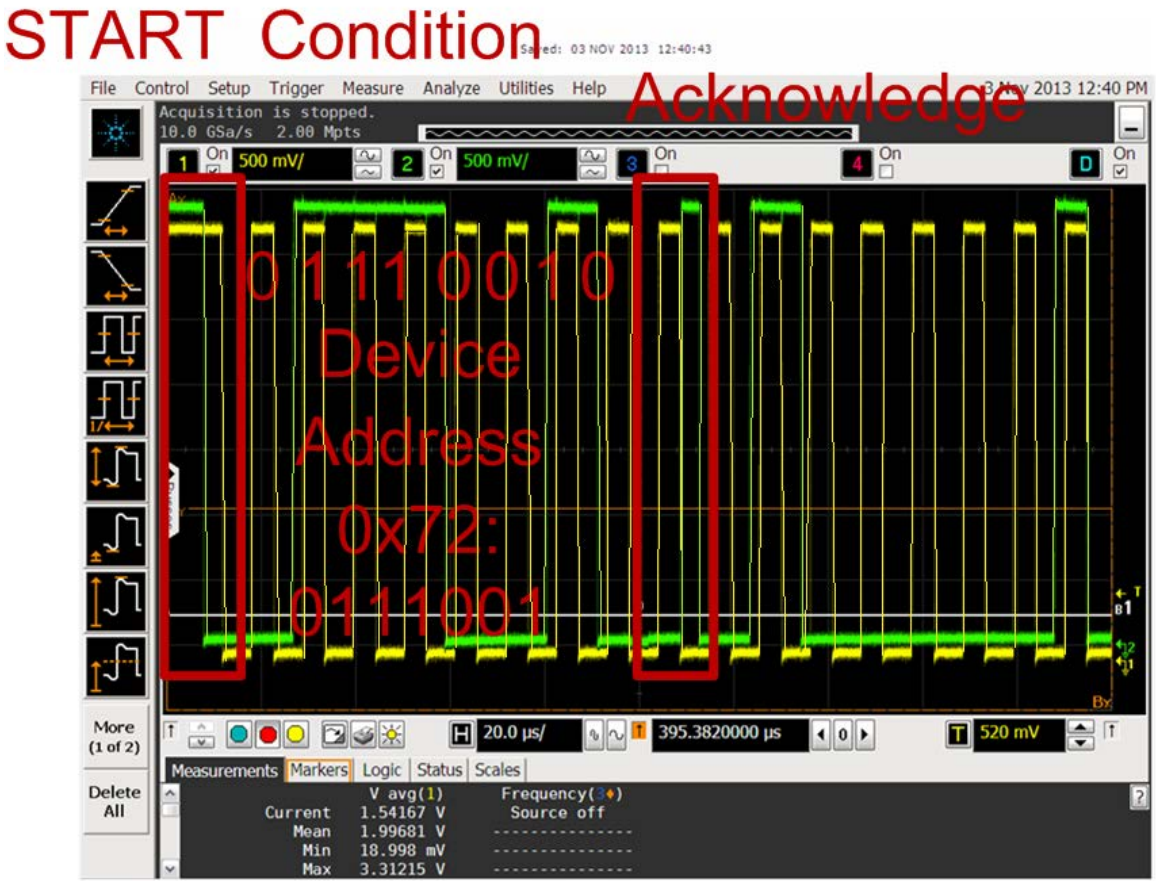


Figure 2.13: Experimental Result of I2C Bus Control Block from Oscilloscope

The I2C control block is also tested by experiments. The green line above is the hdmi_sda port of the FPGA. The yellow line is the hdmi_clk. As shown in Figure 2.13, the AD9889B chip acknowledges the call from master device by pulling down the hdmi_sda line at the ninth positive edge on hdmi_clk line after receiving the correct address.

2.5.2 HDMI Subsystem Test Result

After experiments on the I2C bus, all the HDMI related software and hardware were also tested to make sure that the HDMI output system worked correctly. Four different patterns were generated by the FPGA to verify the correctness of the timing, address and data lines.

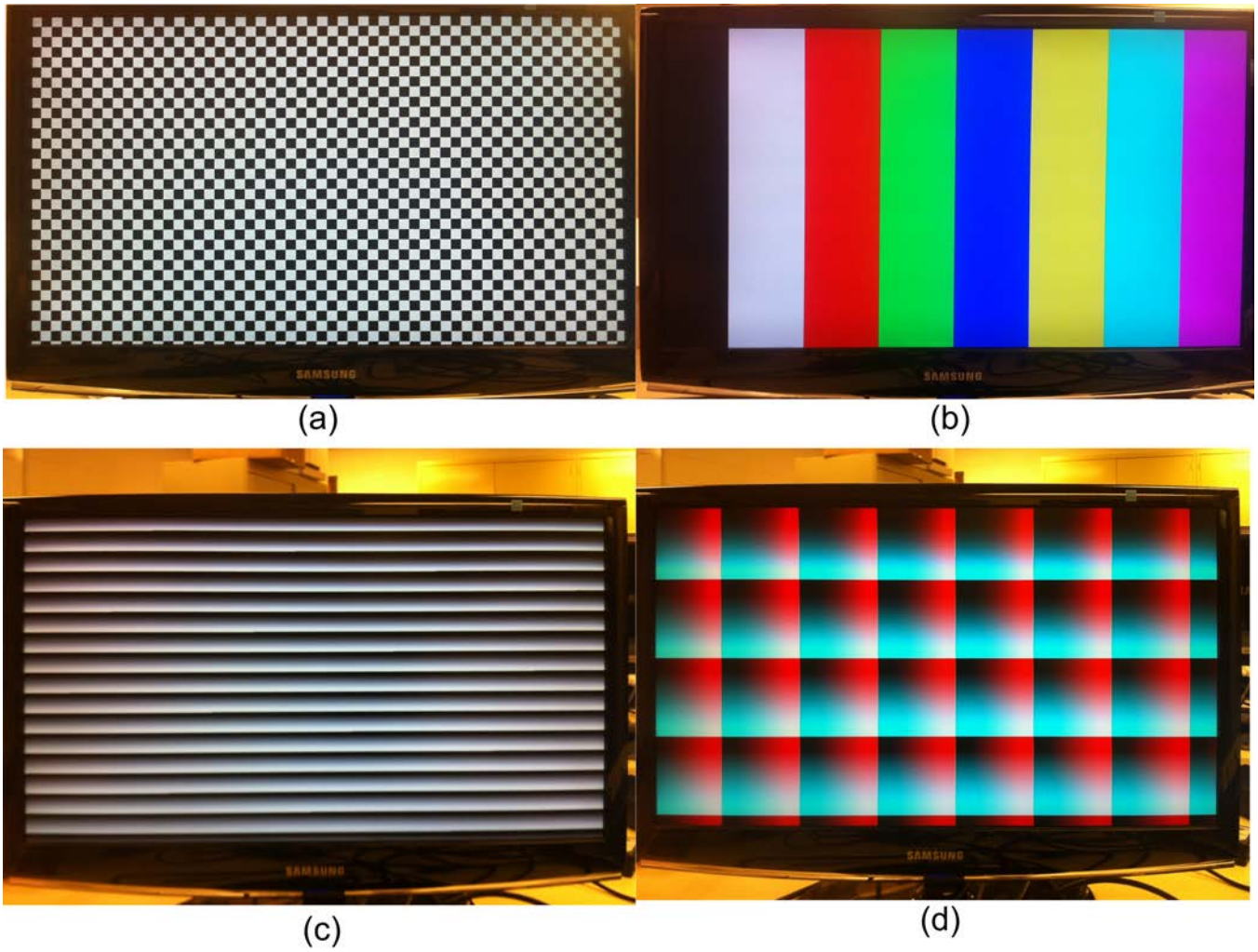


Figure 2.14: Experimental Result of the HDMI Subsystem

In Figure 2.14(a), a check board pattern was generated to test the timing of vertical and horizontal synchronization signals. If the synchronization signals are perfectly aligned with each frame, the first row and column of first white square should be on the boundary of the screen as shown. In Figure 2.14(b), an alternating color-strips pattern was used to test

the RGB output line from FPGA. Eight different combinations of RGB values were used to test the data lines. After that, the grey scale and color spectrum test pattern was also created. The results are shown in Figure 2.14(c) and (d) respectively. These two patterns could detect if there are any data lines are misconnected to each other or ground.As the pattern shown here is uniformly distributed, the HDMI related hardware and control blocks should work fine.

Chapter 3

Image Processing on FPGA

The main image processing tasks are implemented in the FPGA in order to ensure real-time performance. The images from both sensors, visible and NIR, are streamed in the FPGA. The color image is interpolated in order to increase resolution and accuracy of the color information. The NIR image is threshold and superimposed on the color image using a false color. The false color representation of the NIR image is ideal to guide the surgeons' attention to the tumor area as well as assist in identifying additional tumor tissues during the surgery. Since all image processing tasks are implemented in the FPGA, the PC is eliminated from the loop and allows for tethered-free implementation of the goggle system.

This chapter describes the image processing block that is used to generate color and superimposed final images. In particular, an overview of the interpolation algorithm and comparison will be given first. After that, the implementation details, including input design, algorithm related registers and output will be discussed. Finally, the superimposition method and final results will be presented.

3.1 Binary Interpolation for RGB Grid

The color camera used in this system employs a Bayer color pattern. This color pattern is optimized to capture color information that resembles the color sensitivity of the human eye. It uses twice as many green pixels compared to red and blue pixels. In order to generate an accurate color image, the raw color information from the Bayer pattern is typically interpolated in order to increase the spatial resolution as well as increase the accuracy of

the color information. In this section, comparisons of two different interpolation algorithms are discussed. Based on the benefit and cost of each algorithm, the reason of choosing a particular algorithm will be given.

Before talking about the algorithms, the Bayer color pattern in a color imager is shown below.

