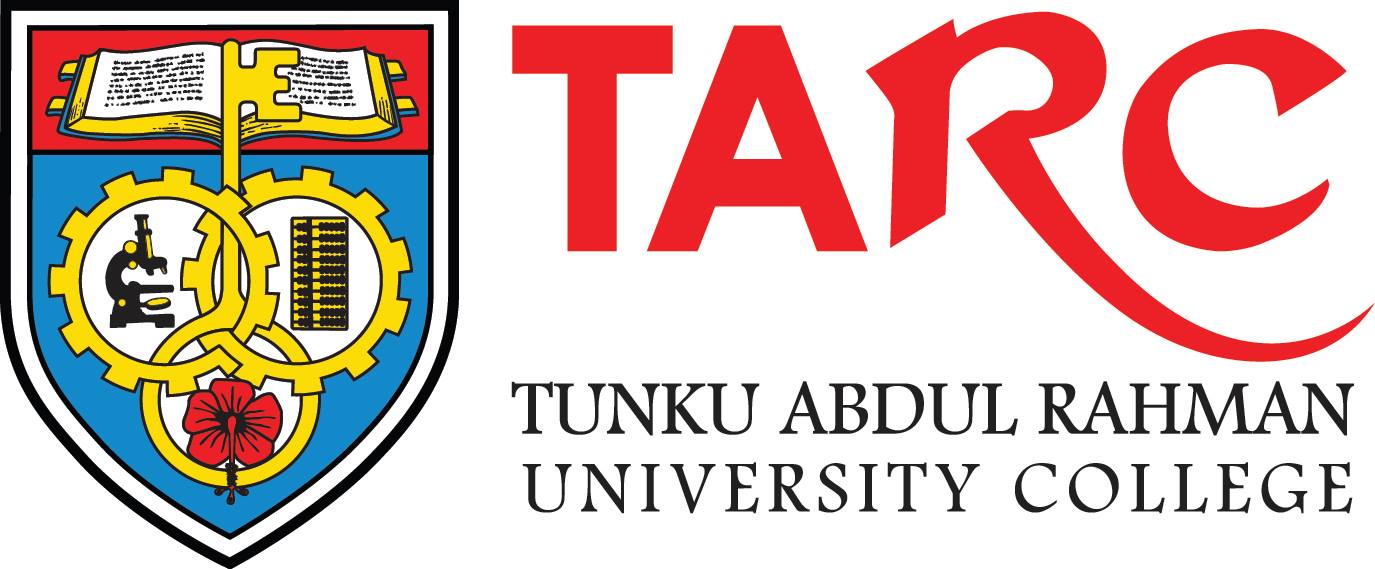
**Development of an FPGA based Iris Recognition System:**

**Image Acquisition and Segmentation**

**By**

**Yap Ken Mun**



**Faculty of Applied Sciences and Computing**

**Tunku Abdul Rahman University College**

**Kuala Lumpur**

**2015/2016**

Final Year Project

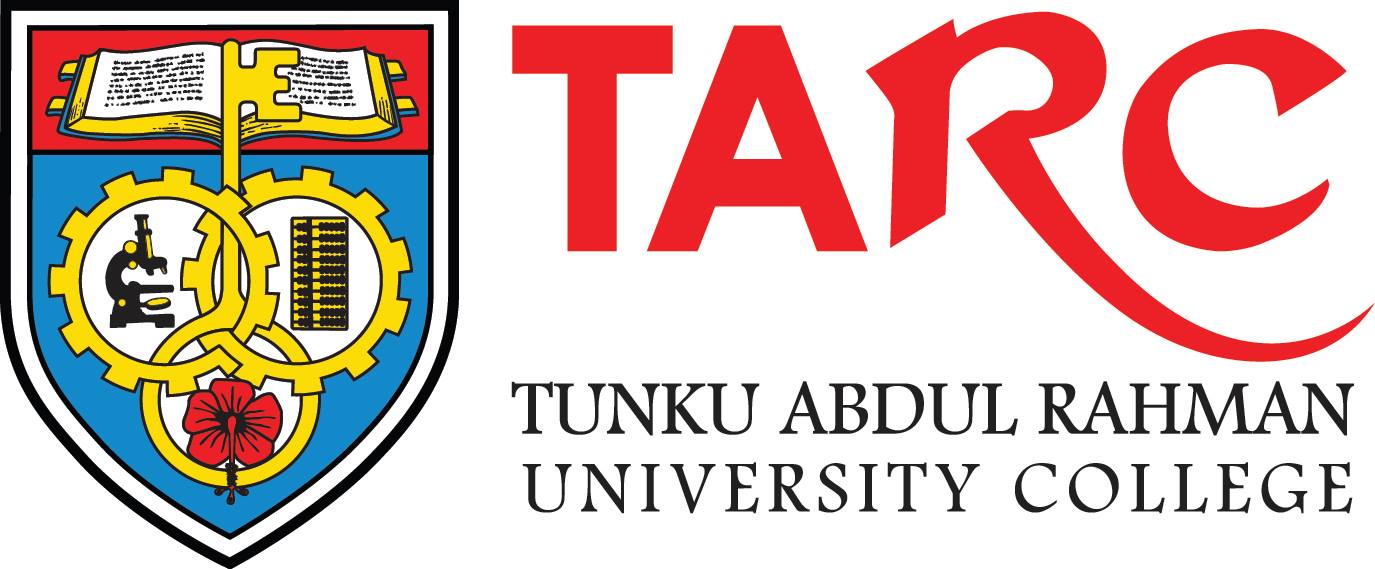
Development of an FPGA based Iris Recognition System:

Image Acquisition and Segmentation

By

Yap Ken Mun

Project supervisor: Miss Michelle Lim Sern Mi



This is a project dissertation submitted to the Faculty of Applied Sciences and Computing in partial fulfillment of the requirement for the award of Bachelor of Science Degree, Tunku Abdul Rahman University College.

Department of Physical Science

Faculty of Applied Sciences and Computing

Tunku Abdul Rahman University College

Kuala Lumpur

Contents

[**Acknowledgement: 1**](#_Toc440464018)

[**Abstract 2**](#_Toc440464019)

[**Chapter 1: Introduction 3**](#_Toc440464020)

[**OBJECTIVES 3**](#_Toc440464021)

[**PROBLEM STATEMENT 3**](#_Toc440464022)

[**Chapter 2: Literature Review 7**](#_Toc440464023)

[**2.1 Conventional Techniques for Image/Iris Recognition 7**](#_Toc440464024)

[2.1.1 Image Acquisition 7](#_Toc440464025)

[2.1.2 Pupil Boundary Detection 8](#_Toc440464026)

[2.1.3 Iris Boundary Detection 9](#_Toc440464027)

[**Chapter 3: Methodology 10**](#_Toc440464028)

[**3.1 CMOS Image Sensor 14**](#_Toc440464029)

[**3.2 I²C Configuration 16**](#_Toc440464030)

[**3.3 CMOS Sensor Data Capture 19**](#_Toc440464031)

[**3.4 RAW to RGB Conversion 21**](#_Toc440464032)

[**3.5 SDRAM 23**](#_Toc440464033)

[**3.6 Image Processing and Segmentation 24**](#_Toc440464034)

[3.6.1 RGB to Grayscale 25](#_Toc440464035)

[3.6.2 Grayscale to Binary Image 26](#_Toc440464036)

[3.6.3 Binary Image to Erosion 27](#_Toc440464037)

[3.6.4 Erosion to Dilation 27](#_Toc440464038)

[3.6.5 Dilation for Pupil Point Detection 28](#_Toc440464039)

[3.6.6 Binary Image for Iris Point Detection 30](#_Toc440464040)

[**3.7 VGA Display 32**](#_Toc440464041)

[**Chapter 4: Results and Discussions 33**](#_Toc440464042)

[**4.1 CMOS Image Sensor 33**](#_Toc440464043)

[**4.2 I2C Configuration 34**](#_Toc440464044)

[**4.3 CMOS Sensor Data Capture 36**](#_Toc440464045)

[**4.4 RAW to RGB 36**](#_Toc440464046)

4.5 Image Processing and Segmentation

[4.5.1 RGB to Grayscale 37](#_Toc440464047)

[4.5.2 Grayscale to Binary Image Conversion of the Pupil 38](#_Toc440464048)

[4.5.3 Binary Image to Erosion 39](#_Toc440464049)

[4.5.4 Dilation 40](#_Toc440464050)

[4.5.5 Pupil and Iris Points Detection 41](#_Toc440464051)

[**4.6 Iris Acquisition and Segmentation 42**](#_Toc440464052)

[**4.7 Integrated Result 44**](#_Toc440464053)

[**4.8 Analysis: 47**](#_Toc440464054)

[**Chapter 5: Conclusion 48**](#_Toc440464055)

[**5.1: Future Trends & Recommendation 48**](#_Toc440464056)

[**References: 49**](#_Toc440464057)

[**Appendix: 50**](#_Toc440464058)

# Acknowledgement:

First, I would like to express my gratitude towards my Final Year Project (FYP) supervisor, Miss Michelle Lim Sern Mi. She had given much guidance, different opinions and ideas to solve problems and much encouragement in my final year project and Bachelor Degree studies. Thanks to her patience on listening and providing us with a good guideline to separate the entire problem into smaller and more manageable portions. She has also guided us during pre-presentation and pre-demonstration which has given me more cconfidence before the actual presentation day. Overall, her suggestions on theory and coding had led us towards the final integration and an on-time completion of this work.

Furthermore, I would like to thank my group mates, Jason Chuah Kwong Hooi and Lee Shyan Feng. They have also given me plenty of support and encouragement when faced with challenges associated with HDL based coding.

# Abstract

This work focuses on the image acquisition and iris segmentation portion of an Iris Recognition System (IRS) which is the initial task before performing iris post-processing, compression and recognition. The proposed iris acquisition uses a 5 Mega pixels Terasic CMOS camera to capture RAW image which is converted into an RGB format before performing segmentation using the Circle Hough Transform technique. This technique includes the process of RGB to grayscale and binary conversion by using thresholding, erosion, dilation for noise reduction and finally iris and pupil localization. This work solves timing problems of conventional iris segmentation schemes by not using Canny Edge detection. All sub-blocks related to the detection and iris pre-processing will be modeled, designed and tested using HDL based testbench in Modelsim before performing final integration, analysis and implementation of the Verilog HDL code into the Cyclone II FPGA for hardware verification purposes. The expected results of this captured and segmented iris is a 640x480 matrix with an accuracy of 71%. Such a contactless biometric recognition system can be applied for augmented security purposes such as criminal recognition and for home security.

## 

# Chapter 1: Introduction

## OBJECTIVES

1. To investigate several different design methods of image acquisition and iris segmentation of the Iris Recognition System (IRS) from past literatures.
2. To model, design and verify using HDL based test-bench for all sub-modules of the image acquisition and iris segmentation using ModelSim of Mentor Graphics environment.
3. To integrate, analyze and perform hardware implementation on all the integrated sub-modules of the image acquisition and iris segmentation design on the Cyclone П FPGA.

## PROBLEM STATEMENT

The proposed work addresses the problems faced by past researches as shown in Figure 1.1 and Figure 1.2 due to the unclear image and timing of Edge Detection. Figure 1.1 shows a conventional block by Terasic, this block represents basic image capture and output display. However, the conventional block in Figure 1.1 suffers from unclear image when displaying data to the VGA. The conventional block shown in Figure 1.2 is the process of Edge Detection. This process leads to timing issues during the processing stage of the iris recognition system.

The proposed architecture will resolve the unclear image of Figure 1 and timing issues of Figure 2. In the proposed architecture, the exposure time will be reduced for a faster capture. This will resolve the problem of blurred images. To resolve the problem of timing, the edge detection step is removed and will instead be replaced with a binary image detection module .

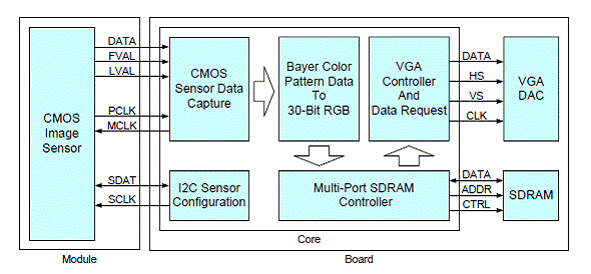


Figure 1.1: Conventional Block Diagram of Image Processing without Segmentation for Iris Recognition (Terasic 2014)

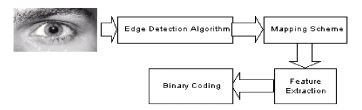
****

Figure 1.2: Conventional Block Diagram of Image Processing with Segmentation for Iris  
Recognition (International Journal of Advances in Engineering & Technology, May 2011)

**1.3** BRIEF BACKGROUND

The first general concept of iris recognition was introduced by Dr. Leonard Flom and Aron Safir in 1980s. Later John Daugman had developed, patented and demonstrated the first actual algorithm that can perform iris recognition (Iris 1999). Iris Recognition System is a biometric system; it is successfully implemented to the security system in airport boarding area to replace the usage of passport. Each person has a specific set of iris code that cannot be copied or be changed, this can help enhance the security system for terrorist or illegal immigrants boarding an airplane (Daughman 2002). In the future, iris recognition system will be installed at Automated Teller Machine (ATM) for user identification, car locking system and other applications that requires high security; especially for banking related devices and highly secured government area.

Iris Recognition System (IRS) in this work is implemented by using a type of Artificial Neural Network (ANN) algorithm for recognition purposes in augmented security systems. The IRS is separated into three blocks i.e. the Image Acquisition and Iris Segmentation Blocks, the Image Post Processing and Compression Blocks and the ANN Blocks. Here, Image Acquisition and Iris Segmentation will be focused, the two main tasks involved are to detect and segment. The following paragraphs will explain each of these tasks in detail.

Detection is the first step in a Real-Time Iris Recognition System (RTIRS). It uses a CMOS camera with a pre-determined distance, angle and with some manual process to capture the human eye. Each individual has a different height, which will influence the accuracy or the clarity of the captured iris image.

Segmentation is the process to separate the iris and pupil. It will begin with edge detection to detect the iris section. Pupil size can be calculated using the Hough Transform Technique. Edge Detection is used to detect the pupil boundary with more ease and accuracy. Circular edge detection is to detect the circle present in an image which has a maximum gray level difference compare to its neighbor. Table 1 shows the different segmentation techniques, where the Hough Transform Techniques has the highest segmentation accuracy of 98.9%.

Table 1: Accuracy of Different Segmentation Techniques (Eng. Nadeer Ataya. 2012)

|  |  |  |
| --- | --- | --- |
| **References** | **Segmentation Techniques** | **Accuracy (%)** |
| Prateek Verma (2014) | Wildes | 86.64 |
| Eng.Nadeer Ataya(2012) | Hough Transform | 98.9 |
| Eng.Nadeer Ataya(2012) | Daughman Integro-Differential | 98.6 |

In a nutshell, this report is a design and development of iris acquisition and segmentation implemented on FPGA. It begins with an introduction to the background, past to current research on the topologies and strategies of conventional design in Chapter 1. Chapter 2 covers a review on past literatures. Chapter 3 proposes the design methodology while the results and discussions are summarized in Chapter 4 and finally Chapter 5 concludes this work.

# Chapter 2: Literature Review

## 2.1 Conventional Techniques for Image/Iris Recognition

### 2.1.1 Image Acquisition

Image quality is one of the most important criteria in any iris recognition system. This will depend all of the following processes which will affect the accuracy of the entire system. A Charge-Couple Device (CCD) camera can be used to capture the iris image with resolution of 640x480 in black and white mode. Furthermore, to capture a consistent iris image, proper lighting and distance of the camera needs to be configured (IJAET 2011).

Moreover, most of the iris biometric systems are still constrained by the image acquisition conditions (Ramadan Gad 2015). There are several iris recognition researches that uses near infrared illumination, within 700-900nm range to capture a clear iris image for their iris database. The famous research institutions for iris recognition includes the Chinese Academy of Sciences (CAS), Institute of Automation (IA) and many more. Table 2.1 shows several Iris Image database by varying research companies using various CCD cameras to capture iris image of 640x480 in size with near infrared (NIR) illumination of 700-900nm in wavelength (Ramadan Gad 2015).

