# Acceleration of the Presenter Detection Algorithms Using FPGA as a Coprocessor for Effective Video Recording

by

Gan Yong Xiang

14813

Dissertation submitter in partial fulfilment of

the requirements for the

Bachelor of Engineering (Hons)

Electrical and Electronics

**JANUARY 2015**

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

## CERTIFICATION OF APPROVAL

**Acceleration of the presenter detection algorithms using FPGA as a coprocessor for effective video recording**

by

Gan Yong Xiang

14813

A project dissertation submitted to the

Electrical and Electronics Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL AND ELECTRONICS)

Approved by,

______________________

(ASSOC. PROF. DR FAWNIZU AZMADI HUSSIN)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

January 2015

# CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____________________

GAN YONG XIANG

# ABSTRACT

In a video recording of a presentation session, presenter detection is required in order for the camera to have the ability to track the face of the presenter. High computation power is required to perform video processing for the detection of the presenter. Instead of using a normal embedded to perform real time video processing, Field Programmable Gate Array (FPGA) can be used as a coprocessor to perform complex computation algorithm. FGPA coprocessors can be used with standard microprocessor or microcontroller to handle complex tasks and subsequently improve the performance. The algorithm will be implemented using hardware instead of software, so that it can be optimized by developing hardware parallel processing. A method of presenter detection algorithm is developed and validated in Matlab. The validation is performing by using sample image and video of the presenter at Chancellor Hall. The execution time of the algorithm is analyzed to find the part which required high computation intensive. The computation intensive parts are implemented as a coprocessor in Altera DE2-115 platform to accelerate the operation of presenter detection. The algorithm is implemented using Verilog and programmed into FPGA. The video frame is captured by a camera module and is sent to the FPGA chip. The video frame is then processed in FPGA and the final output video is driven out to the VGA driver to be displayed on the monitor screen.

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor, Dr Fawnizu for his help in this project. He had given me financial support as well as technical guidance for allowing me to complete my projects. He had been giving me direction on what are to work on and being patience in my effort to learn. He encourages me to be independent and self-learning, which let me to lean more things and helped prepare me for future working life. However, he also tries his best to give me supports with every problem that I faced and helped me to solve the problem.

I like to extend my appreciations to En Badrulnizam for lending the Altera DE2-115 board during this study. Without the FPGA board, I will not able to complement my project successfully.

**Table of Contents**

# LIST OF FIGURE

## LIST OF TABLE

# CHAPTER 1

# INTRODUCTION

## 1.1 Project background

Video recording of a presentation or a video conference in gigantic hall would be more effective if the camera(s) have the ability to locate and track the face of the presenter. Several approaches have been done for face detection. Face detection algorithm is highly demanding on computation power. FPGA provides a good platform for achieving very high performance in complex computation by providing parallelism. In this project, the aim is to implement and optimize the face detection algorithm and more importantly to develop hardware parallel processing engines using FPGA.

## 1.2 Problem statement

During a presentation, the detection of presenter location required complex image analysis, which is computationally expensive. When implemented in an embedded system such as FPGA (Field Programmable Gate Array), the algorithm needs to be optimized. Traditional CPU or microcontrollers have limited abilities to do real time video processing and analytics at the same time. Instead CPU or microcontrollers, the presenter detection algorithms can be implemented in FPGA as a co-processor. The algorithm can be optimized by developing hardware parallel processing engines using FPGA logic and integrating with software algorithms.

## 1.3 Scope of the project

In this project, the presenter detection algorithm is implemented by using hardware description language and programmed into FPGA. FPGA is acted as coprocessor to perform presenter detection operation which required complex computation.

## 1.4 Objectives

The objectives of the project are:

- To investigate and design suitable algorithms for presenter detection during a presentation and video conference.
- To implement and validate the algorithm on an FPGA.
- To optimize the performance of the presenter detection algorithm for use in video tracking in an embedded platform.

# CHAPTER 2

# 2 LITERATURE REVIEW

In order to determine the location of the presenter, face detection is the most common method. Several techniques have been proposed to determine face location from an image. These techniques can be classified into knowledge-based methods, feature-invariant method, template matching method and appearance based method [1].However not all of these technique is able to be implemented in hardware due to its complexity. Other than that, some of the method required floating point operation. Additional hardware is need for the floating point operation that would consequence increase the cost of implementation[2]. M. Ooi [2] suggests that using Reversible component transformation (RCT) to convert RGB to modified YUV. This is due to RCT would be more easier to be implement in hardware due to it does not required floating point.

Today, skin colour segmentation method is defined as the most well liked method used for face detection. Due to the low computational requirement and low complexity of implementation, this technique is classified as an effective and successful method. Generally, the working principle of colour segmentation is by segmenting the image into skin colour and non-skin colour region and further process by categorized the skin colour into different categories. After separation of skin pixel with non-skin pixel, there is always a possibilities that a skin colour is been detect in non-face region. A future processing is needed to reduce the false positive rate as low as possible. Saini and Chand [3] presented a method for face detection using skin colour segmentation and human face feature (knowledge-based approach). In recent year, there are several colour space have been proposed. They involve RGB, HSV, YCbCr and CIE-Lab [4].

Tayal, Lamba and Padhee [5] introduced an approach for automatic face detection using colour based segmentation. They are using HSV colour spaces to perform the segmentation process. A fixed threshold value is used to extract the skin colour region. The histogram is formed by using different sample set of images. The hue component is calculated and a threshold value is fixed from the histogram. HSV model is more preferred due to HSV describe colour based on colour, vibrancy and brightness. This colour model also have the similarly to how the human tend to recognize colour.

Saini and Chand [3] suggest a method for extracting the skin colour region by using RGB and implementation a switching condition. After the skin colour is extracted, there are always noises which make the image imperfectly. In the stage of implementation, Saini and Chand fixed a threshold value to decide the switching condition. If the quality of the image is under the threshold value, the colour space of the image is changed from RGB into HSV to get better quality of image. The colour space is change if the one colour model of image is being distorted.

Tairi et al. [6] extracted skin colour using YUV and RGB colour spaces. This is to improve the segmentation process by using two thresholds. Their algorithm is started by converting RGB colour space into YUV color space. The Y channel is separated from U and V channel to eliminate the effect of luminance. The threshold value is determined by using sample image of skin colours with the help of the colour histogram.

Hu, Pai and Ruan [7] proposed colour segmentation algorithm which suitable for hardware implementation. The floating point operation is not used in the algorithm in order to reduce the computational cost. For the colour space modeling, three chromatic components of RGB colour space is used which required low computational cost.

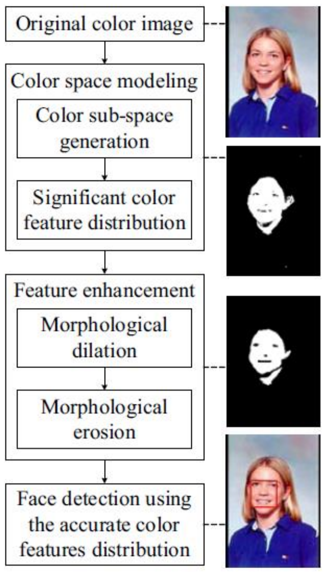

Original color image
Color space modeling
Color sub-space generation
Significant color feature distribution
Feature enhancement
Morphological dilation
Morphological erosion
Face detection using the accurate color features distribution

**Figure 1 Proposed algorithm by Hu, Pai and Ruan**

# CHAPTER 3

# 3 METHODOLOGY

The design method for this project is basically divided into 2 parts.

1. Presenter detection algorithm validate in Matlab
2. Implementation of presenter detection algorithm on the FPGA

The software based presenter detection algorithm is implemented by using Matlab. The Matlab is used to validate and to evaluate the effectiveness of the algorithm. The algorithm is then tested by using some sample image of the presenter in Chancellor hall (CH).

After the algorithm is validate using Matlab, the algorithm which required complex computation will be implemented using hardware description language (HDL). Several stage of the algorithm will be modified during the transition of Matlab to Verilog. This is due to several stage is infeasible to be implemented in Verilog.

After implementing the algorithm in FPGA, the Verilog code will be analysis and synthesis by using Quartus II. A testbench is also created to validate the algorithm on FPGA by using Model Sim. The validate algorithm is then been convert into SRAM Object File (SOF) and programmed to the Cyclone IV in Altera DE2-115 platform.

## 3.1 Presenter detection algorithm

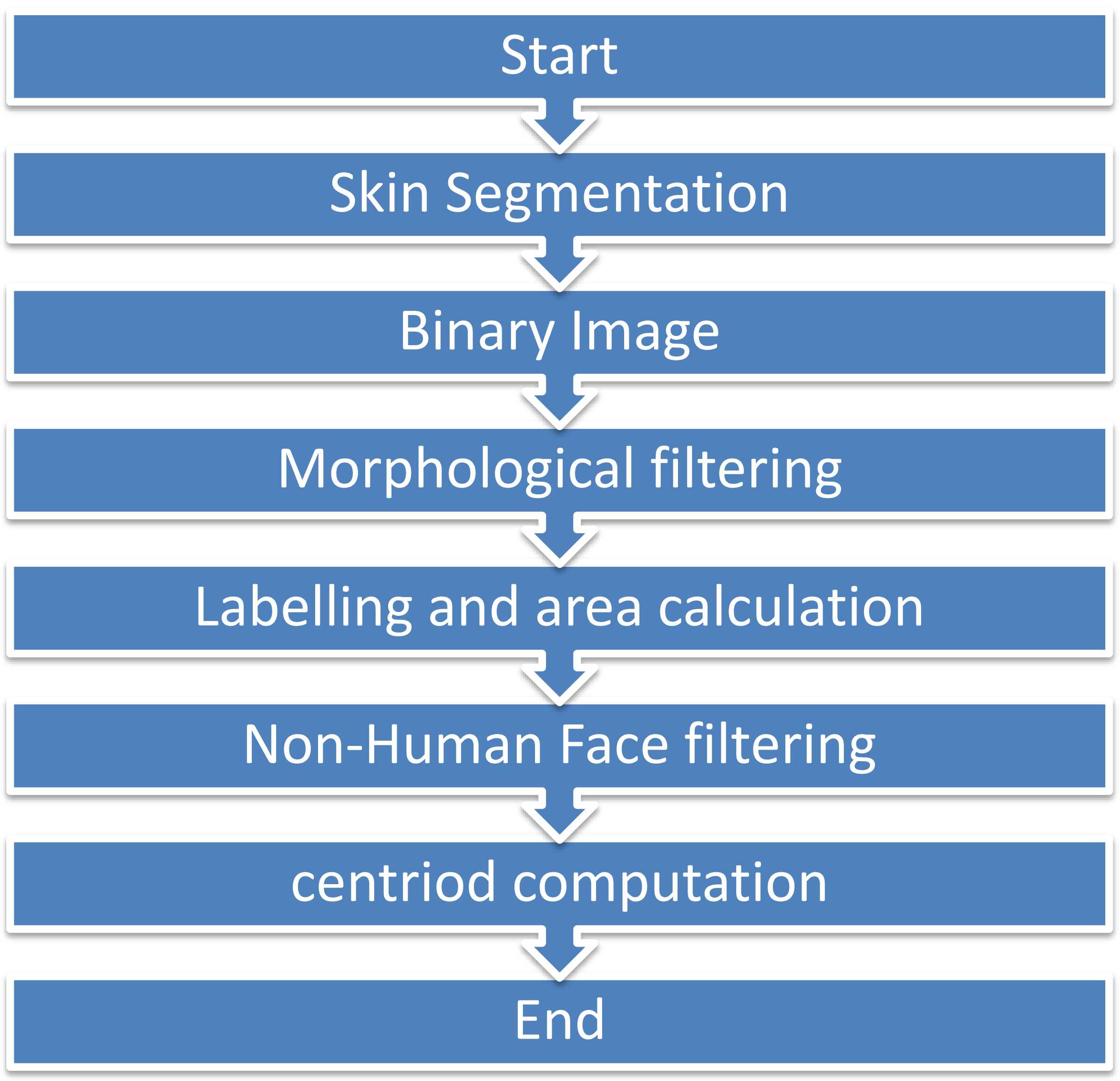


**Figure 2 Presenter Detection Algorithm**

Skin Colour detection is the robust and effective method to detect presenter face in colour image or video. There are several colour models can use for skin colour segmentation such as RGB, HSV and YUV. In this step, the skin like background is not rejected, future processing is needed to remove the background.