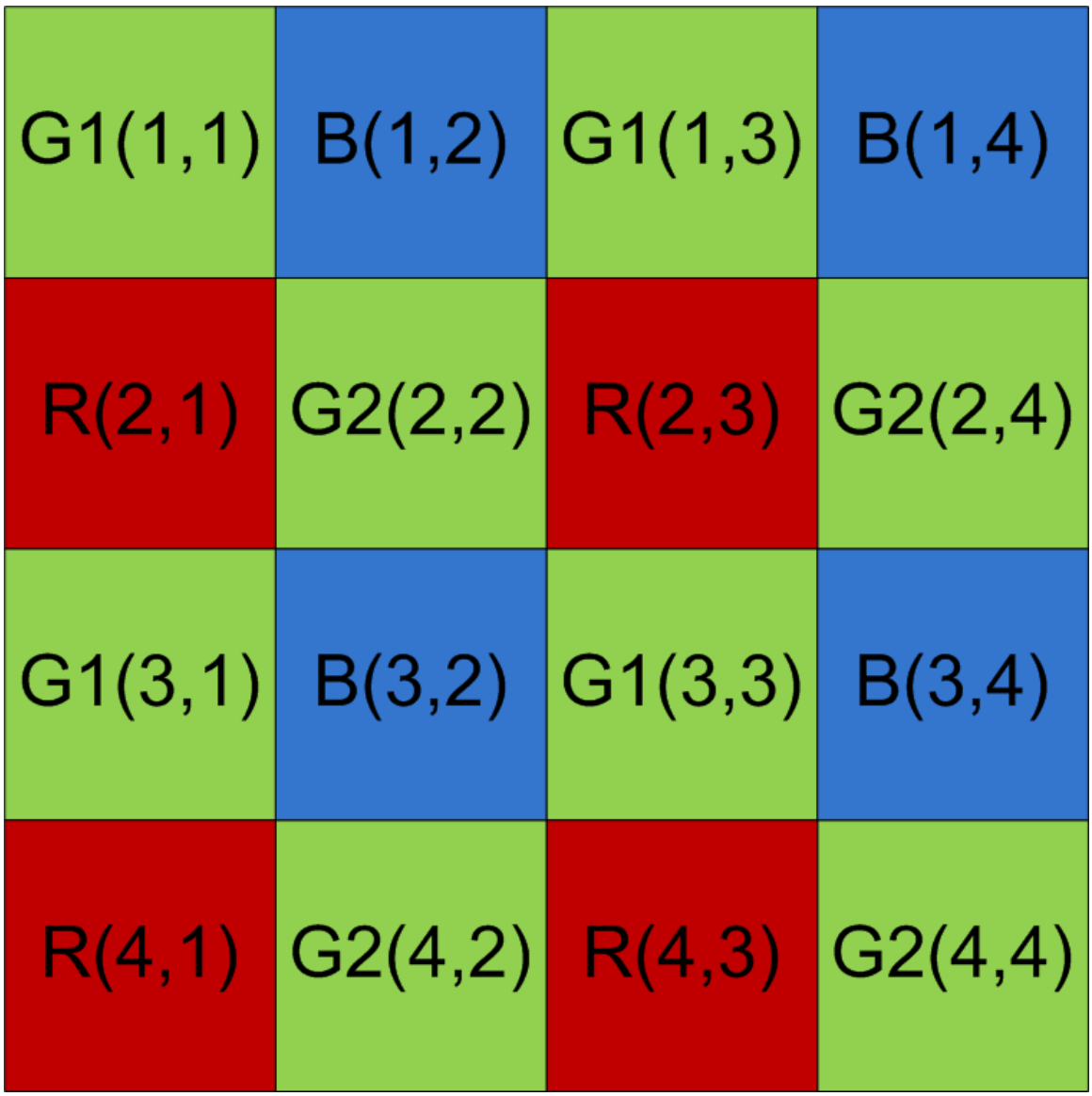


Figure 3.1: RGB Grid of the Visible Sensor

For each frame in our system, the first pixel of the first line will be a pixel with intensity of the green channel, and then the second pixel in the same line will be a blue pixel. If the line number becomes even, like 2, the first pixel will be red, then followed by a green pixel.

3.1.1 2 x 2 Grid Interpolation

For each pixel in the RGB grid, there is only one intensity value of red, green or blue channel. To get a color image, a naive way is to use the color intensity of the neighborhood, which uses a 2x2 grid to interpolate the color information for all 3 channels.

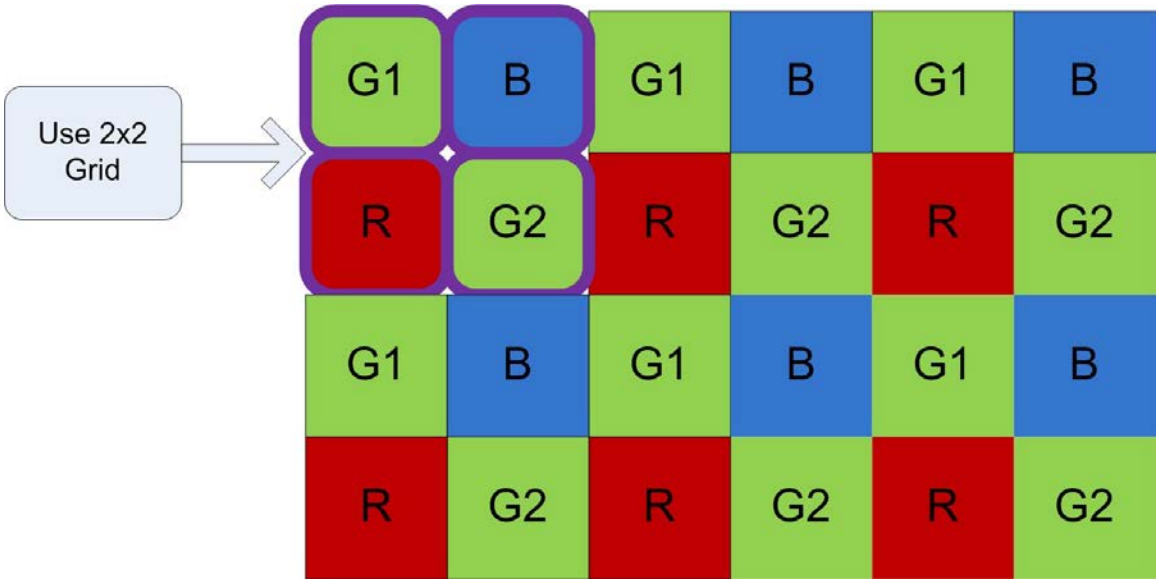


Figure 3.2: 2 x 2 Grid Interpolation Window

For example, when interpolating the color of the first pixel on the above graph, all its neighbors, which are highlighted with purple lines will be used. Those four pixels of the sensor will be treated as one pixel. The red and blue channel values are the intensity of the red and blue pixel in this grid, respectively. The green channel value of the processed pixel is the average of the two green pixels in the grid.

In this way, a color image will be generated very easily. However, the major defect of this method is that the resolution will decrease a lot. In the system, the sensor we used is Aptina MT9V034 with a resolution of 752 x 480. After using the method above, the resolution of

the output color image will be 376 x 240 because every 4 pixels of the sensor will be treated as 1 big pixel.

3.1.2 3 x 3 Slide Window Binary Interpolation

The way to fix the resolution problem mentioned above is to use a slide window for every pixel in the grid and to generate intensity of all 3 channels.

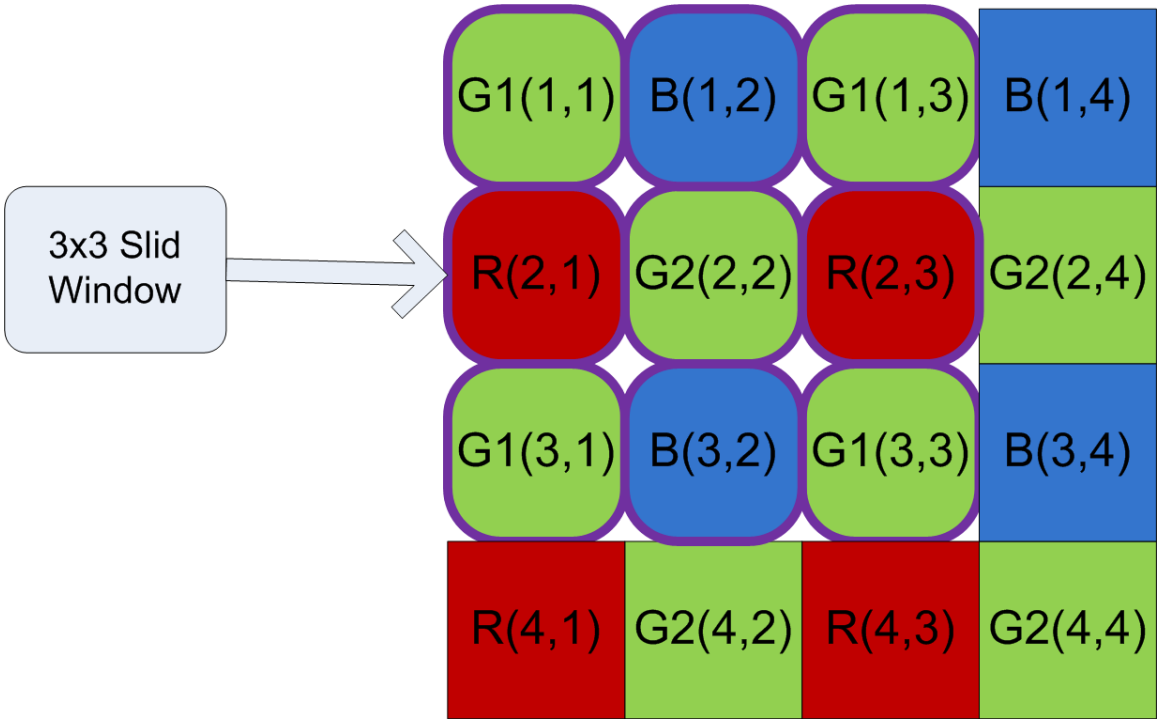


Figure 3.3: 3 x 3 Grid Slide Window

In the figure above, a 3x3 slide window is highlighted with purple lines. The pixel in the middle of the window is a G2 pixel. The green channel of the G2 pixel is known. For that pixel, the red channel could be interpolated by the average of its left and right neighbors, which are red pixels. If the 3x3 slide window is treated like a 3x3 matrix M, the equation for the red channel of G2 pixel is:

$$G2(2,2)_R = (R(2,1) + R(2,3))/2 \tag{3.1}$$

And for the blue channel of the G2 pixel, the equation is similar:

$$G2(2,2)_B = (B(1,2) + B(3,2))/2 \tag{3.2}$$

For the next pixel, a similar method is used.