Table 2.1 Different Iris Database with Varying CCD Camera Usage

|  |  |  |  |
| --- | --- | --- | --- |
| **Database** | **No.of Irises** | **No. of Images** | **Camera used** |
| CASIA 1 | 108 | 756 | CASIA Camera |
| CASIA 3 | 1500 | 22051 | CASIA Camera and OKI irispass-h |
| ICE 2005 | 244 | 2953 | LG2200 |
| ICE 2006 | 480 | 60000 | LG2200 |
| MMU 1 | 90 | 450 | LG Iris Access |
| MMU 2 | 199 | 995 | Panasonic BM-ETU100US Authenticam |
| UBIRIS | 241 | 1877 | Nikon E5700 |
| UPOL | 128 | 384 | SONY DXC-950P 3CCD |

### 2.1.2 Pupil Boundary Detection

The darkest region in the iris image after grayscale conversion is the pupil portion. Firstly, the image is converted into a binary image by using linear threshold transformation to filter the bright pixels. After that, the weight of each pixel is used to approximate the center point of the pupil. The maximum circular summation of the gradient point into the circle is used to calculate the radius. Figure 2.1 shows the flow chart for pupil point detection (Ramadan Gad 2015).

Grayscale

Binary Image

1. weight of centroid pixels
2. maximum circular summation

Figure 2.1 Flow Chart of Pupil Point Detection

Another method for pupil detection uses the same steps in the first and second stage as shown in Figure 2.1. After the 2nd step, the image needs to be added to an average filter to reduce the noise of some pixels. The 3rd step is used to scan the binary image from top left horizontally, using 8-neighbour method moving from left to right to obtain the point of the first and last 8-neighbour that contains the value of 1. The 4th step is to scan vertically from top to bottom and obtain the first top and last bottom point of the 8-neighbour that contains a value of 1. Table 2.2 shows the past result of pupil detection accuracy in different segmentation stages (Ifeanyi and Ghazali YEAR).

|  |  |  |
| --- | --- | --- |
| Stages of Segmentation | Percentage Accuracy | Percentage Error |
| Pupil Localization | 99.10 % | 0.90 % |
| Iris Detection | 98.10 % | 2.00 % |
| Eyelids and Eyelashes Occlusion | 99.40 % | 0.60 % |
| Complete Iris Segmentation | 98.90 % | 1.10 % |

Table 2.2 Accuracy of Pupil Detection for Different Segmentation Stages(Ref)

### 2.1.3 Iris Boundary Detection

Circular Hough Transform is being used to detect the iris boundaries. Canny edge detection draws the boundary of iris; the first step is to smoothen the image using a Gaussian Filter. Next, the gradient needs to be found by obtaining the large magnitude of the image gradient. After that, a pre-defined threshold value is applied to obtain the potential edge. Finally, all edges that are not connected to the strong edges are suppressed (IIUM Year)

After filtering the images using the Circle Hough Transform technique, expressions (1) and (2) indicate how the Hough Transform formula is used to detect the iris boundary. The *r* is the radius from the center to the iris boundaries, *a* and *b* are the center point coordinates and finally *θ* is the angle from 0 to 360 degrees (Ifeanyi and Ghazali Year).

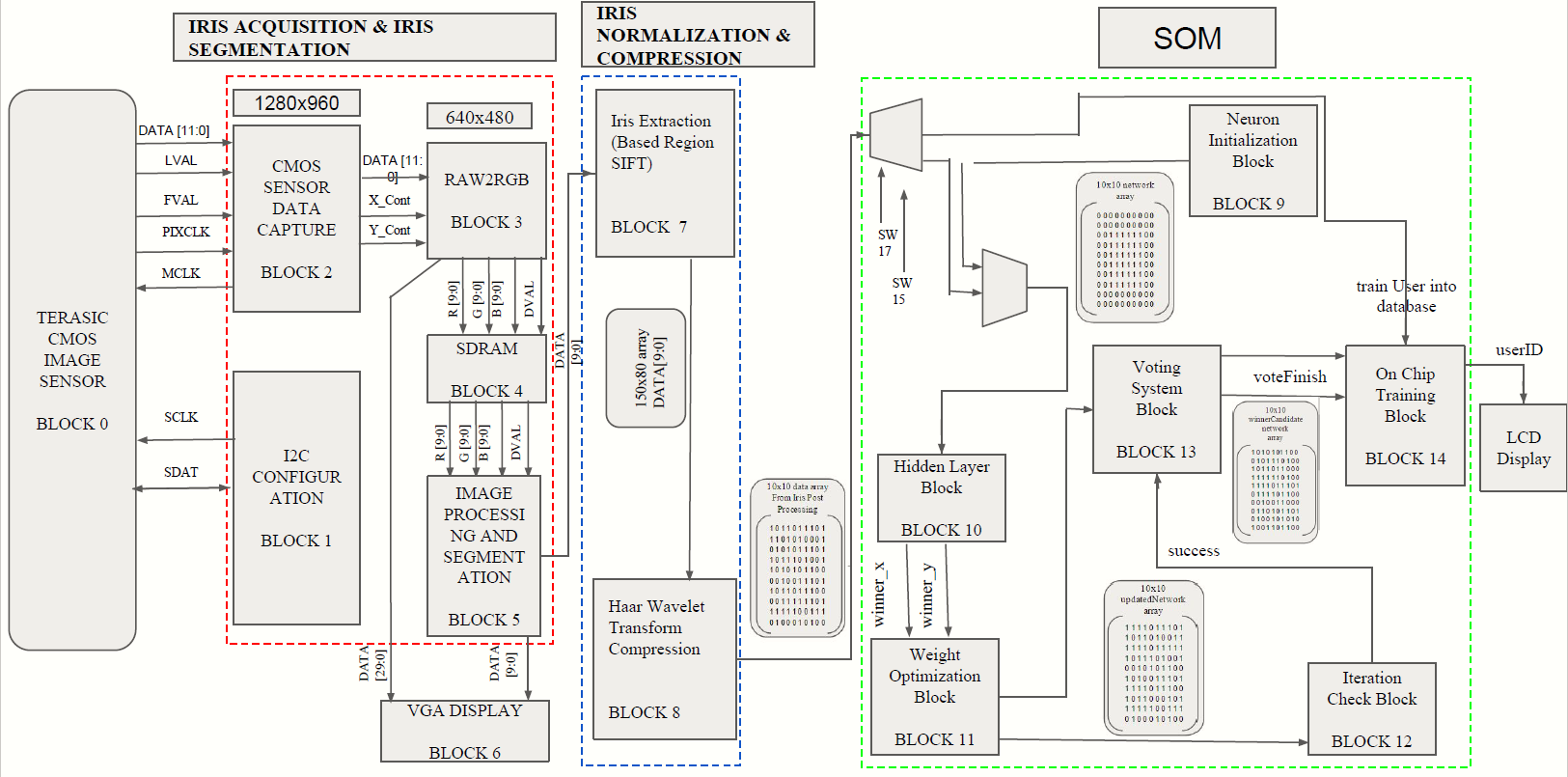
*x = a + r cos (θ)* (2.1)

*y = b + r sin (θ)* (2.2)

# Chapter 3: Methodology

Figure 3.1 shows the overall Iris Recognition System (IRS) block diagram, which is divided into 3 main modules. Firstly, the Iris Acquisition and Segmentation performs image pre-processing and iris/pupil segmentation. Secondly, the Iris Normalization and Iris Compression extract one specific section of the iris image and compress it into a 10x10 matrix. Lastly, the Self-Organizing Map (SOM) trains and stores the compressed iris images as well as performs matching of iris images.

This work focuses on the first IRS module which is the Iris Acquisition and Iris Segmentation unit (IAS) as shown in Figure 3.2. This module consists of six blocks. Firstly, the Terasic CMOS Image Sensor (Block 0) receives command and outputs image data. Secondly, the I²C Configuration (Block 1) communicates using serial protocol and sends command to Block 0. Thirdly, the CMOS Sensor Data Capture (Block 2) either receives image data from Block 0 or captures image data. Next, the RAW to RGB (Block 3) converts RAW images into RGB images while the SDRAM (Block 4) stores these RGB images. The Image Processing and Segmentation (Block 5) on the other hand performs image pre-processing. Lastly, the VGA Display (Block 6) outputs images to the monitor for display. Each sub-block of the Iris Acquisition and Segmentation unit will be fully illustrated with design flows and code fractions in the following sub-sections.

  
Figure 3.1 Complete IRS Block Diagram

640x480

1280x960

TERASIC   
CMOS  
IMAGE  
Sensor

Block 0

IRIS ACQUSITION and IRIS SEGMENTATION

RAW2RGB

30bits

Block 3

Data [11:0]

CMOS  
Sensor  
Data  
Capture  
  
Block 2

DATA [11:0]

Data [29:0]

Y\_Count

LVAL

X\_Count

FVAL

PIXCLK

R [9:0]

G [9:0]

B [9:0]

DVAL

VGA

Display

MCLK

I2C

Configuration

Block 1

SD RAM

Block 4

SDAT

R [9:0]

G [9:0]

B [9:0]

DVAL

SCLK

Data [9:0]

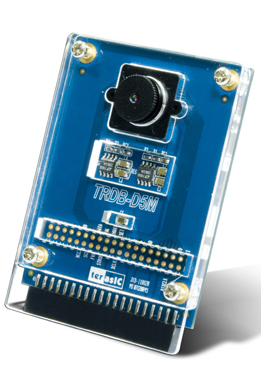
Image Processing   
and Segmentation

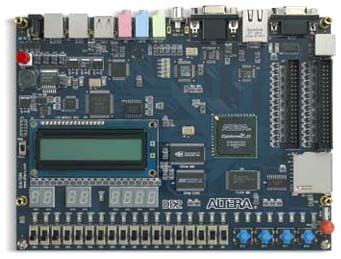
Block 5

Figure 3.2 Iris Acquisitions and Iris Segmentation Block Diagram

## 3.1: CMOS Image Sensor

The CMOS Image Sensor (Block 0) is a CMOS camera with 5 Mega Pixels Figure 3.2 shown the CMOS Camera. This Camera is connected to the Altera DE2 FPGA Board via its GPIO ports as shown in Figure 3.4. Each GPIO pin is responsible for different functions illustrated in Figure 3.5.

  
Figure 3.3 CMOS Camera

  
Figure 3.4 Altera DE2 FPGA Board

GPIO Port (1 and 2)

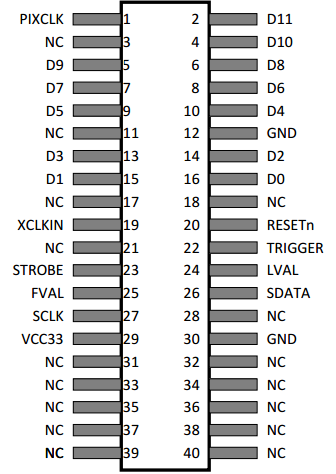
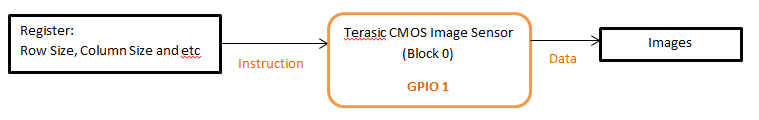
****

Figure 3.5 Altera DE2 GPIO Pin

Block 0 takes commands or instructions from the I²C Configuration Block (Block 1). In each clock cycle Block 1 will output *SCLK* (Pin 27) and *SDATA* (Pin 26) signals to the D5M sensors in Block 0. After the CMOS sensor was successfully configured by the I²C Block, the 12-bit Image Data, Line\_Valid (LVAL), Frame\_Valid (FVAL) and *PIXCLK* signals will then be output to Block 2. Figure 3.6 shows the flow of Block 0 interacting with Blocks 1 and 2. This module originally captures the human eye directly via the Terasic CMOS image sensor. However, due to hardware limitations, image clarity, and possible eye damage due to over exposure to intense lightings, the renowned CASIA version 1 database of human eye images were used as the camera’s input image.

Figure 3.6 Interaction Flow of Blocks 1, 2 and 3

## 3.2 I²C Configuration

The I²C Configuration module (Block 1) outputs relevant signals to the Terasic CMOS Image Sensor (Block 0). Block 1 contains 2 modules i.e. the I²C Controller module and the I²C Configuration module both that provided by Terasic. The first module is the I²C Controller, this module is known as a serial bus protocol, that is to communicate or write/read data from master to slave and slave back to the master. In this module the Altera DE2 FPGA is considered as a master and TRDM\_D5M CMOS sensor as the slave.

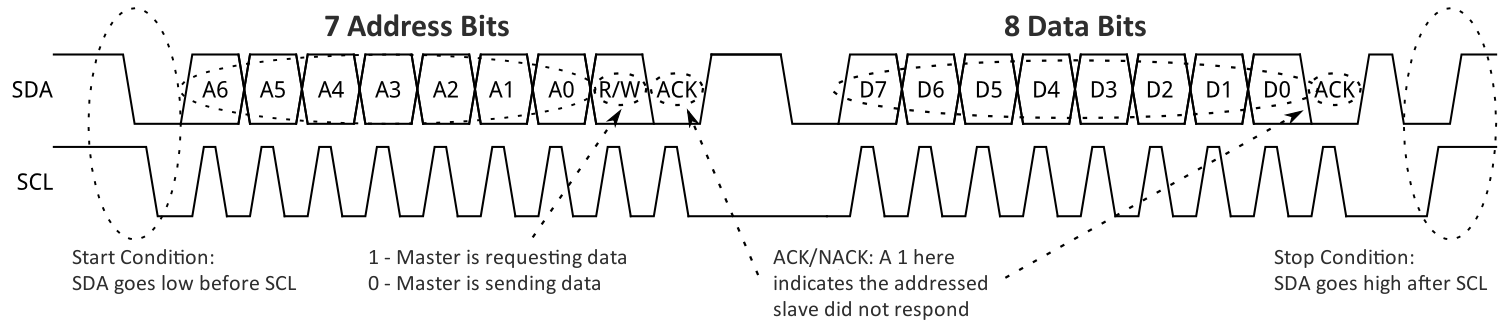


Figure 3.7: I²C Configuration Signal Diagram

Figure 3.7 shows 4 conditions that are needed to set-up an I²C Controller. The first condition is named as Start Condition which is when *SDAT* signal goes low and *SCLK* signal is still at a high pulse, after that *SDAT* signal and *SCLK* signal goes into a low pulse. The second condition is to indicate which slave addresses that the master needs to communicate with. The second last bit is used to indicate reading/writing to the slave and the last bit is for acknowledgement, when the slave obtains the data or address that it requires, the master needs to be acknowledged. The third Condition is to begin sending data from the master to the slave and the last bit is the same as the second condition which represents acknowledgement. Lastly, the Stop Condition is executed when the *SCLK* signal goes into a high pulse before the *SDAT* signal which will considered as a stop signal. Figure 3.8, 3.9 and 3.10 show the Start Condition, Slave Address and Stop Condition in Verilog HDL.

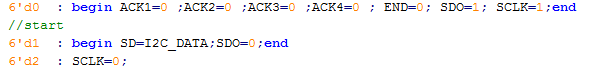
****

Figure 3.8: Start Condition in Verilog HDL Code

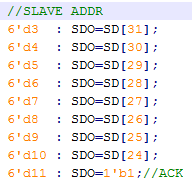
****

Figure 3.9: Slave Address in Verilog HDL Code

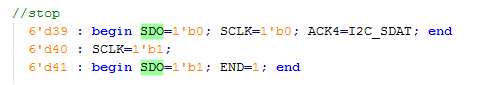
****

Figure 3.10: Stop Condition in Verilog HDL Code

The second part is the I²C Configuration is to send instruction signals to the D5M sensors the CMOS Image Sensor Module (Block 0) to provide output data to the CMOS Sensor Data Capture (Block 1). This module identifies the sizes and start points of rows and columns, the exposure time and many other parameters stated in the TRDM\_D5M datasheet. This configuration contains a default image size of 1280x960 sizes and an exposure time of 0x007c0. Due to the delay shown in the monitor, the exposure time needs to be reducing because the higher the value of exposure time the longer it takes to capture an image. This reduction in exposure time affects the monitor’s output color therefore register values needs to be modified to compensate for this color differences as indicated in Table 3.1.

Table 3.1 I²C CCD Configuration for Color Gain in CCD [Ref Year]

|  |  |  |  |
| --- | --- | --- | --- |
| Register Address | Register Value | | Name |
| Default | Modify Value |
| R0x02B | 0x0008 | 0x0033 | Green1 Gain |
| R0x02C | 0x0008 | 0x0135 | Blue Gain |
| R0x02D | 0x0008 | 0x0339 | Red Gain |
| R0x02E | 0x0008 | 0x0033 | Green2 Gain |

Overall, the I²C Configuration Module begins from I²2 Configuration sub-modules that output a 32-bit address to I²2 Controller. The I²2 Controller will begin separating 32-bits into 4 parts which are the slave address, sub-slave address, and another 16-bits for registers. The first 8-bits of MSB address will be sent out to await acknowlegement. Once an acknowledgement is received then the I2C Controller will continue to send a sub-slave address. This process will continue until the 32-bit has been successfully sent. Figure 3.11 shows the flow chart of the I²C Configuration.