Morphological filtering is used to remove imperfections of binary image. Structuring element is used to probe an image with a small shape. Firstly, erosion is used to remove the background noise, secondly, dilation is used to adds a layer of pixel to the face detected region.

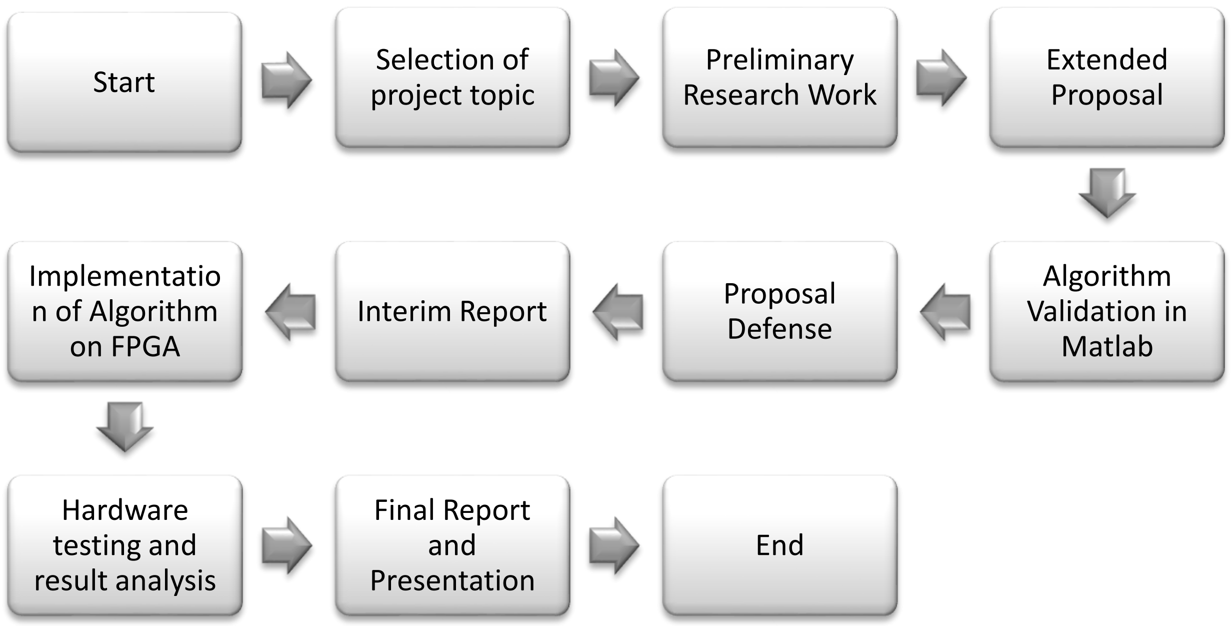
After that, the connected region in the binary image is labeled. The area of the connected region is calculated. At this point, the binary image is contained skin colour region, however not all the region is the face region. Therefore some filtering is needed to remove non-human face region by using human face properties. There are small area filtering, eccentricity filtering and height to width ratio filtering. The smaller area is removed due to face region have larger area compared to other skin area. Next, the shape of face is an oval shape, the eccentricity properties is used to remove the region whose shape likely to line. Lastly, bounding box properties is used to calculate height to width ratio. The region which does not satisfy the maximum threshold value is rejected.

The final stage of the algorithm is centroid computation. The average length of X coordinates and Y Coordinate is calculated to find the centroid of the face.  The centroid point of presenter face is marked.

Tool Used:

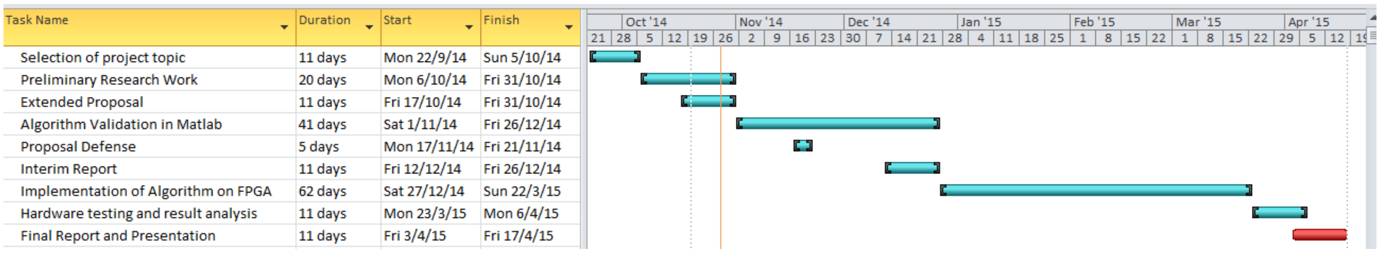
1. DE2-115 development board with digital camera package
2. Matlab
3. Quartus II
4. Model Sim

## 3.2 Project Flow



Start
Selection of project topic
Preliminary Research Work
Extended Proposal
Algorithm Validation in Matlab
Proposal Defense
Interim Report
Implementatio n of Algorithm on FPGA
Hardware testing and result analysis
Final Report and Presentation
End

**Figure 3 Flow Chart**



| Task Name | Duration | Start | Finish |
|---|---|---|---|
| Selection of project topic | 11 days | Mon 22/9/14 | Sun 5/10/14 |
| Preliminary Research Work | 20 days | Mon 6/10/14 | Fri 31/10/14 |
| Extended Proposal | 11 days | Fri 17/10/14 | Fri 31/10/14 |
| Algorithm Validation in Matlab | 41 days | Sat 1/11/14 | Fri 26/12/14 |
| Proposal Defense | 5 days | Mon 17/11/14 | Fri 21/11/14 |
| Interim Report | 11 days | Fri 12/12/14 | Fri 26/12/14 |
| Implementation of Algorithm on FPGA | 62 days | Sat 27/12/14 | Sun 22/3/15 |
| Hardware testing and result analysis | 11 days | Mon 23/3/15 | Mon 6/4/15 |
| Final Report and Presentation | 11 days | Fri 3/4/15 | Fri 17/4/15 |

**Figure 4 Gantt Chart**

## 3.3 The reason of using Altera DE2-115 to implement the project.

Altera DE2-115 is been choose as the platform to implement the project.  5 Mega Pixel Digital Camera Package (D5M) is connected to the 40-pin expansion header of the platform. D5M is used to capture the video and sent to the FPGA Chip. A monitor screen is connected to the VGA port and the final output is displayed on the monitor screen.

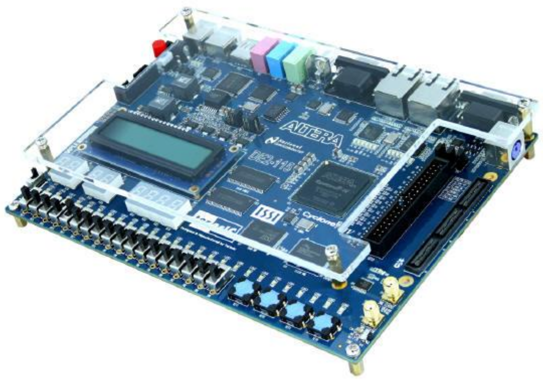


**Figure 5 DE2-115 Development and Education Board**

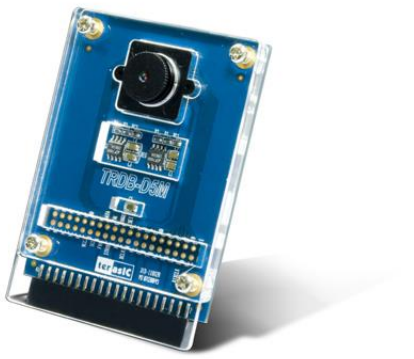


**Figure 6 5 Mega Pixel Digital Camera Package**

Altera platform is chosen in this project because the performance of FPGA is better than CPU for performing complex computation task.  FPGA is a programmable chip and can be configured as a parallel processing device, which mean that, it can perform several task at single clock cycle. Single core CPU executed one task at single clock cycle because it is a sequential processing device.

DE2-115 board is selected as the platform. Cyclone EP4C115 FPGA feature more logic elements, embedded memory and also embedded multiplier. Other than that, DE2-115 platform is consist larger storage of memory device

## 3.4 Expected Performance

At the Matlab stage, a couple of video and image is used to validate the presenter detection algorithm.  If the algorithm is validated, the face position of the presenter will be shown and marked.

At the hardware stage, a camera is used as the input and the monitor screen is used as the output. The camera and the monitor screen will be connected to the FPGA platform. The camera will faced to the presenter face and the face position will be shown in the monitor screen.

# CHAPTER 4

# 4 VALIDATION OF ALGORITHM IN MATLAB

The presenter detection algorithm is validated by using Matlab. The presenter detection algorithm which was described in the methodology part has been proven a robust and effective method. This is due to colour segmentation only requires low computational power requirement.

The modified YUV colour space is chosen to perform colour segmentation. In YUV colour space,it separates it Y channel(intensity) from the U-V channel (chrominance components). YUV colour space would be more effective if compared with RGB colour space, this is due to human skin colour is fall within a certain range of UV component. This means that the brightness of the image will not affect the segmentation process. However the transformation from RGB to YUV colour space required floating point arithmetic. The following equation shows the conversion from RGB to YUV

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = -0.147R - 0.289G + 0.436B$$

$$V = 0.615R - 0.515G - 0.100B$$

In order to avoid the use of floating point arithmetic, reversible component transformation(RCT) is used[2] . The RCT only required integer arithmetic, therefore it is more robust to convert RGB to YUV using RCT. The following equation shows the conversion from RGB to YUV using RCT.

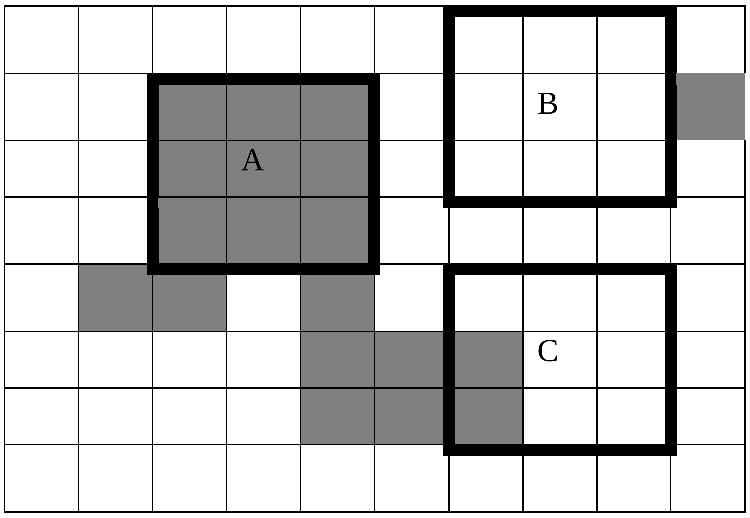
$$Y = \frac{R + 2G + B}{4}$$

$$U = R - G$$

$$V = B - G$$

The input image will be convert to modified YUV colour space using RCT . After the conversion is completed, the skin colour is segmented by using the threshold value of skin-pixel in chrominance range. The skin pixel is fall within the $10 < u < 74$ and $-40 < V < 11$. However, the output of the skin segmentation is not perfect, there might be some noise and skin-colour background is not able to remove in this stage. Therefore, future processing is needed to totally removed these noise. At this state, the detect skin colour region will be converted to white pixel and the non-skin colour region will be converted to black pixel.