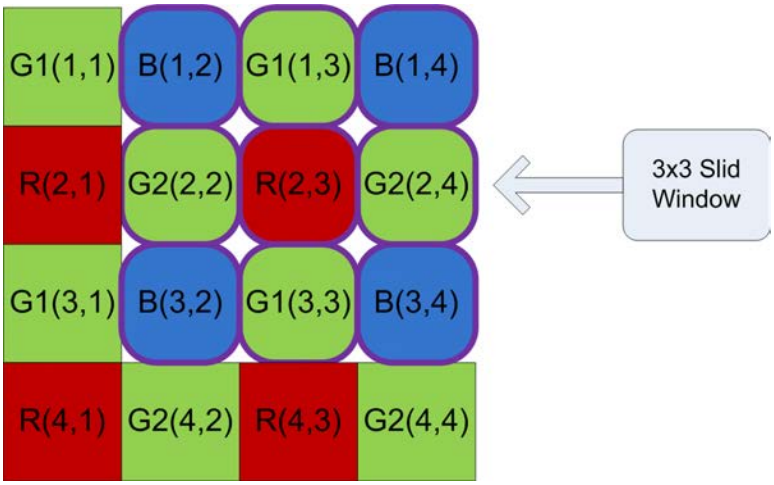


Figure 3.4: 3 x 3 Grid Slide Window of Next Clock Pulse

The second slide window shifts right by only 1 pixel. The middle pixel to be interpolated becomes red. For the value of red channel, the intensity of the pixel can be used directly. The green value could be interpolated as the average of its four green neighbors.

$$R(2,3)_G = (G1(1,3) + G2(2,2) + G2(2,4) + G1(3,3))/4 \tag{3.3}$$

While the blue value of the pixel is the average of its four blue neighbors.

$$R(2,3)_B = (B(1,2) + B(1,4) + B(3,2) + B(3,4))/4 \tag{3.4}$$

When the middle pixel becomes G1 or B, the situation is very similar to G2 and R, as discussed above. If all 4 situations could be handled correctly, all 3 channel intensities could be recovered for each pixel and the resolution will be the original resolution of the sensor.

The downside of this method is:

- Computation will be increased a lot compared with the previous method. However, it is still trivial compared to the computing power of FPGA.
- The boundary condition becomes complicated.

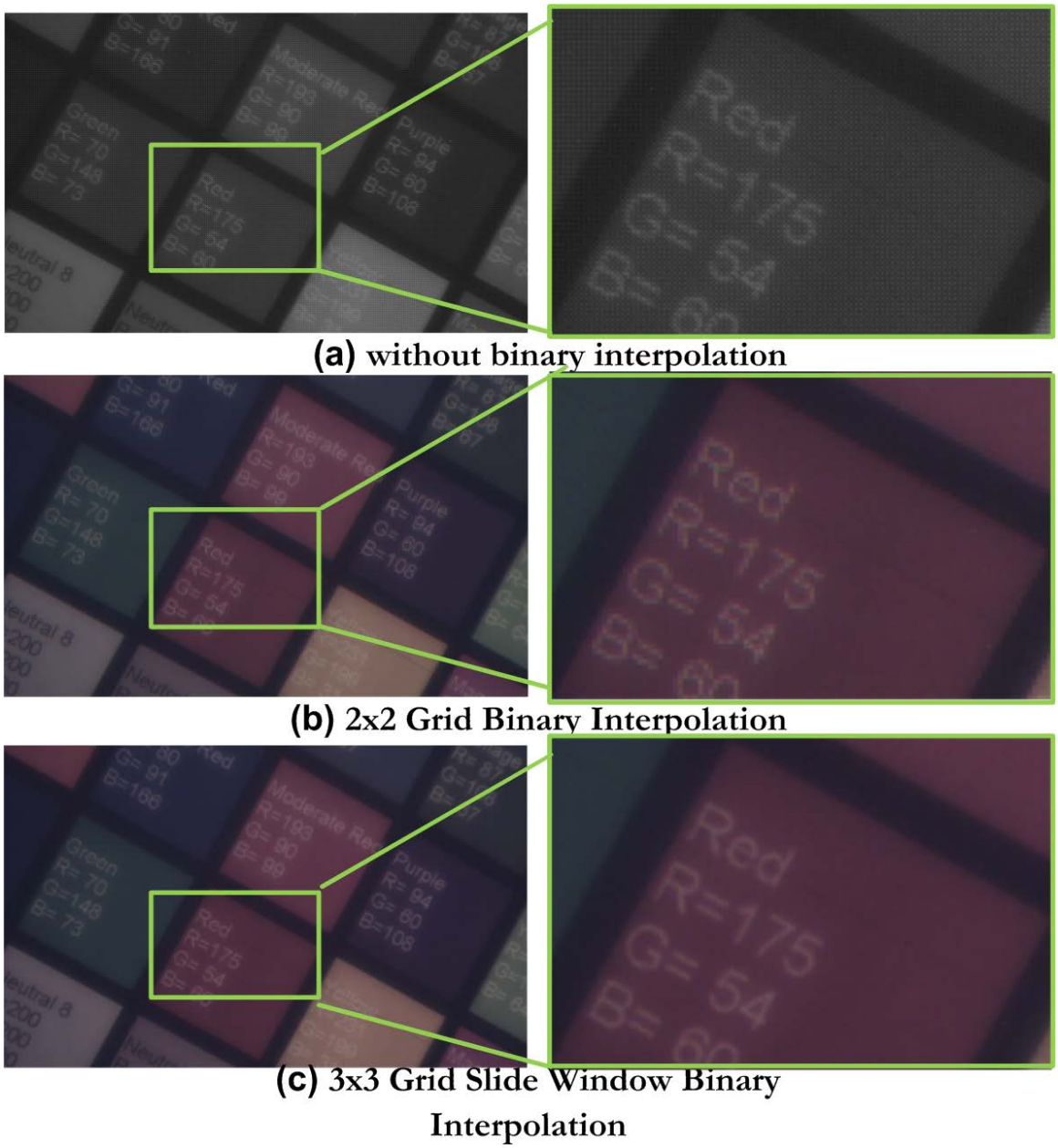


Figure 3.5: Comparison of 2x2 Grid and 3x3 Grid Slide Window Binary Interpolation

Figure 3.5 (a) shows the raw data coming from the visible light sensor without binary interpolation. Figure 3.5 (b) is the result of 2x2 Grid Binary Interpolation, whose resolution is much lower than in Figure 3.5 (c), which is the result of 3x3 Grid Slide Window Binary Interpolation. In Figure 3.5 (b), aliasing effects also appear on the boundaries of blocks and letters. Such effects are eliminated in Figure 3.5 (c) due to the better interpolation algorithm.

The major concern when implementing the 3x3 Grid Slide Window method will be the boundary conditions. To handle the boundary, the first and last columns and rows should be detected correctly. If the pixel is on the boundary, the slide window should be 2x2. 3 neighbors rather than 8 neighbors should be used to compute the RGB information. After having implemented the algorithm without boundary detection, the output image will only have 4 lines on the boundary which are not exactly recovered. However, with a resolution of 752 x 480, these 4 lines could not be observed easily, thus putting no penalty on the quality of the image.

3.2 FPGA Implementation of Slide Window Binary Interpolation

The slide window method is easier said than done with FPGA. Naively, if 3 lines of raw data could be buffered, the computation would be very easy. However, the major concern is how to buffer the data. For the slide window implementation, three lines of data need to be buffered. However, the 2 SDRAM on board were used to read from sensor and write to HDMI port at the same time. Also, the register array on FPGA could not be larger than 100 bytes, otherwise, the routing would take hours to finish and the data read from the buffer would not necessarily be correct.

3.2.1 Input of the Image Processing Block

However, if we take a close look at the slide window algorithm, for each pixel, only a 3x3 matrix is needed to recover the color information of all 3 channels. And if we treat the 3x3

matrix as a shift register, 3 data from each line are needed for each clock cycle. Thus 2 FIFOs could be used to provide the data needed for slide window binary interpolation as shown in the figure below.

