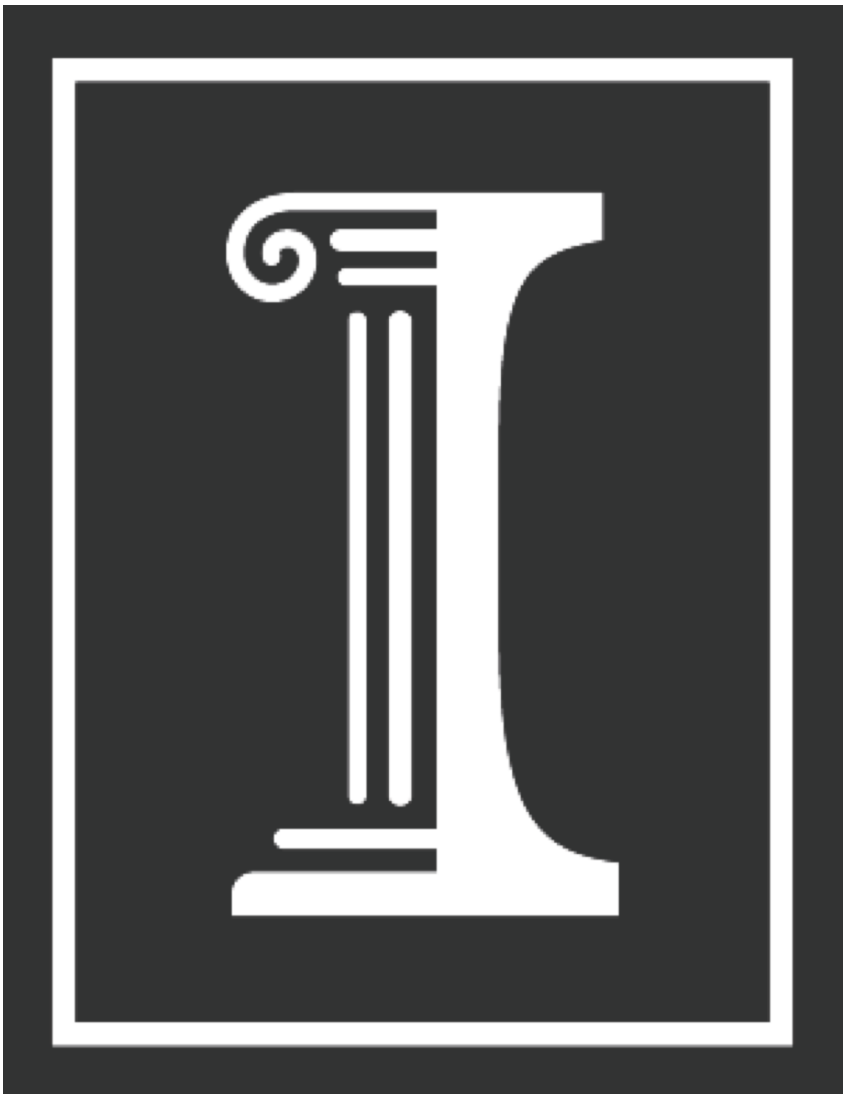
ECE385 Final Project

Simple Neural Network for Face Detection

Haichuan Xu, Yuan Ma

TA: Gene Shiue, Zhenhong Liu



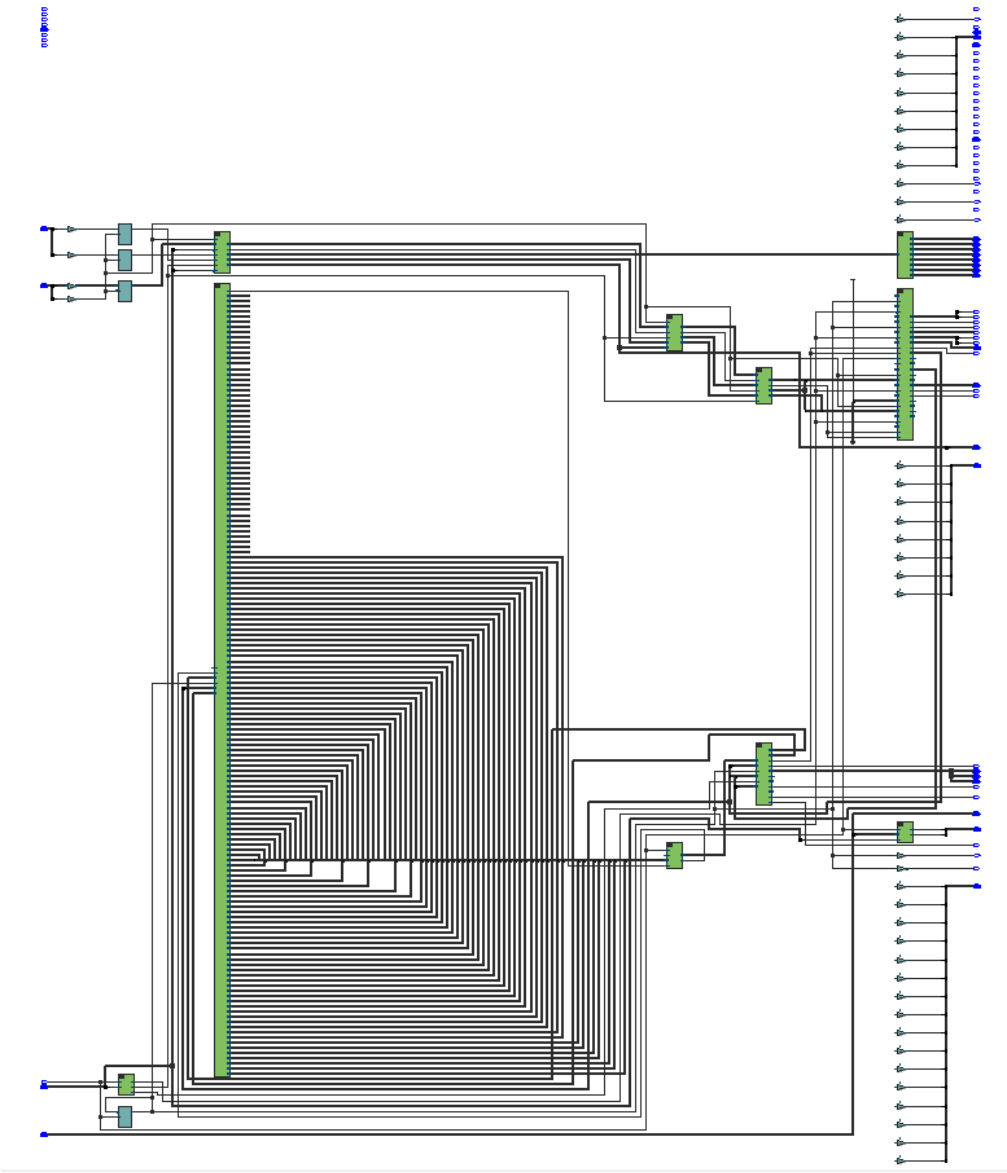
Introduction

This is the final project in the course ECE385. In this project, we plan to design a machine with one convolutional layer and can detect people's face. This machine should also track face and detect multiple faces in a fixed area, as well as some auxiliary