After skin-segmentation state is completed. Morphological filtering is used to remove imperfections of binary image. Matlab function imerobe and imfill is used in these state. imerobe (erosion )is used to remove the background noise, secondly, imfill (dilation) is used to adds a layer of pixel to the face detected region. Structuring element is used to probe an image with a small shape.



A- Structuring element fits the image

B- Structuring element neither fits nor hits the image

C- Structuring element hits the image

The 3x3 square is used as structuring element in erosion operation. This erosion operation is able to remove any size of pixel which smaller than the structure elements. Small noise is able to be removed in this operation. Dilation operation is then used to enlarge the boundary of the white pixel(skin-colour). By using this operation, the missing hole inside the face region is filled in. This allows the detected face region to become one connected region.

The next operation is the labeling of connected region and calculates the number of connected region. Bwlabel build in function in Matlab is used. This function will return the number of connected region and also labeling the connected. The output matric of bwlabel will contain the label for all connected region.

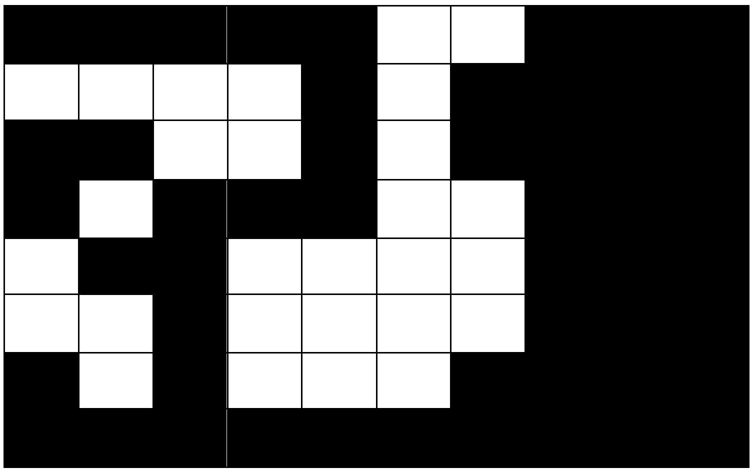


**Figure 7 Input image of bwlabel**

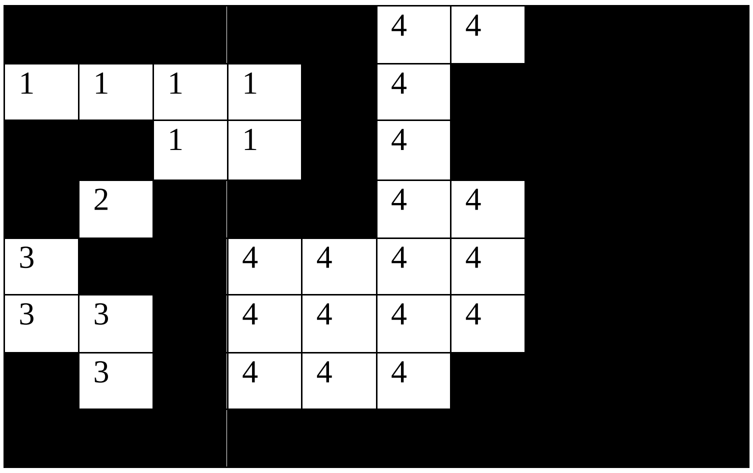


**Figure 8 output image of bwlabel**

After all of the connected region has been labeled. Regionprops function in Matlab is used to measure the properties of the image, including the area, eccentricity, the height as well as the width.

At this stage, the image is filtered using human face feature and the properties of the human face. The first filtering operation is small area filtering. Although some noise is removed during morphological filtering, the small area filtering is still needed. This is due to morphological only able to remove some of the noises but not all. The entire connected skin region is calculated and the maximum area is determined. All the connected area will be compare with the maximum area, only the area larger than 26% of maximum area is determined as the face region. The area less than 26% will be removed.

After the rejection of small area, eccentricity filtering is applied. Eccentricity is defined as how much a conic section varies from being circle. The circle shape is obtained if the eccentricity is equal to zero, whereas the line is obtained for the infinite eccentricity. Based on the human face properties, the shape of the human face is almost same with oval shape. The oval shape is the area whose have the eccentricity greater than 0.89905[8].

After passing above filtering, this stage is filtering based on the height and width of the region. The height and width of the region is obtained by the bounding box properties in Matlab. For the human face region, the height of the area must be greater the width of the area. In order to make sure the height is greater than the width, $\frac{width}{hight} < 1$. By using this properties, if the area of $\frac{width}{hight} > 1$, the area will be removed.

The final stage is to determine the centroid location of human face and light blue asterisk is marked at the center location. In order to find the center location, the average value of the length of width and height of the areas is calculated.

# CHAPTER 5

# 5 IMPLEMENTATION OF PRESENTER DETECTION ALGORITHM ON THE FPGA

## 5.1 Hardware setup

From the run time result, it is found that, Skin Segmentation takes the longest time to perform it job. Therefore skin segmentation is then to be implemented in hardware to accelerate the face detection process. The current hardware algorithm been implemented is shown as below.

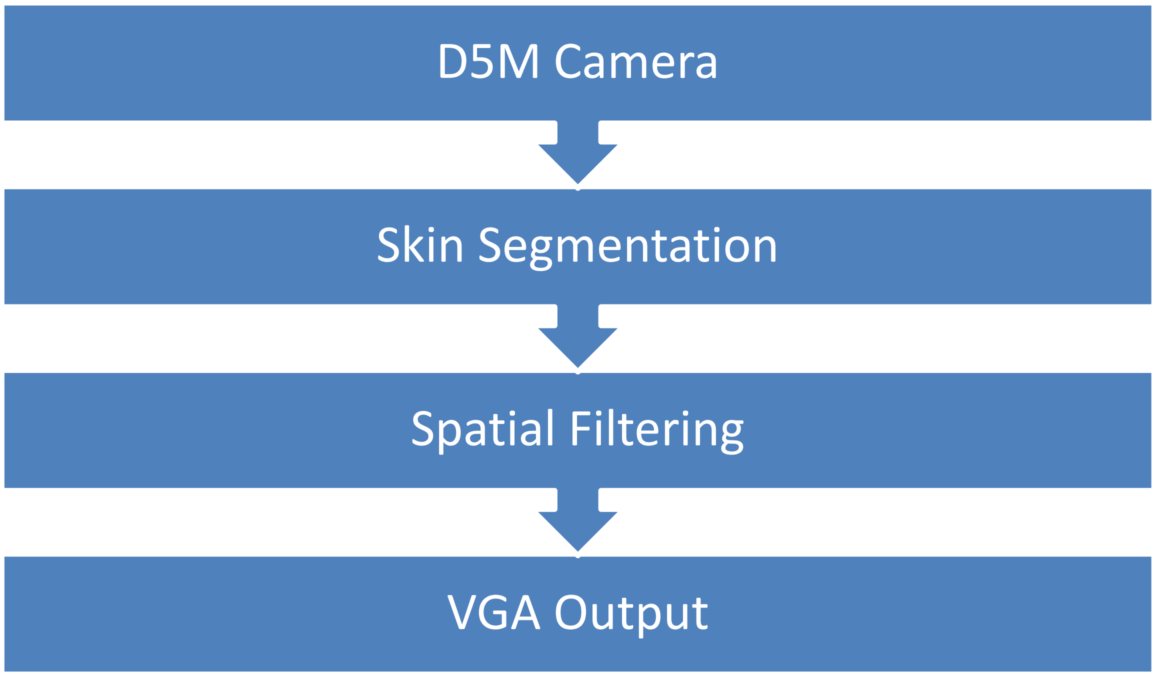


**Figure 9 Presenter detection algorithm on the FPGA**

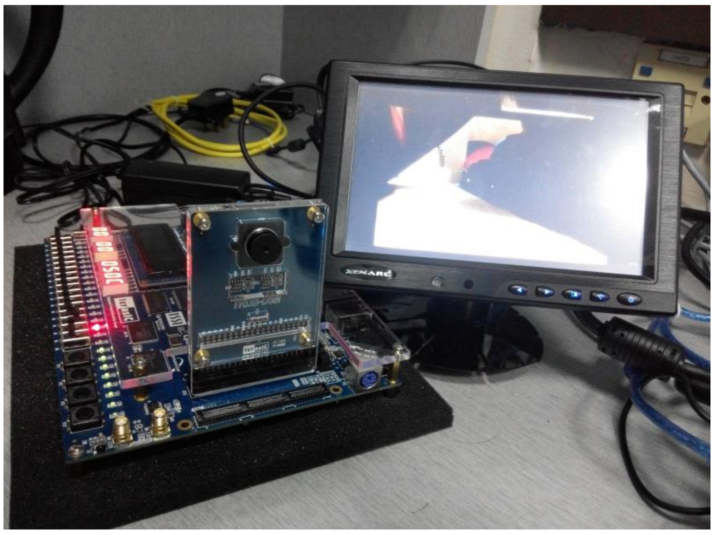


**Figure 10 Hardware Setup**

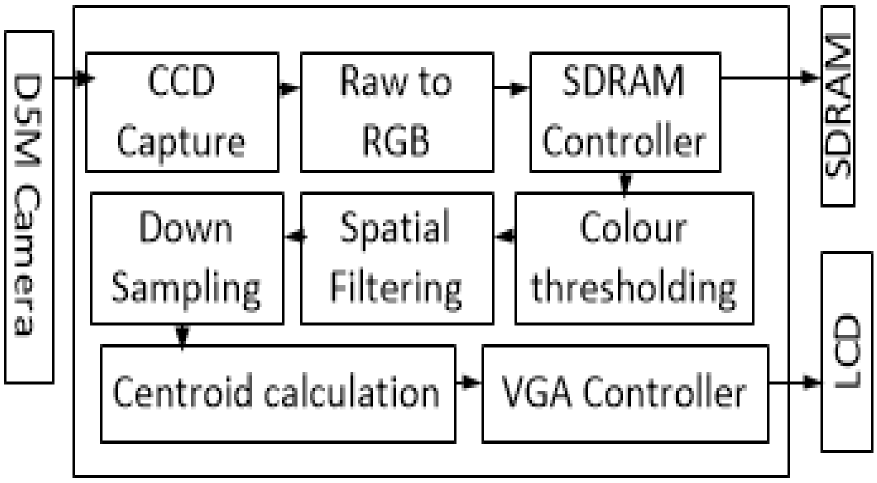
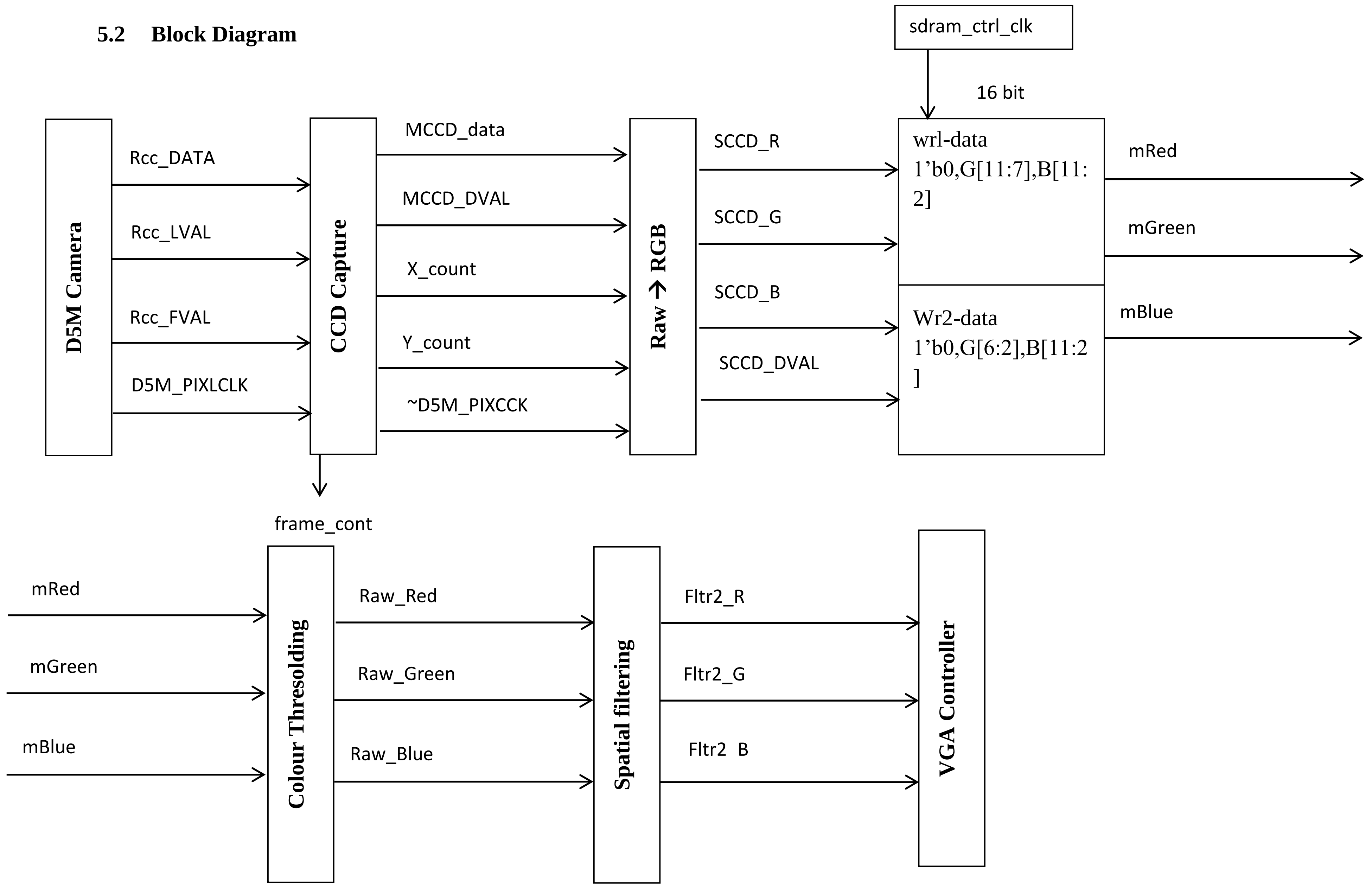