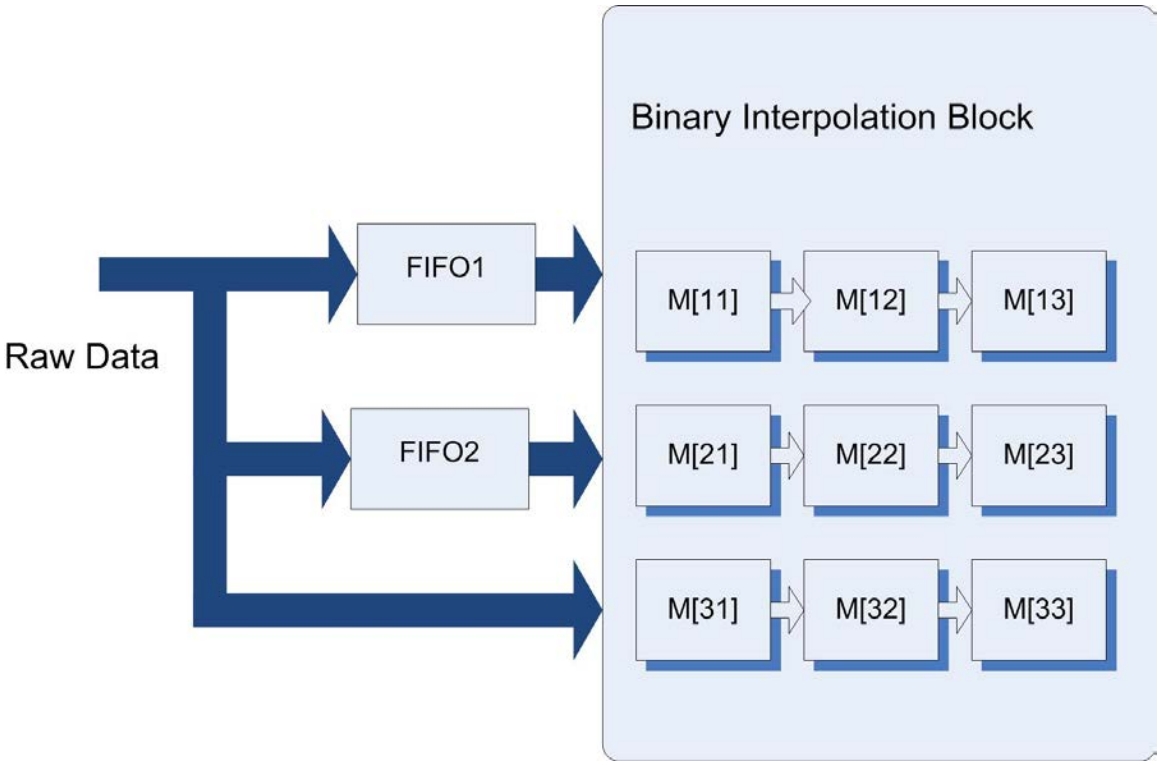


Figure 3.6: The Input of Binary Interpolation Block

The raw data coming from the visible light sensor will be buffered by 2 FIFOs. The write enable signal of the FIFO1 will begin from the first pixel of each frame, while the write enable of the FIFO2 will begin from:

$$FIFO2\_W_rE_n = 752 + 94 = 846 \tag{3.5}$$

So when the third line begins to come from the sensor, the first two lines of data are already inside FIFO1 while the second line of data is inside FIFO2, as shown below.

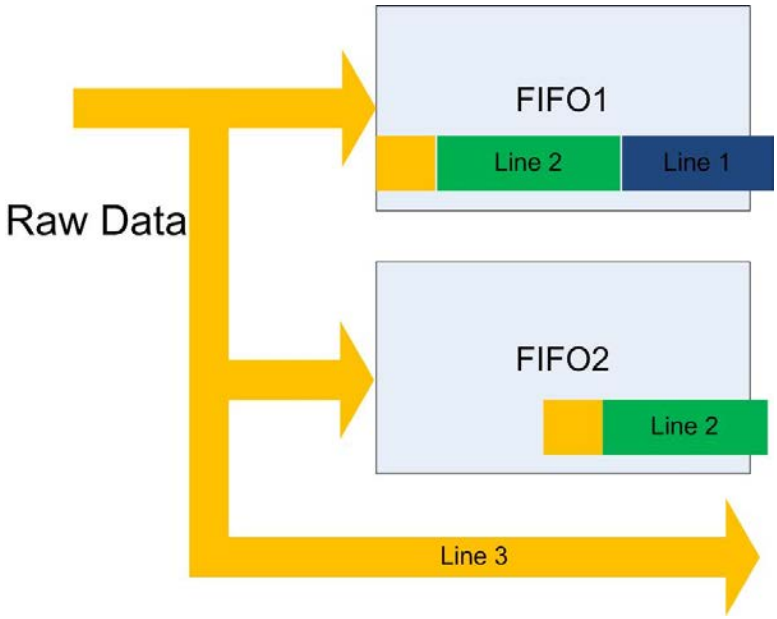


Figure 3.7: The Output Data from FIFOs

If we enable the binary interpolation block and the read ports of the two FIFOs at the beginning of the third row of each frame, then we have:

$$R_d E_n = (752 + 94) * 2 = 1692 \tag{3.6}$$

The binary interpolation block will receive the data from row 1, 2 and 3 at the same time. With the 3x3 register array inside the block, all the data have been acquired to enable that the RGB intensity of the center pixel to be recovered.

3.2.2 Shift Registers in Image Processing Block

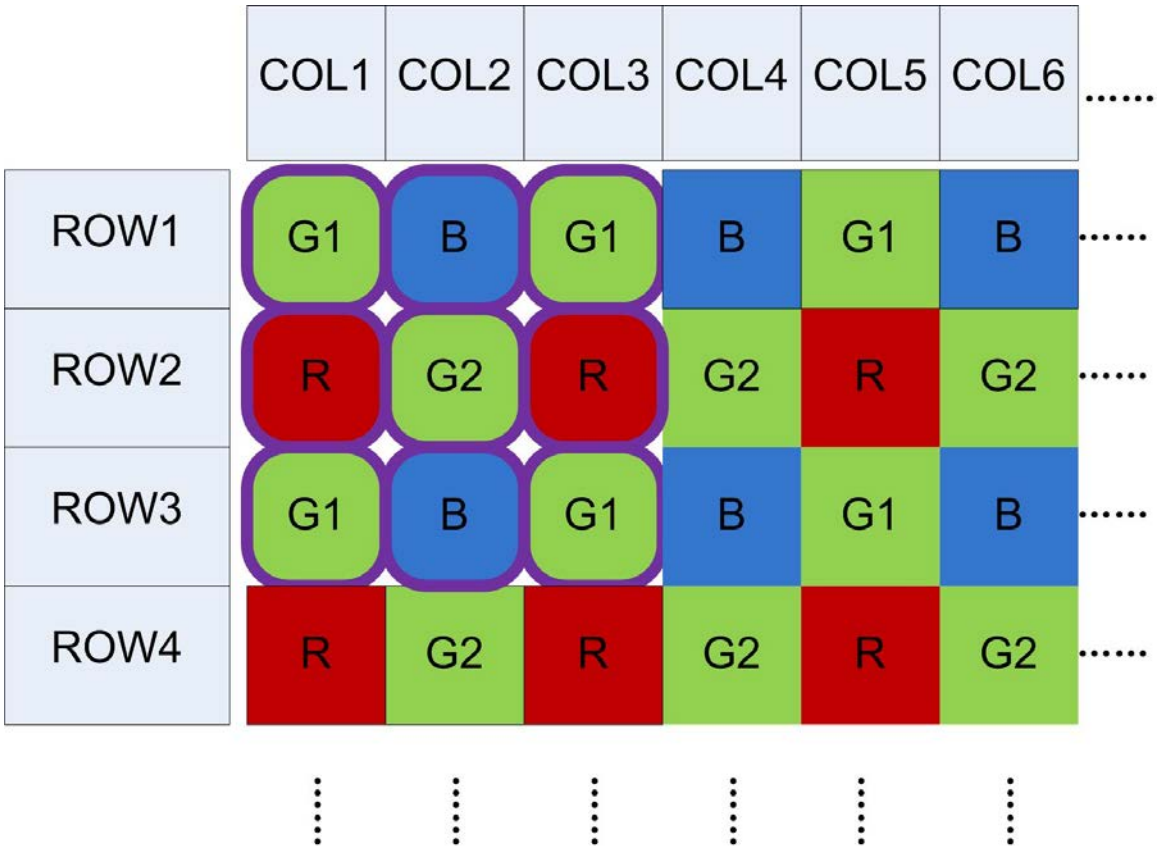


Figure 3.8: The Slide Window of First Pixel of Each Frame

For example, if we assume the pattern shown on the above graph is the first few rows and columns of a frame, then the values in the shift register are shown below:

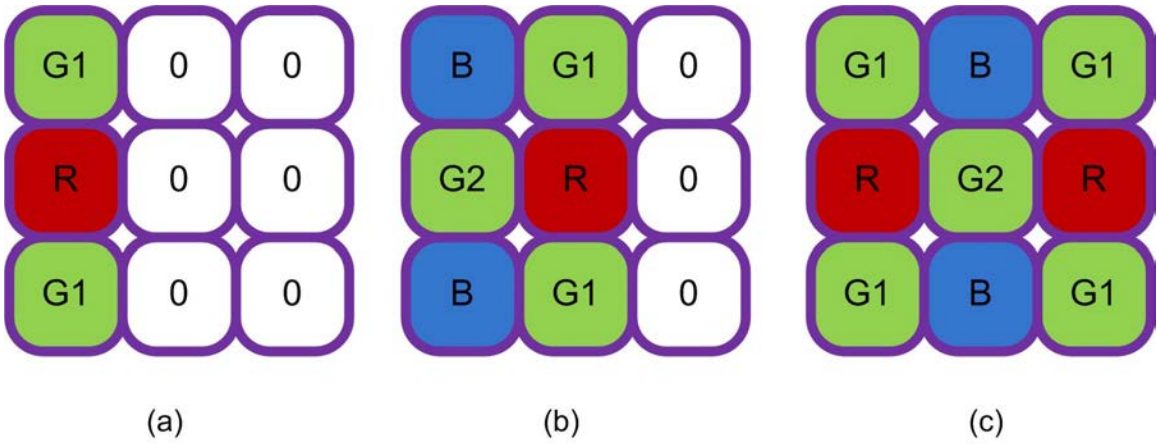


Figure 3.9: The Shift Register Status of First Input

Read enabling of both FIFO1 and FIFO2, along with the enable signal of ColorBinary block, will begin at the first column of 3rd row. Figure 3.9 (a) shows the values in the shift registers after the first clock cycle.

Because the values in the shift registers are set to 0, only the first column of the shift register read the value in. The rest of the columns remain 0. After the 2nd clock cycle, the first two columns of the shift register are filled with input data. The data in the second column comes from the data in the first column.

At the third clock cycle, the shift register is full with 9 data from the first 3 rows and columns of the current frame. The ColorBinary block is able to recover all 3 channels of information of pixel at row 2, column 2. Assuming the coordinates of the input pixel are (xin, yin), the relation between the position of the input pixel and output pixel is:

$$(x_{out}, y_{out}) = (x_{in} - 1, y_{in} - 1) \tag{3.7}$$

So for the pixel on the boundaries of the frame, the output will be a value with no actual meaning.

3.2.3 Binary Interpolation With Shift Registers

After acquiring of all the data needed for binary interpolation, the algorithm to interpolate the data was implemented in ColorBinary block. The first step of the interpolation is to find out what kind of pixel, G1, B, G2 or R, is the middle pixel of the 3x3 shift register matrix. 2 1-bit registers, row and col, are used to find that out. These two registers are initialized as 1, so the row and col value for the first valid pixel of the picture are two logic high values.