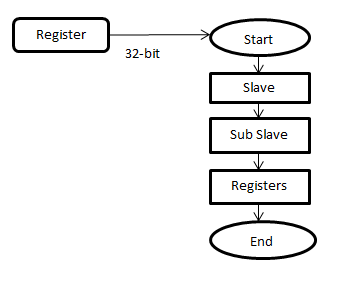


Figure 3.11 I²C Configuration Flow Chart

## 3.3 CMOS Sensor Data Capture

This CMOS Sensor Data Capture (Block 2) obtains a signal to begin image data capture. This module is provided by Terasic. Figure 3.12 shows that there are either valid or non-valid images when the CMOS sensor outputs image data to this data capture block. Using the Frame\_Valid (FVAL) and Line\_Valid (LVAL) signals, the vertical blank, horizontal blank and valid images can be represented. Figure 3.13 shows Verilog HDL code to identify vertical and horizontal blanks. Initially, FVAL equals 0, indicates vertical blank, then when FVAL is 1, a *mCCD\_FVAL* signal will output a 1. Secondly, when FVAL goes back into vertical blank region (0) then the *mCCD\_FVAL* signal goes back into 0 but the *mCCD\_LVAL* signal will rise high or low according to LVAL (horizontal blank). If LVAL is equal to 1 then at the next active-edge of the clock FVAL which is outputting a 1 will render *mFVAL* signal into a 1. This current situation that *mCCD\_FVAL* and *mCCD\_LVAL* is 1 show that it is a valid image. Figure 3.14 shows the flow chart of this module (Block 2), when *mCCD\_FVAL* and *mCCD\_LVAL* is 1 then it will begin to output the image data and image size by using *X\_Count* and *Y\_Count*.

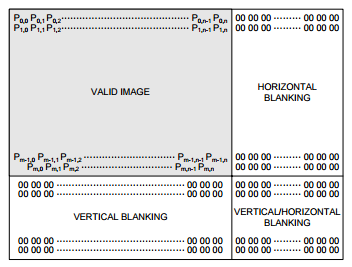


Figure 3.12 Spatial illustration of Image Readout

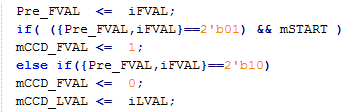
****

Figure 3.13 HDL code for Blank and Valid Image Identification

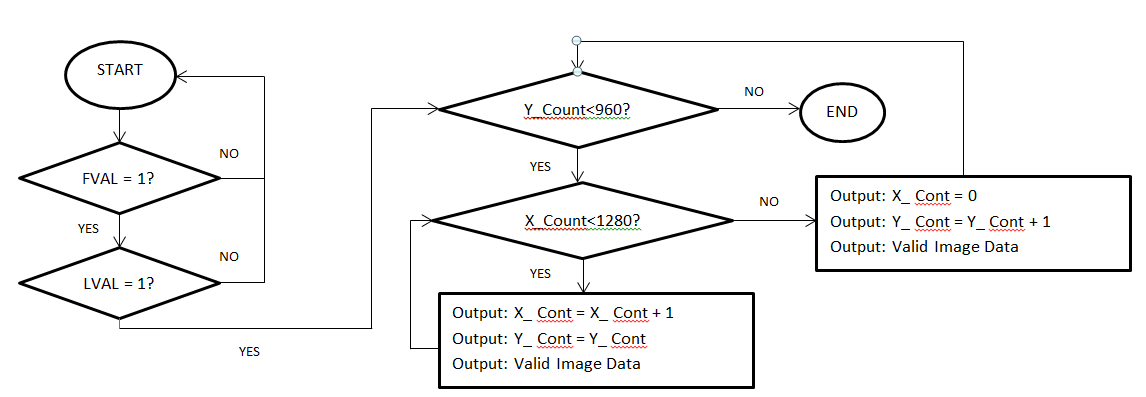


Figure 3.14: Flow Chart for CMOS Sensor Data Capture module

## 3.4 RAW to RGB

This module is a RAW to RGB Converter (Block 3) that combines four pixels with 12-bit data each and converts it into a 30-bit RGB data. This block is provided by Terasic. This module uses a MegaFunction Shift Register as shown in Figure 3.15. This MegaFunction shifts out two taps and a 12-bit data in each clock cycle. Figure 3.16 shows that R and G1 will be shifted out into an orange box. In another clock cycle, R and G1 will then be shifted into the blue boxes while G2 and B will replace the values in the orange boxes. This will continue until the end of the RAW data image. Table 3.2 shows register data represented by orange and blue boxes previously. *mDATA\_0* indicates *column* and *row*, *mDATA\_1* indicates column and *row+1*, *mDATAd\_0* indicates *column-*1 and *row* and lastly *mDATAd\_1* indicates *column-1* and *row-1*.

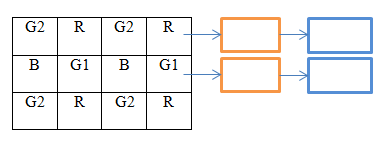


Figure 3.15 MegaFunction Shift Register

Table 3.2: Current RAW Image Data and Previous RAW Image Data Variable

|  |  |
| --- | --- |
| *mDATAd\_0* (01) | *mDATA\_0* (11) |
| *mDATAd\_1* (00) | *mDATA\_1* (10) |

Figure 3.16 shows the RGB Arrangement of a raw image while Figure 3.17 shows the Verilog HDL for the RAW to RGB conversion. Block 1 outputs *X\_Count* and *Y\_Count* to indicate image sizes. This module uses the *X\_Count* and *Y\_Count* data to indicate odd or even number by taking the first bit of the *X* and *Y* count. Firstly, *X* is 0 and *Y* is 0 as is indicated by the Red box. Secondly, *X* is 0 and *Y* is 1 as indicated by the Yellow box. Thirdly, *X* is 1 and *Y* is 0 indicated by the Blue box and Lastly, X is 1 and Y is 1 is indicated by a Purple box. All different priorities of the *X* and *Y* will be combined into an RGB pixel according to the algorithm indicated by the Raw to RGB HDL coding.

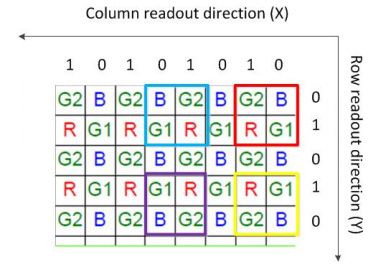


Figure 3.16 Sample RGB Arrangement in a RAW image

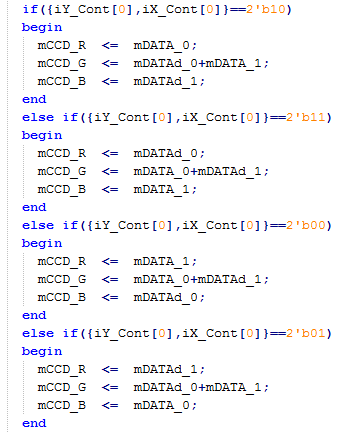


Figure 3.17: Verilog HDL Code for RAW to RGB Conversion

## 3.5 SDRAM

The SDRAM Module (Block 4) acts as a buffer to store the RGB image. The SDRAM is a storage that is organized as a matrix of words. Figure 3.18 shows the capacity of SDRAM, if the SDRAM does not split into more banks then it will slow down the storage speed. In this case Figure 3.19 shows that the SDRAM can be divided into 4 banks that allow operation of one bank while waiting for another. In an Altera DE2 FPGA with SDRAM divided into 4 banks, the read and write can be performed simultaneously. Due to the existence of 4 banks, there is a need to select which bank to be used. This module uses 2 banks where from address 22’h000000 indicates the start of bank 0 and address 22’h100000 indicates the start of bank 1. Since RGB contains 30-bits of data, it has to be separated into 2 banks that is Bank 0 to store 10-bits of Red pixel data and 5-bits of one-half Green pixel data while Bank 1 stores 10-bits of Blue pixel data and the remaining half of the 5-bits Green pixel data.

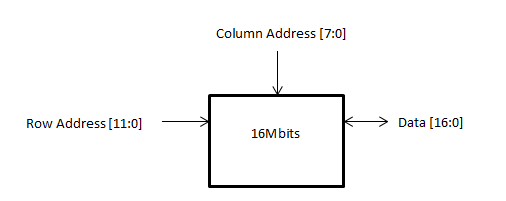


Figure 3.18: Capacity of the SDRAM

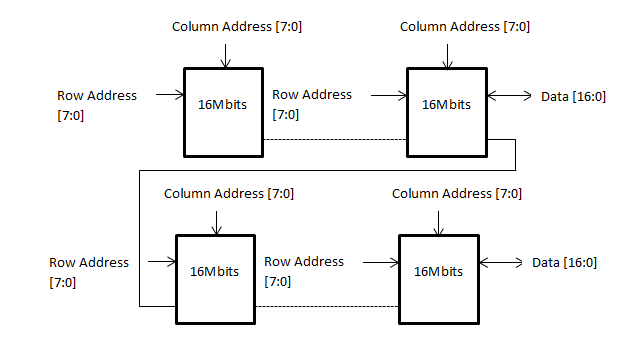


Figure 3.19: 4 Separated Banks of the Reconfigured SDRAM

## 3.6 Image Processing and Segmentation

This Image Processing and Segmentation Module (Block 5) convert image data into grayscale and binary form to ease the iris and pupil segmentation process. This process is separated into 5 steps for pupil detection and 3 steps for iris detection. Figure 3.20 shows the flow of pupil detection while Figure 3.21 shows the flow of iris steps detection. Both the pupil and iris detection converts RGB image from the previous block into grayscale and binary before pupil/iris points can be detected. The pupil point detection requires an extra erosion and dilation step compared to the iris. Each of these steps will be described in the following sub-sections.

RGB to Grayscale

Dilation to detection pupil point

Erosion to Dilation

Binary Image to Erosion

Grayscale to Binary Image

Figure 3.20: Flow of Pupil Point Detection

RGB to Grayscale

Binary Image to detection iris point

Grayscale to Binary Image

Figure 3.21: Flow of Iris Point Detection

### 3.6.1 RGB to Grayscale

This sub-module converts the 30-bit RGB image into 10-bit Grayscale data. Figure 3.22 shows grayscale tone from 0 to 1023, the white is represented as 1023, black is represented as 0 and other tones are known as gray.

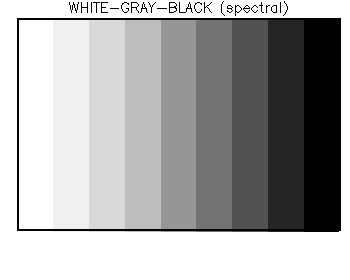


Figure 3.22: Grayscale Color Spectrum

There are several different formulas to convert RGB into Grayscale, first is the lightness method which is an average of the highest color value and lowest color value. The second is an average method that adds up the individual RGB values and divides them by 3. Lastly, the luminosity method that is a weighted average to account for human perception, that is since green is more sensitive to the human green will be more heavily weight than the others. Expressions 2, 3, 4 show the formula for all three methods and Figure 3.23 shows the output images from 3 of these methods (John, 2009).

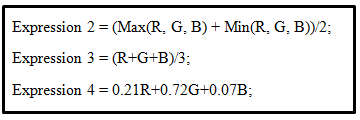
****

Figure 3.23: Grayscale Images Using Different Formula

This module uses the 2nd equation that averages the entire RGB image into grayscale. The first formula causes the important section of the iris to become darker like the sunflower petals. The third formula contains floating values, which requires added complexity to create a floating point unit. Therefore, the averaged method is chosen for its simplicity and average clarity among the three techniques.

### 3.6.2 Grayscale to Binary Image Conversion

This module converts grayscale image into 0 (black) or 1 (white) image. Figure 3.24 shows the Verilog HDL code to convert into a Binary Image. This code contains *iDATA* that is the input data from the grayscale image. The threshold indicates the pre-defined value that is being considered as a black. The threshold value chosen here is 384 based on the experimented environmental lighting intensity.

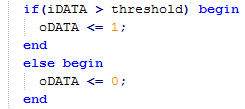


Figure 3.24: HDL Code for Grayscale to Binary Conversion

### 3.6.3 Binary Image to Erosion

This Erosion module reduces the noise and image size. The Erosion module uses the shift register Megafunction shown in Figure 3.25 that outputs 3 values in parallel for each cycle of the clock. When C9 to C1 is being reset, all registers values will be stored as a 1 and being OR-ed together during each cycle, the formula for this operation is as expressed in (3.1).

 (3.1)

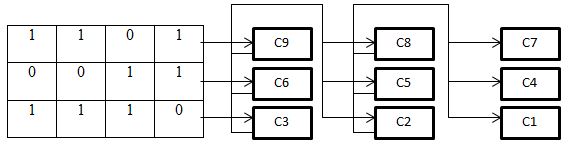


Figure 3.25: Shift Register Megafunction

### 3.6.4 Erosion to Dilation

This Dilation module restores the image size after erosion. Dilation also used shift register Megafunction shown in Figure 3.25 that outputs 3 values in parallel for each clock cycle. When C9 to C1 is resetted, all registers will be stored as 1, all values are AND-ed together during each clock cycle, the formula for dilation is as given in the expression (3.2).

 (3.2)

### 3.6.5 Dilation for Pupil Point Detection

This module locates the coordinate of pupil point as shown in Figure 3.26. After RAW to RGB conversion the image size reduces to 640x480. Figure 3.26 shows how each of the four pupil points were located using the flow chart in Figure 3.27.

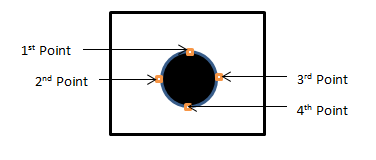
****

Figure 3.26: Image of Pupil Point Detection

The flow chart in Figure 3.27 begin with row by row coordinate checking whereby if the 1st coordinate is detected as a black pixel, it will be outputted as a *TopPoint*. Besides that, the column will be stored as *largestTempCol* , *SmallestTempCol* and a *LargestTempRow*. Three of these temporary storages are being used to compare with the current value. If a black pixel is detected to be smaller than the *SmallestTempCol*, the current value will overwrite the *SmallestTempCol* value. This same method is used to obtain the other three points. After obtaining the last point and providing a signal to indicate that the entire row is made up of white pixels then the search is therefore completed. These four points are being used to find the centre of radius and pupil. The centre coordinates will then be fed into the iris detection and normalization block while the radius value will only be passed to the iris normalization block.

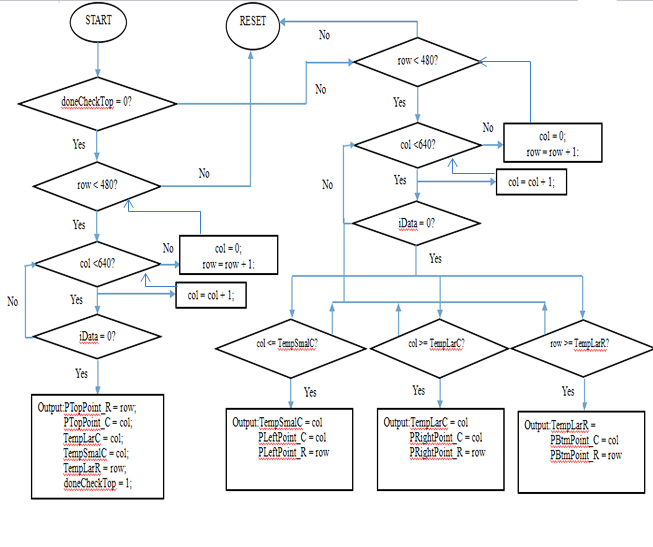


Figure 3.27: Flow Chart for Pupil Point Detection

### 3.6.6 Binary Image for Iris Point Detection

This Module detects the boundaries of the outer iris ring to the left and right side. Figure 3.28 shows the left and right points that must have a condition of being in the same row with the centre point. Figure 3.29 shows the flow chart to find the left and right points of the iris boundaries.

