FYP Final Presentation
“Hardware-Based Face Detection”

Supervisor:

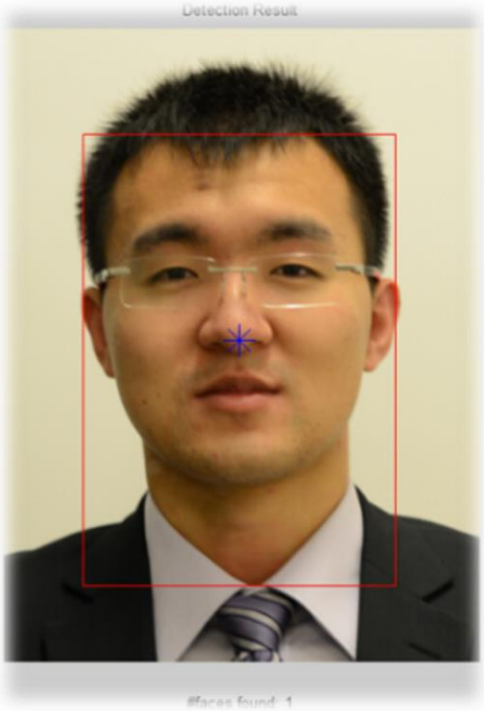
Dr. Benjamin C. Schafer

Project ID: schaferb_20130322183957

Student	Qixuan ZHANG
Student ID	10802747D
Venue	Room DE304
Time	12:40 - 1:00 pm
Date	May 11st, 2015

Outline

- Recall memory
- Project Schedule & Milestones
- Methodology
- Hardware Implementation
- Result & Conclusion
- Further Development
- Q&A



Recall your memory => Face Detection

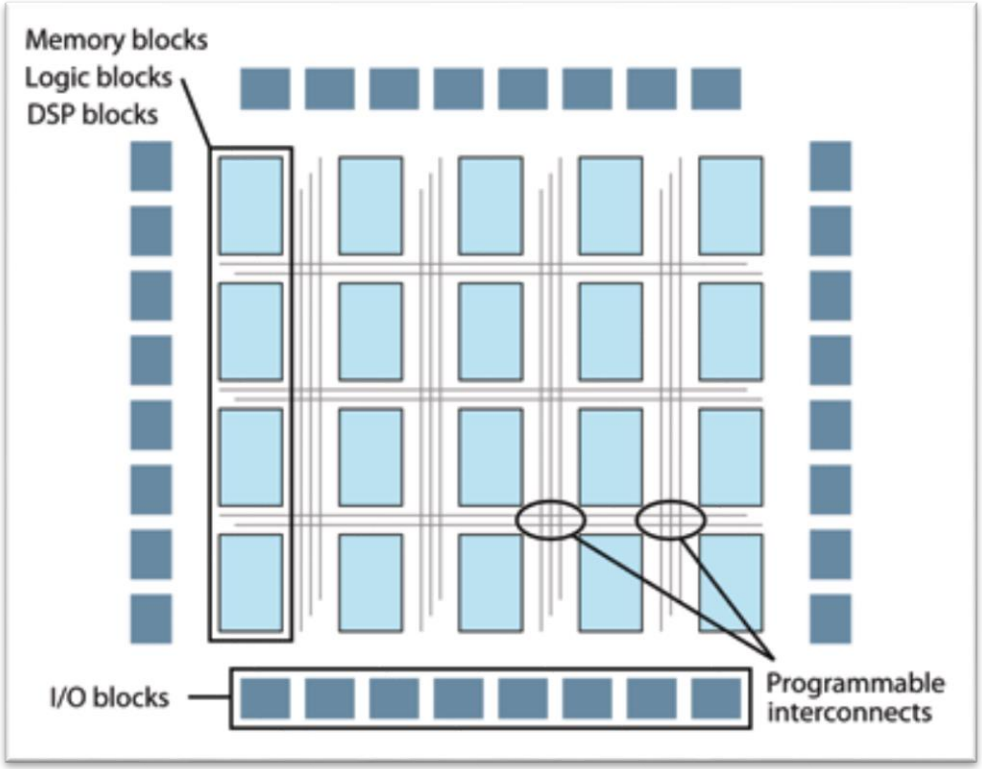
- Wide applications
  - Auto Focusing for Digital Cameras
  - Face Recognition
  - Video surveillance, and etc…

- Skin-Color-Based Face Detection
  - Simple implementation
  - Using fewer FPGA resources

Recall your memory => FPGA

- **Reconfigurable "Versatile Chips"**

Field-Programmable Gate Array (FPGA)



http://www.electronicproducts.com/Digital_ICs/Standard_and_Programmable_Logic/The_evolution_of_FPGA_coprocessing.aspx

Objectives

- To study on the image & video processing techniques, especially ones related to face detection;

- To be familiar with the FPGA system design and develop deeper understanding on its characteristics:
  - Compared to sequential micro-processors

  ✓ **Parallel Execution**

  => Hardware acceleration

  => Faster processing for specific applications

  - Compared to custom ASIC design

  ✓ **Reconfigurable**

  => Faster prototype

  ⇒Low-cost verification

- To implement an FPGA-Based Face Detection System

Project Schedule

Month/Year	Task	Progress
09/2014	Background Learning and Basic Understanding Establishment	✓
10/2014	Trials on C Sobel Filter and get familiar with CWB & QuartusII Tools	✓
11/2014	DE2-115 Board Experiments and Basic Function Implementation like Camera Capture, VGA Display and Simple Image Operations	✓
12/2014	Edge Detection Operation and HLS of the C Sobel Filter Interim Presentation & Report	✓
01/2015	System Design and Operation Flow Trials in Software Approach	✓
02/2015	Hardware Single Modules Design and Functionality Verification	✓
03/2015	System Module Integration and Board Verification	✓
04/2015	Testing, Improvement, Final Report and Presentation	✓