functions such as change camera exposure time and picture capture.

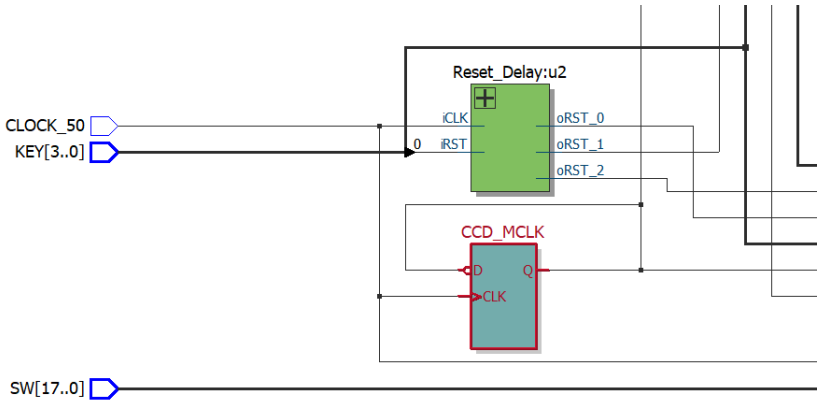
We use the camera TRDB-DC2 to capture the pictures. We plan to crop a middle rectangular section(150pixel * 300pixel) from the camera input to detect faces and we reduce the picture from RGB version to Black and White version. These two changes can reduce the convolutional computation, ensure system stability, and accelerate process by a great amount.