**Figure 11 System Design on FPGA**

## 5.2 Block Diagram



## 5.3 CCD capture

Charge Coupled Device (CCD) is a device which is used for capturing the movement of electrical charge and convert into a binary value. D5M used CCD image sensor to capture the data and this data is received via CCD capture block at the FPGA. CCD block captures the image data pixel by pixel; therefore in order to read 800x600 resolution of an image, 480k clock cycle is needed. However the active image read out by the D5M camera is bordered by horizontal and vertical blanking.

| | |
|---|---|
| Valid Active Image | Horizontal Blanking |
| Vertical Blanking | Vertical/Horizontal Blanking |

**Table 1 Image Readout by D5M Camera**

The output timing diagram needs to be observed to find out the active image region.

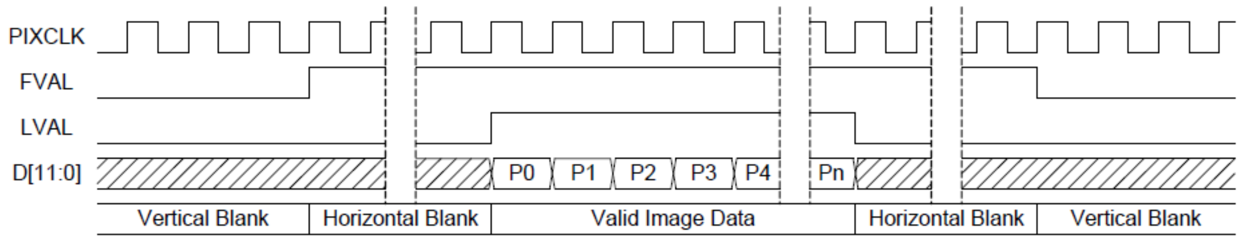


**Figure 12 Output Timing Diagram of D5M camera**

FVAL and LVAL are observed, during valid data image, the FVAL and LVAL are both high. When the FVAL remain high and LVAL become low, horizontal blank is occurred. When FVAL and LVAL are low, it is called horizontal blanking.

## 5.4 RAW to RGB

D5M camera used Bayer pattern to display out it pixels. Bayer pattern format consists of four colors (one red, one blue and two green). There are two green colour due to our human eye is more sensitive to green colour compare with red or blue. Raw to RGB module will separate RAW data to red, green and blue data.

|  |  |
|---|---|
| G1 | R |
| B | G2 |

**Table 2 Bayer Pattern**

## 5.5 SDRAM controller (Double buffering)

The data from the RAW to RGB is transfer to external SDRAM memory through the SDRAM controller. When the VGA sends a read signal to read the data from SDRAM, the data will pass through colour threshold block and spatial filtering block. The FIFO is required due to the Camera Pixel Clock does not synchronize with the VGA clocks. Other than then, the image processing can been optimise by using Double buffering, the filtering block can start to process once one pixel is sent to the SDRAM. Without the Double buffering,  the image only can perform job once one frame has been send to the SDRAM, the image processing job does not able to perform during the data transfer from the camera to SDRAM due to both read and write operation cannot perform at the same time.

## 5.6 Colour thresholding

Due to the normal conversion of RGB to YUB color space required floating point operation, the RGB is convert to modified YUV colour space using reversible component transformation (RCT).

total=mRed+mGreen+mBlue;

YUV_Y=total[11:2]; (used shifts operation to perform division, shift right by 2 = divide by 4)

YUV_U=mRed-mGreen;

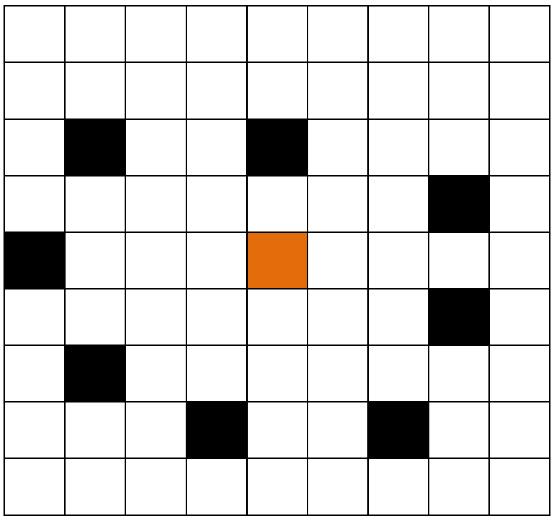
YUV_V=mBlue-mGreen;

Due to 10 bit colour was used in hardware implementation, the U range value is modified to $40<U<296$.

During colour thresolding operation, comparator is used to check for the skin pixel, if the data are located in the range of skin pixel, white pixel is sent out to the VGA, and otherwise a black pixel is sent out if the data are not located in the range of skin pixel. After this operation the video frame was converted to the binary image.

## 5.7 Spatial filtering

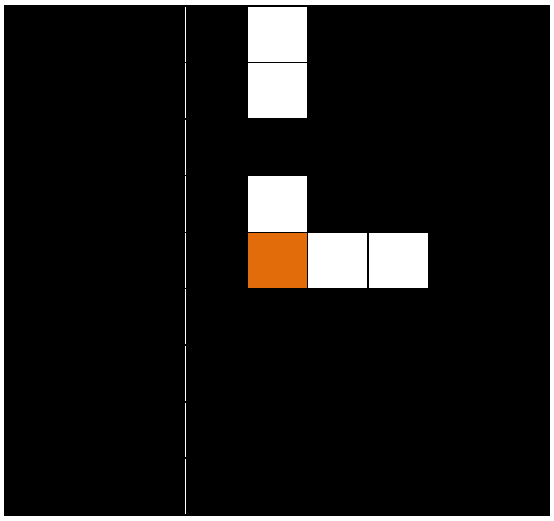
Spatial filtering is one of the image processing operations to check for every pixel and the pixel value is depending by the intensities of neighbourhood pixels. This step was similar to the erosion step which has been validated in Matlab. This operation is used to remove the background noise. In the operation, all pixels are checked with its neighbouring pixels in a 9x9 pixel. The pixel is defined as skin pixel if there are more than 92 % of its neighbours are skin pixel, otherwise it will consider as non-skin pixel.



| Colour | Pixel |
|---|---|
| 🟧 | Skin Pixel to be checked |
| ⬛ | Neighbouring skin pixel |
| ⬜ | Neighbouring non-skin pixel |

$$\text{Percentage of skin pixel} = \frac{9}{81} \times 100\% = 11.11\%$$

It is not a skin pixel.



| Colour | Pixel |
|---|---|
| 🟧 | Skin Pixel to be checked |
| ⬛ | Neighbouring skin pixel |
| ⬜ | Neighbouring non-skin pixel |

$$\text{Percentage of skin pixel} = \frac{76}{81} \times 100\% = 93.82\%$$

It is a skin pixel.

## 5.8 Downsampling

In this block, the 800x600 image is partition into 16x16 blocks, therefore each image will consist of 50x38 of blocks to be labelled as skin and non-skin . 90% of threshold value of skin pixel of each block is set to be considered as skin pixel. This block is used to decrease the size of image and hence reduce the effect of noise. For the final stage, the centroid is calculated based on the average of x axis and y axis.

# CHAPTER 6

## 6 PROPOSED DESIGN FOR OPTIMIZATION

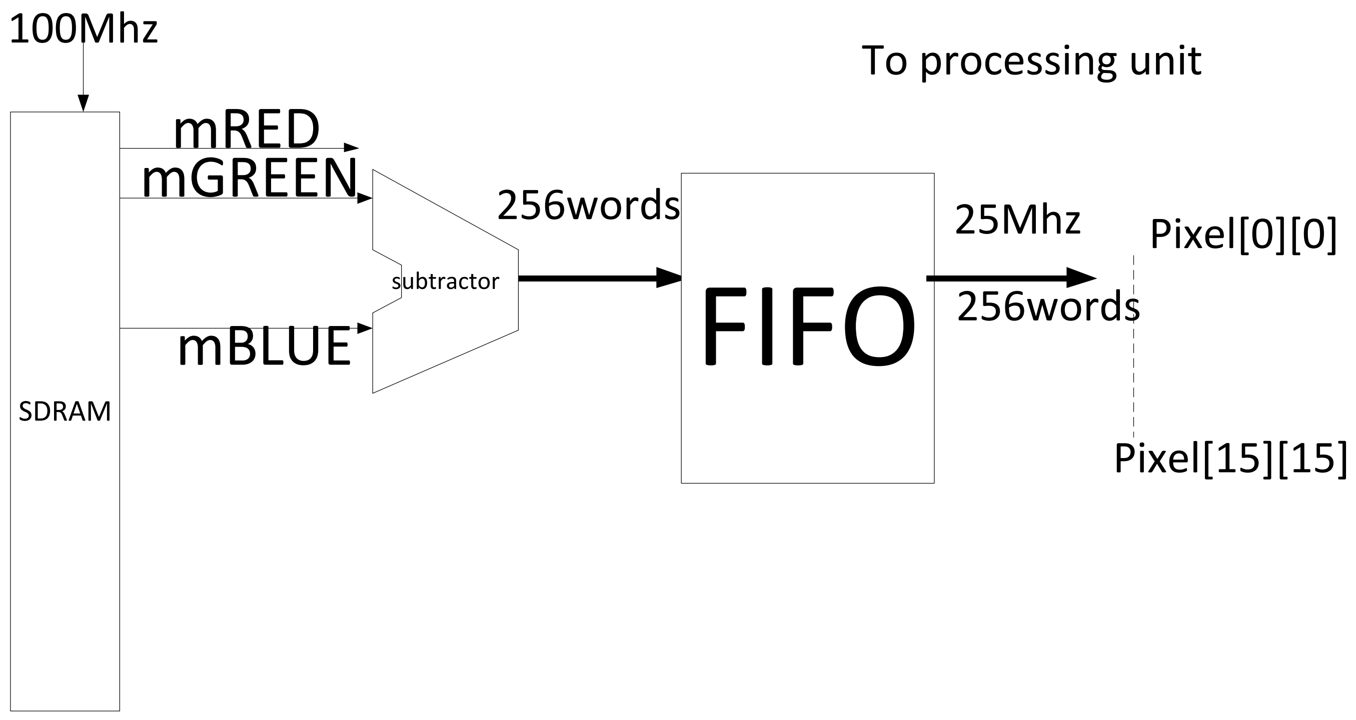


**Figure 13 interfacing between FIFO and SDRAM**

As shown in figure 12, there are two different clock is needed to interface two system. The SDRAM used a higher frequency that the processing unit. A clock domain crossing (CDC) is occurred due to different clock source are used for two system. In order to solve for CDC problem, asynchronous FIFO can be used to read the data and write the data using two different clock.