Major Steps of Project Implementation

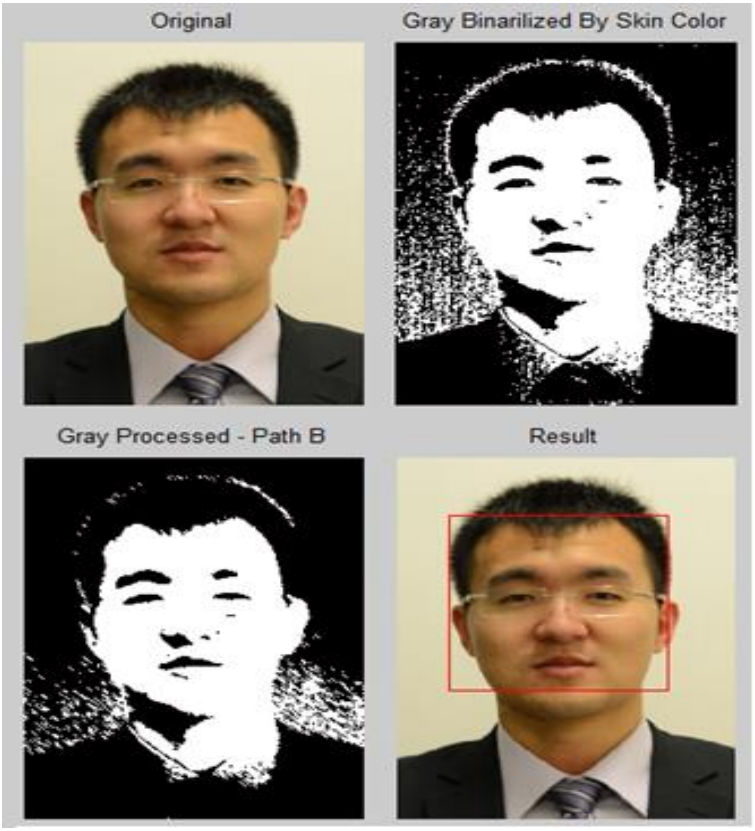
Edge Detection Trial

MATLAB Prototype

Adapted into hardware suitable method

Face Detection System

MATLAB Working Flow



Input Frame Images

↓

Skin Segmentation/Thresholding

↓

Morphological Filtering

↓

Labeling Connect Components

↓

Area Calculation

↓

Spatial Filtering

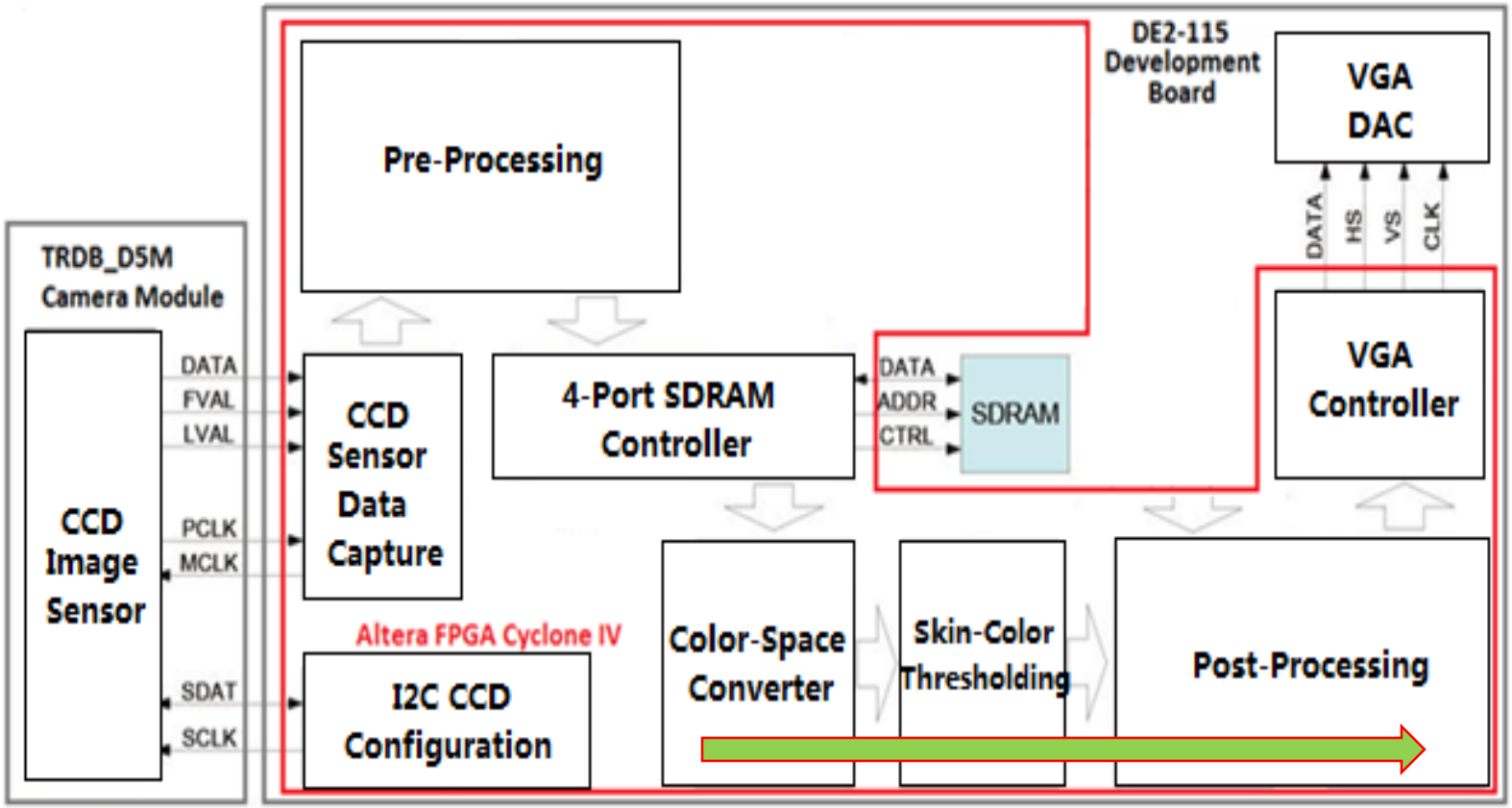
↓

Centroid Computation

↓

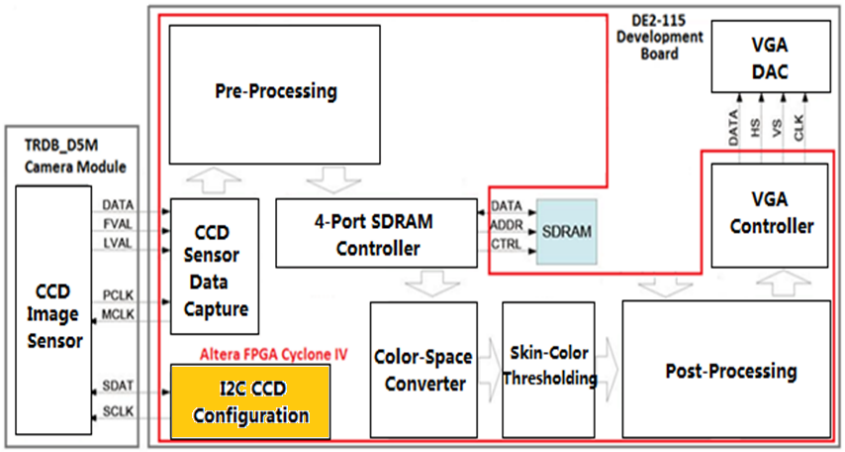
Detection Result

Skin-Color-Based Face Detection System

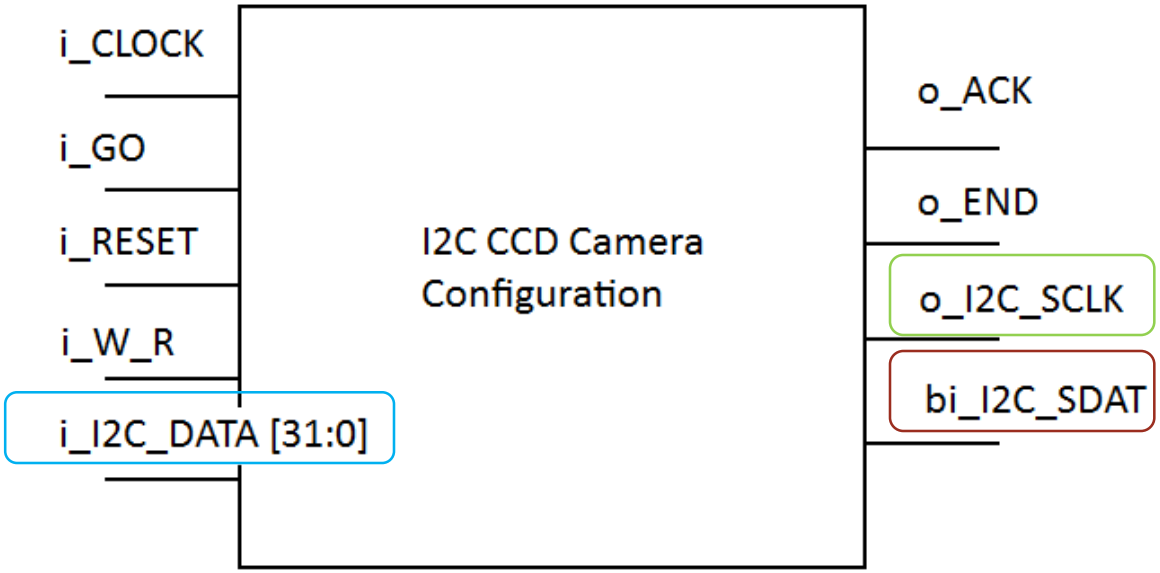


- One **PLL (Phase-Locked Loop)** manages the clock utilization;
- One **Reset_Delay** Module manages the RESET functions;

CCD Camera Configuration By I2C



- Multi-master, multi-slave, single-ended, serial bus
- Used for attaching lower-speed peripherals to processors on computer motherboards or embedded systems
- Two critical bus lines
  - a serial data line (**SDA**)
  - a serial clock line (**SCL**)



CCD Camera Configuration By I2C

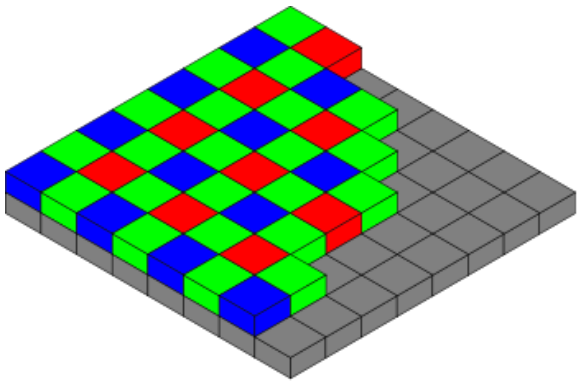
- Look-Up Table (LUT) built according to Hardware Specifications

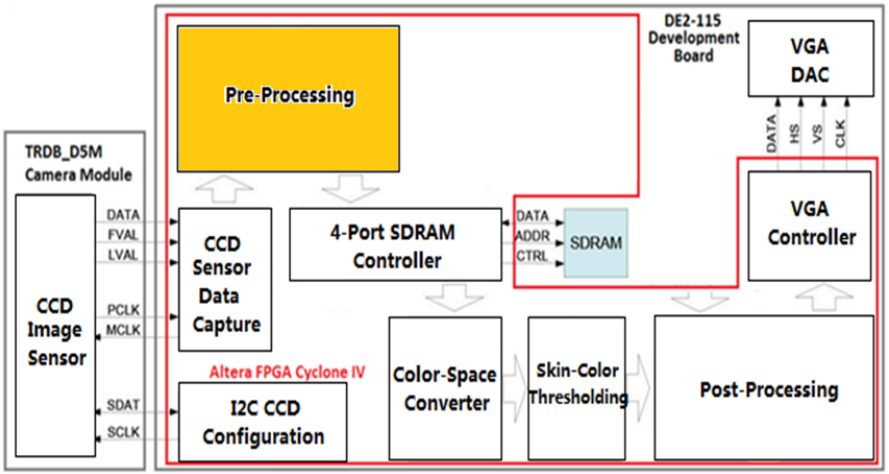
```
239     case(LUT_INDEX)
240     0  :  LUT_DATA <= 24'h000000;
241     1  :  LUT_DATA <= 24'h20c000;          // Mirror Row and Columns
242     2  :  LUT_DATA <= {8'h09,senosr_exposure};// Exposure
243     3  :  LUT_DATA <= 24'h050000;          // H_Blanking
244     4  :  LUT_DATA <= 24'h060019;          // V_Blanking
245     5  :  LUT_DATA <= 24'h0A8000;          // change latch
246     6  :  LUT_DATA <= 24'h2B0013;          // Green 1 Gain
247     7  :  LUT_DATA <= 24'h2C009A;          // Blue Gain
248     8  :  LUT_DATA <= 24'h2D019C;          // Red Gain
249     9  :  LUT_DATA <= 24'h2E0013;          // Green 2 Gain
250     10 :  LUT_DATA <= 24'h100051;          // set up PLL power on
251   `ifdef VGA_640x480p60
252     11 :  LUT_DATA <= 24'h111f04;          // PLL_m_Factor<<8+PLL_n_Divider
253     12 :  LUT_DATA <= 24'h120001;          // PLL_p1_Divider
254   `else
255     11 :  LUT_DATA <= 24'h111805;          // PLL_m_Factor<<8+PLL_n_Divider
256     12 :  LUT_DATA <= 24'h120001;          // PLL_p1_Divider
257   `endif
258     13 :  LUT_DATA <= 24'h100053;          // set USE PLL
259     14 :  LUT_DATA <= 24'h980000;          // disble calibration
260     15 :  LUT_DATA <= 24'hA00000;          // Test pattern control
261     16 :  LUT_DATA <= 24'hA10000;          // Test green pattern value
262     17 :  LUT_DATA <= 24'hA20FFF;          // Test red pattern value
263     18 :  LUT_DATA <= sensor_start_row  ;  // set start row
264     19 :  LUT_DATA <= sensor_start_column ;   // set start column
265     20 :  LUT_DATA <= sensor_row_size;     // set row size
266     21 :  LUT_DATA <= sensor_column_size;     // set column size
267     22 :  LUT_DATA <= sensor_row_mode;     // set row mode in bin mode
268     23 :  LUT_DATA <= sensor_column_mode;     // set column mode    in bin mode
269     24 :  LUT_DATA <= 24'h4901A8;          // row black target
270     default:LUT_DATA  <= 24'h000000;
```

Pre-processing

--- Raw image data processing

- Bayer Pattern => RGB

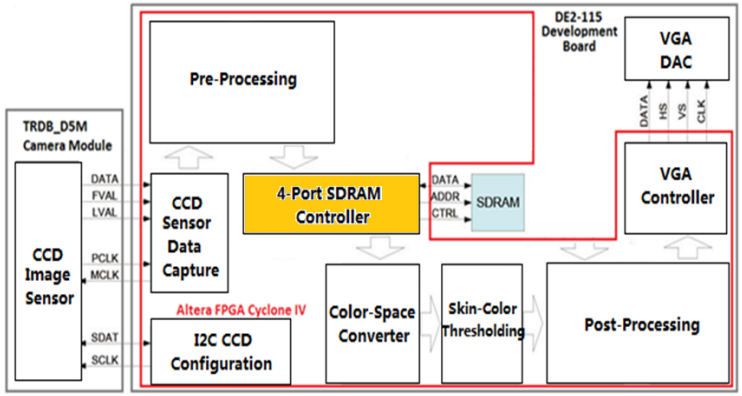




The Bayer arrangement of color filters on the pixel array of an image sensor

Interpolation to get RGB components

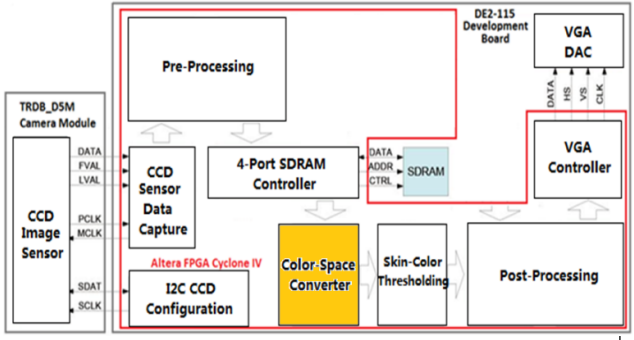
Frame Buffer (Multi-Port SDRAM Controller)



- Multi-Ports
  - 2 writing ports + 2 reading ports
- FIFO Control
  - The video frames are captured real-time and buffered in FIFO
- Critical issues
  - R/W Bandwidth (16-bit each port)
  - Read/Write Synchronization
  - Memory Utilization Efficiency

```
    // FIFO Write Side 1
  WR1_DATA,
    WR1,
    WR1_ADDR,
    WR1_MAX_ADDR,
    WR1_LENGTH,
    WR1_LOAD,
    WR1_CLK,
    // FIFO Write Side 2
  WR2_DATA,
    WR2,
    WR2_ADDR,
    WR2_MAX_ADDR,
    WR2_LENGTH,
    WR2_LOAD,
    WR2_CLK,
    // FIFO Read Side 1
  RD1_DATA,
    RD1,
    RD1_ADDR,
    RD1_MAX_ADDR,
    RD1_LENGTH,
    RD1_LOAD,
    RD1_CLK,
    // FIFO Read Side 2
  RD2_DATA,
    RD2,
    RD2_ADDR,
    RD2_MAX_ADDR,
    RD2_LENGTH,
    RD2_LOAD,
    RD2_CLK,
```