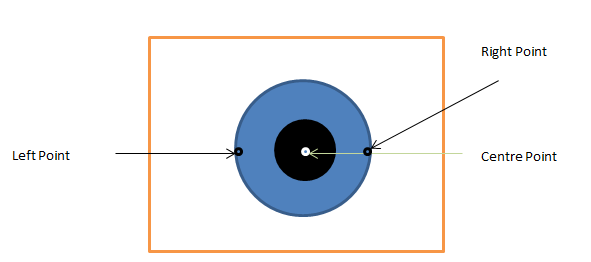


Figure 3.28: Image of Iris Centre Point

The flow chart in Figure 3.29 detects the pixel by checking row by row; once the same centre point row is detected the 1st detected dark pixel is then indicated as a left point. After that, a register is used to store the left point column coordinate. Using the register, continue to compare with other black pixels of different columns. If the current row value is larger than the centre point row value, then the module will output the left and right points to the iris normalization block.



Figure 3.29 Flow Chart of Search Iris Point

## 3.7 VGA Display

VGA Display (Block 6) takes input data from Block 5 or Block 3 to be displayed at the LCD monitor. This VGA controller is configured into a size of 640x480 pixels at an output speed of 25MHz. The VGA requires the configuration of its Horizontal and Vertical Blank signals namely *H\_Sync* and *V\_Sync*. When *H\_Sync* is larger than the pixel parameter of 96 and *V\_Sync* is larger than 2 this would indicate a valid image. If the Horizontal Counter (*H\_Cont*) and Vertical Counter (*V\_Count*) is within the range of 640 as shown in Figure 3.30 this will output the RGB image.

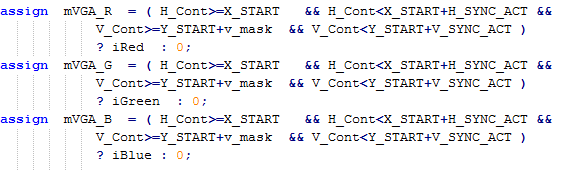


Figure 3.30: Verilog HDL Code of VGA for RGB pixel output

# Chapter 4: Results and Discussions

The results and discussions of this work will be presented in three main sections, firstly in a block by block basis, secondly as an integrated iris acquisition and segmentation sub-block and finally as a fully integrated IRS system. Several parameters will be compared upon in simulated waveform as well as via hardware implementation with the Cyclone II FPGA in different situations. The following sub-sections describe in detail the results and discussions of each block and as an integrated system.

## 4.1 CMOS Image Sensor

Figure 4.1 shows the image capture by the CCD camera using CASIA Database version 1 eye. This image needs to be captured in close proximity due to a minimum iris image size requirement. Proper lighting must be adjusted for the CCD camera to ensure ample exposure to capture the iris image. The CCD camera lens needs to be tuned to its pre-defined focal point to ensure image clarity during iris capture.



Figure 4.1 CASIA Database Eye Image Captured by the CCD Camera

## 4.2 I2C Configuration

Figure 4.2 shows the simulated results that fulfill the start (10 to 15ps) and stop (90 to 95ps) condition. After that, the first slave address is being sent to the slave at 20ps and later at 33 ps, an acknowledgement (*ACK*) signal is received. a. At the simulation time of 35ps, the second sub-slave output has been sent to the slave and another acknowledgement (*ACK*) signal is being received at 52 ps. The last 16-bit address has been sent at 54 ps a while the *ACK* signal was received at 87 ps.

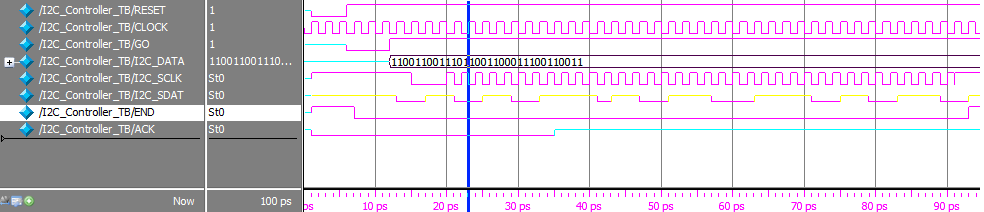


Figure 4.2: Simulated Result of the I²C Controller Module

Table 4.1 shows the captured image outcome of different I²C configurations and settings. The first column shows the default setting provided by Terasic while the second column is the proposed settings. Figure 4.3 shows the result of the default settings made by Terasic and Figure 4.4 shows the result of the proposed settings. The default setting has a high exposure time but a low gain of Red, Green and Blue. This default settings will cause a delay when operating in a free running mode. The proposed settings reduce the exposure time but increase the gain of Red, Green and Blue. The reduction in exposure time improves image clarity as shown in Figure 4.4

Table 4.1: Images with Different Settings

|  |  |  |
| --- | --- | --- |
|  | Figure 4.2  (Default Setting) | Figure 4.3  (Proposed Setting) |
| Row Size | 960 | 960 |
| Column Size | 1280 | 1280 |
| G1 Gain | 11 | 50 |
| G2 Gain | 11 | 50 |
| R Gain | 15 | 824 |
| B Gain | 15 | 307 |
| Exposure Time | 1984 | 1080 |

** **

Figure 4.3 Default Setting Figure 4.4 Proposed Setting

## 4.3 CMOS Sensor Data Capture

The waveform shown in Figure 4.5 indicates the functionality of the CMOS Sensor Data Capture Block. The cursor indicates that when the *iFVAL* signal is low (0) and *iLVAL* is high (1) the module will begin outputting the signals *oY\_Cont*, *oX\_Cont* and *oData*.

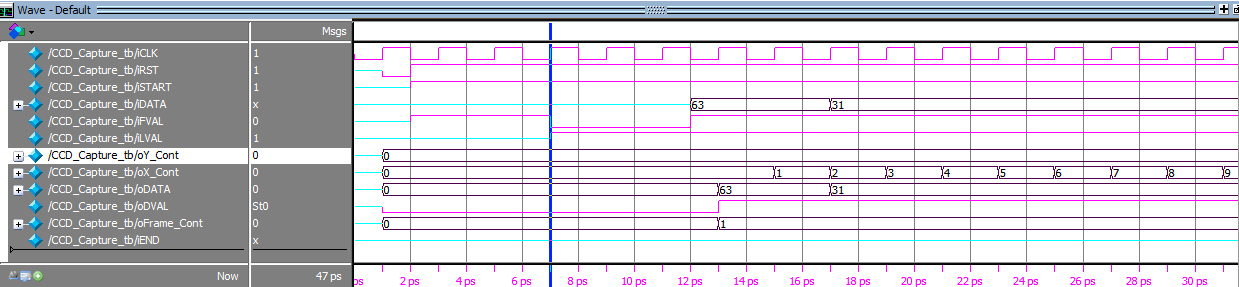
****

Figure 4.5: Results of the CMOS Sensor Data Capture Module

## 4.4 RAW to RGB

Figure 4.6 shows the RGB image converted from the RAW image.

RAW image?? =🡺****Figure 4.6 RGB Image Converted from the RAW Image

**RGB IMAGE**

## 4.5 Image Processing and Segmentation Process

### 4.5.1 RGB to Grayscale

The average method was used to convert the RGB image into a grayscale image. Figure 4.7 shows the RGB image while the converted image in grayscale is shown in Figure 4.8.

****  
Figure 4.7 RGB Image

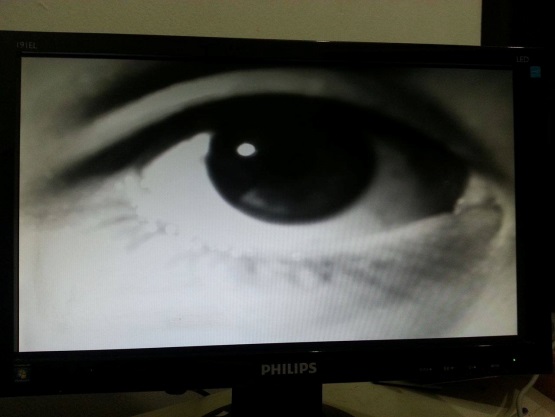
****

Figure 4.8 Grayscale Image

### 4.5.2 Grayscale to Binary Image Conversion of the Pupil

Figure 4.9 shows the output from the grayscale image that has been converted into an ideal binary image.. Different environment changes the binary image output. Figure 4.10 shows the binary image of a partial pupil and Figure 4.11 shows the full iris image in binary. Figure 4.10 shows the result of high light intensity meaning in a very bright environment when the image was captured. Figure 4.11 on the other hand shows the result of low light intensity meaning a very dim environment when the image was captured. Due to the difference in the gain of RGB value and therefore grayscale average values, the final binary image will also be affected. By adjusting the threshold to suit the environmental condition, a positive outcome as shown in Figure 4.9 can be achieved. However, to ensure consistency, either several boundary conditions such as illuminators intensity should be fixed or an automated threshold adjustment module should be in place.

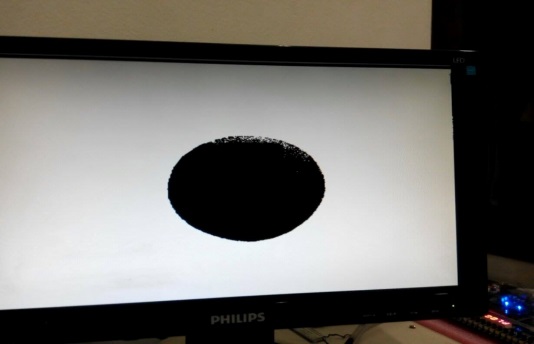


Figure 4.9 Ideal Pupil Binary Image

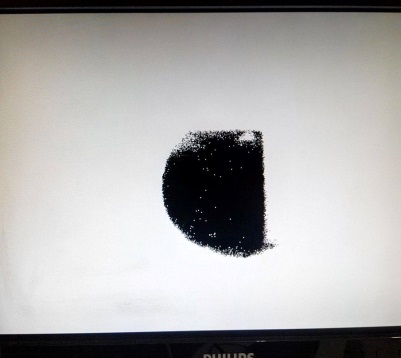


Figure 4.10 Non-ideal Pupil’s Binary Image in High Light Intensity Environment



Figure 4.11 Non-ideal Pupil’s Binary Image in Low Light Intensity Environment

### 4.5.3 Binary Image to Erosion

Figure 4.12 shows the successful erosion that reduces noise and retains the image size. Figure 4.13 shows that the image noise cannot be reduced due to a large amount of dark area that has been wrongly assumed as a valid image. Figure 4.14 shows that the noise can be reduced but the image size has been reduced and slightly distorted as well.



Figure 4.12 Result of Successful Erosion

****Figure 4.13 Result of Unsuccessful Erosion

Noise part

****Figure 4.14 Result of Unsuccessful Erosion

### 4.5.4 Dilation

Figure 4.12 shows a prefect image after erosion which has successfully reduce the image noise but there are some pupil data that is missing. Figure 4.15 shows the result of dilation used to compensate the missing pupil data after the erosion process..

  
Figure 4.15 Result of Dilation Image

### 4.5.5 Pupil and Iris Points Detection

Figure 4.16 shows the display of pupil and iris points on the monitor. The purple arrows represent the pupil point and the black arrow represents the iris points.

  
Figure 4.16 Iris and Pupil Points

Iris Point

Pupil Point

## 4.6 Iris Acquisition and Segmentation

Figure 4.17 shows a pie chart for the recognition success rate and the result indicates 71% success and 29% failure to obtain the Iris and Pupil Points by comparing 10 images. The failure was due to the binary image and the erosion because the threshold was set too high which would result in a smaller image while setting the threshold too low will result in excessive noise in different areas. Figure 4.18 shows the image of unsuccessful iris and pupil point detection. Figure 4.19 shows the image of a successful iris and pupil point detection.

Figure 4.17: Success Rate of Iris and Pupil Point Detection

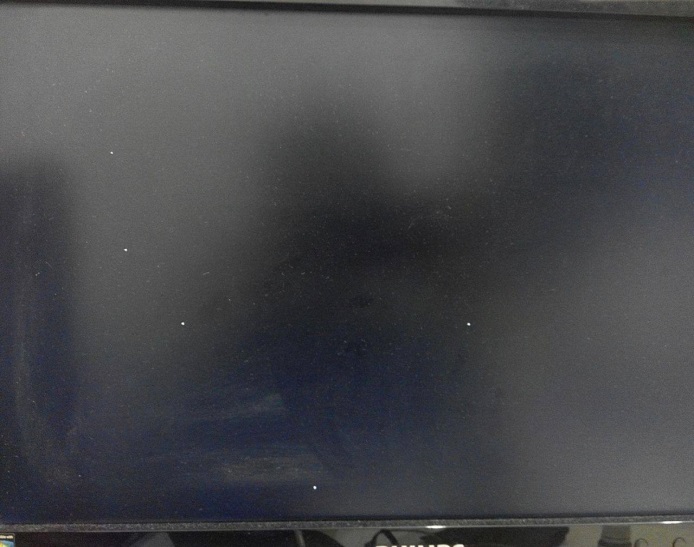
****Figure 4.18: Unsuccessful Pupil and Iris Point Detection



Figure 4.19: Successful Pupil and Iris Point Detection

## 4.7 Integrated Result

Firstly, all eye images from CASIA database version 1 were captured using the CCD Camera. Figure 4.20 to Figure 4.28 show that the result of iris images after being captured and trained into database. There are 3 different users being shown, while Figure 4.20 to Figure 4.22 is known as user 1, Figure 4.23 to Figure 4.25 is known as user 2 and Figure 4.26 to 4.28 is known as user 3. It can be observed that each of the 3 users where trained a total of three times and later stored as three varying samples in the database for matching purposes.



Figure 4.20: User\_1 (Sample 1) Figure 4.21: User\_1 (Sample 2) Figure 4.22: User\_1 (Sample 3)



Figure 4.23: User\_2 (Sample 1) Figure 4.24: User\_2 (Sample 2) Figure 4.25: User\_2 (Sample 3)

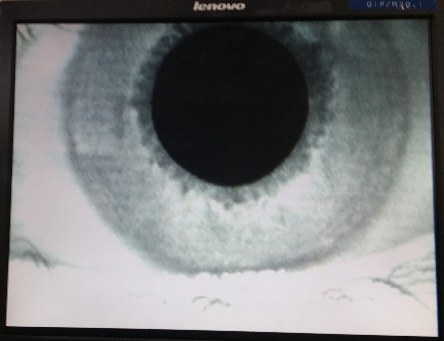
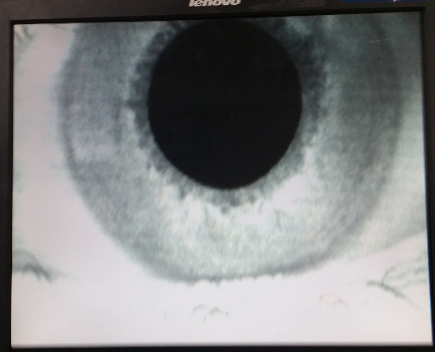
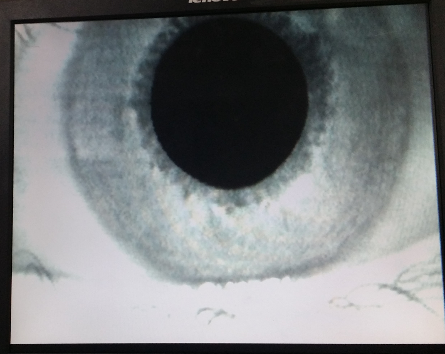


Figure 4.26: User\_3 (Sample 1) Figure 4.27: User\_3 (Sample 2) Figure 4.28: User\_3 (Sample 3)

Table 4.2, on the other hand shows the matching results, to test the recognition accuracyfor a total of 10 conducted tests. User 1 and User 2 can be successfully recognized 40% of the time while User\_3 an be successfully recognized 30% of the time. Overall, all users can be successfully recognized 36.6% of the time. The recognition accuracy for Users 1, 2 and 3 however are 100%, 90% and 100% respectively with an average of ~96.67% based on a total of 10 conducted tests shown in Table 4.2. Figure 4.29 shows the results in a bar chart form that the user success recognized.

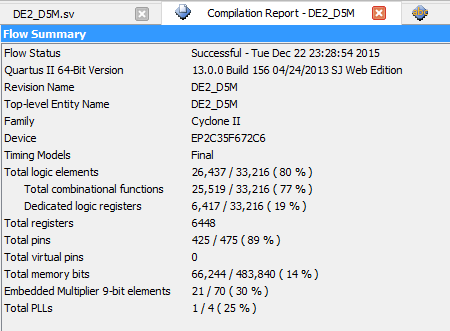
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trained User | Matching Accuracy | | | | | | |  | | | |
| User\_1 | 1 | 1 | x | x | x | x | 1 | | 1 | x | x |
| User\_2 | 2 | 2 | 3 | x | x | 2 | 2 | | x | x | x |
| User\_3 | 3 | x | 3 | 3 | x | x | x | | x | x | x |

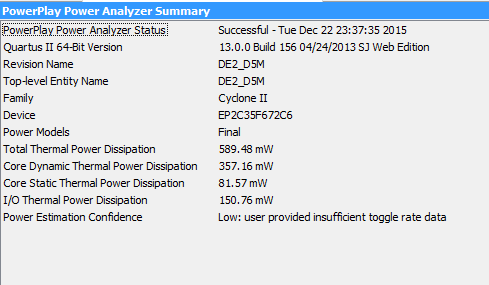
Table 4.2 Matching Results of 3 Users

Figure 4.29: IRS Recognition Probability

## 4.9 Analysis:

Figure 4.30 shows the total logic elements of 80% that has been used which is almost the upper limit for the Altera DE2 Board. Figure 4.31 however, shows the total power dissipation at 589.48 mW for the entire IRS system.