Table 3.1: Input and Output Pixel Mapping for Visible Light RGB Grid

Row	**Col**	**Input Pixel Type**	**Output Pixel Type**
1	1	G1	G2
1	0	B	R
0	1	R	B
0	0	G2	G1

The col register will change every clock cycle when the enable signal of the block is valid, and the row register will change every 846 cycles. Every time the row register changes value, the col register will be reset to default value, 1.

The output pixel type is the pixel type on the up left corner of the input pixel. For different type of the output pixel, the algorithm performs differently, thus generating a uniformly distributed color image.

Because the division on FPGA is bit shift, to simplify the computation, three 10-bit registers - tempR, tempG, tempB - are used to store the value. The eight MSBs will be the final output of the block.

When doing the computation, R and B are similar to each other, and G1 and G2 are also similar. So we take G2 and R as examples to show how the algorithm works.

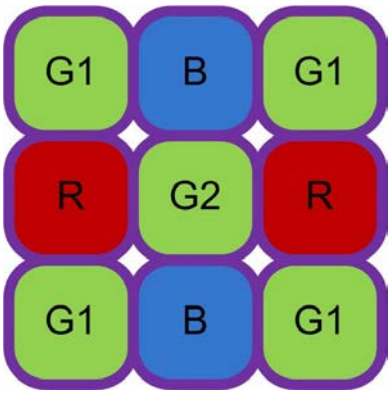


Figure 3.10: An Example of Interpolating G2

When G2 is the middle pixel in the shift register of a 3x3 shift register matrix, as shown above, the green channel value is the intensity of the middle pixel.

$$G2_G = tempG[9:2] = \{M[2,2], 00\} \tag{3.8}$$

The red and blue channels are very similar to equation (3.1) and (3.2).

$$G2_R = tempR[9:2] = \{0, M[2,1], 0\} + \{0, M[2,3], 0\} \tag{3.9}$$

Division by 2 is shifting right by 1 bit in FPGA. Since tempR, tempG, tempB are all 10-bits registers, the input data could be manipulated slightly to make the most significant 8 bits the output. For the blue channel of the G2 pixel, the equation is similar:

$$G2_B = tempR[9:2] = \{0, M[1,2], 0\} + \{0, M[3,2], 0\} \tag{3.10}$$

Another example is that the middle pixel is R pixel, as shown below.

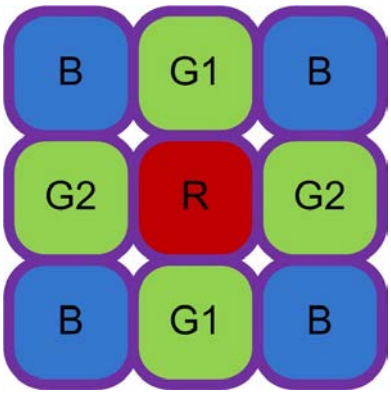


Figure 3.11: An Example of Interpolating R

To compute the intensity of the red channel is easy, because the value is the intensity of the middle pixel. Like equation (3.8), the value needs to be shifted due to the specified output configuration.

$$R_R = tempR[9:2] = \{M[2,2], 00\} \tag{3.11}$$

Because the green and blue channels are the average of 4 pixels, some modifications are needed for the function above to compute properly.

$$R_G = tempG[9:2] = \{00, M[1,2]\} + \{00, M[3,2]\} + \{00, M[2,1]\} + \{00, M[2,3]\} \tag{3.12}$$

$$R_B = tempB[9:2] = \{00, M[1,1]\} + \{00, M[3,1]\} + \{00, M[1,3]\} + \{00, M[3,3]\} \tag{3.13}$$

From equation (3.8) to equation (3.13), only addition is required for all equations to compute the output value, thus FPGA can compute the intensity of all 3 channels of every pixel in real-time.

3.3 FPGA Implementation of Superimposition

After the color image with the same resolution as the NIR image has been recovered, the next step is to fuse them together to generate the final image which contains both visible spectrum and NIR spectrum information. This section will discuss the idea behind superimposing images on FPGA first, and then explain the implementations in detail. For the beam splitter configuration, the disparity graph is shown below.

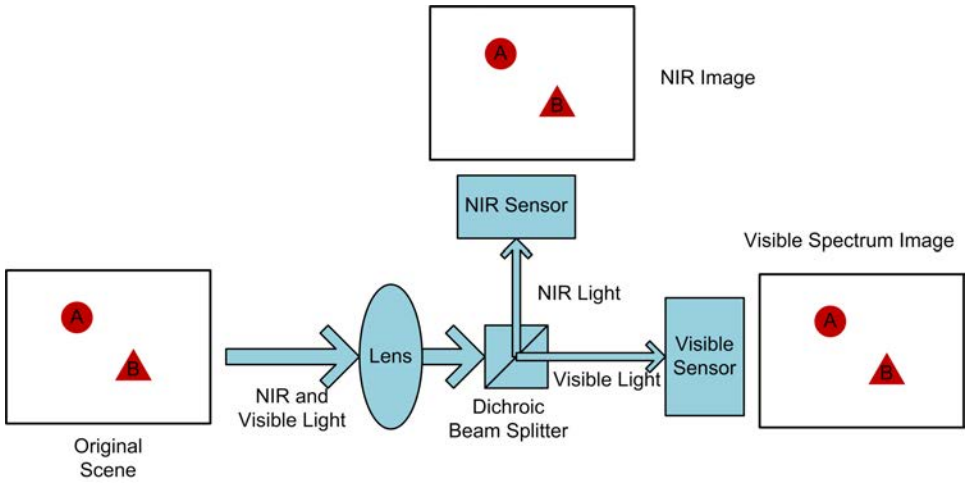


Figure 3.12: The Disparity of the Beam Splitter Configuration

Assuming that points A and B are at the same depth, the relationship of the disparity between them will be:

$$(X,Y)_{A,NIR} - (X,Y)_{A,VIS} = (X,Y)_{B,NIR} - (X,Y)_{B,VIS} \tag{3.14}$$

Because the disparity of the beam splitter configuration is almost constant and independent of the depth of object as shown in the equation above, this section mainly focuses on single PCB configuration.

Although the two sensors in the system are soldered on the same PCB board in the single PCB configuration, there are disparities along both X and Y axes, as shown in the graph below.

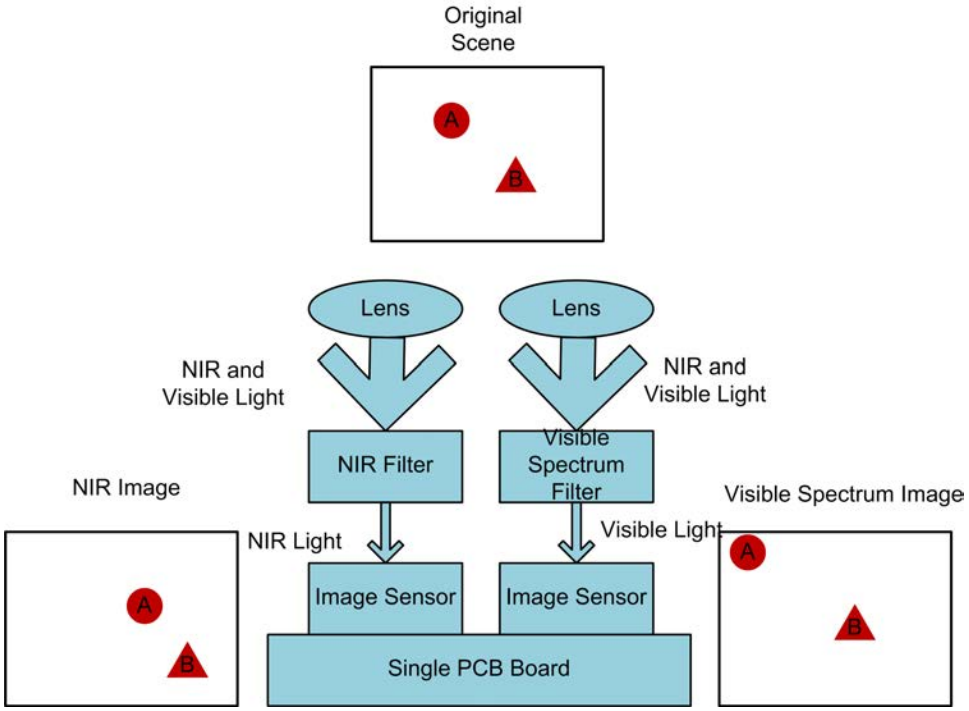


Figure 3.13: The Disparity of the Single PCB Configuration

Assuming that at points A and B are not on the same depth, the relationship of the disparity between them will be:

$$(X,Y)_{A,NIR} - (X,Y)_{A,VIS} \neq (X,Y)_{B,NIR} - (X,Y)_{B,VIS} \tag{3.15}$$

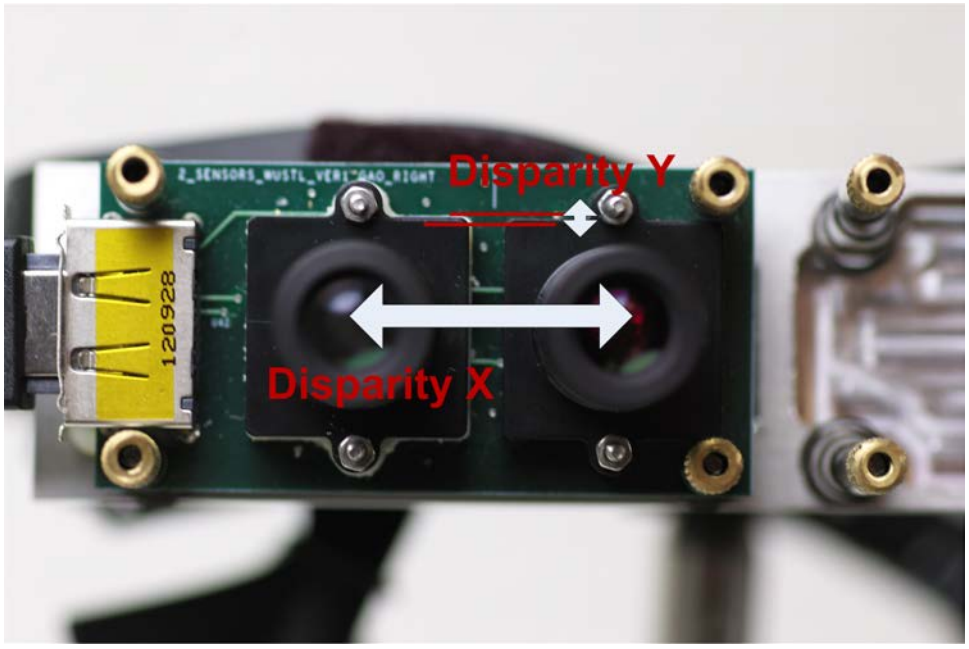


Figure 3.14: The Disparity of The Dual Sensors System

Due to the different locations along the the X axis of the two sensors, there will be disparity along the X axis (disparity_x); and due to the soldering of the sensor chips, there will be disparity along the Y axis (disparity_y). So disparity_x is more than one hundred pixels, while disparity_y is generally only a few pixels.