Color Space Conversion

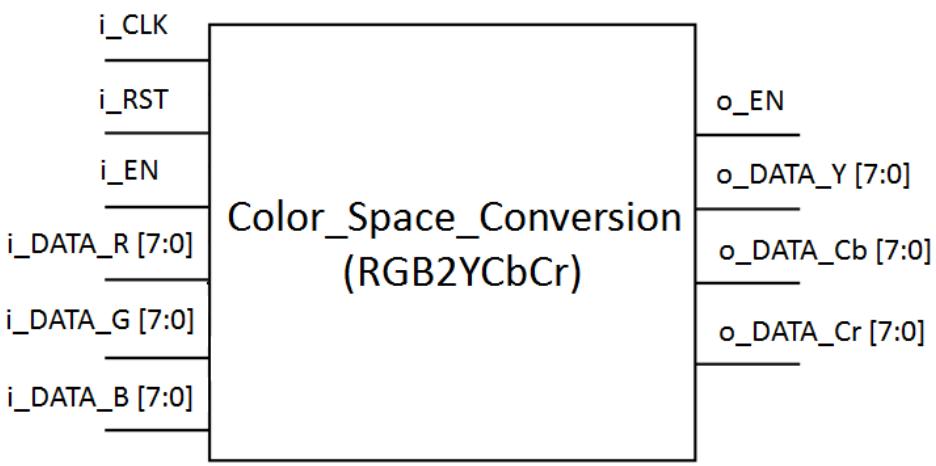


- ✓ RGB ⇔ YCbCr

  => **Luminance** component is separated from **chrominance** component

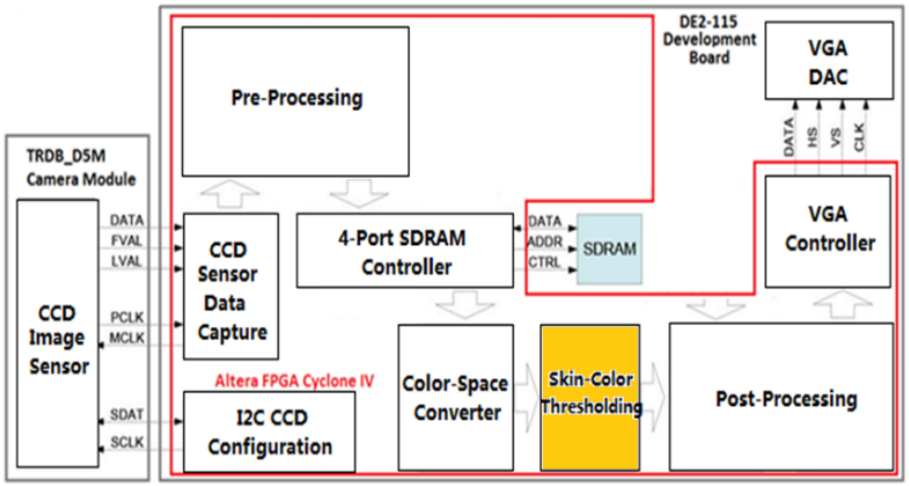
  => used in skin segmentation

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad \begin{array}{l} Y \in [0, 255] \\ C_b \in [0, 255] \\ C_r \in [0, 255] \end{array} \tag{F.15}$$



- Techniques to deal with **FP**
  - Binary representation
  - Shift 10 bits to left => shift back!
  - MAC MegaCores
    (Multiplication & Addition)

Skin Segmentation



- YCbCr Chrominance Component Ranges
  - Theoretical range proposed by previous scholars, D. Chai & K. N. Ngan,

"*Face segmentation using skin color map in videophone applications*"

$$77 < Cb < 127$$

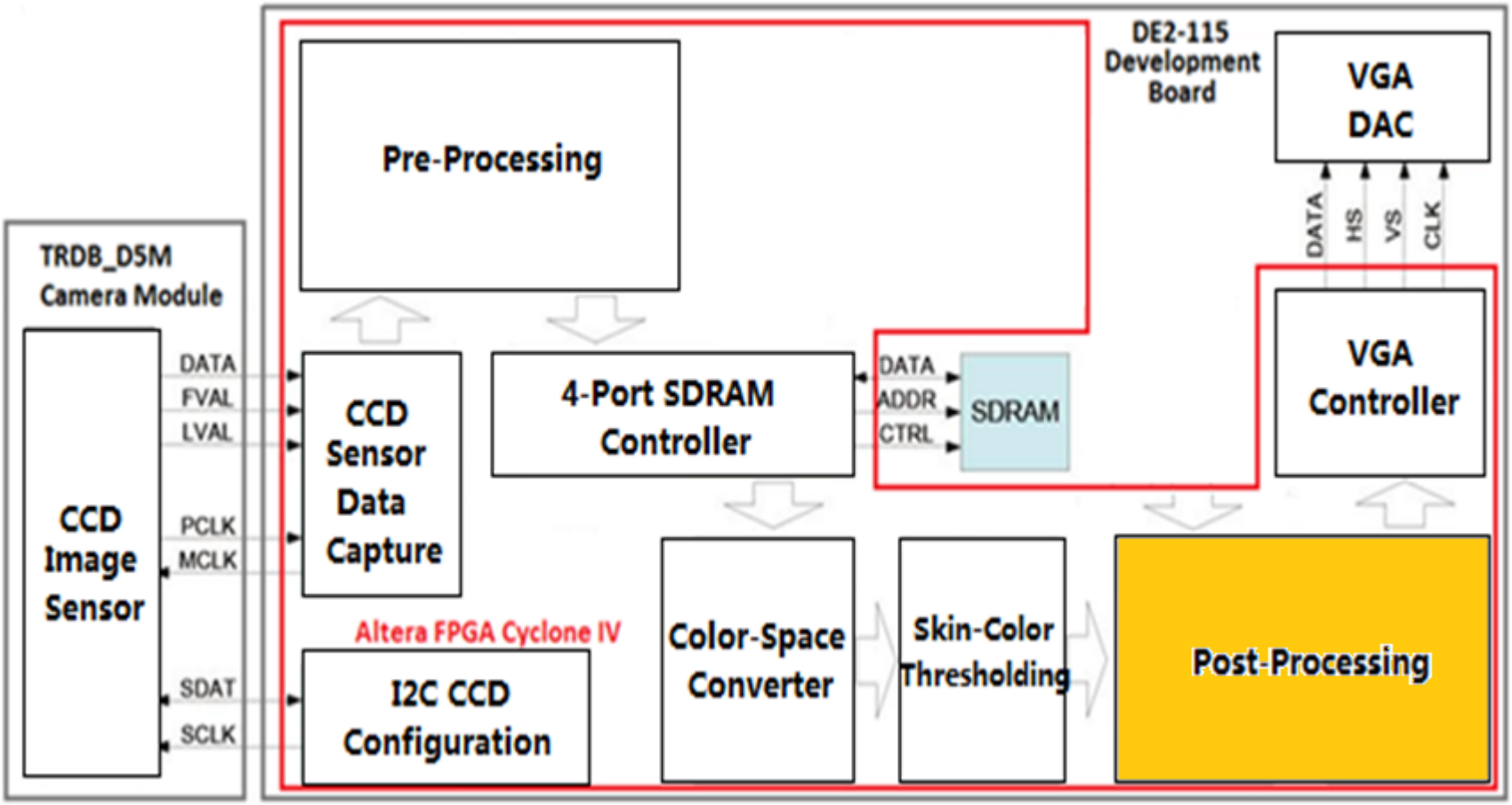
$$133 < Cr < 177$$

  - ***Value range should be adjusted according to the real environment setting for the reason of lighting noise***

Post-Processing

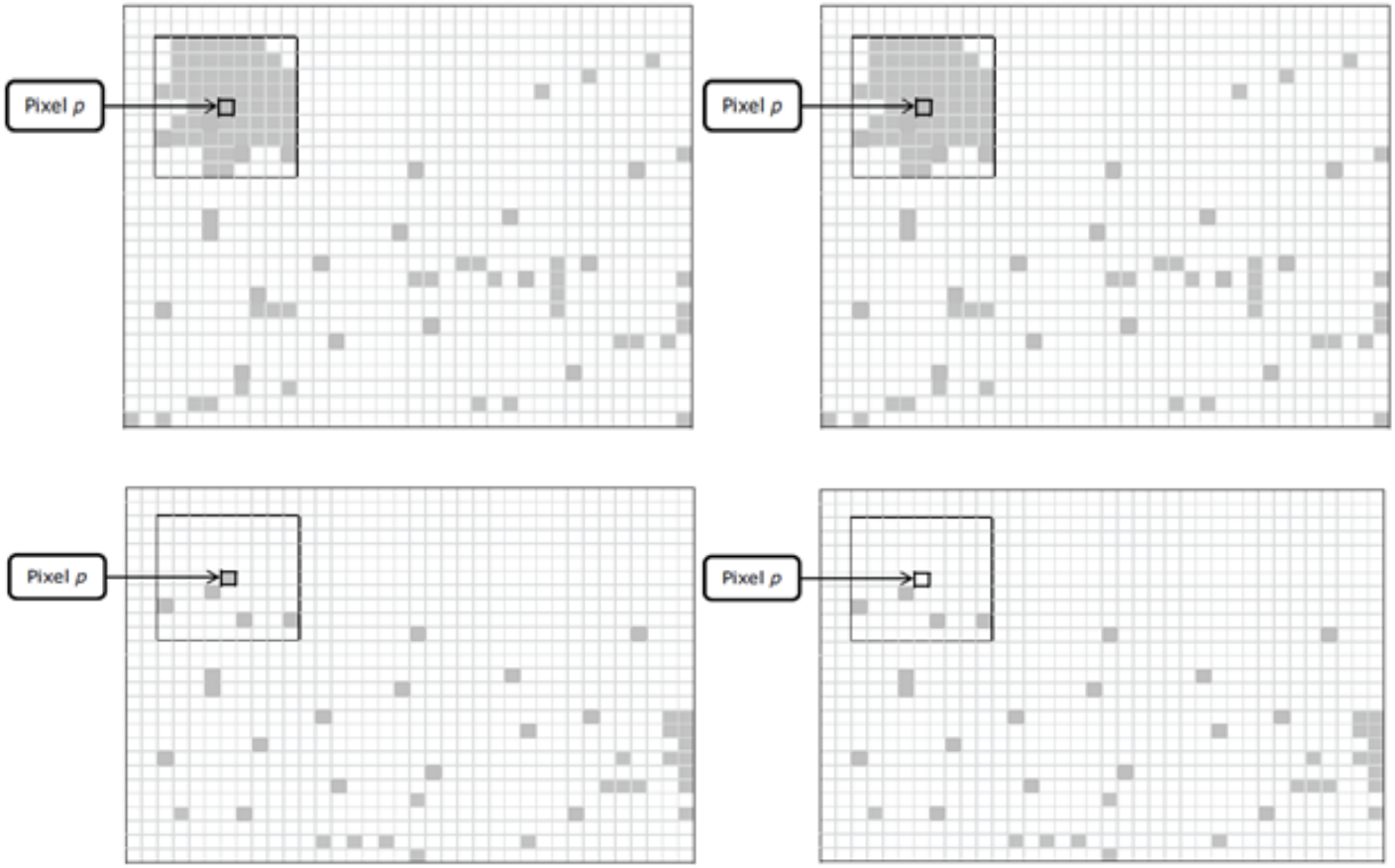
=>

Spatial Filtering + Temporal Filtering + Centroid Computation



Spatial Filtering

- **Window Size** & **Threshold** should be adjusted for better performance
  - 9 Rows * 9 Columns = 81 pixels
  - Threshold = 78



Temporal Filtering

- Time-Weighted Average Module

  (learnt from the averaging module by Prof. Bruce Lund, Cornell University)

  - Weighting for the old average and later pixel data

$$\text{out\_avg} = \frac{3}{4}\,\text{in\_avg} + \frac{1}{4}\,\text{pixel\_data}$$

Multiplication

$$\text{out}_{\text{avg}} = \text{in\_avg} - \frac{1}{4}\,\text{in\_avg} + \frac{1}{4}\,\text{pixel\_data}$$

Bit Shifting

$$\text{out\_avg} = \text{in\_avg} - \text{in\_avg} >> 2 + \text{pixel\_data} >> 2$$

Frame Pixel	t	t+1	t+2	t+3	t+4	t+5	t+6	t+7	t+8	t+9	t+10	t+11	t+12
Before Filtering	0	1	1	1	0	1	1	1	1	0	0	1	1
Average	0.00	0.25	0.44	0.58	0.43	0.58	0.68	0.76	0.82	0.62	0.46	0.60	0.70
After Filtering	0	1	1	1	1	1	1	1	1	1	1	1	1