  
Figure 4.30: Total Logic Elements of the IRS

  
Figure 4.31: Total Power Dissipation of the IRS

# Chapter 5: Conclusion

This work focuses on the initial Iris Capture and Pre-processing portion of a complete IRS system. A near optimal Iris Acquisition and Segmentation architecture has been proposed using a 5 Mega Pixel CCD Camera by capturing CASIA Database Version 1 eye that has gone through image processing and segmentation. The main aim is to obtain a faster recognition speed and enhanced recognition accuracy. The proposed Iris Acquisition and Segmentation will be modeled, designed and simulated using Verilog HDL based test-bench of all sub-modules in ModelSim software of Mentor Graphics environment. The proposed iris acquisition and segmentation method, is to firstly capture a clear eye image, convert this image into a 30-bit RGB then a 10-bit grayscale, and finally a binary image and later filter this image, the last task is the pupil and iris point detection. Finally, this Iris Acquisition and Segmentation results in a speed reduction of at least 2 times compared to conventional designs at 1.8s but the accuracy of segmentation is reduced to 71% compared to conventional designs. Such contactless IRS system can be applied in normal household or law enforcement agencies for security augmentation purposes.

## 5.1: Future Trends & Recommendation

IRS is considered expensive compared to other systems with similar applications. To ensure the accuracy is high and secure, a good camera with Near-Infrared LED to capture clear iris images is needed. An automated binary image threshold adjustment can be created to obtain a perfect iris and pupil image. For better reliability, the IRS can be fitted with backup rechargeable batteries or renewable energy harvested from the environment to ensure a non-interrupted operation when there is an absence of electricuty.

# References:

1. Naveen Singh, Dilip Gandhi, Krishna Pal Singh (May 2011). Iris Recognition System Using A Canny Edge Detection and A Circular Hough Transform. [Online] Available at:http://www.ijaet.org/media/0001/27IRIS-RECOGNITION-SYSTEM-USING-A-CANNY-EDGE-DETECTION-AND-A-CIRCULAR-HOUGH-TRANSFORM-Copyright-IJAET.pdf. [Accessed 28 November 2015]
2. Fpga4fun.com – A simple controller. 2016. fpga4fun.com – A simple controller. [ONLINE] Available at:http://www.fpga4fun.com/SDRAM2.html. [Accessed 27 December 2015]
3. E.G.M Petrakis. Binary Image Analysis. [Online] Available at:http://www.intelligence.tuc.gr/~petrakis/courses/computervision/binary.pdf
4. John D. Cook (August 2009). Three algorithms for converting color to grayscale. [ONLINE] Available at:http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/ [Accessed 28 November 2015]
5. Sr.Sagaya Mary James (March 2015). Iris Recognition Process. [Online] Available at:http://www.ijana.in/Special%20Issue/file26.pdf. [Accessed 22 December 2015]
6. Xiaofan Bao, Jiayuan Wang (2013). Real-time “Photoshop” An FPGA-based Real-time Morphological Image Processing. [Online] Available at:http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/s2013/xb46\_jw937/xb46\_jw937/. [Accessed 15 November 2015]
7. Ifeanyi Ugbaga Nkole, Ghazali Bin Sulong. An Enhanced Iris Segmentation Algorithms Using Circle Hough Transform. [Online] Available at:http://informatics.fsktm.um.edu.my/cameraready/Informatics\_005.pdf. [Accessed 19 November 2015]
8. CASIA?????

# Appendix:

Main Module:

module DE2\_D5M

(

//////////////////// Clock Input ////////////////////

CLOCK\_27, // 27 MHz

CLOCK\_50, // 50 MHz

EXT\_CLOCK, // External Clock

//////////////////// Push Button ////////////////////

KEY, // Pushbutton[3:0]

//////////////////// DPDT Switch ////////////////////

SW, // Toggle Switch[17:0]

//////////////////// 7-SEG Dispaly ////////////////////

HEX0, // Seven Segment Digit 0

HEX1, // Seven Segment Digit 1

HEX2, // Seven Segment Digit 2

HEX3, // Seven Segment Digit 3

HEX4, // Seven Segment Digit 4

HEX5, // Seven Segment Digit 5

HEX6, // Seven Segment Digit 6

HEX7, // Seven Segment Digit 7

//////////////////////// LED ////////////////////////

LEDG, // LED Green[8:0]

LEDR, // LED Red[17:0]

//////////////////////// UART ////////////////////////

UART\_TXD, // UART Transmitter

UART\_RXD, // UART Receiver

//////////////////////// IRDA ////////////////////////

IRDA\_TXD, // IRDA Transmitter

IRDA\_RXD, // IRDA Receiver

///////////////////// SDRAM Interface ////////////////

DRAM\_DQ, // SDRAM Data bus 16 Bits

DRAM\_ADDR, // SDRAM Address bus 12 Bits

DRAM\_LDQM, // SDRAM Low-byte Data Mask

DRAM\_UDQM, // SDRAM High-byte Data Mask

DRAM\_WE\_N, // SDRAM Write Enable

DRAM\_CAS\_N, // SDRAM Column Address Strobe

DRAM\_RAS\_N, // SDRAM Row Address Strobe

DRAM\_CS\_N, // SDRAM Chip Select

DRAM\_BA\_0, // SDRAM Bank Address 0

DRAM\_BA\_1, // SDRAM Bank Address 0

DRAM\_CLK, // SDRAM Clock

DRAM\_CKE, // SDRAM Clock Enable

//////////////////// Flash Interface ////////////////

FL\_DQ, // FLASH Data bus 8 Bits

FL\_ADDR, // FLASH Address bus 22 Bits

FL\_WE\_N, // FLASH Write Enable

FL\_RST\_N, // FLASH Reset

FL\_OE\_N, // FLASH Output Enable

FL\_CE\_N, // FLASH Chip Enable

//////////////////// SRAM Interface ////////////////

SRAM\_DQ, // SRAM Data bus 16 Bits

SRAM\_ADDR, // SRAM Address bus 18 Bits

SRAM\_UB\_N, // SRAM High-byte Data Mask

SRAM\_LB\_N, // SRAM Low-byte Data Mask

SRAM\_WE\_N, // SRAM Write Enable

SRAM\_CE\_N, // SRAM Chip Enable

SRAM\_OE\_N, // SRAM Output Enable

//////////////////// ISP1362 Interface ////////////////

OTG\_DATA, // ISP1362 Data bus 16 Bits

OTG\_ADDR, // ISP1362 Address 2 Bits

OTG\_CS\_N, // ISP1362 Chip Select

OTG\_RD\_N, // ISP1362 Write

OTG\_WR\_N, // ISP1362 Read

OTG\_RST\_N, // ISP1362 Reset

OTG\_FSPEED, // USB Full Speed, 0 = Enable, Z = Disable

OTG\_LSPEED, // USB Low Speed, 0 = Enable, Z = Disable

OTG\_INT0, // ISP1362 Interrupt 0

OTG\_INT1, // ISP1362 Interrupt 1

OTG\_DREQ0, // ISP1362 DMA Request 0

OTG\_DREQ1, // ISP1362 DMA Request 1

OTG\_DACK0\_N, // ISP1362 DMA Acknowledge 0

OTG\_DACK1\_N, // ISP1362 DMA Acknowledge 1

//////////////////// LCD Module 16X2 ////////////////

LCD\_ON, // LCD Power ON/OFF

LCD\_BLON, // LCD Back Light ON/OFF

LCD\_RW, // LCD Read/Write Select, 0 = Write, 1 = Read

LCD\_EN, // LCD Enable

LCD\_RS, // LCD Command/Data Select, 0 = Command, 1 = Data

LCD\_DATA, // LCD Data bus 8 bits

//////////////////// SD\_Card Interface ////////////////

SD\_DAT, // SD Card Data

SD\_DAT3, // SD Card Data 3

SD\_CMD, // SD Card Command Signal

SD\_CLK, // SD Card Clock

//////////////////// USB JTAG link ////////////////////

TDI, // CPLD -> FPGA (data in)

TCK, // CPLD -> FPGA (clk)

TCS, // CPLD -> FPGA (CS)

TDO, // FPGA -> CPLD (data out)

//////////////////// I2C ////////////////////////////

I2C\_SDAT, // I2C Data

I2C\_SCLK, // I2C Clock

//////////////////// PS2 ////////////////////////////

PS2\_DAT, // PS2 Data

PS2\_CLK, // PS2 Clock

//////////////////// VGA ////////////////////////////

VGA\_CLK, // VGA Clock

VGA\_HS, // VGA H\_SYNC

VGA\_VS, // VGA V\_SYNC

VGA\_BLANK, // VGA BLANK

VGA\_SYNC, // VGA SYNC

VGA\_R, // VGA Red[9:0]

VGA\_G, // VGA Green[9:0]

VGA\_B, // VGA Blue[9:0]

//////////// Ethernet Interface ////////////////////////

ENET\_DATA, // DM9000A DATA bus 16Bits

ENET\_CMD, // DM9000A Command/Data Select, 0 = Command, 1 = Data

ENET\_CS\_N, // DM9000A Chip Select

ENET\_WR\_N, // DM9000A Write

ENET\_RD\_N, // DM9000A Read

ENET\_RST\_N, // DM9000A Reset

ENET\_INT, // DM9000A Interrupt

ENET\_CLK, // DM9000A Clock 25 MHz

//////////////// Audio CODEC ////////////////////////

AUD\_ADCLRCK, // Audio CODEC ADC LR Clock

AUD\_ADCDAT, // Audio CODEC ADC Data

AUD\_DACLRCK, // Audio CODEC DAC LR Clock

AUD\_DACDAT, // Audio CODEC DAC Data

AUD\_BCLK, // Audio CODEC Bit-Stream Clock

AUD\_XCK, // Audio CODEC Chip Clock

//////////////// TV Decoder ////////////////////////

TD\_DATA, // TV Decoder Data bus 8 bits

TD\_HS, // TV Decoder H\_SYNC

TD\_VS, // TV Decoder V\_SYNC

TD\_RESET, // TV Decoder Reset

//////////////////// GPIO ////////////////////////////

GPIO\_0, // GPIO Connection 0

GPIO\_1 // GPIO Connection 1

);

//////////////////////// Clock Input ////////////////////////

input CLOCK\_27; // 27 MHz

input CLOCK\_50; // 50 MHz

input EXT\_CLOCK; // External Clock

//////////////////////// Push Button ////////////////////////

input [3:0] KEY; // Pushbutton[3:0]

//////////////////////// DPDT Switch ////////////////////////

input [17:0] SW; // Toggle Switch[17:0]

//////////////////////// 7-SEG Dispaly ////////////////////////

output [6:0] HEX0; // Seven Segment Digit 0

output [6:0] HEX1; // Seven Segment Digit 1

output [6:0] HEX2; // Seven Segment Digit 2

output [6:0] HEX3; // Seven Segment Digit 3

output [6:0] HEX4; // Seven Segment Digit 4

output [6:0] HEX5; // Seven Segment Digit 5

output [6:0] HEX6; // Seven Segment Digit 6

output [6:0] HEX7; // Seven Segment Digit 7

//////////////////////////// LED ////////////////////////////

output [8:0] LEDG; // LED Green[8:0]

output [17:0] LEDR; // LED Red[17:0]

//////////////////////////// UART ////////////////////////////

output UART\_TXD; // UART Transmitter

input UART\_RXD; // UART Receiver

//////////////////////////// IRDA ////////////////////////////

output IRDA\_TXD; // IRDA Transmitter

input IRDA\_RXD; // IRDA Receiver

/////////////////////// SDRAM Interface ////////////////////////

inout [15:0] DRAM\_DQ; // SDRAM Data bus 16 Bits

output [11:0] DRAM\_ADDR; // SDRAM Address bus 12 Bits

output DRAM\_LDQM; // SDRAM Low-byte Data Mask

output DRAM\_UDQM; // SDRAM High-byte Data Mask

output DRAM\_WE\_N; // SDRAM Write Enable

output DRAM\_CAS\_N; // SDRAM Column Address Strobe

output DRAM\_RAS\_N; // SDRAM Row Address Strobe

output DRAM\_CS\_N; // SDRAM Chip Select

output DRAM\_BA\_0; // SDRAM Bank Address 0

output DRAM\_BA\_1; // SDRAM Bank Address 0

output DRAM\_CLK; // SDRAM Clock

output DRAM\_CKE; // SDRAM Clock Enable

//////////////////////// Flash Interface ////////////////////////

inout [7:0] FL\_DQ; // FLASH Data bus 8 Bits

output [21:0] FL\_ADDR; // FLASH Address bus 22 Bits

output FL\_WE\_N; // FLASH Write Enable

output FL\_RST\_N; // FLASH Reset

output FL\_OE\_N; // FLASH Output Enable

output FL\_CE\_N; // FLASH Chip Enable

//////////////////////// SRAM Interface ////////////////////////

inout [15:0] SRAM\_DQ; // SRAM Data bus 16 Bits

output [17:0] SRAM\_ADDR; // SRAM Address bus 18 Bits

output SRAM\_UB\_N; // SRAM High-byte Data Mask

output SRAM\_LB\_N; // SRAM Low-byte Data Mask

output SRAM\_WE\_N; // SRAM Write Enable

output SRAM\_CE\_N; // SRAM Chip Enable

output SRAM\_OE\_N; // SRAM Output Enable

//////////////////// ISP1362 Interface ////////////////////////

inout [15:0] OTG\_DATA; // ISP1362 Data bus 16 Bits

output [1:0] OTG\_ADDR; // ISP1362 Address 2 Bits

output OTG\_CS\_N; // ISP1362 Chip Select

output OTG\_RD\_N; // ISP1362 Write

output OTG\_WR\_N; // ISP1362 Read

output OTG\_RST\_N; // ISP1362 Reset

output OTG\_FSPEED; // USB Full Speed, 0 = Enable, Z = Disable

output OTG\_LSPEED; // USB Low Speed, 0 = Enable, Z = Disable

input OTG\_INT0; // ISP1362 Interrupt 0

input OTG\_INT1; // ISP1362 Interrupt 1

input OTG\_DREQ0; // ISP1362 DMA Request 0

input OTG\_DREQ1; // ISP1362 DMA Request 1

output OTG\_DACK0\_N; // ISP1362 DMA Acknowledge 0

output OTG\_DACK1\_N; // ISP1362 DMA Acknowledge 1

//////////////////// LCD Module 16X2 ////////////////////////////

inout [7:0] LCD\_DATA; // LCD Data bus 8 bits

output LCD\_ON; // LCD Power ON/OFF

output LCD\_BLON; // LCD Back Light ON/OFF

output LCD\_RW; // LCD Read/Write Select, 0 = Write, 1 = Read

output LCD\_EN; // LCD Enable

output LCD\_RS; // LCD Command/Data Select, 0 = Command, 1 = Data

//////////////////// SD Card Interface ////////////////////////

inout SD\_DAT; // SD Card Data

inout SD\_DAT3; // SD Card Data 3

inout SD\_CMD; // SD Card Command Signal

output SD\_CLK; // SD Card Clock

//////////////////////// I2C ////////////////////////////////

inout I2C\_SDAT; // I2C Data

output I2C\_SCLK; // I2C Clock

//////////////////////// PS2 ////////////////////////////////

input PS2\_DAT; // PS2 Data

input PS2\_CLK; // PS2 Clock

//////////////////// USB JTAG link ////////////////////////////

input TDI; // CPLD -> FPGA (data in)

input TCK; // CPLD -> FPGA (clk)

input TCS; // CPLD -> FPGA (CS)

output TDO; // FPGA -> CPLD (data out)

//////////////////////// VGA ////////////////////////////

output VGA\_CLK; // VGA Clock

output VGA\_HS; // VGA H\_SYNC

output VGA\_VS; // VGA V\_SYNC

output VGA\_BLANK; // VGA BLANK

output VGA\_SYNC; // VGA SYNC

output [9:0] VGA\_R; // VGA Red[9:0]

output [9:0] VGA\_G; // VGA Green[9:0]

output [9:0] VGA\_B; // VGA Blue[9:0]