However, the disparities will be constant for a fixed depth of object. In this section, the method to fuse images on FPGA with a given disparity will be presented. No matter what kind of configuration is used, either the beam splitter configuration or the single PCB configuration given both X and Y axis disparities, a fused image can be generated in real-time on FPGA.

The general method of computing the disparity_x and disparity_y is to first find the features in the images from each sensor and then use the feature descriptor to match the features from different images. After that, the transformation of the matched features can be computed. However, since this method requires more memory and computation power than the board can afford, a simple method based on the particular system design is used.

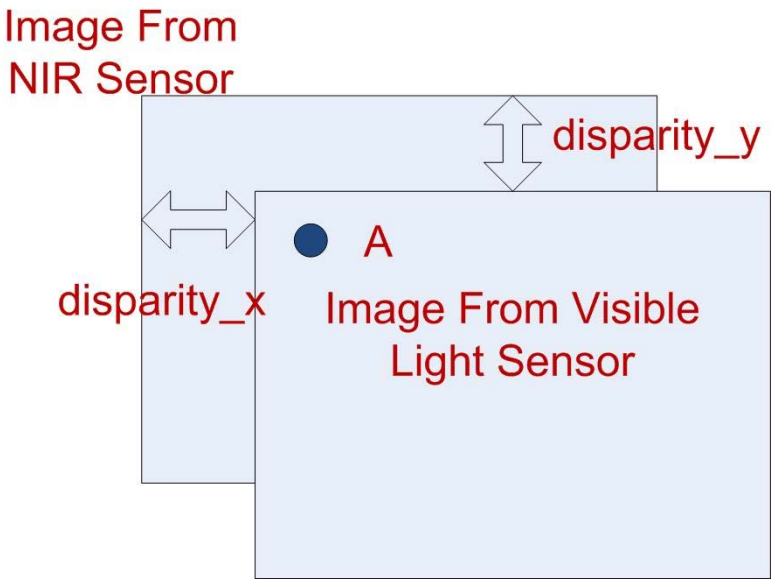


Figure 3.15: The Disparity of the Output from Dual Sensors System

In the above graph, the point A is on images from both sensors. Let the coordinates of A be (Xnir, Ynir) and (Xvis, Ynis) for NIR image and visible light sensor, respectively. The origins of both images are at the top left corner of the image. Then the disparity_x and disparity_y could be defined as:

$$disparity\_x = X_{nir} - X_{vis} \tag{3.16}$$

$$disparity\_y = Y_{nir} - Y_{vis} \tag{3.17}$$

In our system, disparity_x is always positive and the value is greater than 100 and less than 200. Also, disparity_y is a small positive number. If both disparity_x and disparity_y are positive, the problem can be solved easily. Another FIFO could be created to buffer the data coming from the NIR sensor, and a delay time could be added to the FIFO when reading the data out. The delay time could be defined as:

$$delay = 846 * disparity\_y + disparity\_x \tag{3.18}$$

Where 846 is the number of pixels in one row, so that the output NIR pixel of the FIFO corresponds to the same pixel in the visible light image. Thus, the output of the FIFO is used as an input to the image processing block. Before outputing the interpolated pixel value, a threshold will be compared with the pixel intensity of NIR sensor. If the intensity is larger than the threshold, the pixel will be highlighted as red in the final image, and the interpolated value will be ignored in this case.

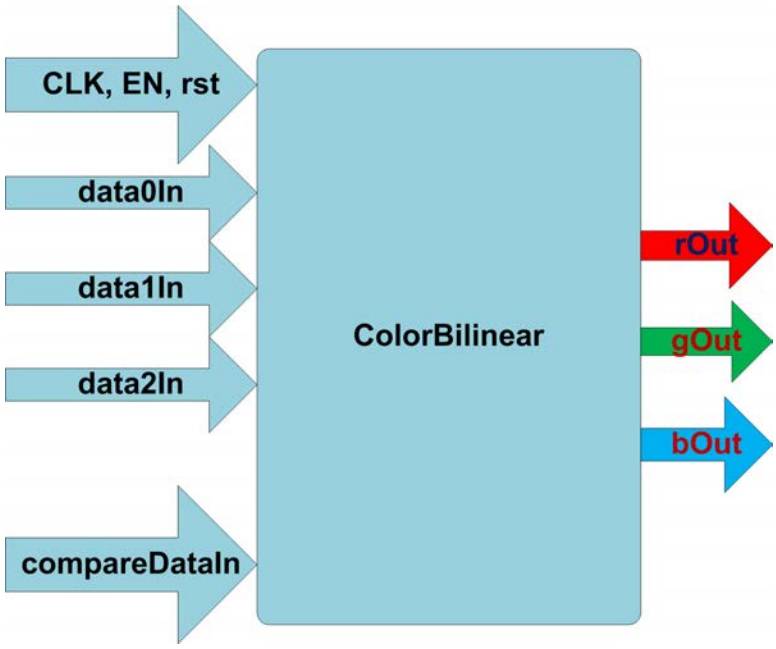


Figure 3.16: Block Diagram of Image Processing Block

The block diagram of the image processing block is shown above. There are four data input ports–data1In, data2In, data3In and compareDataIn–as introduced above. The data1In, data2In and data3In are pixel values from the same column but adjacent rows, and the compareDataIn is the intensity of the corresponding pixel from the NIR sensor.

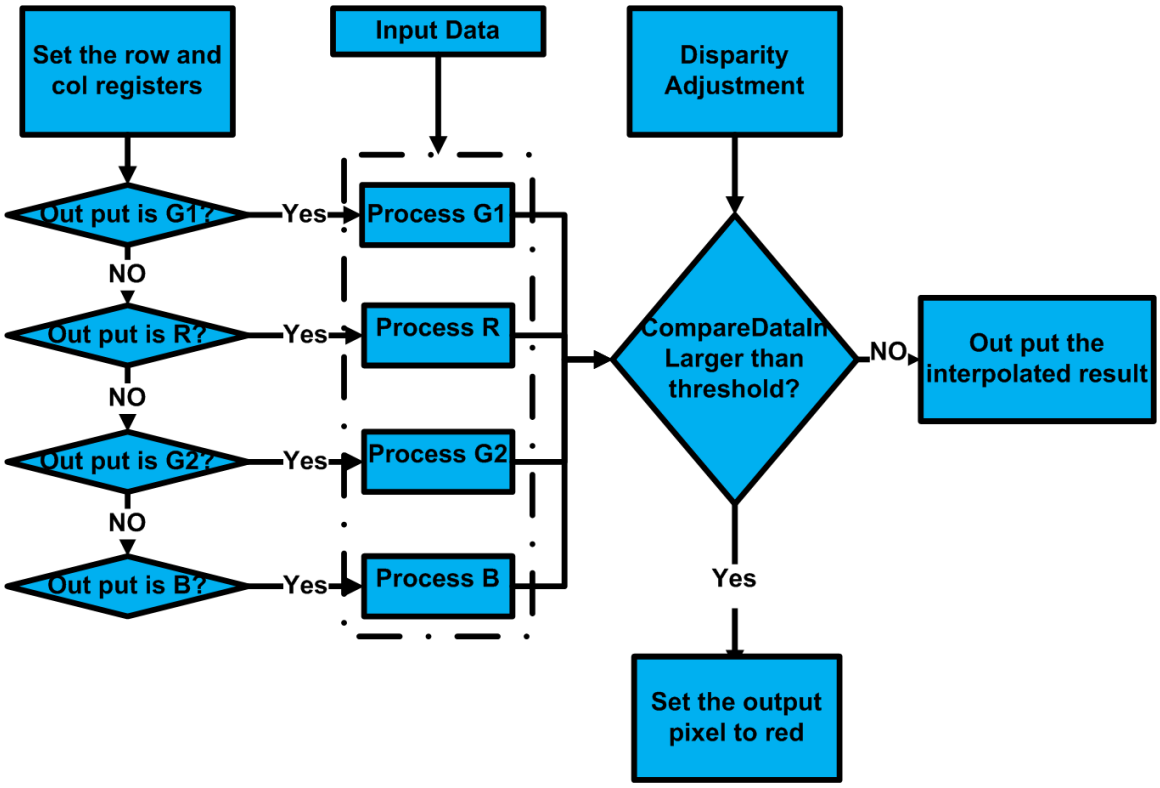


Figure 3.17: Flow Chart of Image Processing Block

The output of the block is the RGB value of a certain pixel as shown above. If compareDataIn is larger than the threshold, the red channel of output will be the intensity of compareDataIn. Green and blue channels are small constant numbers in this case, so the pixel is highlighted as red.

3.4 Conclusion

In this chapter, the whole process of the image processing block has been illustrated. The system was tested by tracking two 850nm LEDs on the same plane. Because the test was done on single PCB configuration, the disparities in both x and y directions depend on the distance between sensor and object, and a graphic user interface has been created to manually adjust the disparity_x and disparity_y for certain measurements to match the LEDs on both NIR and visible spectrum images.