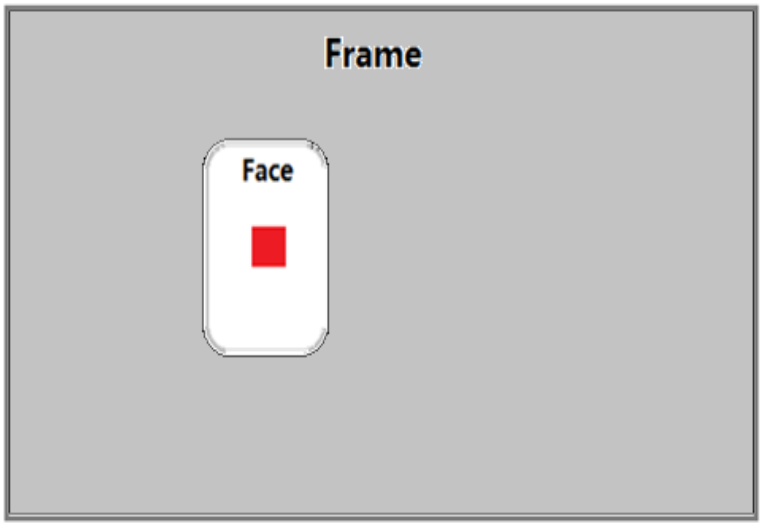
Centroid Computation

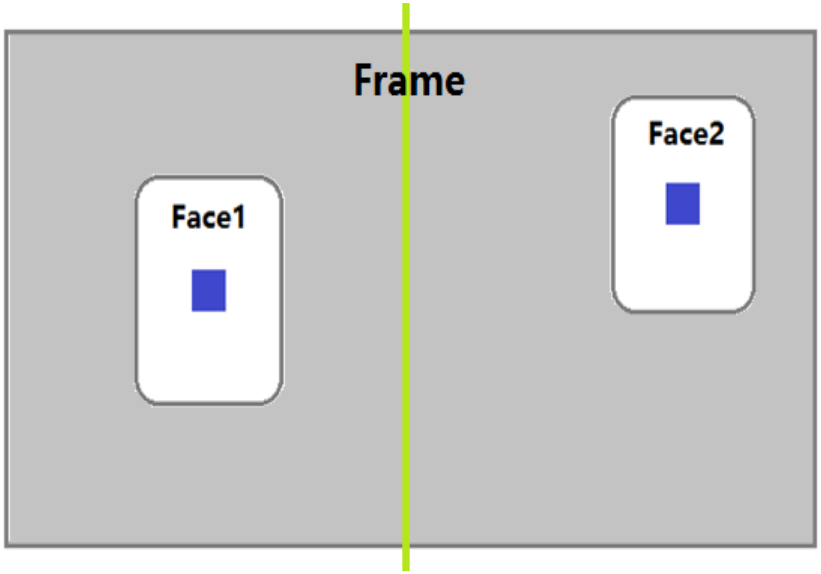
--- To mark the faces

--- For further facial feature verification

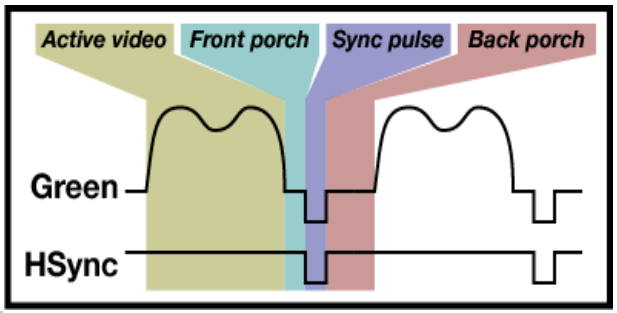
- Calculate the centroids of the candidate regions
  - *By averaging the sum of X,Y Coordinates*

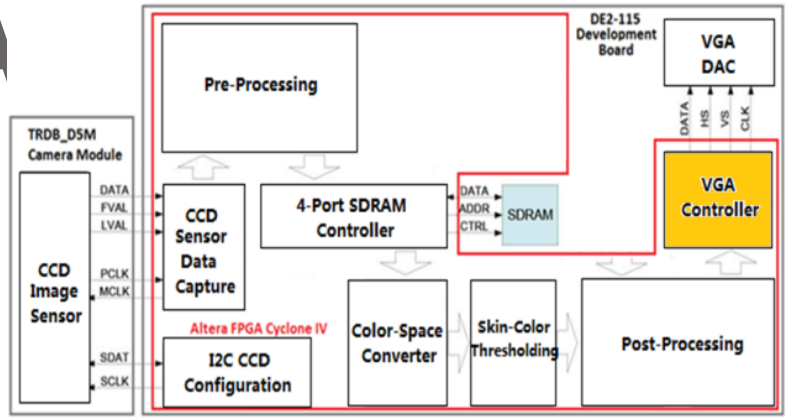
- At most two faces can be detected!
  - *Assume the two faces can only be aligned horizontally*





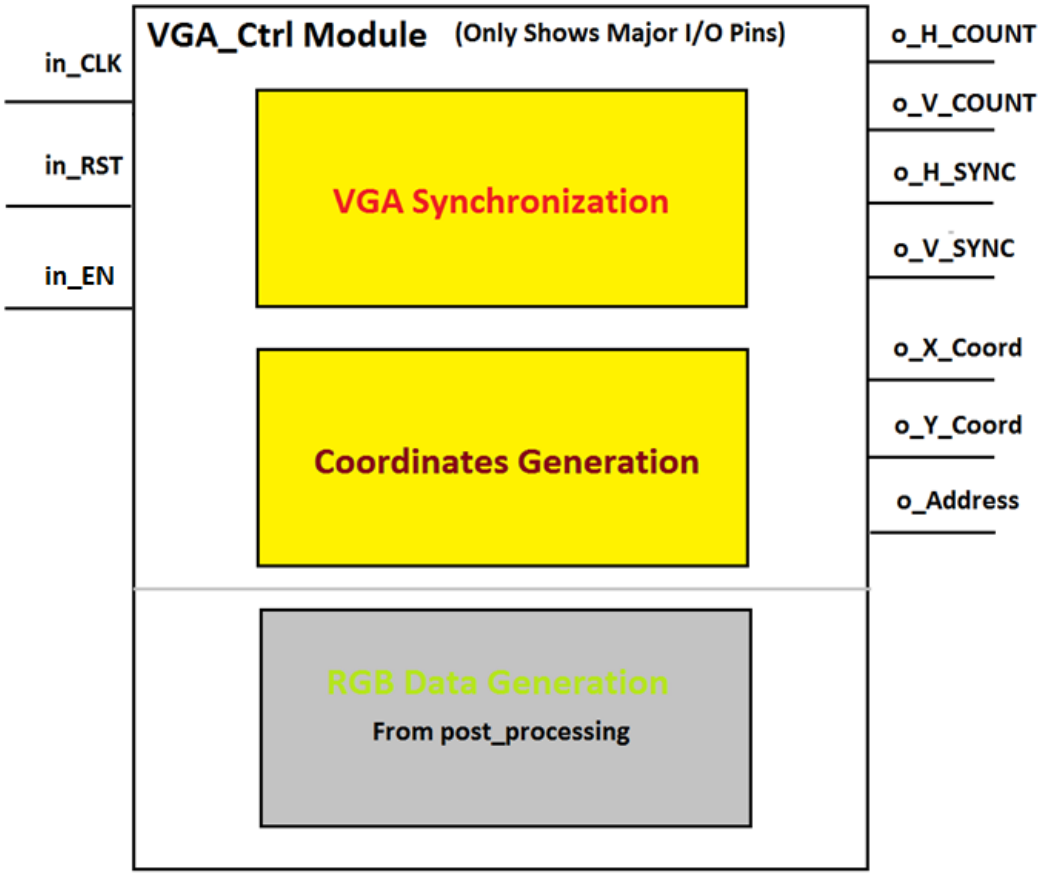
VGA Control by FPGA






```
////////////////////////////////////////////////////////////////////////////////
//      Horizontal      Parameter
parameter       H_FRONT =       16;
parameter       H_SYNC  =       96;
parameter       H_BACK  =       48;
parameter       H_ACT   =       640;
parameter       H_BLANK =       H_FRONT+H_SYNC+H_BACK;
parameter       H_TOTAL =       H_FRONT+H_SYNC+H_BACK+H_ACT;
////////////////////////////////////////////////////////////////////////////////
//      Vertical Parameter
parameter       V_FRONT =       11;
parameter       V_SYNC  =       2;
parameter       V_BACK  =       31;
parameter       V_ACT   =       480;
parameter       V_BLANK =       V_FRONT+V_SYNC+V_BACK;
parameter       V_TOTAL =       V_FRONT+V_SYNC+V_BACK+V_ACT;
////////////////////////////////////////////////////////////////////////////////
```

VGA Display + Pixel Coordinates Synchronization



- Used to record the **corresponding coordinates** for **each operating pixels**;
- Post-processing causes certain cycles **delay**

  => Coordinate signals are **delayed** for specific cycles;

Hardware Implementation Result

Hardware Synthesis Report

Module	Total logic elements	Total combinational functions	Dedicated logic registers	Total registers	Total memory bits	Fmax (Slow 1200mV 85C)	Fmax (Slow 1200mV 0C)
Post_Processing	**18,606 / 114,480 (16 %)**	**18,520 / 114,480 (16 %)**	**6,865 / 114,480 (6 %)**	**6865**	**0 / 3,981,312 (0 %)**	**9.74 MHz**	**10.8 MHz**
skin_seg	41 / 114,480 (< 1 %)	11 / 114,480 (< 1 %)	31 / 114,480 (< 1 %)	31	0 / 3,981,312 (0 %)	N/A	N/A
RAW2RGB	102 / 114,480 (< 1 %)	93 / 114,480 (< 1 %)	72 / 114,480 (< 1 %)	72	30,672 / 3,981,312 (< 1 %)	249.63 MHz	272.7 MHz
Reset_Delay	54 / 114,480 (< 1 %)	54 / 114,480 (< 1 %)	37 / 114,480 (< 1 %)	37	0 / 3,981,312 (0 %)	218.05 MHz	239.98 MHz
CCD_Capture	89 / 114,480 (< 1 %)	88 / 114,480 (< 1 %)	80 / 114,480 (< 1 %)	80	0 / 3,981,312 (0 %)	272.03 MHz	295.86 MHz
RGB2YCbCr	481 / 114,480 (< 1 %)	457 / 114,480 (< 1 %)	195 / 114,480 (< 1 %)	195	0 / 3,981,312 (0 %)	185.98 MHz	208.55 MHz
Sdram_Control	1,645 / 114,480 (1 %)	1,394 / 114,480 (1 %)	745 / 114,480 (< 1 %)	745	32,768 / 3,981,312 (< 1 %)	144.7 MHz	160.05 MHz
VGA_Ctrl	104 / 114,480 (< 1 %)	104 / 114,480 (< 1 %)	26 / 114,480 (< 1 %)	26	0 / 3,981,312 (0 %)	408.0 MHz	447.43 MHz
I2C_CCD_Config	280 / 114,480 (< 1 %)	266 / 114,480 (< 1 %)	131 / 114,480 (< 1 %)	131	0 / 3,981,312 (0 %)	229.15 MHz	247.4 MHz
Complete Design	**20,728 / 114,480 (18 %)**	**20,391 / 114,480 (18 %)**	**7,955 / 114,480 (7 %)**	**7955**	**59,404 / 3,981,312 (1 %)**	**166.0 MHz**	**183.42 MHz**

Conclusion

--- Performance Analysis

- ***Functionality Achieved***
  - Good Functionality in Ideal Environment
    (lighting conditions, simple background)

- ***Hardware or FPGA Resources Optimized***
  - Use optimized MegaCores for Altera Devices
  - Use fewer logic elements (18%)

- Drawbacks

  --- For the limitation of the algorithms adopted
  - Luminance Conditions
  - Facial Views
  - Skin-Color Objects

  ⇒ ***Finding good thresholds for different environments is hard.***
  ⇒ ***Picking bad thresholds yields many false positives.***

Further Issues To Be Considered

- **Aspect I – Algorithm View**

  Advanced Machine Learning Algorithms can be adopted for higher accuracy;

- **Aspect II – Hardware View**

  Customized hardware may be used, such as camera module, memory and DSPs.

- **Aspect III – Design Flow View**

  MATLAB to HDL Coder, Complete HLS Design Flow

- **Aspect IV – Performance Comparison View**

  Use the same algorithm for both software and hardware implementations and make comparisons.

- **Aspect V – Application View**

  Face Detection System with real applications, like security check and etc.