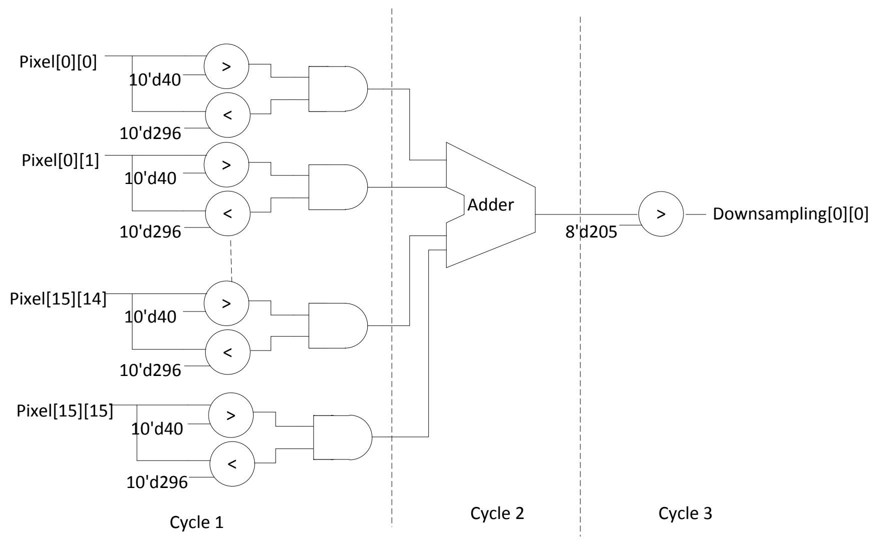


**Figure 14 Colour Threshold Unit**

The processing unit for face detection is represented by the colour threshold unit and down sampling binary image builder. Modified YUV is used for colour thresholding and the thresholding value is 10<u<74 [2]. 10 bit of U range is used, therefore, the thresholding value is modified to 40<U<296. Three state of pipelining state is implemented. The first state is to extract the skin colour pixel. The second state is to count the number of skin pixel. The last state is to compare the total number of skin pixel with a certain value.

For 800x600 image, there will be 50x38 of 16x16 pixel. Therefore the clock cycle needed to process one frame of 800x600 is 50x38=1900 clock cycle. This circuit has optimised the required clock cycles from 480,000 to 1900 clock cycle.

# CHAPTER 7

# 7 RESULTS AND DISCUSSION

## 7.1 Mathlab

The presenter detection algorithm is tested by using the following image

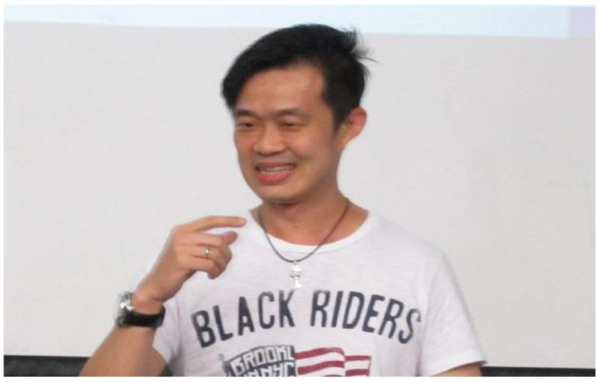


**Figure 15 input image of sample 1**

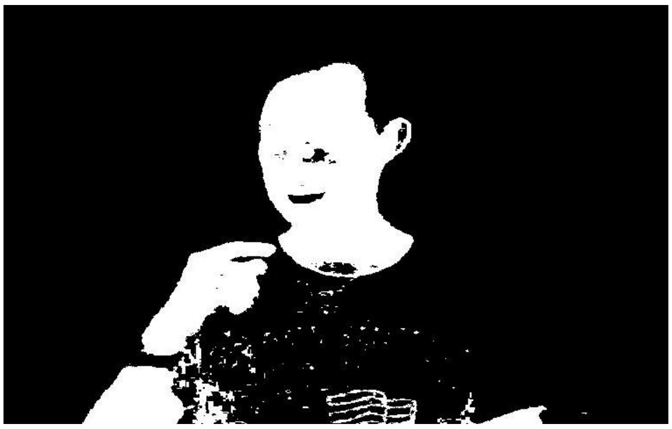


**Figure 16 colour segmentation result of sample 1**

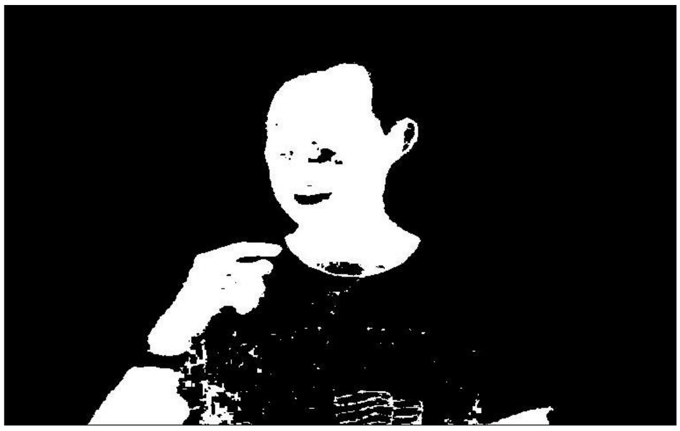


**Figure 17 eroded result of sample 1**

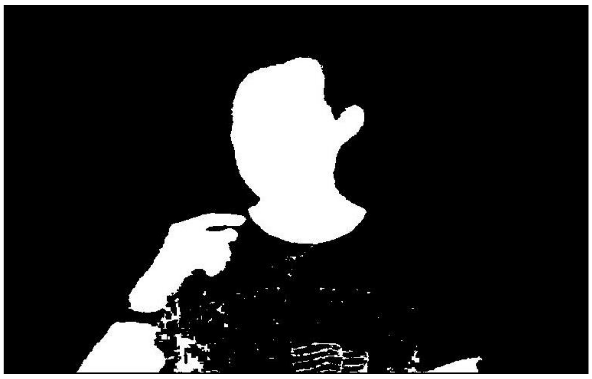


**Figure 18 dilation result of sample 1**

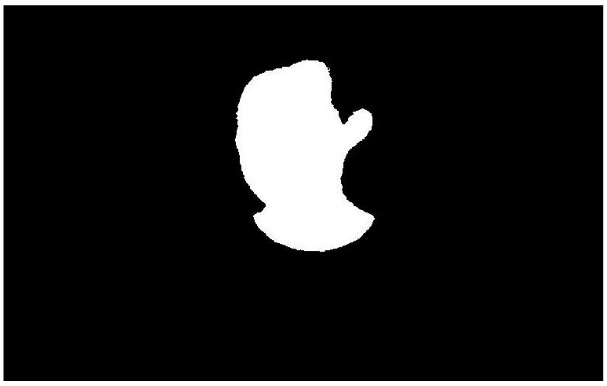


**Figure 19 Filtering result of sample 1**

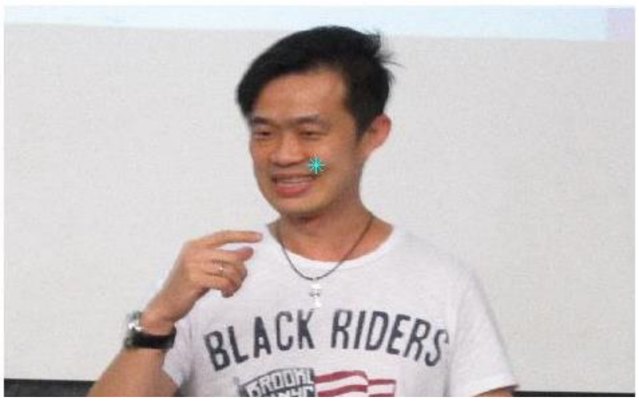


**Figure 20 Detection result of sample 1**

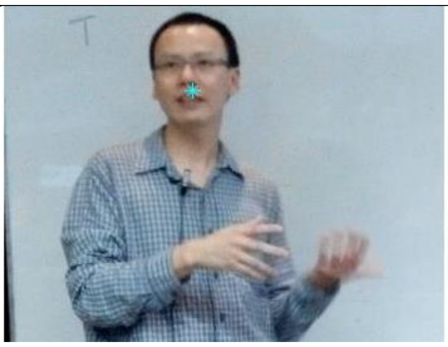
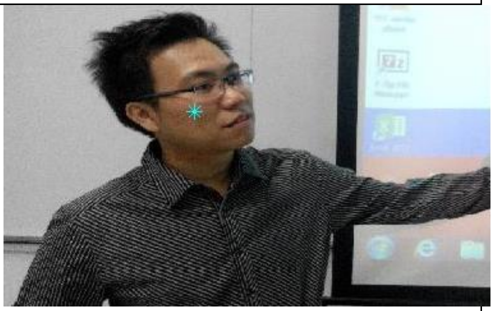
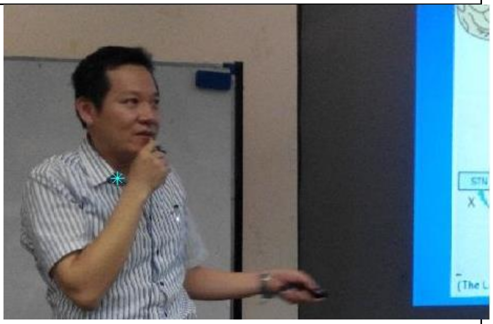
| Sample | Input Image | Detection Result |
|---|---|---|
| 2 | |  |
| 3 | |  |
| 4 | |  |

**Table 3 Matlab Result**

| Number of image | Face detected | False Face detection |
|---|---|---|
| 4 | 75% | 25% |

**Table 4 face detection result**

The run time is based on the sample image 1(1448x2296). The time is obtained by using *tic* and *toc* function of Matlab.

| Operation | Run Time(seconds) |
|---|---|
| Skin Segmentation | 3.078961 |
| Morphological filtering | 0.457322 |
| Area Labelling | 0.025064 |
| Non-Human face filtering | 0.444274 |
| Centroid computation | 0.076275 |

**Table 5 Executed time of the algorithm**

## 7.2 Hardware implementation

The current result was able to separate the skin colour with non-skin colour. Other than that, the system also able to remove the back ground noise by spatial filtering.