Our algorithm for face detection follows the basic convolutional-layer computation in neural network: we use a mask to cover the input image into pieces and each piece can be calculated with a confidence level. We then define a confidence threshold and declare there is a face in which section the confidence is greater than that confidence level.

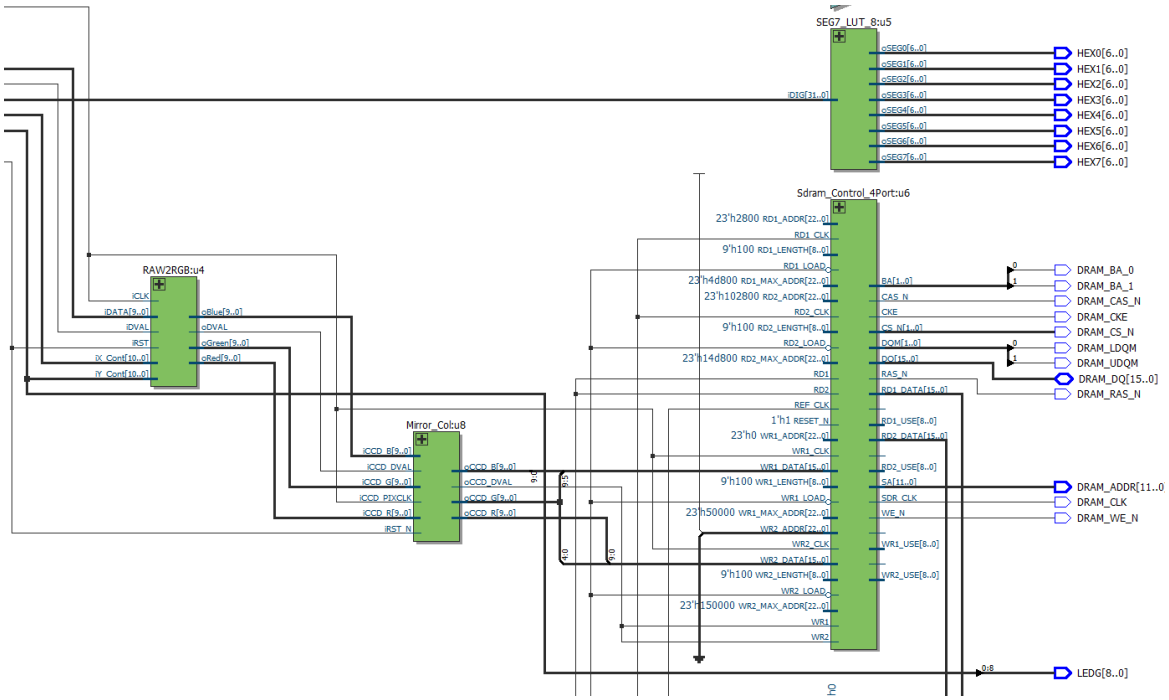
Top Level Diagram



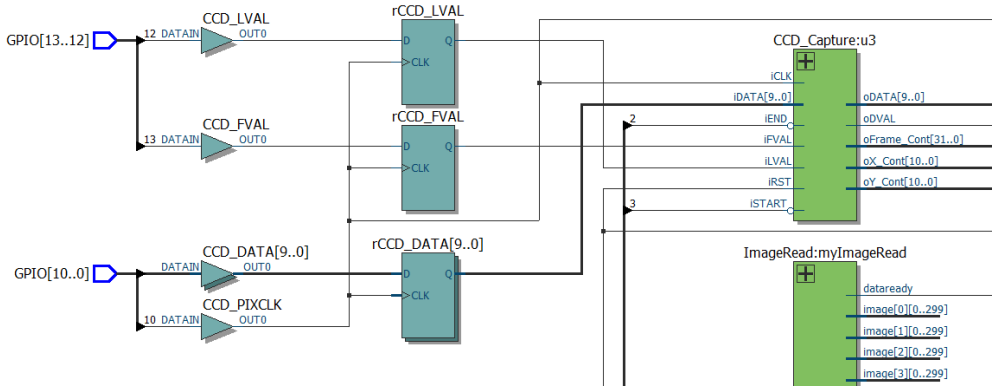
Peripheral



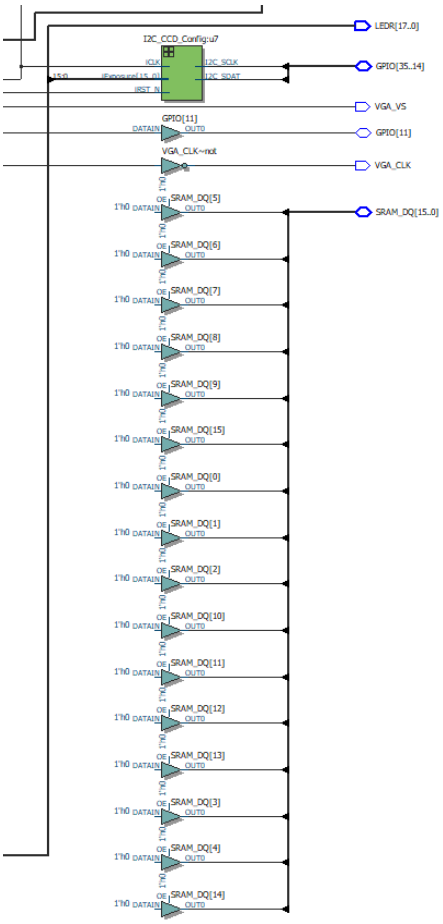
CCD Data Processing



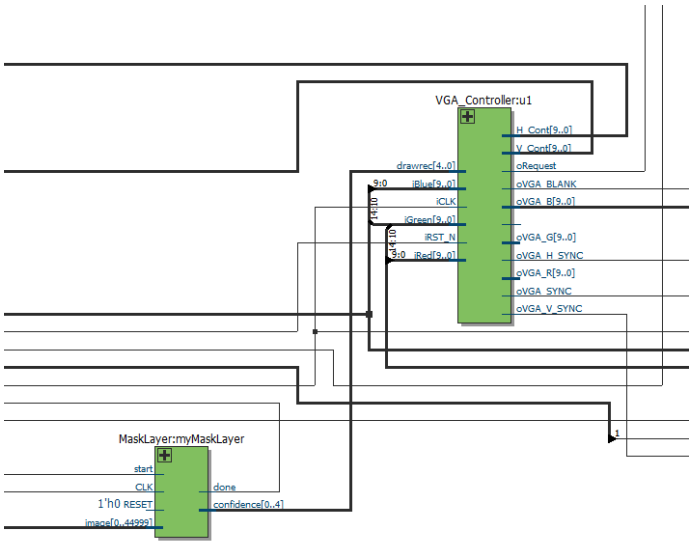
CCD Capture



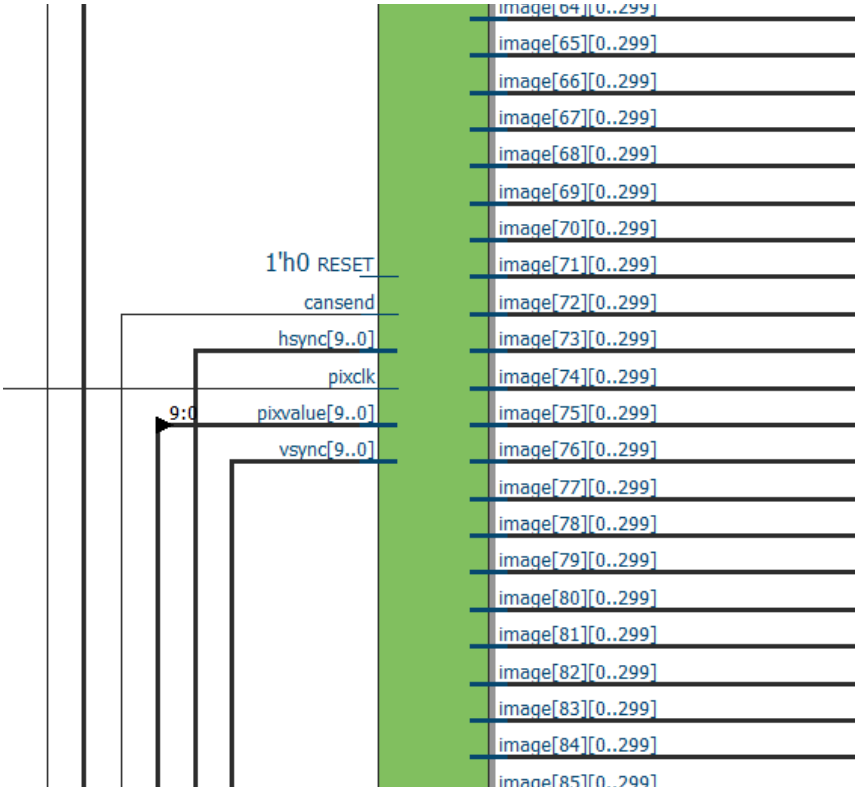
I2C_CCD_Interface



Mask Computation and VGA Controller



On-Chip Register Array



Written Description of Major Components

Written Description of Camera Input

In this project, we use the code offered from the Altera official website. To adapt the code for DE-2 115 board, we first make sure that all the pins are correctly connected. Notice that in the RAW2RGB module, we adapt 36 GPIO pins to 3 10-bit-long RGB signals and in Mirror_Col module, we reflex the image read from the camera from left to right. I2C_CCD and I2C_Controller