References

- Kwok-Wai Wong, Kin-Man Lam and Wan-Chi Siu, "*An Efficient Algorithm for Human Face Detection and Facial Feature Extraction under Different Conditions*", Pattern Recognition, Vol. 34, pp.1993-2004, U.S.A, 2001
- Erik Hjelmas and Boon Kee Low, *"Face Detection: A Survey"*, Computer Vision and Image Understanding, Vol. 83, No. 3, pp.236-274, Sept. 2001
- Rein-Lien Hsu, Mohamed A. Mottaleb and Anil K. Jain, *"Face Detection in Color Images"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 24, No. 5, May 2002, [Online], http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1000242&tag=1
- Paul Viola and Michael J. Jones, *"Robust Real-Time Face Detection"*, International Journal of Computer Vision 57(2), pp. 137-154, 2004
- Ramsri Goutham Golla, *"Real-time Face Detection and Tracking"*, Arizona State University
- Xilinx training materials on FPGA design - Field Programmable Gate Array, October 2014, [Online] http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm
- J.U. Cho, S. Mirzaei, J. Oberg and R. Kastner, *"FPGA-Based Face Detection System Using Haar Classifiers"*, International Symposium on Field Programmable Gate Arrays (FPGA), Feb. 2009
- Yu Wei, Xiong Bing and Charayaphan Chareonsak, *"FPGA Implementation of AdaBoost Algorithm for Detection of Face Biometrics"*, IEEE International Workshop on Biomedical Circuits & Systems, 2004
- W. Meeus, K. Van Beeck, T. Goedemé, J. Meel and D. Stroobandt, *"An Overview of Today's High-Level Synthesis Tools"*, Design Automation Embedded System, Springer Science + Business Media, LLC 2012
- K. Wakabayashi and Benjamin C. Schafer, *""All-in-C" Behavioral Synthesis and Verification with CyberWorkBench, From C to Tape-Out with No Pain and A Lot of Gain"*, High-Level Synthesis from Algorithm to Digital Circuit, Springer, Chapter 7, 2008
- Yu-ting Pai, Shanq-jang Ruan, Mon-chau Shie and Yi-chi Liu, "*A Simple and Accurate Color Face Detection Algorithm in Complex Background*", ICME, 2006, 2012 IEEE International Conference on Multimedia and Expo, 2012 IEEE International Conference on Multimedia and Expo 2006, pp. 1545-1548

References

- D. Ghimire and J. Lee, *"A Robust Face Detection Method Based on Skin Color and Edges"*, Journal of Information Process System, Vol. 9, No. 1, March 2013
- J. Cho, B. Benson, S. Cheamanukul and R. Kastner, *"Increased Performace of FPGA-Based Color Classification System"*, Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium, May 2010
- S. Paschalakis and M. Bober, *"Real-Time Face Detection and Tracking for Mobile Videoconferencing"*, Real-Time Imaging, Vol. 10, No. 2, pp. 81-94, April 2004
- NEC CyberWorkBench Product Brochure, NEC Official Company Website, Oct. 2014, [Online], http://www.nec.com/en/global/prod/cwb/pdf/212_0920_CyberWorkBench_Eng_03.pdf
- Ming-Hsuan Yang, David J. Kriegman and Narendra Ahuja, *"Detecting Faces in Images: A Survey"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, Jan. 2002
- Thu Thao Nguyen, *"Design Project Report - Real Time Face Detection and Tracking"*, ECE MEng Design Project, Cornell University, December, 2012
- S. L. Phung, A. Bouzerdoum, and D. Chai, *"A Novel Skin Color Model in YCbCr Color Space and Its Application to Human Face Detection"*, IEEE Image Processing, Proceedings, International Conference, Volume I, 2002
- Douglas Chai and King N. Ngan, *"Face Segmentation Using Skin-Color Map in Videophone Applications"*, IEEE Transactions on circuits and systems for video technology, Volume 9, No. 4, June 1999, [Online], http://www.ee.cuhk.edu.hk/~knngan/TCSVT_v9_n4_p551-564.pdf
- Prof. Bruce Land, *"ECE5760 Advanced Micro-controllers Course Portal – Final Projects"*, Cornell University, [Online], http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/
- Chen-Chiung Hsieh, Dung-Hua Liou and Wei-Ru Lai, *"Enhanced Face-Based Adaptive Skin Color Model"*, Journal of Applied Science and Engineering, Vol. 15, No. 2, pp. 167-176, 2012, [Online], http://www2.tku.edu.tw/~tkjse/15-2/10-IE9920.pdf

END

Thank you for your listening!

If no questions…..

Have a nice day! ☺

Recall memory => Face Detection

- Wide applications
  - Auto Focusing for Digital Cameras
  - Face Recognition
  - Video surveillance, and etc…

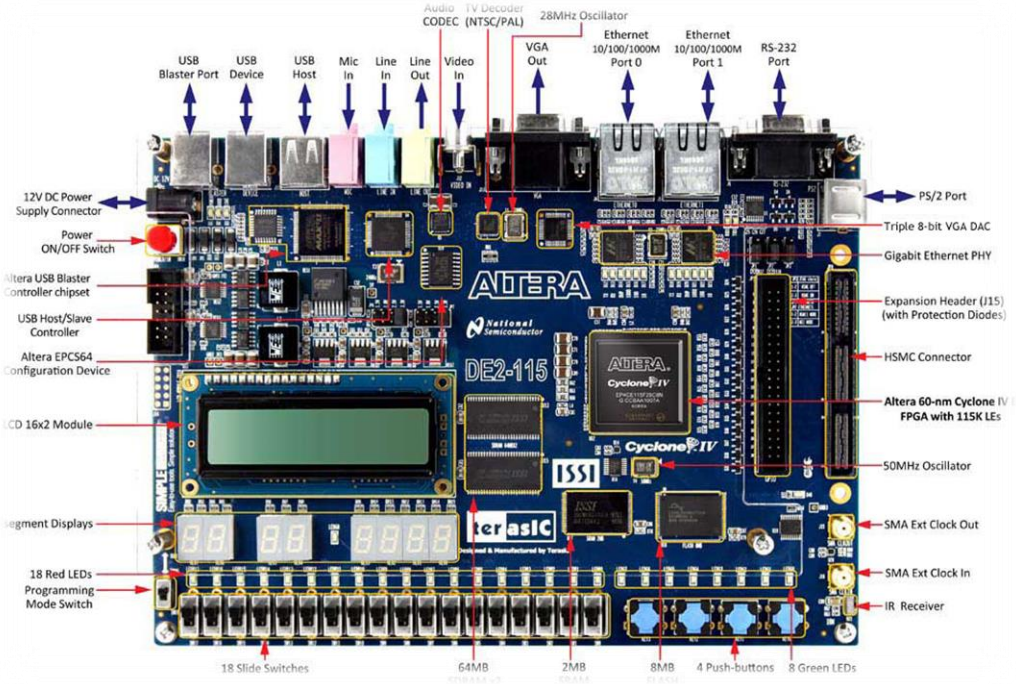
- Viola-Jones AdaBoost Face Detection, Neural Network Based Face Detection, PCA-Based Face Detection
  - Complicated
  - Resource-demanding

- Skin-Color-Based Face Detection
  - Simple implementation
  - Using fewer FPGA resources (Memory & LEs)

Hardware Summary

Function	Hardware Components	Core Modules
Processing and Control	Terasic FPGA DE2-115 Development and Education Board (Altera Cyclone IV 4CE115 FPGA device)	Control Unit, Processing Unit, Memory Unit and Communication Unit
Picture Capture	Terasic TRDB_D5M - 5 Mega Pixel Digital Camera Package	Picture Data Collection Unit
Display	PC Display with VGA Port	VGA Display Unit

Table 3 – Hardware list and corresponding tasks



EDA Tools Summary

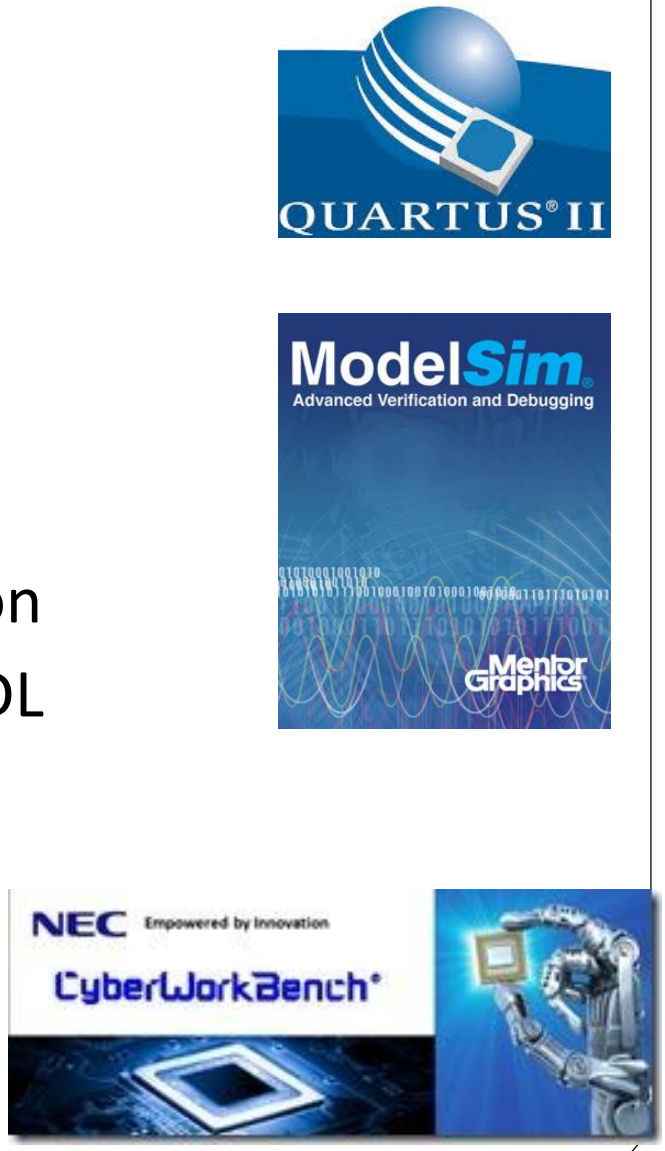
- Altera Quartus II 13.1

  => Verilog HDL Coding
- ModelSim-Altera Starter Edition

  => Simulation
- NEC CyberWorkBench (CWB)

  => HLS + Preliminary Simulation

  => From C to Verilog HDL



Edge Detection Trial

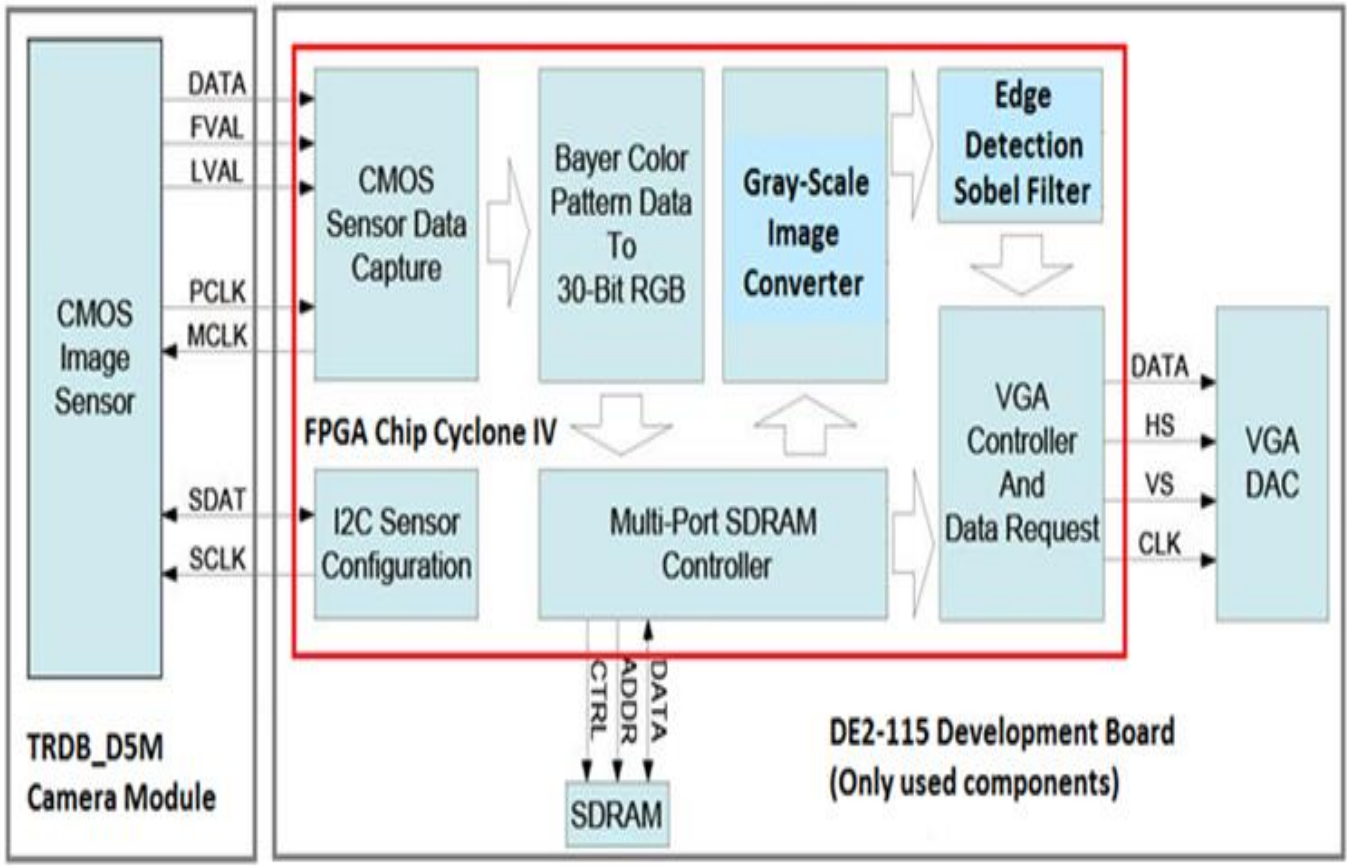
$$\nabla_1 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \qquad \nabla_2 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$\frac{\partial I}{\partial x} \rightarrow \text{horizontal} \qquad \frac{\partial I}{\partial y} \rightarrow \text{vertical}$$