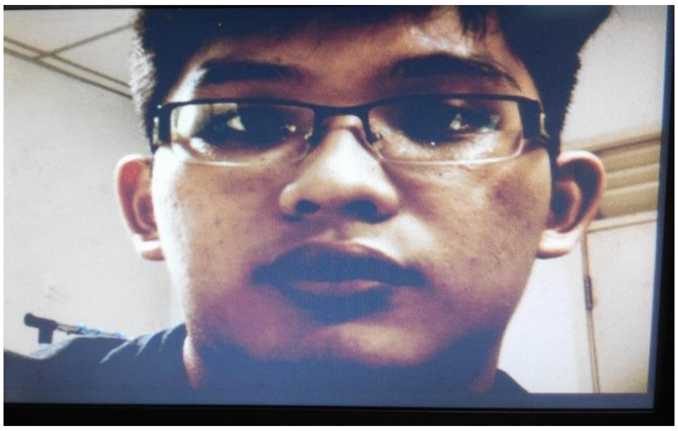


**Figure 21 Input Image without any filtering**

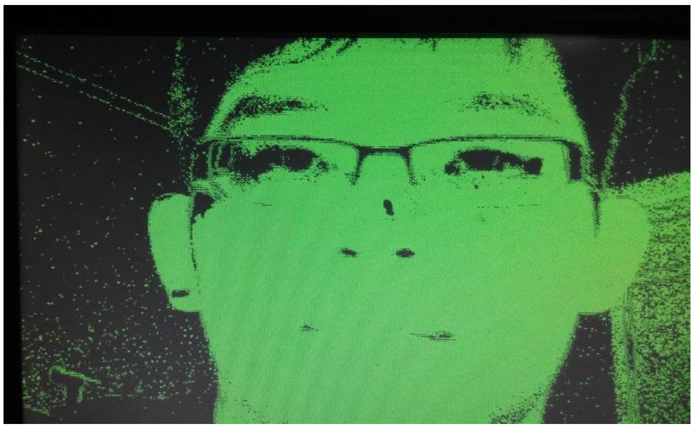


**Figure 22 Skin segmentation**

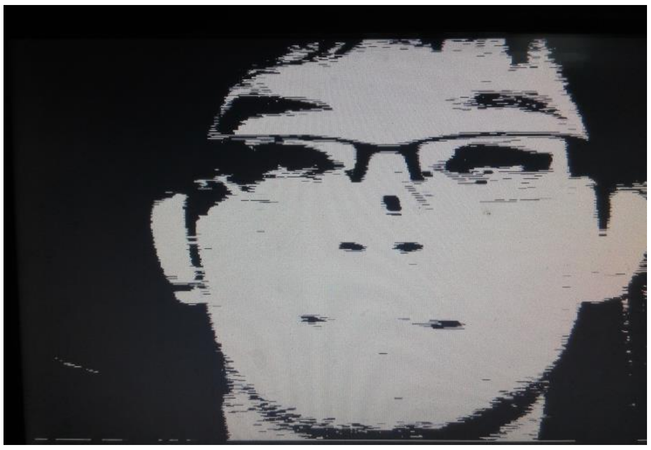


**Figure 23 Spatial Filtering**

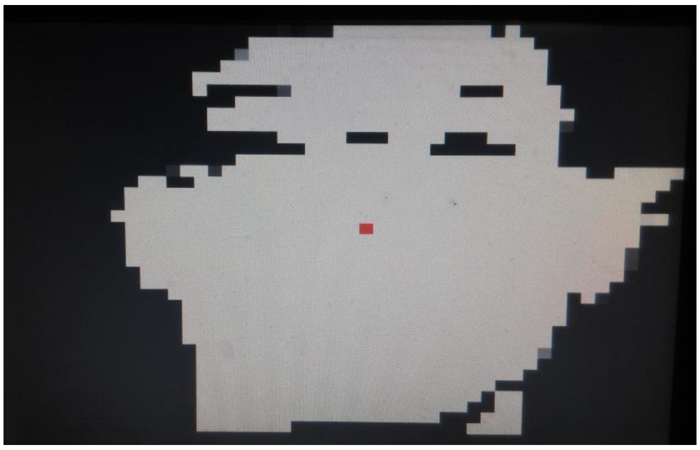
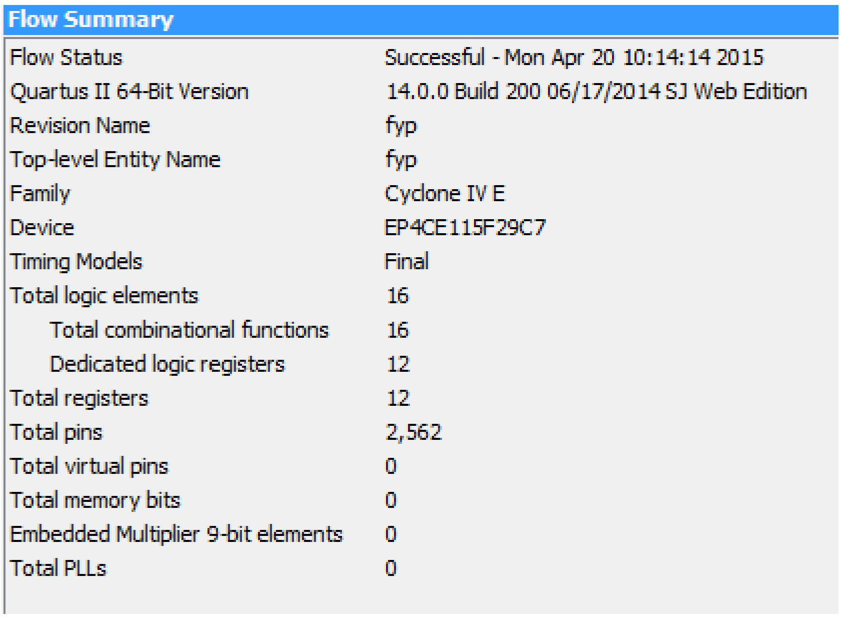


**Figure 24 Down sampling and Centroid calculation**

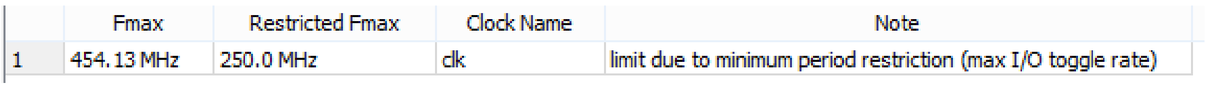
## 7.3 Proposed circuit Result



**Flow Summary**

| | |
|---|---|
| Flow Status | Successful - Mon Apr 20 10:14:14 2015 |
| Quartus II 64-Bit Version | 14.0.0 Build 200 06/17/2014 SJ Web Edition |
| Revision Name | fyp |
| Top-level Entity Name | fyp |
| Family | Cyclone IV E |
| Device | EP4CE115F29C7 |
| Timing Models | Final |
| Total logic elements | 16 |
| Total combinational functions | 16 |
| Dedicated logic registers | 12 |
| Total registers | 12 |
| Total pins | 2,562 |
| Total virtual pins | 0 |
| Total memory bits | 0 |
| Embedded Multiplier 9-bit elements | 0 |
| Total PLLs | 0 |

**Figure 25 Analysis & Synthesis report**



| | Fmax | Restricted Fmax | Clock Name | Note |
|---|---|---|---|---|
| 1 | 454.13 MHz | 250.0 MHz | clk | limit due to minimum period restriction (max I/O toggle rate) |

**Figure 26 Fmax Summary**

# CHAPTER 8

# 8 CONCLUSION AND RECOMMENDATION

## 8.1 Conclusion

The beneficial of this project will obviously toward video recording of presenter in presenter hall. The video recording becomes more efficient as the camera having the ability to track the face of the presenter. The algorithm which required complex computation is implemented in hardware instead of software. The hardware implementation provides a good platform to perform real time video processing.

Other than that, the cost of implementation in FPGA is relatively low. Hardware/software co-design also can be developed in FPGA. FPGA can be connected to CPU or microcontroller as a co-processor. The use of FPGA also is a way to shorten the development time. A Application-Specific Integrated Circuits (ASICs) can take more than one years to be implemented.

## 8.2 Recommendation

For future work, a motor can be connected to the FPGA to move the camera based on the location of the presenter. The speed of the system will be good as FPGA provide a good platform to perform real time task. Other than that, the algorithm can be improved to increase the accuracy and reduce the false positive rate.

# REFERENCES

[1] M.-H. Yang, N. Ahuja, and D. Kriegman, "A survey on face detection methods," 1999.

[2] M. Ooi, "Hardware implementation for face detection on Xilinx Virtex-II FPGA using the reversible component transformation colour space," in *Electronic Design, Test and Applications, 2006. DELTA 2006. Third IEEE International Workshop on*, 2006, pp. 6 pp.-46.

[3] H. K. Saini and O. Ch, "Skin Segmentation Using RGB Color Model and Implementation of Switching Conditions," *Skin,* vol. 3, pp. 1781-1787, 2013.

[4] S. L. Phung, A. Bouzerdoum, and D. Chai Sr, "Skin segmentation using color pixel classification: analysis and comparison," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 148-154, 2005.

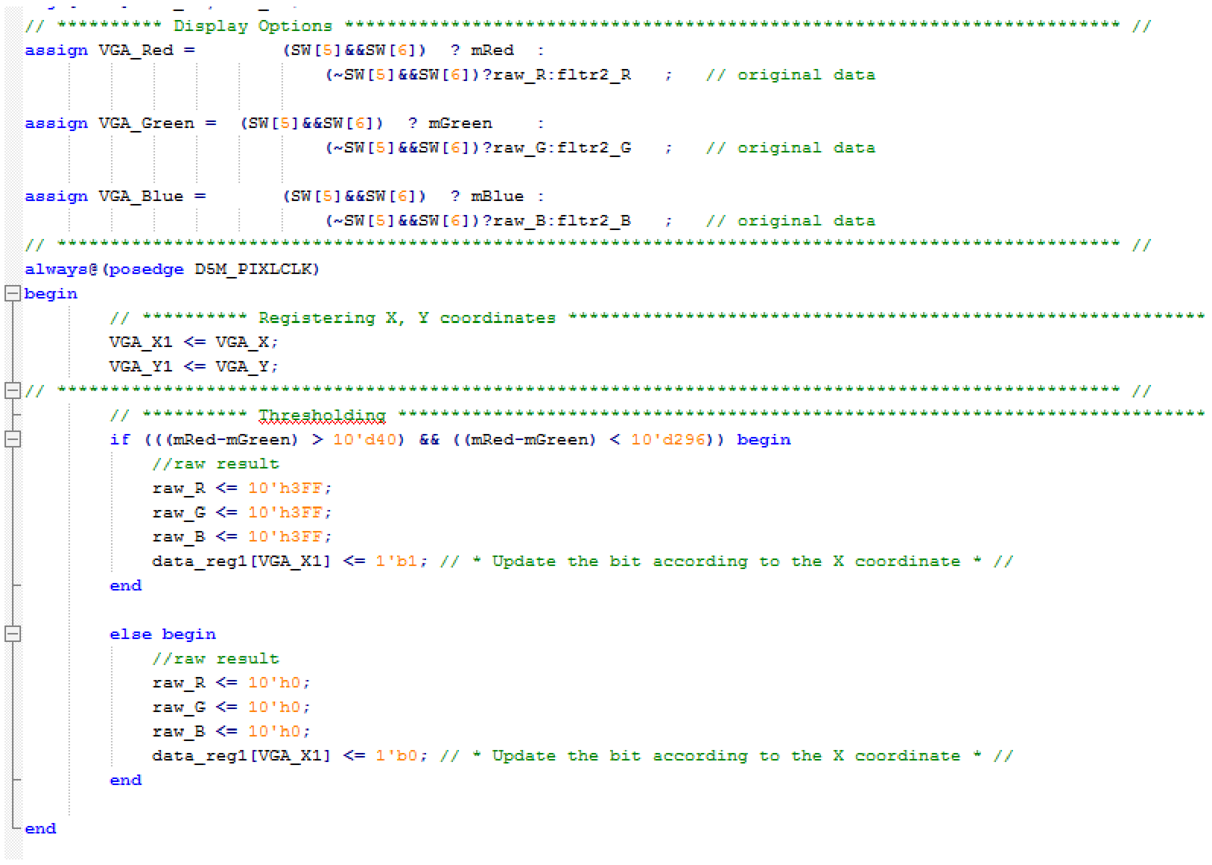
[5] Y. Tayal, R. Lamba, and S. Padhee, "Automatic face detection using color based segmentation," *International Journal of Scientific and Research Publications,* vol. 2, 2012.

[6] Z. H. Al Tairi and M. I. Saripan, "Skin segmentation using YUV and RGB color spaces," *Journal of information processing systems,* vol. 10, pp. 283-299, 2014.