establish the proper I2C interface to connect the TRDB-DC2 camera with the DE2-115 board. The CCD_Capture module aims to read KEY input on the board and control the SDRAM either to output the updated information or keep the original information.

Memory Implemented in Design

We primarily used two types of memory in the design, the SDRAM and on-chip memory. The SDRAM is used to capture a particular frame for further inspection, and the on-chip memory is used to store the middle 300*150 frame data for further processing by the computation core in real time.

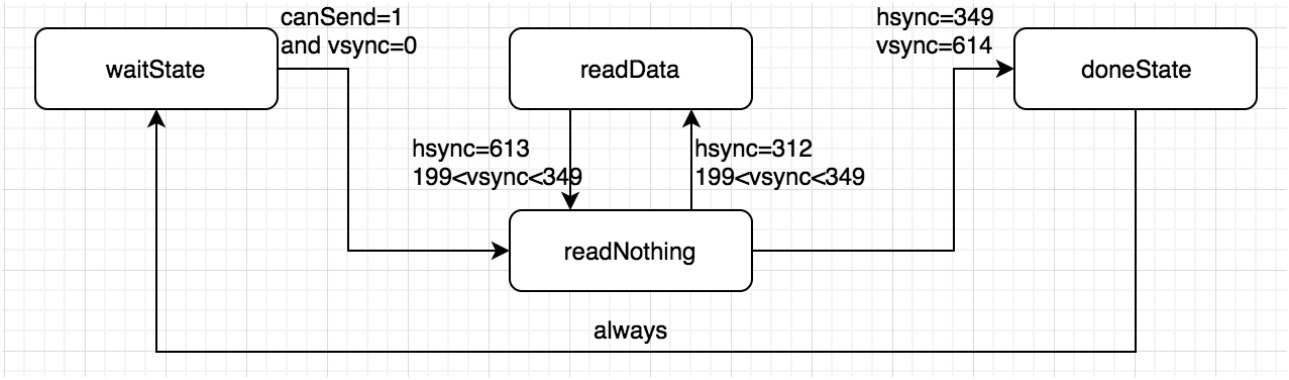
SDRAM

The major strength of SDRAM is its large storing capacity. Since the full RGB channel image data of a whole frame is relatively large, and capture performance (time delay) is not required, we choose to use SDRAM to store the whole frame full RGB image data. The image data captured by the CCD camera first goes into the RAW2RGB module to convert to RGB

data, then goes into the mirror_column module to mirror/reverse the image data, then the data goes in the SDram_control_4Port module to store into SDRAM. The data is finally read by the VGA_controller upon request.

On-chip Register Array

On the other hand, since we want to process and display the processed data as soon as possible after the frame input of the CCD camera, the performance of computation core is extremely important. We chose to use on-chip memory to store the data required for processing. In order to speed up the processing, we compressed the RGB data to black and white data and stored it in an on-chip register array and pushed it to the computational core. Notice that to update this register array and only read from the middle 300*150 section from the input image, a proper finite state machine is needed as below.



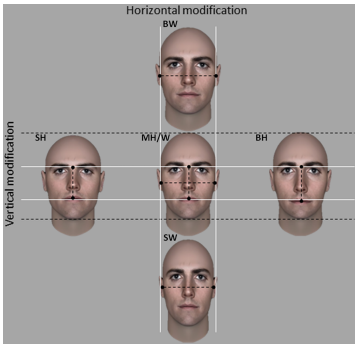
Note that the canSend signal denotes computation core process has completed and request for new image, hsync and vsync signals denotes pixel position passed in from the VGA controller, and dataReady signal(not included in the above) denotes that the middle 150*300 section is stored successfully. Also note that to cooperate well with the finite state machine in computation core, doneState is automatically transfer to waitState, and the dataReady signal is active only during doneState.