$$|\nabla| = \sqrt{\nabla_2^1 + \nabla_2^2} \text{ ——— gradient magnitude}$$

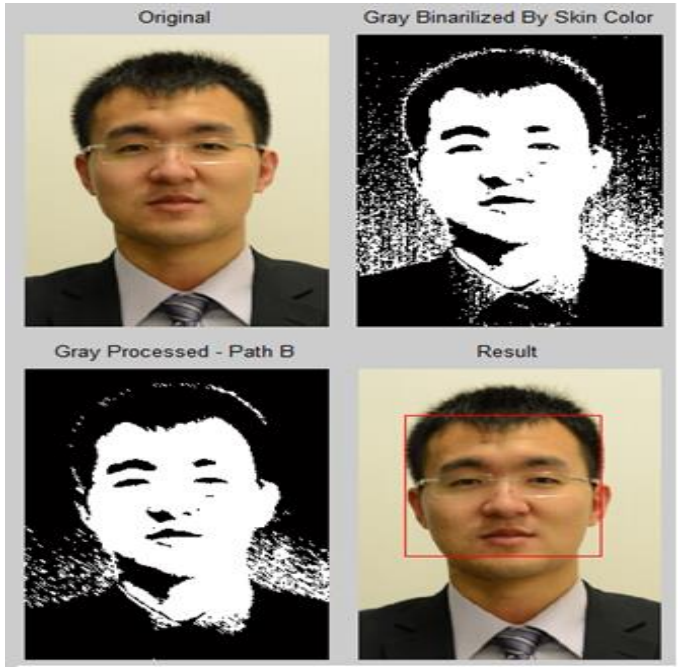
- ⇒ **Sobel Filter**
  - – **Spatial Derivative**

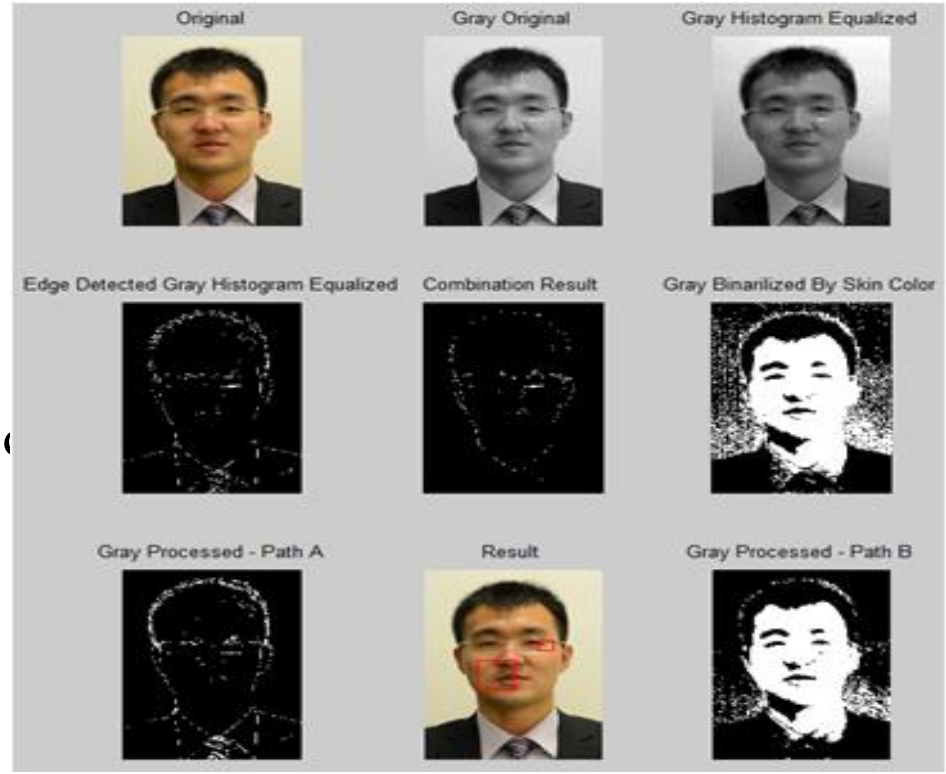
    (Horizontal + Vertical)



MATLAB Result

Pure Skin-Color-Based Face Detec



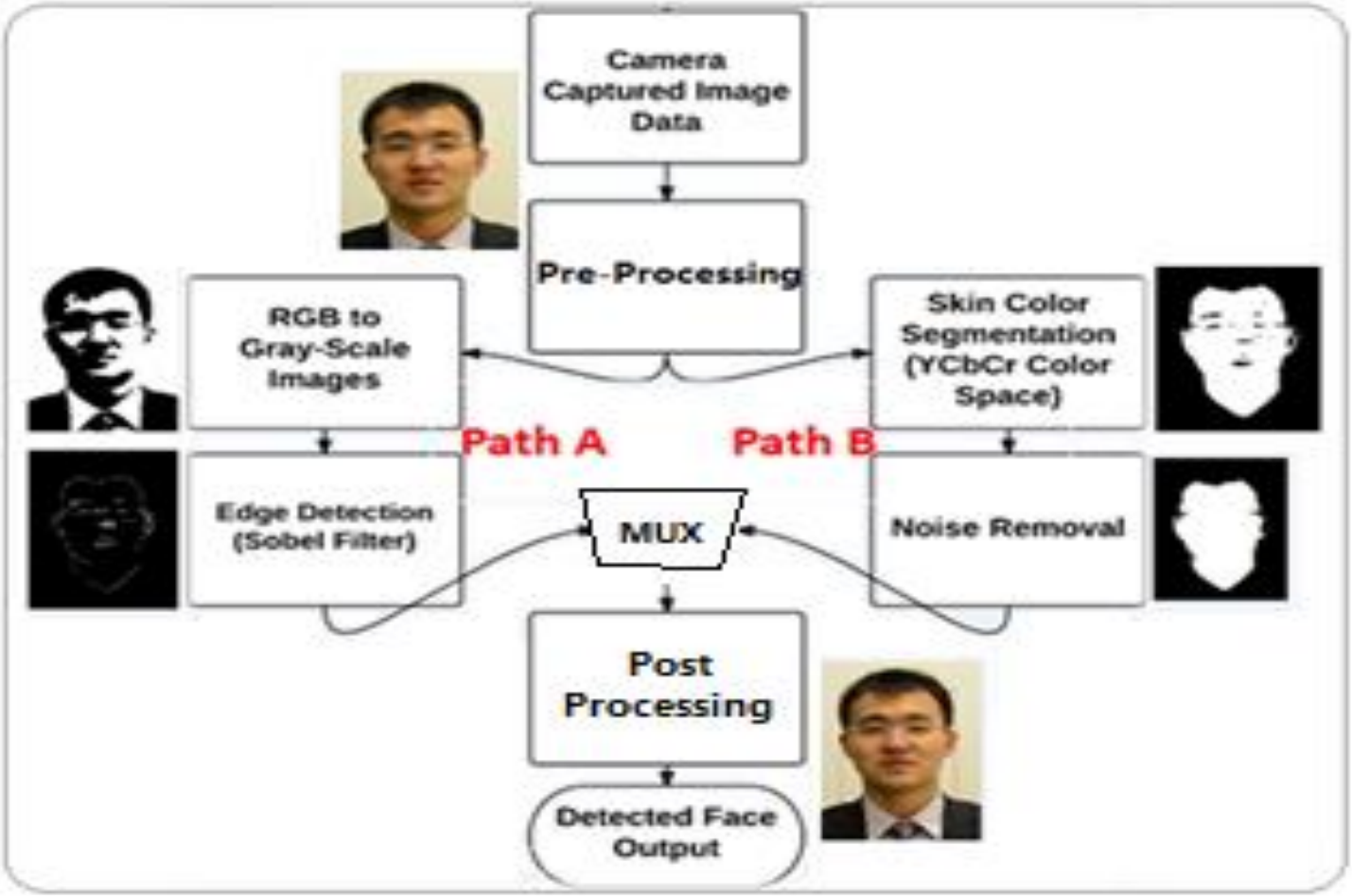


Combined-two-path method

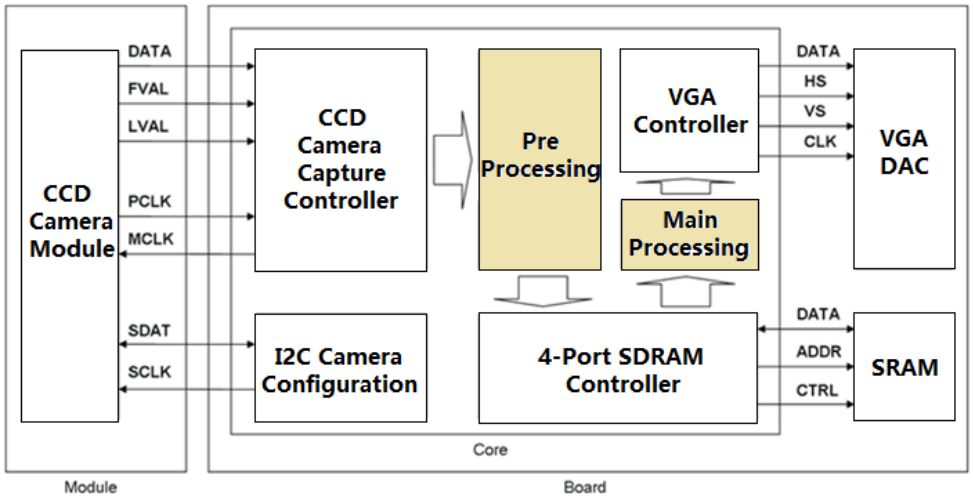
Hardware Implementation

=> Detection Method was adjusted for hardware consideration.