//////////////// Ethernet Interface ////////////////////////////

inout [15:0] ENET\_DATA; // DM9000A DATA bus 16Bits

output ENET\_CMD; // DM9000A Command/Data Select, 0 = Command, 1 = Data

output ENET\_CS\_N; // DM9000A Chip Select

output ENET\_WR\_N; // DM9000A Write

output ENET\_RD\_N; // DM9000A Read

output ENET\_RST\_N; // DM9000A Reset

input ENET\_INT; // DM9000A Interrupt

output ENET\_CLK; // DM9000A Clock 25 MHz

//////////////////// Audio CODEC ////////////////////////////

inout AUD\_ADCLRCK; // Audio CODEC ADC LR Clock

input AUD\_ADCDAT; // Audio CODEC ADC Data

inout AUD\_DACLRCK; // Audio CODEC DAC LR Clock

output AUD\_DACDAT; // Audio CODEC DAC Data

inout AUD\_BCLK; // Audio CODEC Bit-Stream Clock

output AUD\_XCK; // Audio CODEC Chip Clock

//////////////////// TV Devoder ////////////////////////////

input [7:0] TD\_DATA; // TV Decoder Data bus 8 bits

input TD\_HS; // TV Decoder H\_SYNC

input TD\_VS; // TV Decoder V\_SYNC

output TD\_RESET; // TV Decoder Reset

//////////////////////// GPIO ////////////////////////////////

inout [35:0] GPIO\_0; // GPIO Connection 0

inout [35:0] GPIO\_1; // GPIO Connection 1

///////////////////////////////////////////////////////////////////

//=============================================================================

// REG/WIRE declarations

//=============================================================================

// CCD

wire [11:0] CCD\_DATA;

wire CCD\_SDAT;

wire CCD\_SCLK;

wire CCD\_FLASH;

wire CCD\_FVAL;

wire CCD\_LVAL;

wire CCD\_PIXCLK;

wire CCD\_MCLK; // CCD Master Clock

wire [15:0] Read\_DATA1;

wire [15:0] Read\_DATA2;

wire VGA\_CTRL\_CLK;

wire [11:0] mCCD\_DATA;

wire mCCD\_DVAL;

wire mCCD\_DVAL\_d;

wire [15:0] X\_Cont;

wire [15:0] Y\_Cont;

wire [9:0] X\_ADDR;

wire [31:0] Frame\_Cont;

wire DLY\_RST\_0;

wire DLY\_RST\_1;

wire DLY\_RST\_2;

wire Read;

reg [11:0] rCCD\_DATA;

reg rCCD\_LVAL;

reg rCCD\_FVAL;

wire [11:0] sCCD\_R;

wire [11:0] sCCD\_G;

wire [11:0] sCCD\_B;

wire sCCD\_DVAL;

wire [9:0] VGA\_R; // VGA Red[9:0]

wire [9:0] VGA\_G; // VGA Green[9:0]

wire [9:0] VGA\_B; // VGA Blue[9:0]

reg [1:0] rClk;

wire sdram\_ctrl\_clk;

//=============================================================================

// Structural coding

//=============================================================================

assign CCD\_DATA[0] = GPIO\_1[13];

assign CCD\_DATA[1] = GPIO\_1[12];

assign CCD\_DATA[2] = GPIO\_1[11];

assign CCD\_DATA[3] = GPIO\_1[10];

assign CCD\_DATA[4] = GPIO\_1[9];

assign CCD\_DATA[5] = GPIO\_1[8];

assign CCD\_DATA[6] = GPIO\_1[7];

assign CCD\_DATA[7] = GPIO\_1[6];

assign CCD\_DATA[8] = GPIO\_1[5];

assign CCD\_DATA[9] = GPIO\_1[4];

assign CCD\_DATA[10]= GPIO\_1[3];

assign CCD\_DATA[11]= GPIO\_1[1];

assign GPIO\_1[16] = CCD\_MCLK;

assign CCD\_FVAL = GPIO\_1[22];

assign CCD\_LVAL = GPIO\_1[21];

assign CCD\_PIXCLK = GPIO\_1[0];

assign GPIO\_1[19] = 1'b1; // tRIGGER

assign GPIO\_1[17] = DLY\_RST\_1;

/\*

assign LEDR = SW;

assign LEDG = Y\_Cont;

\*/

assign VGA\_CTRL\_CLK= rClk[0];

assign VGA\_CLK = ~rClk[0];

always@(posedge CLOCK\_50) rClk <= rClk+1;

always@(posedge CCD\_PIXCLK)

begin

rCCD\_DATA <= CCD\_DATA;

rCCD\_LVAL <= CCD\_LVAL;

rCCD\_FVAL <= CCD\_FVAL;

end

VGA\_Controller u1 ( // Host Side

.oRequest(Read),

.iRed(wDISP\_R),

.iGreen(wDISP\_G),

.iBlue(wDISP\_B),

// VGA Side

.oVGA\_R(VGA\_R),

.oVGA\_G(VGA\_G),

.oVGA\_B(VGA\_B),

.oVGA\_H\_SYNC(VGA\_HS),

.oVGA\_V\_SYNC(VGA\_VS),

.oVGA\_SYNC(VGA\_SYNC),

.oVGA\_BLANK(VGA\_BLANK),

// Control Signal

.iCLK(VGA\_CTRL\_CLK),

.iRST\_N(DLY\_RST\_2)

);

Reset\_Delay u2 ( .iCLK(CLOCK\_50),

.iRST(KEY[0]),

.oRST\_0(DLY\_RST\_0),

.oRST\_1(DLY\_RST\_1),

.oRST\_2(DLY\_RST\_2)

);

CCD\_Capture u3 ( .oDATA(mCCD\_DATA),

.oDVAL(mCCD\_DVAL),

.oX\_Cont(X\_Cont),

.oY\_Cont(Y\_Cont),

.oFrame\_Cont(Frame\_Cont),

.iDATA(rCCD\_DATA),

.iFVAL(rCCD\_FVAL),

.iLVAL(rCCD\_LVAL),

.iSTART(!KEY[3]),

.iEND(!KEY[2]),

.iCLK(CCD\_PIXCLK),

.iRST(DLY\_RST\_2)

);

RAW2RGB u4 ( .iCLK(CCD\_PIXCLK),

.iRST(DLY\_RST\_1),

.iDATA(mCCD\_DATA),

.iDVAL(mCCD\_DVAL),

.oRed(sCCD\_R),

.oGreen(sCCD\_G),

.oBlue(sCCD\_B),

.oDVAL(sCCD\_DVAL),

.iX\_Cont(X\_Cont),

.iY\_Cont(Y\_Cont)

);

SEG7\_LUT\_8 u5 ( .oSEG0(HEX0),.oSEG1(HEX1),

.oSEG2(HEX2),.oSEG3(HEX3),

.oSEG4(HEX4),.oSEG5(HEX5),

.oSEG6(HEX6),.oSEG7(HEX7),

.iDIG(Frame\_Cont[31:0])

);

sdram\_pll u6 (

.inclk0(CLOCK\_50),

.c0(sdram\_ctrl\_clk),

.c1(DRAM\_CLK)

);

assign CCD\_MCLK = rClk[0];

Sdram\_Control\_4Port u7 ( // HOST Side

.REF\_CLK(CLOCK\_50),

.RESET\_N(1'b1),

.CLK(sdram\_ctrl\_clk),

// FIFO Write Side 1

.WR1\_DATA({1'b0,sCCD\_G[11:7],sCCD\_B[11:2]}),

.WR1(sCCD\_DVAL),

.WR1\_ADDR(0),

.WR1\_MAX\_ADDR(640\*480),

.WR1\_LENGTH(9'h100),

.WR1\_LOAD(!DLY\_RST\_0),

.WR1\_CLK(~CCD\_PIXCLK),

// FIFO Write Side 2

.WR2\_DATA( {1'b0,sCCD\_G[6:2],sCCD\_R[11:2]}),

.WR2(sCCD\_DVAL),

.WR2\_ADDR(22'h100000),

.WR2\_MAX\_ADDR(22'h100000+640\*480),

.WR2\_LENGTH(9'h100),

.WR2\_LOAD(!DLY\_RST\_0),

.WR2\_CLK(~CCD\_PIXCLK),

// FIFO Read Side 1

.RD1\_DATA(Read\_DATA1),

.RD1(Read),

.RD1\_ADDR(0),

.RD1\_MAX\_ADDR(640\*480),

.RD1\_LENGTH(9'h100),

.RD1\_LOAD(!DLY\_RST\_0),

.RD1\_CLK(~VGA\_CTRL\_CLK),

// FIFO Read Side 2

.RD2\_DATA(Read\_DATA2),

.RD2(Read),

.RD2\_ADDR(22'h100000),

.RD2\_MAX\_ADDR(22'h100000+640\*480),

.RD2\_LENGTH(9'h100),

.RD2\_LOAD(!DLY\_RST\_0),

.RD2\_CLK(~VGA\_CTRL\_CLK),

// SDRAM Side

.SA(DRAM\_ADDR),

.BA({DRAM\_BA\_1,DRAM\_BA\_0}),

.CS\_N(DRAM\_CS\_N),

.CKE(DRAM\_CKE),

.RAS\_N(DRAM\_RAS\_N),

.CAS\_N(DRAM\_CAS\_N),

.WE\_N(DRAM\_WE\_N),

.DQ(DRAM\_DQ),

.DQM({DRAM\_UDQM,DRAM\_LDQM})

);

assign UART\_TXD = UART\_RXD;

I2C\_CCD\_Config u8 ( // Host Side

.iCLK(CLOCK\_50),

.iRST\_N(DLY\_RST\_2),

.iZOOM\_MODE\_SW(SW[16]),

.iEXPOSURE\_ADJ(KEY[1]),

.iEXPOSURE\_DEC\_p(SW[0]),

// I2C Side

.I2C\_SCLK(GPIO\_1[24]),

.I2C\_SDAT(GPIO\_1[23])

);

wire [9:0] wVGA\_R = Read\_DATA2[9:0];

wire [9:0] wVGA\_G = {Read\_DATA1[14:10],Read\_DATA2[14:10]};

wire [9:0] wVGA\_B = Read\_DATA1[9:0];

//Coding

wire [9:0] gDATA;

wire gCCD\_DVAL;

wire wGFlag;

RGB2GRAY u9 (

.oDVAL(gCCD\_DVAL),

.oDATA(gDATA),

.oFlag(wGFlag),

.iRed(wVGA\_R),

.iGreen(wVGA\_G),

.iBlue(wVGA\_B),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iDVAL(Read),

);

wire bDATA;

wire bCCD\_DVAL;

PupilBinaryImage u10 (

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iDVAL(gCCD\_DVAL),

.iDATA(gDATA),

.oDATA(bDATA),

.oDVAL(bCCD\_DVAL),

);

wire oErosion;

wire eCCD\_DVAL;

Erosion u11 (

.iDATA(bDATA),

.iDVAL(bCCD\_DVAL),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.oDATA(oErosion),

.oDVAL(eCCD\_DVAL),

);

wire oDilation;

wire dCCD\_DVAL;

Dilation u12 (

.iDATA(oErosion),

.iDVAL(eCCD\_DVAL),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.oDATA(oDilation),

.oDVAL(dCCD\_DVAL),

);

wire [9:0] woC\_1;

wire [9:0] woR\_1;

wire [9:0] woC\_2;

wire [9:0] woR\_2;

wire [9:0] woC\_3;

wire [9:0] woR\_3;

wire [9:0] woC\_4;

wire [9:0] woR\_4;

wire [9:0] woC\_PC;

wire [9:0] woC\_PR;

wire [9:0] woPRadius;

wire wPFlagCoor;

wire oDetectPupil;

wire dpCCD\_DVAL;

Detect\_OuterPupil u13(

.iEND(SW[17]),

.iDATA(oDilation),

.iDVAL(dCCD\_DVAL),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.oC\_1(woC\_1), //top

.oR\_1(woR\_1),

.oC\_2(woC\_2),//left

.oR\_2(woR\_2),

.oC\_3(woC\_3),//right

.oR\_3(woR\_3),

.oC\_4(woC\_4),//btm

.oR\_4(woR\_4),

.oC\_PC(woC\_PC),

.oC\_PR(woC\_PR),

.oFlag(wPFlagCoor),

.oDATA(oDetectPupil),

.oDVAL(dpCCD\_DVAL),

.oPRadius(woPRadius),

);

wire [9:0]pPupilCoor;

wire ppCCD\_DVAL;

Output\_Point u14(

.iDATA(ibDATA),

.iDVAL(ibCCD\_DVAL),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iC\_1(woC\_1),

.iR\_1(woR\_1),

.iC\_2(woC\_2),

.iR\_2(woR\_2),

.iC\_3(woC\_3),

.iR\_3(woR\_3),

.iC\_4(woC\_4),

.iR\_4(woR\_4),

.iC\_PC(woC\_PC),

.iC\_PR(woC\_PR),

.iI\_R1(woI\_R1),

.iI\_C1(woI\_C1),

.iI\_R2(woI\_R2),

.iI\_C2(woI\_C2),

.oDVAL(ppCCD\_DVAL),

.oDATA(pPupilCoor),

);

wire ibData;

wire ibCCD\_DVAL;

IrisBinaryImage u15 (

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iDVAL(gCCD\_DVAL),

.iDATA(gDATA),

.oDATA(ibDATA),

.oDVAL(ibCCD\_DVAL),

);

wire [9:0] woI\_R1;

wire [9:0] woI\_C1;

wire [9:0] woI\_R2;

wire [9:0] woI\_C2;

wire [9:0] woIRadius;

wire wIFlagCoor;

Detect\_InnerIris u16(

.iDATA(ibDATA),

.iDVAL(ibCCD\_DVAL),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iC\_PC(woC\_PC),

.iC\_PR(woC\_PR),

.oFlag(wIFlagCoor),

.oI\_R1(woI\_R1), //left

.oI\_C1(woI\_C1),

.oI\_R2(woI\_R2), //right

.oI\_C2(woI\_C2),

.oIRadius(woIRadius),

);

wire cgCCD\_DVAL;

wire [9:0] cgDATA;

GRAYCOPY u18 (

.oDVAL(cgCCD\_DVAL),

.oDATA(cgDATA),

.iRed(wVGA\_R),

.iGreen(wVGA\_G),

.iBlue(wVGA\_B),

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iDVAL(Read),

.iSignal(wSignal)

);

wire [9:0] wNDATA;

wire signalToCompress;

irisNormalization u17(

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iDVAL(cgCCD\_DVAL),

.iPupilFlag(wPFlagCoor),

.iIrisFlag(wIFlagCoor),

.iCaptured(!KEY[2]),

.iStart(!KEY[3]),

.iDATA(cgDATA),

.pupil\_radius(woPRadius),

.iris\_radius(woIRadius),

.centre\_pointX(woC\_PC),

.centre\_pointY(woC\_PR),

.ograySignal(wSignal),

.oDATA(wNDATA),

.oSignal(signalToCompress),

);

assign LEDR[9] = signalToCompress;

assign LEDR[0] = wSignal;

wire arrayToSOM[0:9][0:9];

wire startSOM;

irisCompression compress(

.iCLK(VGA\_CTRL\_CLK),

.iRST(DLY\_RST\_2),

.iDATA(wNDATA),

.compressSignal(signalToCompress),

.oArray(arrayToSOM),

.oSignal(startSOM),

);

assign LEDR[10] = startSOM;

SOM som(

.CLOCK\_50(CLOCK\_50),

.iRST(DLY\_RST\_2),

.bRST(SW[7]),

.startRecognition(SW[17]),

.startTraining(SW[15]),

.trainUser(SW[14]),

.storeControl(SW[13]),

.storeUser(SW[12:9]),

.data(arrayToSOM),

.LED(LEDR[17:14]),

);

\*/

//output image

reg [9:0] pupilBinaryImage;

always@(posedge CLOCK\_50)begin

if(bDATA == 1)begin

pupilBinaryImage = 10'd1023;

end

else begin

pupilBinaryImage = 10'd0;

end

end

reg [9:0] erosionImage;

always@(posedge CLOCK\_50)begin

if(oErosion == 1)begin

erosionImage = 10'd1023;

end

else begin

erosionImage = 10'd0;

end

end

reg [9:0] dilationImage;

always@(posedge CLOCK\_50)begin

if(oDilation == 1)begin

dilationImage = 10'd1023;

end

else begin

dilationImage = 10'd0;

end

end

reg [9:0] irisBinaryImage;

always@(posedge CLOCK\_50)begin

if(ibDATA == 1)begin

irisBinaryImage = 10'd1023;

end

else begin

irisBinaryImage = 10'd0;

end

end

wire [9:0] wDISP\_R =

SW[6] ? pPupilCoor :