[7] K.-T. Hu, Y.-T. Pai, S.-J. Ruan, and E. Naroska, "A hardware-efficient color segmentation algorithm for face detection," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*, 2010, pp. 688-691.

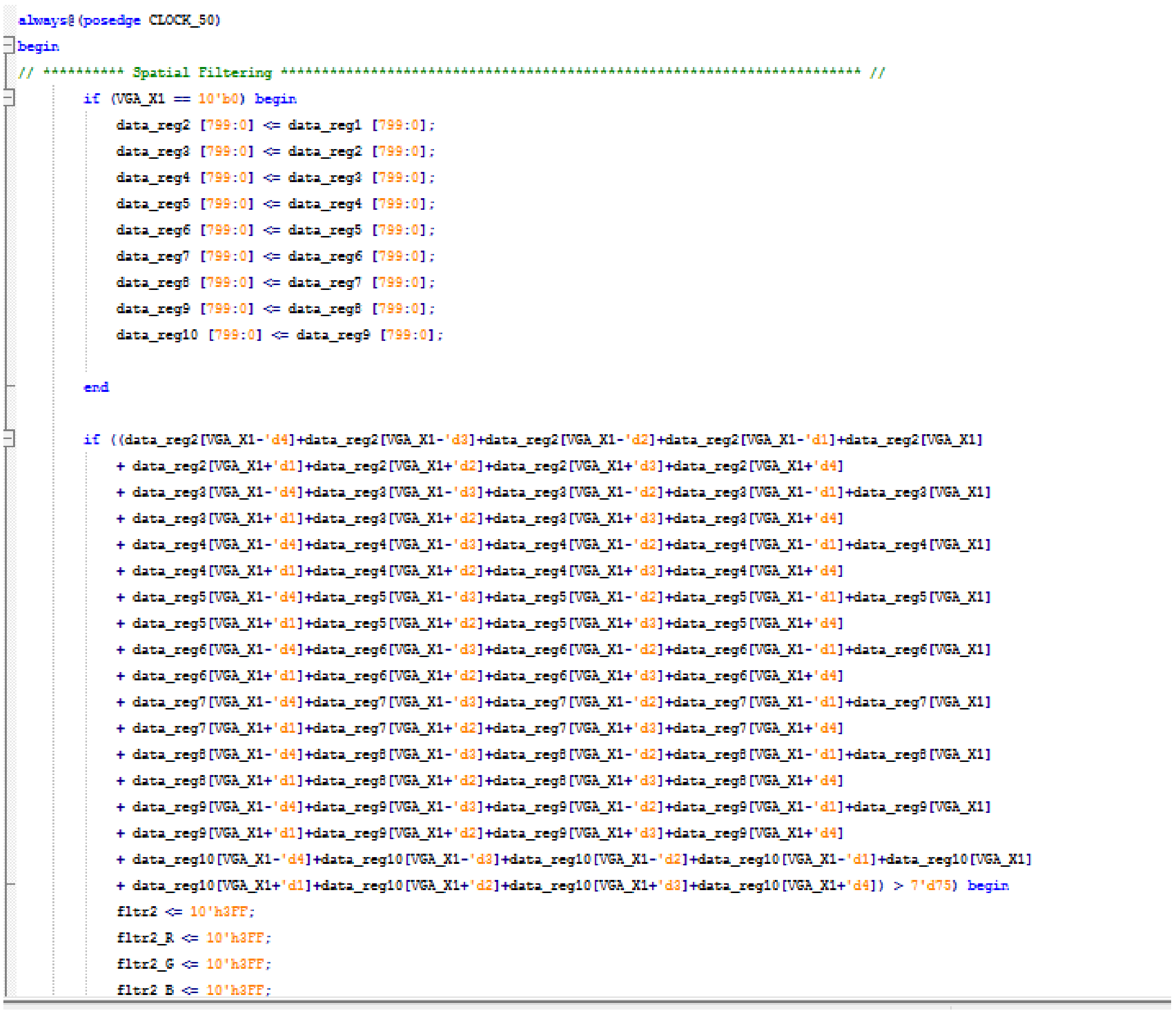
[8] D.S.Raghuvanshi and D.Agrawal. "Human Face Detection by Using Skin Color Segmentation, Face Features and Regions Properties". *International Journal of Computer Applications* 38(9):14-17, January 2012.

# APPENDICES



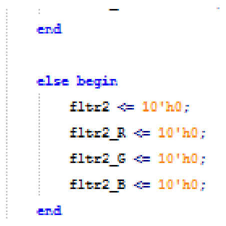
```verilog
// ********** Display Options ************************************************************************** //
assign VGA_Red =        (SW[5]&&SW[6])  ? mRed   :
                            (~SW[5]&&SW[6])?raw_R:fltr2_R   ;   // original data

assign VGA_Green =  (SW[5]&&SW[6])  ? mGreen    :
                            (~SW[5]&&SW[6])?raw_G:fltr2_G   ;   // original data

assign VGA_Blue =       (SW[5]&&SW[6])  ? mBlue :
                            (~SW[5]&&SW[6])?raw_B:fltr2_B   ;   // original data
// ************************************************************************************************** //
always@(posedge D5M_PIXLCLK)
begin
        // ********** Registering X, Y coordinates ***********************************************************
        VGA_X1 <= VGA_X;
        VGA_Y1 <= VGA_Y;
// ************************************************************************************************** //
        // ********** Thresholding ***********************************************************************
        if (((mRed-mGreen) > 10'd40) && ((mRed-mGreen) < 10'd296)) begin
            //raw result
            raw_R <= 10'h3FF;
            raw_G <= 10'h3FF;
            raw_B <= 10'h3FF;
            data_reg1[VGA_X1] <= 1'b1; // * Update the bit according to the X coordinate * //
        end

        else begin
            //raw result
            raw_R <= 10'h0;
            raw_G <= 10'h0;
            raw_B <= 10'h0;
            data_reg1[VGA_X1] <= 1'b0; // * Update the bit according to the X coordinate * //
        end

end
```

**Figure 27 Colour thresolding Verilog code**



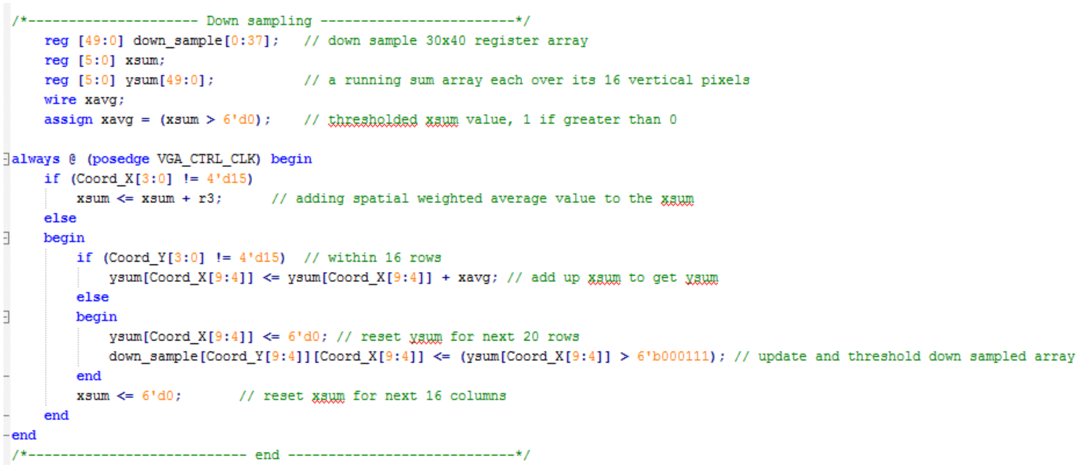
```
always@(posedge CLOCK_50)
begin
// ********** Spatial Filtering *************************************************************************** //
        if (VGA_X1 == 10'b0) begin
            data_reg2 [799:0] <= data_reg1 [799:0];
            data_reg3 [799:0] <= data_reg2 [799:0];
            data_reg4 [799:0] <= data_reg3 [799:0];
            data_reg5 [799:0] <= data_reg4 [799:0];
            data_reg6 [799:0] <= data_reg5 [799:0];
            data_reg7 [799:0] <= data_reg6 [799:0];
            data_reg8 [799:0] <= data_reg7 [799:0];
            data_reg9 [799:0] <= data_reg8 [799:0];
            data_reg10 [799:0] <= data_reg9 [799:0];

        end

        if ((data_reg2[VGA_X1-'d4]+data_reg2[VGA_X1-'d3]+data_reg2[VGA_X1-'d2]+data_reg2[VGA_X1-'d1]+data_reg2[VGA_X1]
            + data_reg2[VGA_X1+'d1]+data_reg2[VGA_X1+'d2]+data_reg2[VGA_X1+'d3]+data_reg2[VGA_X1+'d4]
            + data_reg3[VGA_X1-'d4]+data_reg3[VGA_X1-'d3]+data_reg3[VGA_X1-'d2]+data_reg3[VGA_X1-'d1]+data_reg3[VGA_X1]
            + data_reg3[VGA_X1+'d1]+data_reg3[VGA_X1+'d2]+data_reg3[VGA_X1+'d3]+data_reg3[VGA_X1+'d4]
            + data_reg4[VGA_X1-'d4]+data_reg4[VGA_X1-'d3]+data_reg4[VGA_X1-'d2]+data_reg4[VGA_X1-'d1]+data_reg4[VGA_X1]
            + data_reg4[VGA_X1+'d1]+data_reg4[VGA_X1+'d2]+data_reg4[VGA_X1+'d3]+data_reg4[VGA_X1+'d4]
            + data_reg5[VGA_X1-'d4]+data_reg5[VGA_X1-'d3]+data_reg5[VGA_X1-'d2]+data_reg5[VGA_X1-'d1]+data_reg5[VGA_X1]
            + data_reg5[VGA_X1+'d1]+data_reg5[VGA_X1+'d2]+data_reg5[VGA_X1+'d3]+data_reg5[VGA_X1+'d4]
            + data_reg6[VGA_X1-'d4]+data_reg6[VGA_X1-'d3]+data_reg6[VGA_X1-'d2]+data_reg6[VGA_X1-'d1]+data_reg6[VGA_X1]
            + data_reg6[VGA_X1+'d1]+data_reg6[VGA_X1+'d2]+data_reg6[VGA_X1+'d3]+data_reg6[VGA_X1+'d4]
            + data_reg7[VGA_X1-'d4]+data_reg7[VGA_X1-'d3]+data_reg7[VGA_X1-'d2]+data_reg7[VGA_X1-'d1]+data_reg7[VGA_X1]
            + data_reg7[VGA_X1+'d1]+data_reg7[VGA_X1+'d2]+data_reg7[VGA_X1+'d3]+data_reg7[VGA_X1+'d4]
            + data_reg8[VGA_X1-'d4]+data_reg8[VGA_X1-'d3]+data_reg8[VGA_X1-'d2]+data_reg8[VGA_X1-'d1]+data_reg8[VGA_X1]
            + data_reg8[VGA_X1+'d1]+data_reg8[VGA_X1+'d2]+data_reg8[VGA_X1+'d3]+data_reg8[VGA_X1+'d4]
            + data_reg9[VGA_X1-'d4]+data_reg9[VGA_X1-'d3]+data_reg9[VGA_X1-'d2]+data_reg9[VGA_X1-'d1]+data_reg9[VGA_X1]
            + data_reg9[VGA_X1+'d1]+data_reg9[VGA_X1+'d2]+data_reg9[VGA_X1+'d3]+data_reg9[VGA_X1+'d4]
            + data_reg10[VGA_X1-'d4]+data_reg10[VGA_X1-'d3]+data_reg10[VGA_X1-'d2]+data_reg10[VGA_X1-'d1]+data_reg10[VGA_X1]
            + data_reg10[VGA_X1+'d1]+data_reg10[VGA_X1+'d2]+data_reg10[VGA_X1+'d3]+data_reg10[VGA_X1+'d4]) > 7'd75) begin
            fltr2 <= 10'h3FF;
            fltr2_R <= 10'h3FF;
            fltr2_G <= 10'h3FF;
            fltr2_B <= 10'h3FF;
```