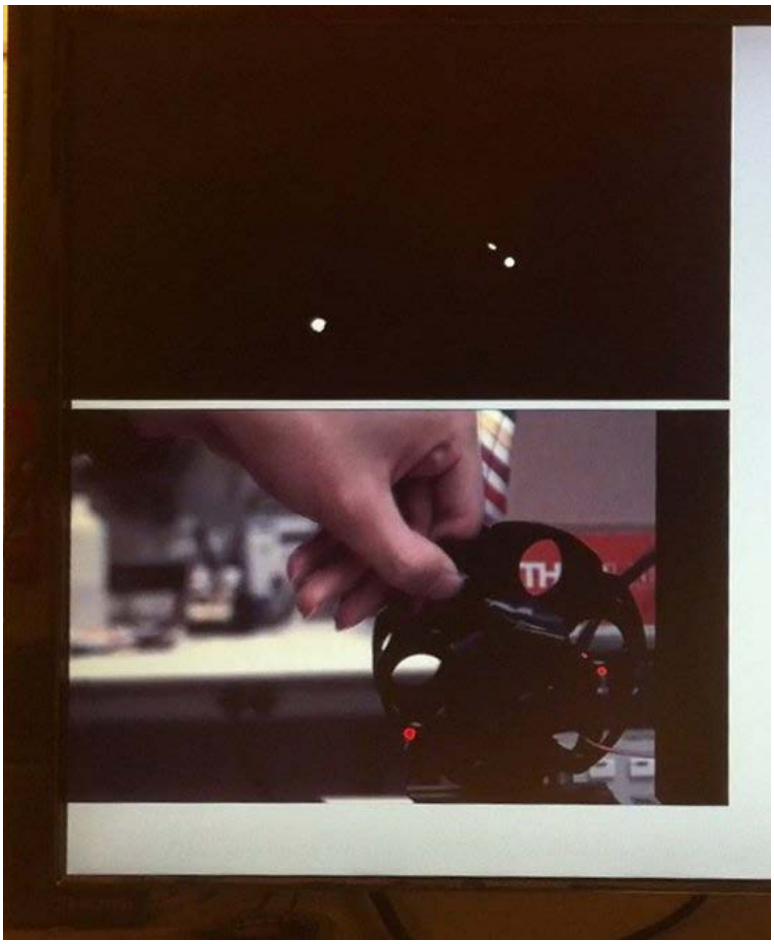


Figure 3.18: System Test Result with 850nm LEDs

Figure 3.18 shows the test result of the current system. All the image processing is done on the FPGA and the output is a screen shot of an HDMI monitor used to show the fused image. The upper part of the image is the output of an NIR sensor with an 800nm high pass filter. The white spot shows the location of the 850nm LEDs. The lower part of the graph is the fused image. The red spot overlaps very well with the actual LEDs' positions in the output image from viable light sensor.

The data flow of the whole FPGA design is shown in Figure 3.19.

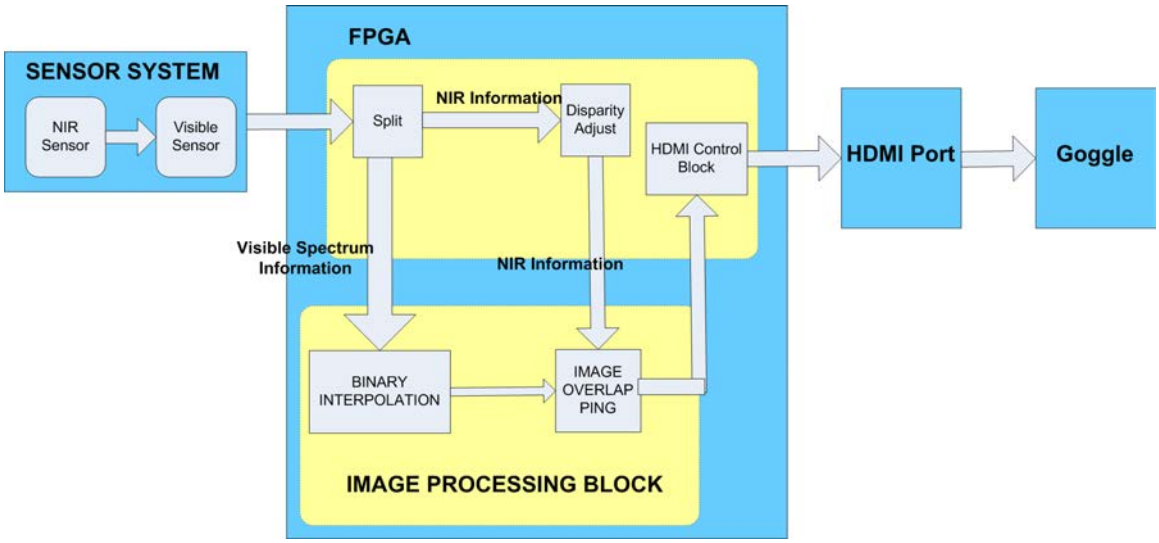


Figure 3.19: Data Flow Block Diagram of the System

The data coming from the slave sensor is 8-bits per cycle at 25MHz. The master sensor will combine the data into its own data package, and the output of the sensor system will be 16-bits width at 25MHz. Then the incoming data will be split into visible spectrum data and NIR data. Visible spectrum data will be passed to the image processing block for binary interpolation, while NIR data will be adjusted for disparity before going to the image processing block for superimposing. Then the final output of the image processing block will be 24-bits width RGB value. The output will go to the HDMI port for display.

Chapter 4

Goggle Imaging System Conclusion

This thesis describes a compact, real-time intraoperative system that can assist surgeons to identify tumor margins by using both near infrared and visible spectrum data. To provide a real-time display, an HDMI interface on a custom PCB board driven by FGPA was implemented and tested. The interface is able to output high definition resolution images of the surgical site at 24Hz.

To process images during surgery, a binary interpolation algorithm and superimposing block are implemented on an FPGA. The algorithm is optimized to run in real-time on FPGA and at the same frequency as the HDMI interface, hence online identification of tumors and tumor margins will be provided to surgeons.

The system is tested by observing the fluorescent dye which is injected into patients. The experimental configuration is shown below.

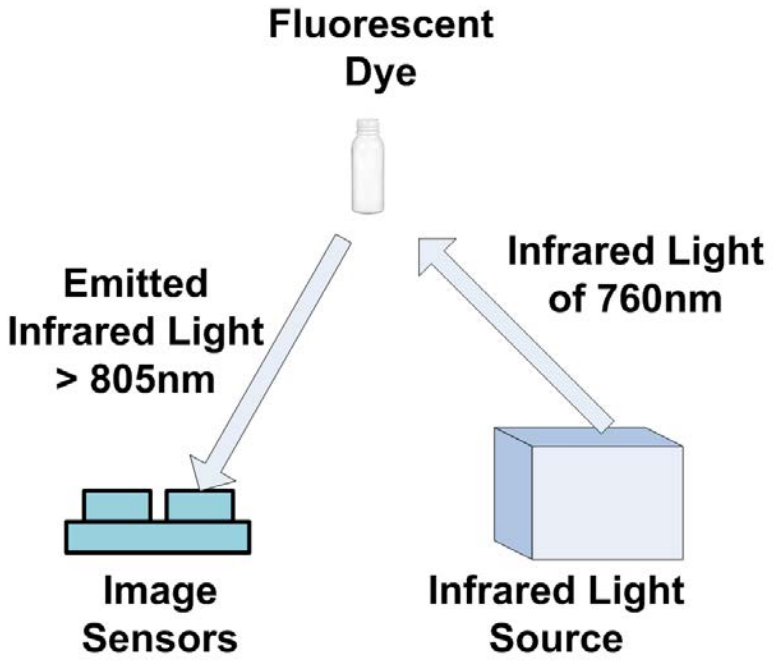


Figure 4.1: System Test Configuration

An infrared light source is used to shine 760nm infrared light onto the florescent dye. Then the dye will emit light with a wavelength higher than 805nm. With a long pass filter on top of the image sensor, the emitted light will be captured by the sensor system. The actual picture of the experimental setup is shown below.

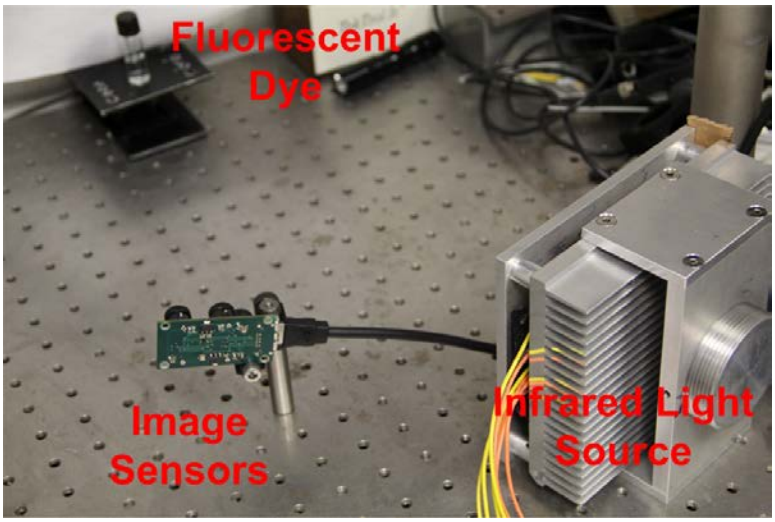


Figure 4.2: Actual System Test Configuration

The output of the system is shown below.