SW[5] ? irisBinaryImage:

SW[4] ? dilationImage :

SW[3] ? erosionImage :

SW[2] ? pupilBinaryImage:

SW[1] ? gDATA :

wVGA\_R;

wire [9:0] wDISP\_G =

SW[6] ? pPupilCoor :

SW[5] ? irisBinaryImage:

SW[4] ? dilationImage :

SW[3] ? erosionImage :

SW[2] ? pupilBinaryImage:

SW[1] ? gDATA :

wVGA\_G;

wire [9:0] wDISP\_B =

SW[6] ? pPupilCoor :

SW[5] ? irisBinaryImage:

SW[4] ? dilationImage :

SW[3] ? erosionImage :

SW[2] ? pupilBinaryImage:

SW[1] ? gDATA :

wVGA\_B;

endmodule

**I²C Configuration:**

module I2C\_CCD\_Config ( // Host Side

iCLK,

iRST\_N,

iZOOM\_MODE\_SW,

iEXPOSURE\_ADJ,

iEXPOSURE\_DEC\_p,

// I2C Side

I2C\_SCLK,

I2C\_SDAT

);

// Host Side

input iCLK;

input iRST\_N;

input iZOOM\_MODE\_SW;

// I2C Side

output I2C\_SCLK;

inout I2C\_SDAT;

// Internal Registers/Wires

reg [15:0] mI2C\_CLK\_DIV;

reg [31:0] mI2C\_DATA;

reg mI2C\_CTRL\_CLK;

reg mI2C\_GO;

wire mI2C\_END;

wire mI2C\_ACK;

reg [23:0] LUT\_DATA;

reg [5:0] LUT\_INDEX;

reg [3:0] mSetup\_ST;

////////////// CMOS sensor registers setting //////////////////////

input iEXPOSURE\_ADJ;

input iEXPOSURE\_DEC\_p;

parameter default\_exposure = 16'h0438;

parameter exposure\_change\_value = 16'd200;

reg [24:0] combo\_cnt;

wire combo\_pulse;

reg [1:0] izoom\_mode\_sw\_delay;

reg [3:0] iexposure\_adj\_delay;

wire exposure\_adj\_set;

wire exposure\_adj\_reset;

reg [15:0] senosr\_exposure;

wire [23:0] sensor\_start\_row;

wire [23:0] sensor\_start\_column;

wire [23:0] sensor\_row\_size;

wire [23:0] sensor\_column\_size;

wire [23:0] sensor\_row\_mode;

wire [23:0] sensor\_column\_mode;

assign sensor\_start\_row = iZOOM\_MODE\_SW ? 24'h010036 : 24'h010000;

assign sensor\_start\_column = iZOOM\_MODE\_SW ? 24'h020010 : 24'h020000;

assign sensor\_row\_size = iZOOM\_MODE\_SW ? 24'h0303BF : 24'h03077F;

assign sensor\_column\_size = iZOOM\_MODE\_SW ? 24'h0404FF : 24'h0409FF;

assign sensor\_row\_mode = iZOOM\_MODE\_SW ? 24'h220000 : 24'h220011;

assign sensor\_column\_mode = iZOOM\_MODE\_SW ? 24'h230000 : 24'h230011;

always@(posedge iCLK or negedge iRST\_N)

begin

if (!iRST\_N)

begin

iexposure\_adj\_delay <= 0;

end

else

begin

iexposure\_adj\_delay <= {iexposure\_adj\_delay[2:0],iEXPOSURE\_ADJ};

end

end

assign exposure\_adj\_set = ({iexposure\_adj\_delay[0],iEXPOSURE\_ADJ}==2'b10) ? 1 : 0 ;

assign exposure\_adj\_reset = ({iexposure\_adj\_delay[3:2]}==2'b10) ? 1 : 0 ;

always@(posedge iCLK or negedge iRST\_N)

begin

if (!iRST\_N)

senosr\_exposure <= default\_exposure;

else if (exposure\_adj\_set|combo\_pulse)

begin

if (iEXPOSURE\_DEC\_p)

begin

if ((senosr\_exposure < exposure\_change\_value)||

(senosr\_exposure == 16'h0))

senosr\_exposure <= 0;

else

senosr\_exposure <= senosr\_exposure - exposure\_change\_value;

end

else

begin

if (((16'hffff -senosr\_exposure) <exposure\_change\_value)||

(senosr\_exposure == 16'hffff))

senosr\_exposure <= 16'hffff;

else

senosr\_exposure <= senosr\_exposure + exposure\_change\_value;

end

end

end

always@(posedge iCLK or negedge iRST\_N)

begin

if (!iRST\_N)

combo\_cnt <= 0;

else if (!iexposure\_adj\_delay[3])

combo\_cnt <= combo\_cnt + 1;

else

combo\_cnt <= 0;

end

assign combo\_pulse = (combo\_cnt == 25'h1fffff) ? 1 : 0;

wire i2c\_reset;

assign i2c\_reset = iRST\_N & ~exposure\_adj\_reset & ~combo\_pulse ;

/////////////////////////////////////////////////////////////////////

// Clock Setting

parameter CLK\_Freq = 50000000; // 50 MHz

parameter I2C\_Freq = 20000; // 20 KHz

// LUT Data Number

parameter LUT\_SIZE = 25;

///////////////////// I2C Control Clock ////////////////////////

always@(posedge iCLK or negedge i2c\_reset)

begin

if(!i2c\_reset)

begin

mI2C\_CTRL\_CLK <= 0;

mI2C\_CLK\_DIV <= 0;

end

else

begin

if( mI2C\_CLK\_DIV < (CLK\_Freq/I2C\_Freq) )

mI2C\_CLK\_DIV <= mI2C\_CLK\_DIV+1;

else

begin

mI2C\_CLK\_DIV <= 0;

mI2C\_CTRL\_CLK <= ~mI2C\_CTRL\_CLK;

end

end

end

////////////////////////////////////////////////////////////////////

I2C\_Controller u0 ( .CLOCK(mI2C\_CTRL\_CLK), // Controller Work Clock

.I2C\_SCLK(I2C\_SCLK), // I2C CLOCK

.I2C\_SDAT(I2C\_SDAT), // I2C DATA

.I2C\_DATA(mI2C\_DATA), // DATA:[SLAVE\_ADDR,SUB\_ADDR,DATA]

.GO(mI2C\_GO), // GO transfor

.END(mI2C\_END), // END transfor

.ACK(mI2C\_ACK), // ACK

.RESET(i2c\_reset)

);

////////////////////////////////////////////////////////////////////

////////////////////// Config Control ////////////////////////////

//always@(posedge mI2C\_CTRL\_CLK or negedge iRST\_N)

always@(posedge mI2C\_CTRL\_CLK or negedge i2c\_reset)

begin

if(!i2c\_reset)

begin

LUT\_INDEX <= 0;

mSetup\_ST <= 0;

mI2C\_GO <= 0;

end

else if(LUT\_INDEX<LUT\_SIZE)

begin

case(mSetup\_ST)

0: begin

mI2C\_DATA <= {8'hBA,LUT\_DATA};

mI2C\_GO <= 1;

mSetup\_ST <= 1;

end

1: begin

if(mI2C\_END)

begin

if(!mI2C\_ACK)

mSetup\_ST <= 2;

else

mSetup\_ST <= 0;

mI2C\_GO <= 0;

end

end

2: begin

LUT\_INDEX <= LUT\_INDEX+1;

mSetup\_ST <= 0;

end

endcase

end

end

////////////////////////////////////////////////////////////////////

///////////////////// Config Data LUT //////////////////////////

always

begin

case(LUT\_INDEX)

0 : LUT\_DATA <= 24'h000000;

1 : LUT\_DATA <= 24'h20c000; // Mirror Row and Columns

2 : LUT\_DATA <= {8'h09,senosr\_exposure};// Exposure

3 : LUT\_DATA <= 24'h050000; // H\_Blanking

4 : LUT\_DATA <= 24'h060019; // V\_Blanking

5 : LUT\_DATA <= 24'h0A8000; // change latch

6 : LUT\_DATA <= 24'h2B0032; // Green 1 Gain

7 : LUT\_DATA <= 24'h2C0134; // Blue Gain

8 : LUT\_DATA <= 24'h2D0338; // Red Gain

9 : LUT\_DATA <= 24'h2E0032; // Green 2 Gain

10 : LUT\_DATA <= 24'h100051; // set up PLL power on

11 : LUT\_DATA <= 24'h111807; // PLL\_m\_Factor<<8+PLL\_n\_Divider

12 : LUT\_DATA <= 24'h120002; // PLL\_p1\_Divider

13 : LUT\_DATA <= 24'h100053; // set USE PLL

14 : LUT\_DATA <= 24'h980000; // disble calibration

15 : LUT\_DATA <= 24'hA00000; // Test pattern control

16 : LUT\_DATA <= 24'hA10000; // Test green pattern value

17 : LUT\_DATA <= 24'hA20FFF; // Test red pattern value

18 : LUT\_DATA <= sensor\_start\_row ; // set start row

19 : LUT\_DATA <= sensor\_start\_column ; // set start column

20 : LUT\_DATA <= sensor\_row\_size; // set row size

21 : LUT\_DATA <= sensor\_column\_size; // set column size

22 : LUT\_DATA <= sensor\_row\_mode; // set row mode in bin mode

23 : LUT\_DATA <= sensor\_column\_mode; // set column mode in bin mode

24 : LUT\_DATA <= 24'h4901A8; // row black target

default:LUT\_DATA <= 24'h000000;

endcase

end

endmodule

**I²C Controller:**

module I2C\_Controller (

CLOCK,

I2C\_SCLK,//I2C CLOCK

I2C\_SDAT,//I2C DATA

I2C\_DATA,//DATA:[SLAVE\_ADDR,SUB\_ADDR,DATA]

GO, //GO transfor

END, //END transfor

ACK, //ACK

RESET

);

input CLOCK;

input [31:0]I2C\_DATA;

input GO;

input RESET;

inout I2C\_SDAT;

output I2C\_SCLK;

output END;

output ACK;

reg SDO;

reg SCLK;

reg END;

reg [31:0]SD;

reg [6:0]SD\_COUNTER;

wire I2C\_SCLK=SCLK | ( ((SD\_COUNTER >= 4) & (SD\_COUNTER <=39))? ~CLOCK :0 );

wire I2C\_SDAT=SDO?1'bz:0 ;

reg ACK1,ACK2,ACK3,ACK4;

wire ACK=ACK1 | ACK2 |ACK3 |ACK4;

//--I2C COUNTER

always @(negedge RESET or posedge CLOCK ) begin

if (!RESET) SD\_COUNTER=6'b111111;

else begin

if (GO==0)

SD\_COUNTER=0;

else

if (SD\_COUNTER < 41) SD\_COUNTER=SD\_COUNTER+1;

end

end

//----

always @(negedge RESET or posedge CLOCK ) begin

if (!RESET) begin SCLK=1;SDO=1; ACK1=0;ACK2=0;ACK3=0;ACK4=0; END=1; end

else

case (SD\_COUNTER)

6'd0 : begin ACK1=0 ;ACK2=0 ;ACK3=0 ;ACK4=0 ; END=0; SDO=1; SCLK=1;end

//start

6'd1 : begin SD=I2C\_DATA;SDO=0;end

6'd2 : SCLK=0;

//SLAVE ADDR

6'd3 : SDO=SD[31];

6'd4 : SDO=SD[30];

6'd5 : SDO=SD[29];

6'd6 : SDO=SD[28];

6'd7 : SDO=SD[27];

6'd8 : SDO=SD[26];

6'd9 : SDO=SD[25];

6'd10 : SDO=SD[24];

6'd11 : SDO=1'b1;//ACK

//SUB ADDR

6'd12 : begin SDO=SD[23]; ACK1=I2C\_SDAT; end

6'd13 : SDO=SD[22];

6'd14 : SDO=SD[21];

6'd15 : SDO=SD[20];

6'd16 : SDO=SD[19];

6'd17 : SDO=SD[18];

6'd18 : SDO=SD[17];

6'd19 : SDO=SD[16];

6'd20 : SDO=1'b1;//ACK

//DATA

6'd21 : begin SDO=SD[15]; ACK2=I2C\_SDAT; end

6'd22 : SDO=SD[14];

6'd23 : SDO=SD[13];

6'd24 : SDO=SD[12];

6'd25 : SDO=SD[11];

6'd26 : SDO=SD[10];

6'd27 : SDO=SD[9];

6'd28 : SDO=SD[8];

6'd29 : SDO=1'b1;//ACK

//DATA

6'd30 : begin SDO=SD[7]; ACK3=I2C\_SDAT; end

6'd31 : SDO=SD[6];

6'd32 : SDO=SD[5];

6'd33 : SDO=SD[4];

6'd34 : SDO=SD[3];

6'd35 : SDO=SD[2];

6'd36 : SDO=SD[1];

6'd37 : SDO=SD[0];

6'd38 : SDO=1'b1;//ACK

//stop

6'd39 : begin SDO=1'b0; SCLK=1'b0; ACK4=I2C\_SDAT; end

6'd40 : SCLK=1'b1;

6'd41 : begin SDO=1'b1; END=1; end

endcase

end

**endmodule**

**CCD\_Capture:**

module CCD\_Capture( oDATA,

oDVAL,

oX\_Cont,

oY\_Cont,

oFrame\_Cont,

iDATA,

iFVAL,

iLVAL,

iSTART,

iEND,

iCLK,

iRST

);

input [11:0] iDATA;

input iFVAL;

input iLVAL;

input iSTART;

input iEND;

input iCLK;

input iRST;

output [11:0] oDATA;

output [15:0] oX\_Cont;

output [15:0] oY\_Cont;

output [31:0] oFrame\_Cont;

output oDVAL;

reg Pre\_FVAL;

reg mCCD\_FVAL;

reg mCCD\_LVAL;

reg [11:0] mCCD\_DATA;

reg [15:0] X\_Cont;

reg [15:0] Y\_Cont;

reg [31:0] Frame\_Cont;

reg mSTART;

parameter COLUMN\_WIDTH = 1280;

assign oX\_Cont = X\_Cont;

assign oY\_Cont = Y\_Cont;

assign oFrame\_Cont = Frame\_Cont;

assign oDATA = mCCD\_DATA;

assign oDVAL = mCCD\_FVAL&mCCD\_LVAL;

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

mSTART <= 0;

else

begin

if(iSTART)

mSTART <= 1;

if(iEND)

mSTART <= 0;

end

end

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

begin

Pre\_FVAL <= 0;

mCCD\_FVAL <= 0;

mCCD\_LVAL <= 0;

X\_Cont <= 0;

Y\_Cont <= 0;

end

else

begin

Pre\_FVAL <= iFVAL;

if( ({Pre\_FVAL,iFVAL}==2'b01) && mSTART )

mCCD\_FVAL <= 1;

else if({Pre\_FVAL,iFVAL}==2'b10)

mCCD\_FVAL <= 0;

mCCD\_LVAL <= iLVAL;

if(mCCD\_FVAL)

begin

if(mCCD\_LVAL)

begin

if(X\_Cont<(COLUMN\_WIDTH-1))

X\_Cont <= X\_Cont+1;

else

begin

X\_Cont <= 0;

Y\_Cont <= Y\_Cont+1;

end

end

end

else

begin

X\_Cont <= 0;

Y\_Cont <= 0;

end

end

end

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

Frame\_Cont <= 0;

else

begin

if( ({Pre\_FVAL,iFVAL}==2'b01) && mSTART )

Frame\_Cont <= Frame\_Cont+1;

end

end

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

mCCD\_DATA <= 0;

else if (iLVAL)

mCCD\_DATA <= iDATA;

else

mCCD\_DATA <= 0;

end

endmodule

**RAW2RGB:**

module RAW2RGB( oRed,

oGreen,

oBlue,

oDVAL,

iX\_Cont,

iY\_Cont,

iDATA,

iDVAL,

iCLK,

iRST

);

input [10:0] iX\_Cont;

input [10:0] iY\_Cont;

input [11:0] iDATA;

input iDVAL;

input iCLK;

input iRST;

output [11:0] oRed;

output [11:0] oGreen;

output [11:0] oBlue;

output oDVAL;

wire [11:0] mDATA\_0;

wire [11:0] mDATA\_1;

reg [11:0] mDATAd\_0;

reg [11:0] mDATAd\_1;

reg [11:0] mCCD\_R;

reg [12:0] mCCD\_G;

reg [11:0] mCCD\_B;

reg mDVAL;

assign oRed = mCCD\_R[11:0];

assign oGreen = mCCD\_G[12:1];

assign oBlue = mCCD\_B[11:0];

assign oDVAL = mDVAL;