Issues to consider	Software (MATLAB)	Hardware (Verilog)
Memory Resources & Access	The whole frame image are stored and any pixel can be accessed whenever we want;	Specific pixel data in a frame cannot always be stored so they cannot be accessed whenever we want;
Code Implementation & Timing	Sequential execution;	Parallel execution;
	Delay---Wait for the preceding executions;	Lower CLK frequency but tasks can be divided into several cycles and many tasks can be handled at the same time (Pipelines);


Camera Captured Image Data
Pre-Processing
RGB to Gray-Scale Images
Skin Color Segmentation (YCbCr Color Space)
Path A
Path B
Edge Detection (Sobel Filter)
MUX
Noise Removal
Post Processing
Detected Face Output

Basic System Considerations



- Tradeoff
  - Store 10-bit R, G, B data
  - Color Space Conversion after Frame Buffer
    - More bits and more memory
    - Higher video acquisition rate
    - Better synchronization of the pixel coordinates

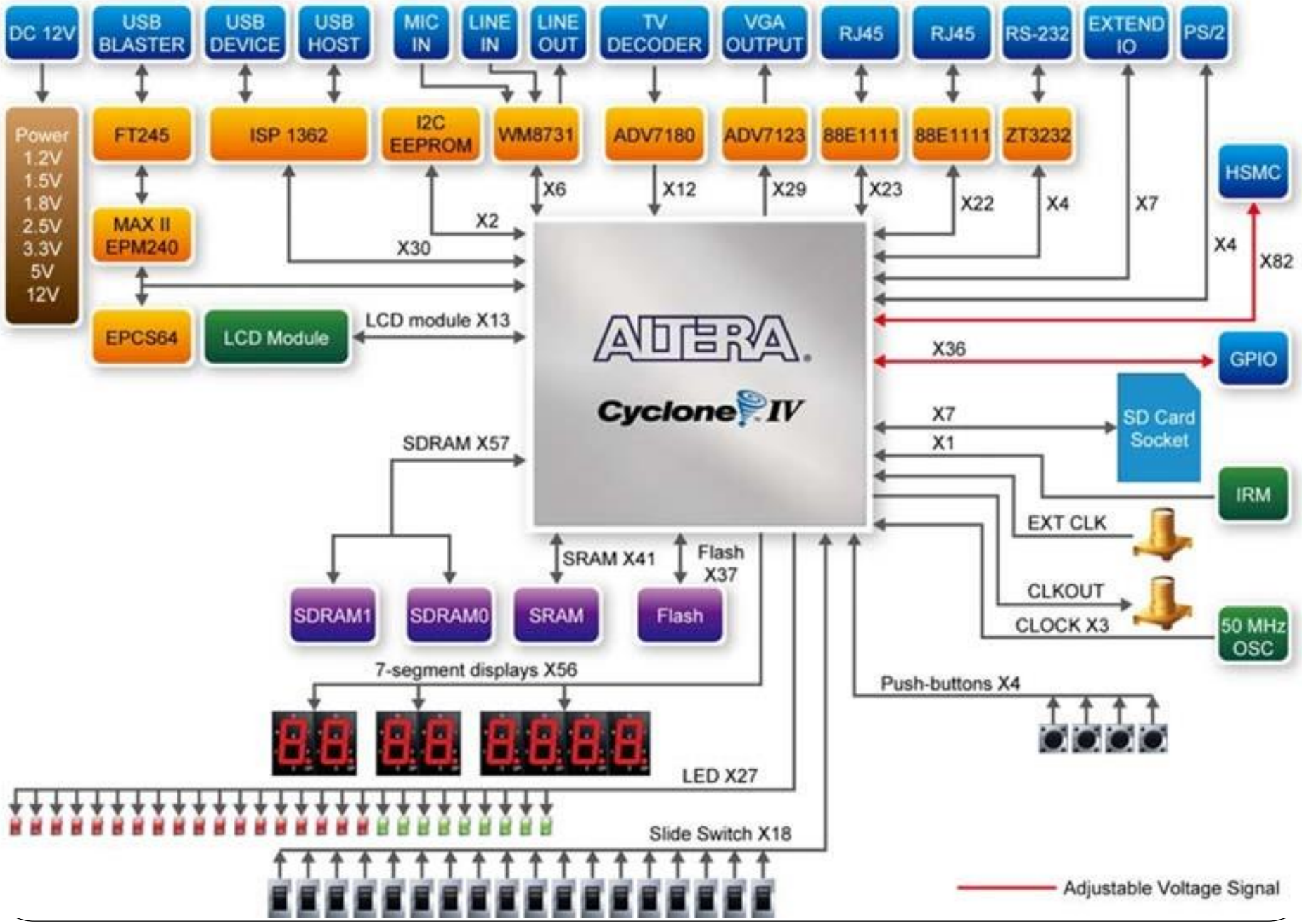
VGA Display + Pixel Coordinates Synchronization

```
always@(posedge iCLK or
begin
        if(!iRST_N)
        begin
                H_Cont
                oVGA_H_S
        end
        else
        begin
                if(H_Cor
                H_Cont
                else
                H_Cont
                //
                if(H_Cor
                oVGA_H_S
                if(H_Cor
                oVGA_H_S
        end
end

//      Vertical Generat
always@(posedge oVGA_H_S
begin
        if(!iRST_N)
        begin
                V_Cont
                oVGA_V_SYNC_2clk                <=      1;
        end
        else
        begin
                if(V_Cont<V_TOTAL)
                V_Cont  <=      V_Cont+1'b1;
                else
                V_Cont  <=      0;
                //      Vertical Sync
                if(V_Cont==V_FRONT-1)                   //      Front porch end
                oVGA_V_SYNC_2clk        <=      1'b0;
                if(V_Cont==V_FRONT+V_SYNC-1)    //      Sync pulse end
                oVGA_V_SYNC_2clk        <=      1'b1;
        end
end
```

```
//////////////////////////////////////////////////////////////////////////////
//      Horizontal      Parameter
parameter       H_FRONT =       16;
parameter       H_SYNC  =       96;
parameter       H_BACK  =       48;
parameter       H_ACT   =       640;
parameter       H_BLANK =       H_FRONT+H_SYNC+H_BACK;
parameter       H_TOTAL =       H_FRONT+H_SYNC+H_BACK+H_ACT;
//////////////////////////////////////////////////////////////////////////////
//      Vertical Parameter
parameter       V_FRONT =       11;
parameter       V_SYNC  =       2;
parameter       V_BACK  =       31;
parameter       V_ACT   =       480;
parameter       V_BLANK =       V_FRONT+V_SYNC+V_BACK;
parameter       V_TOTAL =       V_FRONT+V_SYNC+V_BACK+V_ACT;
//////////////////////////////////////////////////////////////////////////////
```

```
assign  oRequest        =       ((H_Cont>=H_BLANK && H_Cont<H_TOTAL)    && (V_Cont>=V_BLANK && V_Cont<V_TOTAL));
assign  oCurrent_X      =       (H_Cont>=H_BLANK)       ?       H_Cont-H_BLANK  :       11'h0   ;
assign  oCurrent_Y      =       (V_Cont>=V_BLANK)       ?       V_Cont-V_BLANK  :       11'h0   ;
```


DC 12V
USB BLASTER
USB DEVICE
USB HOST
MIC IN
LINE IN
LINE OUT
TV DECODER
VGA OUTPUT
RJ45
RJ45
RS-232
EXTEND IO
PS/2
Power 1.2V 1.5V 1.8V 2.5V 3.3V 5V 12V
FT245
ISP 1362
I2C EEPROM
WM8731
ADV7180
ADV7123
88E1111
88E1111
ZT3232
HSMC
MAX II EPM240
X6
X12
X29
X23
X22
X4
X7
X2
X30
X4
X82
EPCS64
LCD Module
LCD module X13
ALTERA
Cyclone IV
X36
GPIO
X7
X1
SD Card Socket
SDRAM X57
IRM
EXT CLK
SRAM X41
Flash X37
CLKOUT
CLOCK X3
50 MHz OSC
SDRAM1
SDRAM0
SRAM
Flash
7-segment displays X56
Push-buttons X4
LED X27
Slide Switch X18
Adjustable Voltage Signal

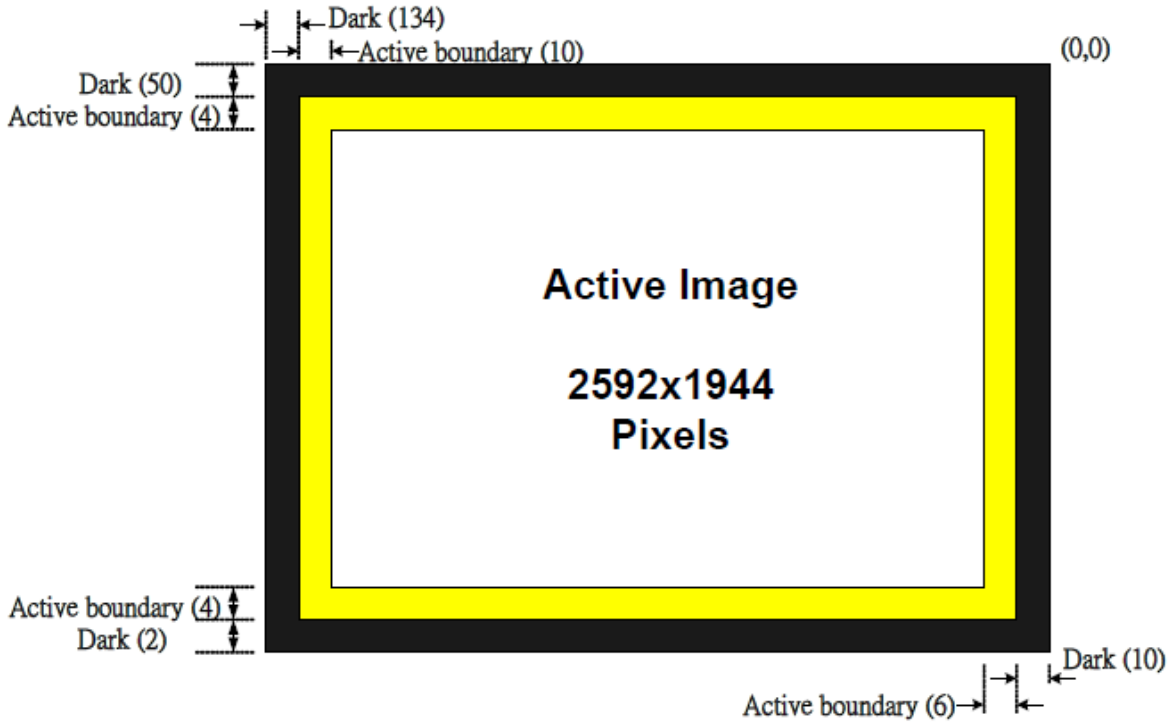


Figure 1.1 Pixel Array Description

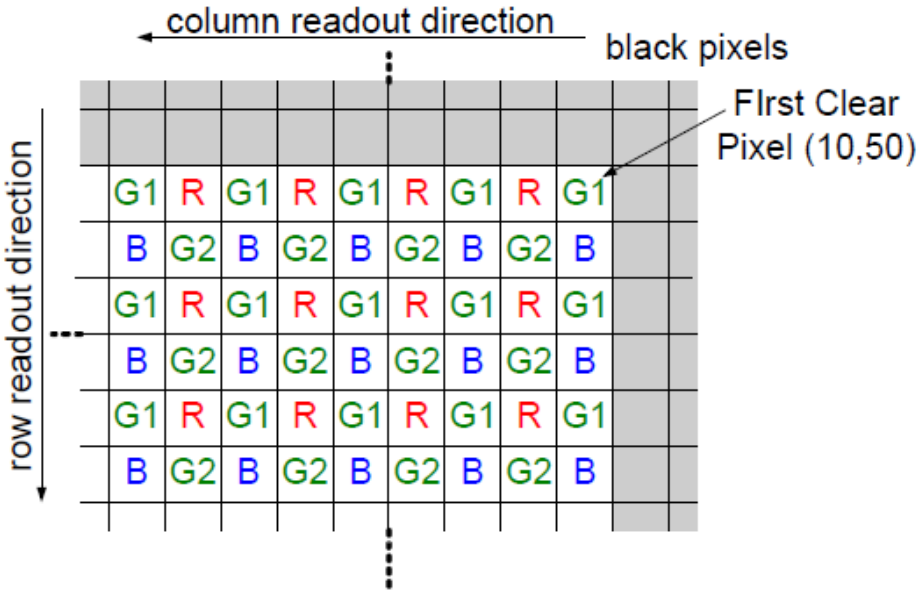


Figure 1.2 Pixel Color Pattern Detail (Top Right Corner)

Image Enhancement

- Can be added for better detection performance
- Luminance Enhancement
  - linear lighting correction
  - nonlinear lighting processing
  - Proposed enhancement uses nonlinear transfer function based on a local approach in HSV color space
- Contrast Enhancement
  - Histogram Equalization
  - Gaussian convolution in HSV color space

Noise Removal

- Low-pass filters and median filters are used most often for noise suppression or smoothing, while high-pass filters are typically used for image enhancement.

Low-Pass Filter

=> Remove High-Frequency Noise

Median Filter

=> A nonlinear process

- to reduce impulsive, or salt-and-pepper noise
- to preserve edges in an image

CWB HLS Flow of Design

(1) Describe ANSI-C and modify for HW.