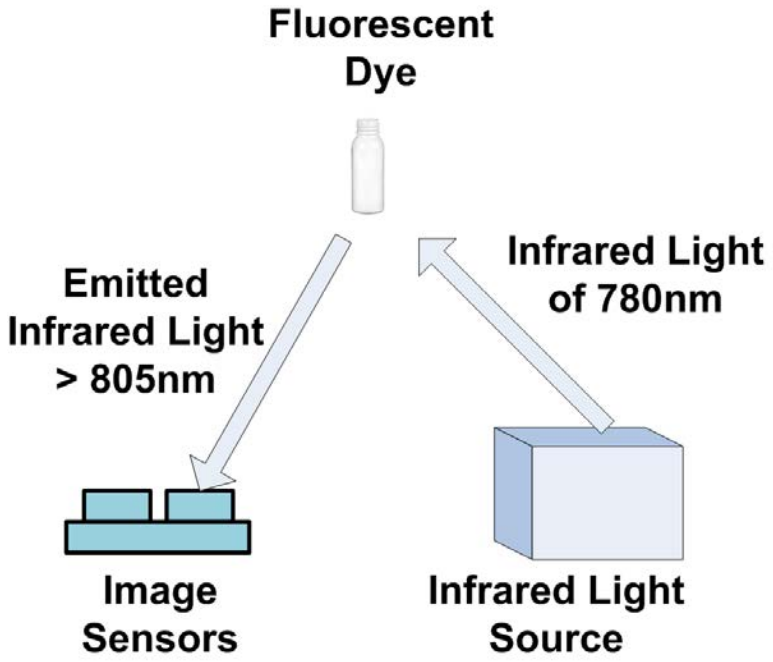


Figure 4.3: HDMI Output Result of the System when Observing Fluorescent Dye

The color image is the real-time processed final image that contains both visible and infrared spectrum information. The red color indicates the infrared information. It can be seen from the image that the two channels are well aligned, which means that the system will enable surgeons to identify tumors and tumor margins during surgery in real-time. As this method facilitates rapid scanning of the surgical site, the cancer recurrence risk will be much lower than with a traditional offline identification method, such as CT, MRI or PET.

Compared to the previous generation, an image processing block with interpolation and superimposing algorithms is developed on an FPGA, which removes the necessity of a PC-in-the-loop for image processing tasks without sacrificing image quality. To further improve the performance and weight of the system, a real-time display block through the HDMI interface was developed based on a custom HDMI PCB and an I2C control block on FPGA. All necessary image processing and transmission blocks have been developed on FPGA, which allows for a compact, easily portable and lightweight imaging system for intraoperative purposes.

In conclusion, the goggle system is able to process images in both the visible and near infrared spectrum in real-time, thus providing feedback to surgeons during the surgery. The system is based on FPGA and no PC is required in the loop, so the cost is very low and the weight is light, making it accessible to developing countries and rural areas. All of the following tasks have been achieved: 1) Real-time acquisition of both visible and near infrared spectrum images from a pair of CMOS imaging detectors; 2) Real-time image processing for both near infrared and color images in order to enhance the captured information; and 3) Real-time display of fused visible-near infrared images in high definition (HDMI) format on a goggle device.

Bibliography

[1] Adam Q Liu Y FAU Bauer, Walter J Bauer AQ FAU Akers, Gail Akers WJ FAU Sudlow, Kexian Sudlow G FAU Liang, Duanwen Liang K FAU Shen, Mikhail Y Shen D FAU Berezin, Joseph P Berezin MY FAU Culver, Samuel Culver JP FAU Achilefu, and Achilefu S. Hands-free, wireless goggles for near-infrared fluorescence and real-time image-guided surgery.

[2] John M. Kurtz, Robert Amalric, Henri Brandone, Yves Ayme, Jocelyne Jacquemier, Jean-Claude Pietra, Daniel Hans, Jean-Franois Pollet, Claude Bressac, and Jean-Maurice Spitalier. Local recurrence after breast-conserving surgery and radiotherapy. frequency, time course, and prognosis. *Cancer*, 63(10):1912–1917, May 1989.

[3] Anne Moyer. Psychosocial outcomes of breast-conserving surgery versus mastectomy: A meta-analytic review., 1997.

[4] N Noguchi M FAU Katev, I Katev N FAU Miyazaki, and Miyazaki I. Clinical and histologic determinations of adequate breast resection in breast-conserving surgery. a review.

[5] Lisa Jacobs. Positive margins: The challenge continues for breast surgeons. 15(5):1271–1272–, 2008.

[6] S.; Achacoso N.; Haque R.; Nekhlyudov L.; Fletcher S.; Quesenberry C.; Habel L. Collins, L.; Schnitt. utcome of women with ductal carcinoma in situ (dcis) treated with breast-conserving surgery alone: A case-control study of 225 patients from the cancer research network. *Modern Pathology*, 22:34A–35A, 2009.

[7] Frank A Vicini, Larry L Kestin, Neal S Goldstein, Peter Y Chen, Jane Pettinga, Robert C Frazier, and Alvaro A Martinez. Impact of young age on outcome in patients with ductal carcinoma-in-situ treated with breast-conserving therapy. *Journal of clinical oncology*, 18(2):296–296, 2000.

[8] Giorgio Zavagno, Elena Goldin, Roberto Mencarelli, Giovanni Capitanio, Paola Del Bianco, Renato Marconato, Simone Mocellin, Giorgia Marconato, Valentina Belardinelli, and Francesca Marcon. Role of resection margins in patients treated with breast conservation surgery. *Cancer*, 112(9):1923–1931, 2008.

[9] Charles Kunos, Larry Latson, Beth Overmoyer, Paula Silverman, Robert Shenk, Timothy Kinsella, and Janice Lyons. Breast conservation surgery achieving¿=2 mm tumor-free margins results in decreased localregional recurrence rates. *The breast journal*, 12(1):28–36, 2006.

[10] Silvia Skripenova and Lester J Layfield. Initial margin status for invasive ductal carcinoma of the breast and subsequent identification of carcinoma in reexcision specimens. *Archives of pathology & laboratory medicine*, 134(1):109–114, 2010.

[11] Shigeki Arii, S Tanaka, Y Mitsunori, N Nakamura, A Kudo, N Noguchi, and T Irie. Surgical strategies for hepatocellular carcinoma with special reference to anatomical hepatic resection and intraoperative contrast-enhanced ultrasonography. *Oncology*, 78(Suppl. 1):125–130, 2010.

[12] Mark G van Vledder, Michael S Torbenson, Timothy M Pawlik, Emad M Boctor, Ulrike M Hamper, Kelly Olino, and Michael A Choti. The effect of steatosis on echogenicity of colorectal liver metastases on intraoperative ultrasonography. *Archives of Surgery*, 145(7):661–, 2010.

[13] Osamu Ukimura, Koji Okihara, Kazumi Kamoi, Yoshio Naya, Atsushi Ochiai, and Tsuneharu Miki. Intraoperative ultrasonography in an era of minimally invasive urology. *International journal of urology*, 15(8):673–680, 2008.

[14] Robert A Kane. Intraoperative ultrasonography history, current state of the art, and future directions. *Journal of Ultrasound in Medicine*, 23(11):1407–1420, 2004.

[15] Markus Rudin and Ralph Weissleder. Molecular imaging in drug discovery and development. *Nature Reviews Drug Discovery*, 2(2):123–131, 2003.

[16] Lee G Wilke, J Brown, Torre M Bydlon, Stephanie A Kennedy, Lisa M Richards, Marlee K Junker, Jennifer Gallagher, William T Barry, Joseph Geradts, and Nimmi Ramanujam. Rapid noninvasive optical imaging of tissue composition in breast tumor margins. *The American Journal of Surgery*, 198(4):566–574, 2009.

[17] Susan L Troyan and John V Frangioni. The flare((tm)) intraoperative near-infrared fluorescence imaging system: a first-in-human clinical trial in breast cancer sentinel lymph node mapping. *Annals of surgical oncology*, 16(10):2943–2952, 2009.

[18] Matthew D Keller, Shovan K Majumder, and Anita Mahadevan-Jansen. Spatially offset raman spectroscopy of layered soft tissues. *Optics letters*, 34(7):926–928, 2009.

[19] J. Q.; Barry W. T.; Geradts J.; Wilke L. G.; Kennedy S. A.; Richards L. M.; Junker M. K.; Ramanujam N. Bydlon, T. M.; Brown. Rapid optical imaging of breast tumor margins: Final results from a 100-patient clinical study. *Cancer Research*, 69:770S–771S, 2009.

[20] Kumar R Bhushan, Preeti Misra, Fangbing Liu, Sanjeev Mathur, Robert E Lenkinski, and John V Frangioni. Detection of breast cancer microcalcifications using a dual-modality spect/nir fluorescent probe. *Journal of the American Chemical Society*, 130(52):17648–17649, 2008.

[21] Ana Carolina de Miranda Marzullo, Osmar Pinto Neto, Renata Andrade Bitar, Herculano da Silva Martinho, and Airton Abraho Martin. Ft-raman spectra of the border of infiltrating ductal carcinoma lesions. *Photomedicine and laser surgery*, 25(5):455–460, 2007.

[22] Adam M Zysk and Stephen A Boppart. Computational methods for analysis of human breast tumor tissue in optical coherence tomography images. *Journal of biomedical optics*, 11(5):054015–054015, 2006.

[23] Abigail S Haka, Zoya Volynskaya, Joseph A Gardecki, Jon Nazemi, Joanne Lyons, David Hicks, Maryann Fitzmaurice, Ramachandra R Dasari, Joseph P Crowe, and Michael S Feld. In vivo margin assessment during partial mastectomy breast surgery using raman spectroscopy [? q1: Running head: Raman margin assessment at partial mastectomy. short title ok? q1]. *Cancer Research*, 66(6):3317–3322, 2006.

[24] Robert E Lenkinski, Muneeb Ahmed, Atif Zaheer, John V Frangioni, and S Nahum Goldberg. Near-infrared fluorescence imaging of microcalcification in an animal model of breast cancer¡ sup¿ 1¡/sup¿. *Academic radiology*, 10(10):1159–1164, 2003.

[25] Karel J Zuzak, Michael D Schaeberle, Mark T Gladwin, Richard O Cannon, and Ira W Levin. Noninvasive determination of spatially resolved and time-resolved tissue perfusion in humans during nitric oxide inhibition and inhalation by use of a visible-reflectance hyperspectral imaging technique. *Circulation*, 104(24):2905–2910, 2001.

Real-time Image Processing on FPGA, Zhang, M.S. 2014