Line\_Buffer u0 ( .clken(iDVAL),

.clock(iCLK),

.shiftin(iDATA),

.taps0x(mDATA\_1),

.taps1x(mDATA\_0) );

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

begin

mCCD\_R <= 0;

mCCD\_G <= 0;

mCCD\_B <= 0;

mDATAd\_0<= 0;

mDATAd\_1<= 0;

mDVAL <= 0;

end

else

begin

mDATAd\_0 <= mDATA\_0;

mDATAd\_1 <= mDATA\_1;

mDVAL <= {iY\_Cont[0]|iX\_Cont[0]} ? 1'b0 : iDVAL;

if({iY\_Cont[0],iX\_Cont[0]}==2'b10)

begin

mCCD\_R <= mDATA\_0;

mCCD\_G <= mDATAd\_0+mDATA\_1;

mCCD\_B <= mDATAd\_1;

end

else if({iY\_Cont[0],iX\_Cont[0]}==2'b11)

begin

mCCD\_R <= mDATAd\_0;

mCCD\_G <= mDATA\_0+mDATAd\_1;

mCCD\_B <= mDATA\_1;

end

else if({iY\_Cont[0],iX\_Cont[0]}==2'b00)

begin

mCCD\_R <= mDATA\_1;

mCCD\_G <= mDATA\_0+mDATAd\_1;

mCCD\_B <= mDATAd\_0;

end

else if({iY\_Cont[0],iX\_Cont[0]}==2'b01)

begin

mCCD\_R <= mDATAd\_1;

mCCD\_G <= mDATAd\_0+mDATA\_1;

mCCD\_B <= mDATA\_0;

end

end

end

endmodule

**RGB2Grayscale:**

module RGB2GRAY(

oDVAL,

oDATA,

oFlag,

iRed,

iGreen,

iBlue,

iCLK,

iRST,

iDVAL,

);

input iDVAL;

input iCLK;

input iRST;

output reg[9:0] oDATA;

output reg oDVAL;

output oFlag;

input [9:0] iRed;

input [9:0] iGreen;

input [9:0] iBlue;

reg [20:0] counter;

reg tempFlag;

assign oFlag = tempFlag;

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

begin

oDVAL <= 0;

oDATA <= 10'b0;

counter <= 0;

tempFlag <= 0;

end

else

begin

oDVAL <= iDVAL;

if(counter < 307199)begin

if(iDVAL)begin

oDATA = (iRed+iGreen+iBlue)/3;

counter = counter + 1;

tempFlag = 0;

end

end

else begin

counter = 0;

tempFlag = 1;

end

end

end

endmodule

**PupilBinary:**

module PupilBinaryImage(

iCLK, //VGA Clock

iRST,

iDVAL,

iDATA,

oDATA,

oDVAL,

);

parameter threshold = 10'd487;//d278;

input iCLK;

input iRST;

input iDVAL;

input[9:0] iDATA;

output reg oDATA;

output reg oDVAL;

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

begin

oDVAL <= 0;

oDATA <= 0;

end

else

begin

oDVAL <= iDVAL;

if(iDVAL) begin

if(iDATA > threshold) begin

oDATA <= 1;

end

else begin

oDATA <= 0;

end

end

else begin

oDATA <= 0;

end

end

end

endmodule

**Iris Binary:**

module IrisBinaryImage(

iCLK, //VGA Clock

iRST,

iDVAL,

iDATA,

oDATA,

oDVAL,

);

parameter threshold = 10'd712;

input iCLK;

input iRST;

input iDVAL;

input[9:0] iDATA;

output reg [9:0]oDATA;

output reg oDVAL;

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)

begin

oDVAL <= 1'b0;

oDATA <= 0;

end

else

begin

oDVAL <= iDVAL;

if(iDVAL)

begin

if(iDATA > threshold)

oDATA <= 1;

else

oDATA <= 0;

end

else

oDATA <= 0;

end

end

endmodule

**Erosion:**

/////////////////////////////////////////////////////////////////

// This Module Is To Reduce The Unwanted Image/Noise //

// ------------------------------------------------------------//

// Purpose: To reduce image size //

// Input : Pupil Binary Image //

// Output : Erosion Image In Binary Form //

/////////////////////////////////////////////////////////////////

module Erosion(

iDATA,

iDVAL,

iCLK,

iRST,

oDATA,

oDVAL

);

input iDATA;

input iDVAL;

input iCLK;

input iRST;

output reg oDATA;

output reg oDVAL;

reg C1;

reg C2;

reg C3;

reg C4;

reg C5;

reg C6;

reg C7;

reg C8;

reg C9;

wire outLine1;

wire outLine2;

wire outLine3;

Line\_Buffer\_Erosion erosion(

.clken(iDVAL),

.clock(iCLK),

.shiftin(iDATA),

.taps0x(outLine1),

.taps1x(outLine2),

.taps2x(outLine3)

);

always@(posedge iCLK or negedge iRST)begin

if(!iRST)begin

C1 <= 1;

C2 <= 1;

C3 <= 1;

C4 <= 1;

C5 <= 1;

C6 <= 1;

C7 <= 1;

C8 <= 1;

C9 <= 1;

end

else begin

oDVAL <= iDVAL;

C9 <= outLine1;

C8 <= C9;

C7 <= C8;

C6 <= outLine2;

C5 <= C6;

C4 <= C5;

C3 <= outLine3;

C2 <= C3;

C1 <= C2;

if(iDVAL)begin

oDATA <= (C9 | C8 | C7 | C6 | C5 | C4 | C3 | C2 | C1);

end

else begin

oDATA <= 0;

end

end

end

endmodule

**Dilation:**

module Dilation(

iDATA,

iDVAL,

iCLK,

iRST,

oDATA,

oDVAL

);

input iDATA;

input iDVAL;

input iCLK;

input iRST;

output reg oDATA;

output reg oDVAL;

reg C1;

reg C2;

reg C3;

reg C4;

reg C5;

reg C6;

reg C7;

reg C8;

reg C9;

wire outLine1;

wire outLine2;

wire outLine3;

Line\_Buffer\_Dilation dilation(

.clken(iDVAL),

.clock(iCLK),

.shiftin(iDATA),

.taps0x(outLine1),

.taps1x(outLine2),

.taps2x(outLine3)

);

always@(posedge iCLK or negedge iRST)begin

if(!iRST)begin

C1 <= 1;

C2 <= 1;

C3 <= 1;

C4 <= 1;

C5 <= 1;

C6 <= 1;

C7 <= 1;

C8 <= 1;

C9 <= 1;

end

else begin

oDVAL <= iDVAL;

C9 <= outLine1;

C8 <= C9;

C7 <= C8;

C6 <= outLine2;

C5 <= C6;

C4 <= C5;

C3 <= outLine3;

C2 <= C3;

C1 <= C2;

if(iDVAL)begin

oDATA <= (C9 & C8 & C7 & C6 & C5 & C4 & C3 & C2 & C1);

end

else begin

oDATA <= 0;

end

end

end

endmodule

**Detect Pupil Point:**

module Detect\_OuterPupil(

iDATA,

iDVAL,

iCLK,

iRST,

iEND,

oC\_1,

oR\_1,

oC\_2,

oR\_2,

oC\_3,

oR\_3,

oC\_4,

oR\_4,

oC\_PC,

oC\_PR,

oFlag,

oDATA,

oDVAL,

oPRadius,

);

input iEND;

input iDATA;

input iDVAL;

input iCLK;

input iRST;

reg [9:0] r;

reg [9:0] c;

output [9:0]oC\_1;

output [9:0]oR\_1;

output [9:0]oC\_2;

output [9:0]oR\_2;

output [9:0]oC\_3;

output [9:0]oR\_3;

output [9:0]oC\_4;

output [9:0]oR\_4;

output [9:0]oC\_PC;

output [9:0]oC\_PR;

output [9:0]oPRadius;

output oFlag;

output reg oDATA;

output reg oDVAL;

//register

reg foundFirstRowBlack;

reg unsigned [9:0] temp1;

reg unsigned [9:0] temp2;

reg unsigned [9:0] temp3;

reg unsigned [9:0] temp4;

reg unsigned [9:0] temp5;

reg unsigned [9:0] temp6;

reg unsigned [9:0] temp7;

reg unsigned [9:0] temp8;

reg unsigned [9:0] temp9;

reg unsigned [9:0] temp10;

reg unsigned [9:0] temp11;

reg unsigned [9:0] temp12;

reg unsigned [9:0] temp13;

reg unsigned [9:0] temp14;

reg unsigned [9:0] temp15;

reg unsigned [9:0] temp16;

reg unsigned [9:0] temp17;

reg unsigned [9:0] temp18;

reg unsigned [9:0] storeSmallestCol;

reg unsigned [9:0] storeLargestCol;

reg unsigned [9:0] storeLargestRow;

reg [9:0] endFirstCol;

reg doneFirstSearch;

reg SmallestFlag;

reg doneChecking;

reg [9:0] counter;

reg previousRowFLag;

reg [9:0] previousRow;

reg [9:0] storeRow;

reg tempFlag;

reg tempFlag1;

assign oC\_1 = temp1; //top

assign oR\_1 = temp2; //top

assign oC\_2 = temp9; //left

assign oR\_2 = temp10;//left

assign oC\_3 = temp11;//right

assign oR\_3 = temp12;//right

assign oC\_4 = temp13;//btm

assign oR\_4 = temp14;//btm

assign oC\_PC = temp15;

assign oC\_PR = temp16;

assign oPRadius = temp17;

assign oFlag = tempFlag;

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)begin

r <= 0;

c <= 0;

temp1 <= 0;

temp2 <= 0;

temp3 <= 0;

temp4 <= 0;

temp5 <= 0;

temp6 <= 0;

temp7 <= 0;

temp8 <= 0;

temp9 <= 0;

temp10 <= 0;

temp11 <= 0;

temp12 <= 0;

temp13 <= 0;

temp14 <= 0;

temp15 <= 0;

temp16 <= 0;

temp17 <= 0;

temp18 <= 0;

tempFlag <= 0;

tempFlag1 <= 0;

foundFirstRowBlack <= 0;

storeSmallestCol <= 0;

storeLargestCol <= 0;

storeLargestRow <= 0;

doneFirstSearch <= 0;

SmallestFlag <= 0;

counter <= 0;

endFirstCol <= 0;

previousRowFLag <= 0;

doneChecking <= 0;

previousRow <= 0;

storeRow <= 0;

end

else begin

if(iDVAL)begin

if(r<480)begin

if(c<640)begin

if(doneFirstSearch == 0)begin

if(foundFirstRowBlack == 0)begin

if(r>=130 && r<=372)begin

if(c>=100 && c<=487)begin

if(iDATA == 0)begin

temp1 <= c;

temp2 <= r;

foundFirstRowBlack = 1;

storeSmallestCol = c;

storeLargestCol = c;

endFirstCol = c + 1;

previousRow = r;

storeLargestRow = r;

previousRowFLag = 0;

end

end

end

end

if(foundFirstRowBlack == 1)begin

endFirstCol = endFirstCol + 1;

end

end

if(doneFirstSearch == 1)begin

if(doneChecking == 0)begin

if(r>=130 && r<=472)begin

if(c>=100 && c<=627)begin

if(counter == 627) begin

doneChecking = 1;

end

if(iDATA == 1 && r > previousRow)begin

counter = counter + 1;

previousRowFLag = 0;

end

else if(iDATA == 0)begin

if(c <= storeSmallestCol)begin

temp3 = c;

temp4 = r;

storeSmallestCol = c;

storeRow = r;

end

if(c >= storeLargestCol && r == storeRow)begin

temp5 = c;

temp6 = r;

storeLargestCol = c;

end

if(c == temp1 && r>=storeLargestRow)begin

temp7 = c;

temp8 = r;

storeLargestRow = r;

end

if(previousRowFLag == 0)begin

previousRow = r + 1;

previousRowFLag = 1;

end

counter = 0;

end

end

end

end

if(doneChecking == 1)begin

temp9 = temp3;

temp10 = temp4;

temp11 = temp5;

temp12 = temp6;

temp13 = temp7;

temp14 = temp8;

temp15 = temp3+((temp5-temp3)/2);

temp16 = temp2+((temp8-temp2)/2);

if(((temp5-temp3)/2) > ((temp8-temp2)/2))begin

temp17 = (temp5-temp3)/2;

end

else begin

temp17 = ((temp8-temp2)/2);

end

tempFlag = doneChecking;

end

end

c = c + 1;

end

if(endFirstCol == 640)begin

doneFirstSearch = 1;

end

if(c == 640)begin

c = 0;

r = r + 1;

end

end // r

if(r == 480) begin

c = 0;

r = 0;

foundFirstRowBlack = 0;

storeSmallestCol = 0;

storeLargestCol = 0;

counter = 0;

doneFirstSearch = 0;

endFirstCol = 0;

previousRowFLag = 0;

doneChecking = 0;

storeLargestCol = 0;

storeLargestRow = 0;

storeRow = 0;

tempFlag = 0;

end

end

end

end

endmodule

**Detect Iris Point:**

module Detect\_InnerIris(

iDATA,

iDVAL,

iCLK,

iRST,

iC\_PC,

iC\_PR,

oFlag,

oI\_R1,

oI\_C1,

oI\_R2,

oI\_C2,

oIRadius,

);

input [9:0] iDATA;

input iDVAL;

input iCLK;

input iRST;

input [9:0]iC\_PC;

input [9:0]iC\_PR;

output [9:0] oI\_R1;

output [9:0] oI\_C1;

output [9:0] oI\_R2;

output [9:0] oI\_C2;

output [9:0] oIRadius;

output oFlag;

reg [9:0] temp1;

reg [9:0] temp2;

reg [9:0] temp3;

reg unsigned[9:0] temp4;

reg unsigned[9:0] temp6;

reg unsigned[9:0] temp7;

reg unsigned[9:0] temp8;

reg unsigned[9:0] temp9;

reg unsigned[9:0] temp10;

reg unsigned[9:0] temp11;

reg [9:0] r;

reg [9:0] c;

reg [9:0] storeLargestCOL;

reg [9:0] storeSmallestCOL;

reg firstSearch;

reg [9:0]counter;

reg smallestCOLFlag;

reg largestCOLFlag;

reg [9:0] smallestCOL;

reg [9:0] largestCOL;

reg tempFlag;

assign oI\_R1 = temp6;//left

assign oI\_C1 = temp7;//left

assign oI\_R2 = temp10;//right

assign oI\_C2 = temp11;//right

assign oIRadius = temp4;

assign oFlag = tempFlag;

always@(posedge iCLK or negedge iRST)

begin

if(!iRST)begin

r <= 0;

c <= 0;

temp4 <= 0;

temp6 <= 0;

temp7 <= 0;

temp8 <= 0;

temp9 <= 0;

temp10<= 0;

temp11<= 0;

tempFlag <= 0;

counter <= 0;

smallestCOLFlag <= 0;

largestCOLFlag <= 0;

smallestCOL <= 0;

largestCOL <= 0;

end

else begin

if(iDVAL == 1)begin

if(r<480)begin

if(c<640)begin

if(largestCOLFlag == 0)begin

if(r>=100 && r<=472)begin

if(c>=150 && c<=597)begin

if(r == iC\_PR)begin

if(smallestCOLFlag == 0)begin

if(iDATA == 0)begin

temp6 = r;

temp7 = c;

smallestCOL = c;

smallestCOLFlag = 1;

end

end

if(smallestCOLFlag == 1)begin

if(iDATA == 0)begin

if(c>smallestCOL)begin

temp8 = r;

temp9 = c;

smallestCOL = c;

end

end

end

end

if(r > iC\_PR)begin

largestCOLFlag = 1;

end

end

end

end

if(largestCOLFlag == 1)begin

temp10 = temp8;

temp11 = temp9;

if((temp9-iC\_PC) > (iC\_PC - temp7))begin

temp4 = temp9 - iC\_PC;

end

else begin

temp4 = iC\_PC - temp7 ;

end

tempFlag = largestCOLFlag;

end

c = c + 1;

end

if(c == 640)begin

c = 0;

r = r + 1;

end

end

if(r == 480)begin

r = 0;

c = 0;

smallestCOLFlag = 0;

largestCOLFlag = 0;

smallestCOL = 0;

tempFlag = 0;

end

end

end

end

endmodule