[Steps]

A. Add input and output variables.

B. Modify descriptions which cannot be synthesized in HW.
e.g. 'recursive call', 'dynamic memory allocation', 'system call', etc.

C. Add bit width declaration (although not necessary)
The bit width declaration of inputs and outputs is recommended to achieve smaller area designs. Other internal variables are optimized automatically.

Behavioral Synthesis

(1) Specify clock frequency
(2) Specify scheduling mode
(3) Specify CWB libraries, FLIB and BLIB (Refer to the Section. 1-2-1)
(4) Synthesize the design

Verification

Verification in CWB

(1) Simulation based verification (behavioral level, cycle accurate level)
(2) Formal verification (Property checker, C-RTL equivalence prover)

Backup Pages for Q&A

```
12  module sobel (input_row_a00 ,input_row_a01 ,input_row_a02 ,output_row ,CLOCK ,RESET ;
13  input    [0:7]    input_row_a00 ;    // line#= ../../sobel.c:57
14  input    [0:7]    input_row_a01 ;    // line#= ../../sobel.c:57
15  input    [0:7]    input_row_a02 ;    // line#= ../../sobel.c:57
16  output   [0:7]    output_row ;       // line#= ../../sobel.c:58
17  input              CLOCK ;
18  input              RESET ;
```

```
43  #ifdef C
44
45  #include "stdio.h"
46  #include "stdlib.h"
47
48  unsigned char input_row[SIZE_BUFFER];
49  unsigned char output_row;
50
51  /* Global variables */
52  unsigned char line_buffer[SIZE_BUFFER][SIZE_BUFFER];
53
54  #else
55  /* Entity declaration : Inputs and outputs with their bitwidths */
56  //defined for CWB HW-C
57  in ter(0:8) input_row[SIZE_BUFFER];        //8-bit for each pixel-> 0-255
58  out ter(0:8) output_row;                   //8-bit for each pixel-> 0-255
59
60  /* Global variables */
61  var(0:8) line_buffer[SIZE_BUFFER][SIZE_BUFFER];
62  #endif
63  ////////////////////////////////////////////////////////////////
64
65  #ifdef C
66          void sobel(){
67  #else
68          process sobel(){        //defined for CWB HW-C
69  #endif
70  //local variables declaration
71    unsigned int X, Y;
72    int sumX, sumY;
73    int SUM, rowOffset, colOffset;
74
75    char Gx[3][3] ={{-1 ,-2 ,-1},
76                    { 0, 0, 0},
77                    { 1, 2, 1}};
78
79
80    char Gy[3][3] ={{-1, 0, 1},
81                    {-2, 0, 2},
82                    {-1, 0, 1}};
```

Synthesis Report By Quartus II

Flow Summary	
Flow Status	Successful - Fri Jan 09 00:12:45 2015
Quartus II 32-bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	DE2_115_CAMERA
Top-level Entity Name	DE2_115_CAMERA
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	2,255 / 114,480 (2 %)
Total combinational functions	1,912 / 114,480 (2 %)
Dedicated logic registers	1,376 / 114,480 (1 %)
Total registers	1376
Total pins	428 / 529 (81 %)
Total virtual pins	0
Total memory bits	74,656 / 3,981,312 (2 %)
Embedded Multiplier 9-bit elements	30 / 532 (6 %)
Total PLLs	1 / 4 (25 %)

Target Device

LEs Usage

Regs

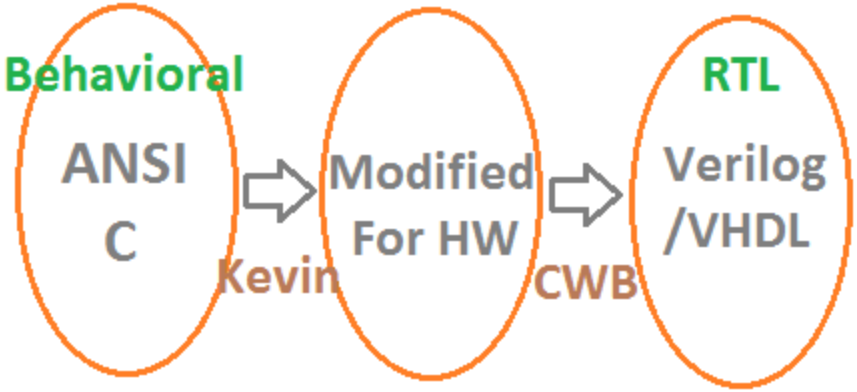
Mem

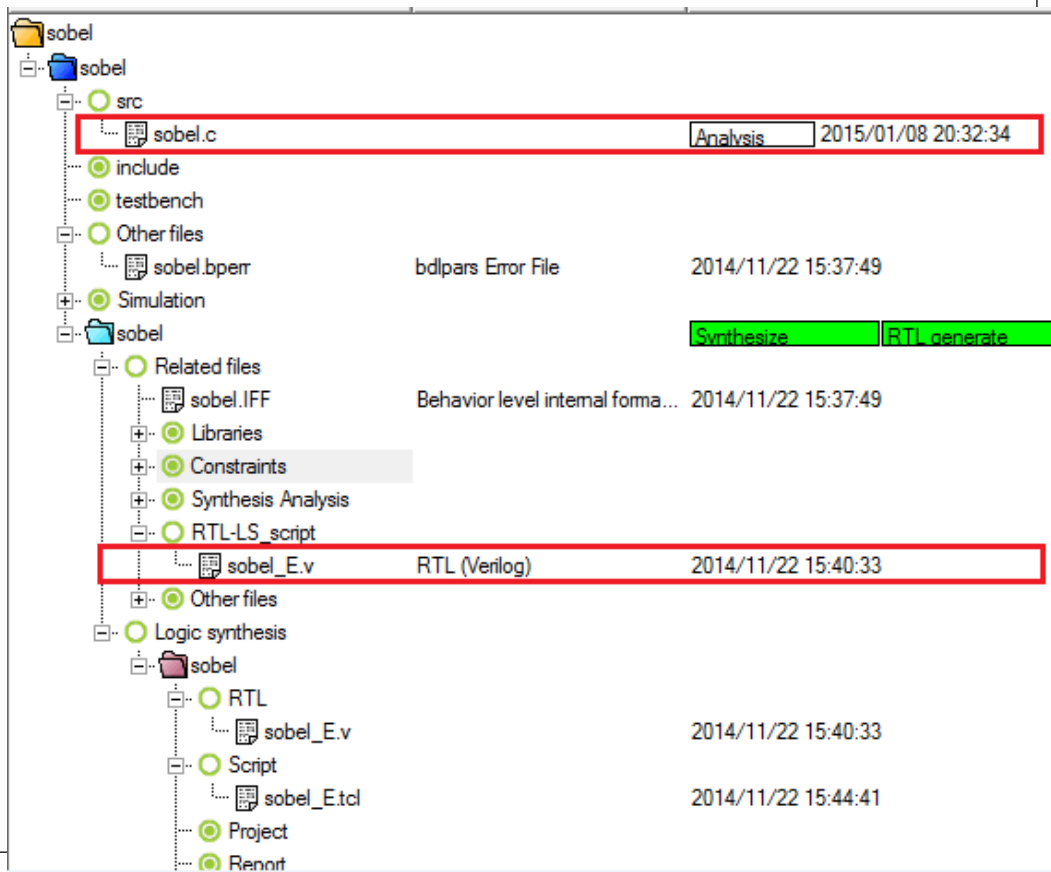
Altera IP or Megafunctions Used in The Project

Entity	IP Component Name	Version	IP File	Vendo
Line_Buffer1	Shift register (RAM-based)	9.1	v/Line_Buffer1.qip	Altera
sdram_pll	ALTPLL	9.1	v/sdram_pll.qip	Altera
LineBuffer_3	Shift register (RAM-based)	13.0	LineBuffer_3.qip	Altera
PA_3	PARALLEL_ADD	N/A	H:/Final Year Project/Kevin Codes/FPGA/DE2_115_CAMERA_Sobel/PA_3.v	Altera
SQRT	ALTSQRT	N/A	H:/Final Year Project/Kevin Codes/FPGA/DE2_115_CAMERA_Sobel/SQRT.v	Altera
MAC_3	ALTMULT_ACCUM (MAC)	13.0	MAC_3.qip	Altera
Sdram_RD_FIFO	FIFO	N/A	H:/Final Year Project/Kevin Codes/FPGA/DE2_115_CAMERA_Sobel/Sdram_Control/Sdram_RD_FIFO.v	Altera
Sdram_WR_FIFO	FIFO	N/A	H:/Final Year Project/Kevin Codes/FPGA/DE2_115_CAMERA_Sobel/Sdram_Control/Sdram_WR_FIFO.v	Altera
Line_Buffer3	Shift register (RAM-based)	13.0	Line_Buffer3.qip	N/A
LineBuffer3	Shift register (RAM-based)	13.0	LineBuffer3.qip	N/A

High-Level Synthesis By NEC CyberWorkBench (CWB)

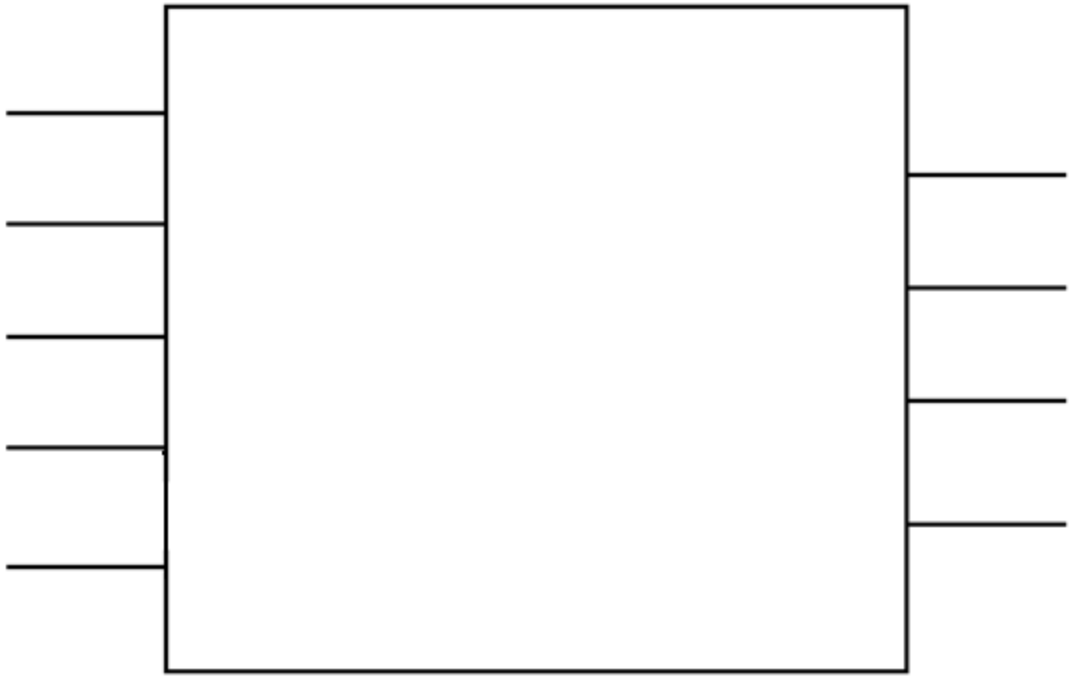
- From C to Verilog HDL
- Preliminary simulation & verification





Three steps to achieve breakthrough performance (From Xilinx)

- 1. Utilize the dedicated resources
  - Dedicated resources are faster than a LUT/Flip-Flop implementation and consume less power
  - Typically built with the CORE Generator tool and instantiated
  - DSP48E, FIFO, Block RAM, ISERDES, OSERDES, EMAC, and MGT, for example
- 2. Write the code for performance
  - Use synchronous design methodology
  - Ensure the code is written optimally for critical paths
  - Pipeline when necessary
- 3. Drive your synthesis tool
  - Try different optimization techniques
  - Add critical timing constraints in synthesis
  - Preserve hierarchy
  - Apply full and correct constraints
  - Use high effort



Post_Processing	18,606 / 114,480 (16 %)	18,520 / 114,480 (16 %)	6,865 / 114,480 (6 %)	6865	0 / 3,98 0 %