**Figure 28 Spatial Filtering Verilog code Part 1**



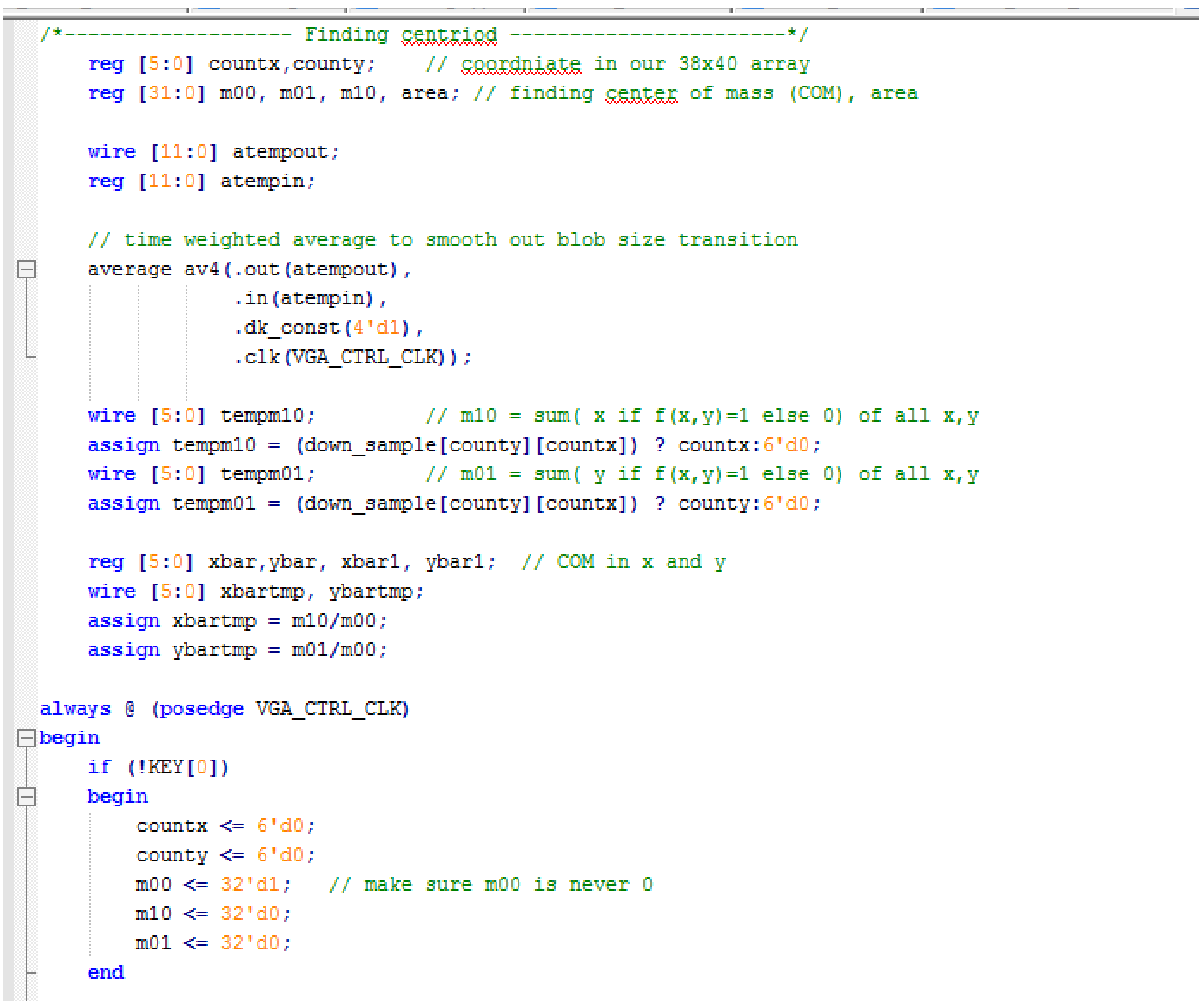
```
        end

        else begin
            fltr2 <= 10'h0;
            fltr2_R <= 10'h0;
            fltr2_G <= 10'h0;
            fltr2_B <= 10'h0;
        end
```

**Figure 29 Spatial Filtering Verilog code Part 2**



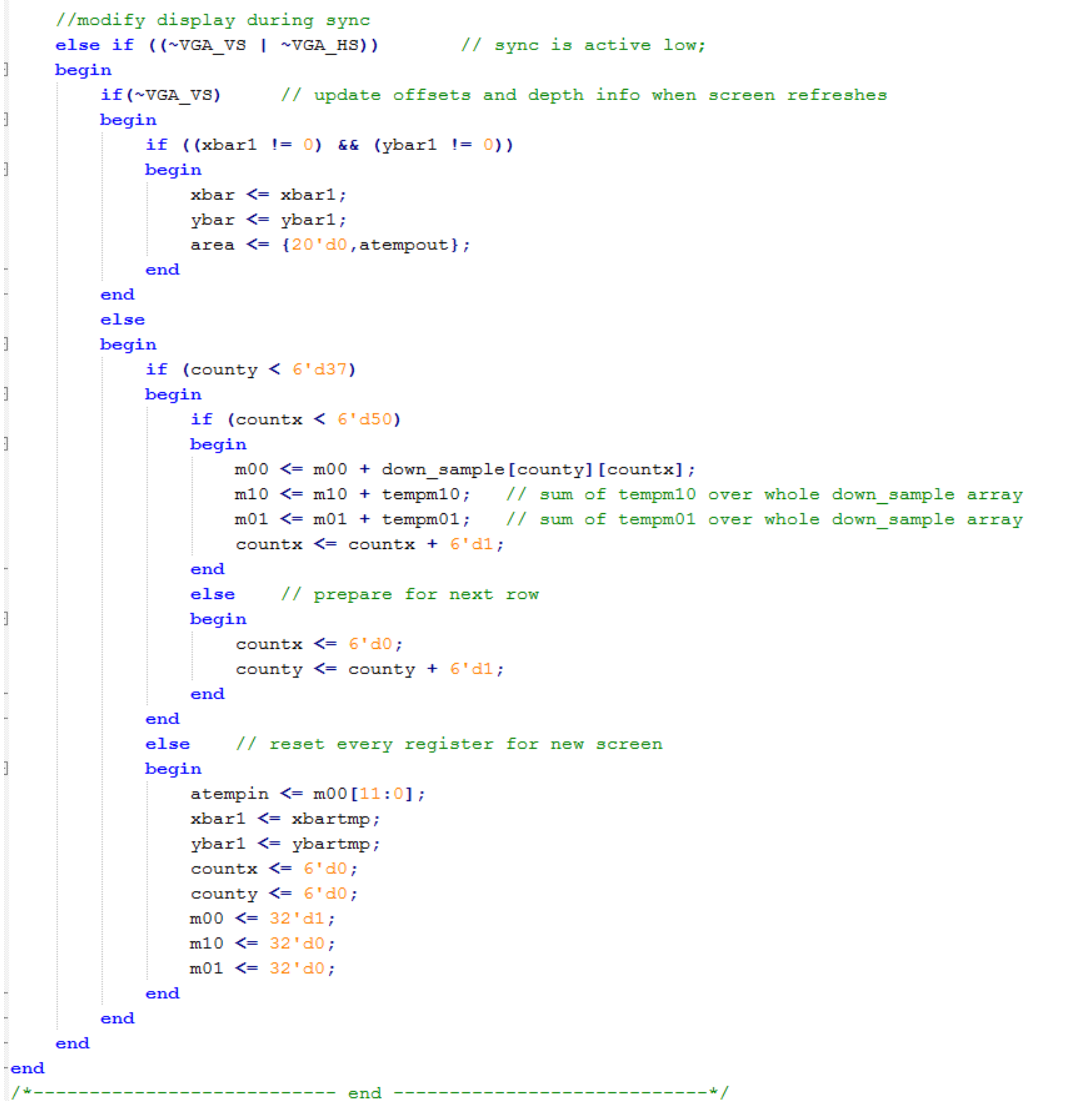
```
/*--------------------- Down sampling -----------------------*/
    reg [49:0] down_sample[0:37];   // down sample 30x40 register array
    reg [5:0] xsum;
    reg [5:0] ysum[49:0];           // a running sum array each over its 16 vertical pixels
    wire xavg;
    assign xavg = (xsum > 6'd0);    // thresholded xsum value, 1 if greater than 0

always @ (posedge VGA_CTRL_CLK) begin
    if (Coord_X[3:0] != 4'd15)
        xsum <= xsum + r3;      // adding spatial weighted average value to the xsum
    else
    begin
        if (Coord_Y[3:0] != 4'd15)  // within 16 rows
            ysum[Coord_X[9:4]] <= ysum[Coord_X[9:4]] + xavg; // add up xsum to get ysum
        else
        begin
            ysum[Coord_X[9:4]] <= 6'd0; // reset ysum for next 20 rows
            down_sample[Coord_Y[9:4]][Coord_X[9:4]] <= (ysum[Coord_X[9:4]] > 6'b000111); // update and threshold down sampled array
        end
        xsum <= 6'd0;       // reset xsum for next 16 columns
    end
end
/*-------------------------- end ---------------------------*/
```

**Figure 30 Down Sampling verilog code**



```
/*-------------------- Finding centriod ------------------------*/
    reg [5:0] countx,county;    // coordniate in our 38x40 array
    reg [31:0] m00, m01, m10, area; // finding center of mass (COM), area

    wire [11:0] atempout;
    reg [11:0] atempin;

    // time weighted average to smooth out blob size transition
    average av4(.out(atempout),
                .in(atempin),
                .dk_const(4'd1),
                .clk(VGA_CTRL_CLK));

    wire [5:0] tempm10;         // m10 = sum( x if f(x,y)=1 else 0) of all x,y
    assign tempm10 = (down_sample[county][countx]) ? countx:6'd0;
    wire [5:0] tempm01;         // m01 = sum( y if f(x,y)=1 else 0) of all x,y
    assign tempm01 = (down_sample[county][countx]) ? county:6'd0;

    reg [5:0] xbar,ybar, xbar1, ybar1;  // COM in x and y
    wire [5:0] xbartmp, ybartmp;
    assign xbartmp = m10/m00;
    assign ybartmp = m01/m00;

always @ (posedge VGA_CTRL_CLK)
begin
    if (!KEY[0])
    begin
        countx <= 6'd0;
        county <= 6'd0;
        m00 <= 32'd1;   // make sure m00 is never 0
        m10 <= 32'd0;
        m01 <= 32'd0;
    end
```

**Figure 31 Centroid Calculation verilog code part 1**



```
    //modify display during sync
    else if ((~VGA_VS | ~VGA_HS))       // sync is active low;
    begin
        if(~VGA_VS)      // update offsets and depth info when screen refreshes
        begin
            if ((xbar1 != 0) && (ybar1 != 0))
            begin
                xbar <= xbar1;
                ybar <= ybar1;
                area <= {20'd0,atempout};
            end
        end
        else
        begin
            if (county < 6'd37)
            begin
                if (countx < 6'd50)
                begin
                    m00 <= m00 + down_sample[county][countx];
                    m10 <= m10 + tempm10;   // sum of tempm10 over whole down_sample array
                    m01 <= m01 + tempm01;   // sum of tempm01 over whole down_sample array
                    countx <= countx + 6'd1;
                end
                else    // prepare for next row
                begin
                    countx <= 6'd0;
                    county <= county + 6'd1;
                end
            end
            else    // reset every register for new screen
            begin
                atempin <= m00[11:0];
                xbar1 <= xbartmp;
                ybar1 <= ybartmp;
                countx <= 6'd0;
                county <= 6'd0;
                m00 <= 32'd1;
                m10 <= 32'd0;
                m01 <= 32'd0;
            end
        end
    end
end
/*--------------------------- end ----------------------------*/
```

**Figure 32  Centroid Calculation verilog code part 2**