With these two types of memory working together, we are able to both store large amount of full image data and speed up the computational processing core.

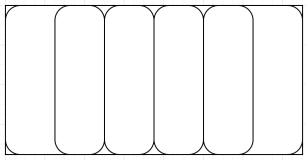
Written description of the convolutional layer

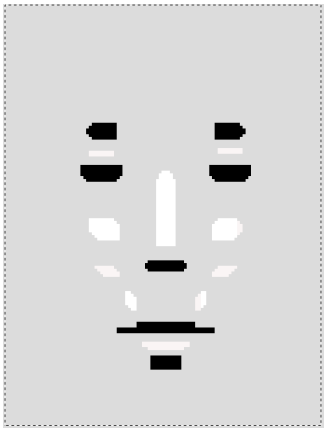
We design to build one layer of convolutional computation and hardcode all the weights and mask. This means we need to use a mask that can look and recognize a face by one step mask convolution. A suited mask, thus, is the crucial problem we need first to solve.

Mask Generation, Weight and Confidence Threshold

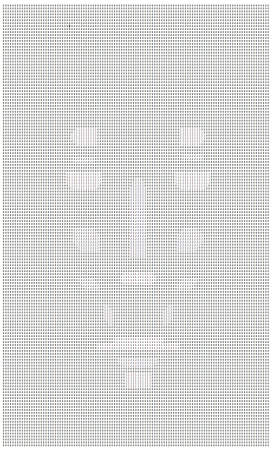


After research, we find a proper way to draw a promising mask. As the picture shows on the left (*How components of facial width to height ratio differently contribute to the perception of social traits* Manuela Costa , Guillaume Lio, Alice Gomez, Angela Sirigu Published: February 24, 2017 https://doi.org/10.1371/journal.pone.0172739), we need to maintain some important ratio in our mask, such as the ratio of face width to the distance from the eye level to the mouth level. Also, we notice that the cheek and the nose always reflect lights, yet eye blow and mouth absorb reflections. We then draw our mask as below with the size of 113 * 150. And we can cover the input image(10*300) well by declaring five mask calculation, as show below.



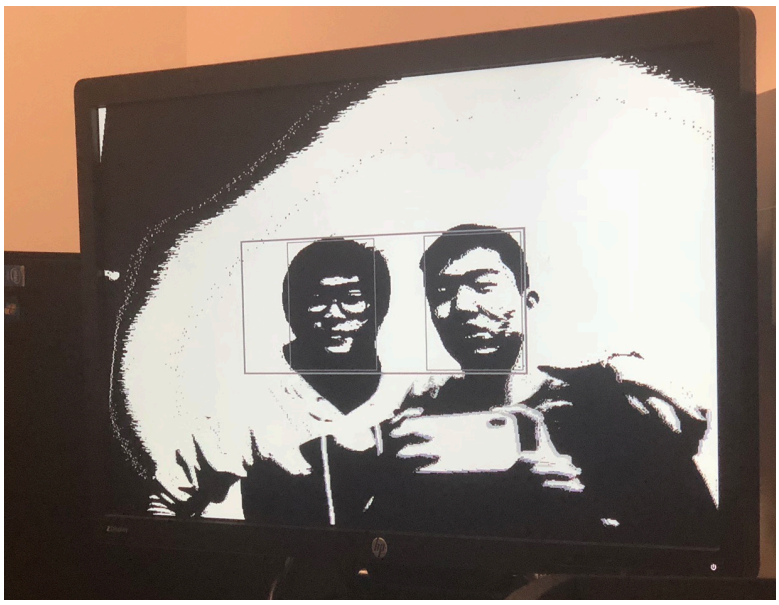


We decided to assign zero weight for the grey area of the mask and 1 for the black region if that input pixel is also black and 1 for the white region if the input pixel is also white. To make this computation happen, we first convert this mask to number array. On the right is the number array in two dimention version. We temporarily assign the white pixel as 2, black pixel as 1, else zero. In this way, we are able to extract all the pixel position of black region in the mask, as well as the white region in the mask. This makes each confidence split into two kinds: confidence of black region, and confidence of white region. In the real time computation, given the input image is already converted to binary level and 1 represents white and 0 represent black, we add all the pixel value at white region to calculate the confidence of white, add the inverse value of all the pixel at black region to calculate the confidence of black. After tens of trials, we found that when both the black confidence and the white confidence are above 250, this machine reaches its best performance. Thus we set both black and white confidence threshold to 250.



Parallel Computation

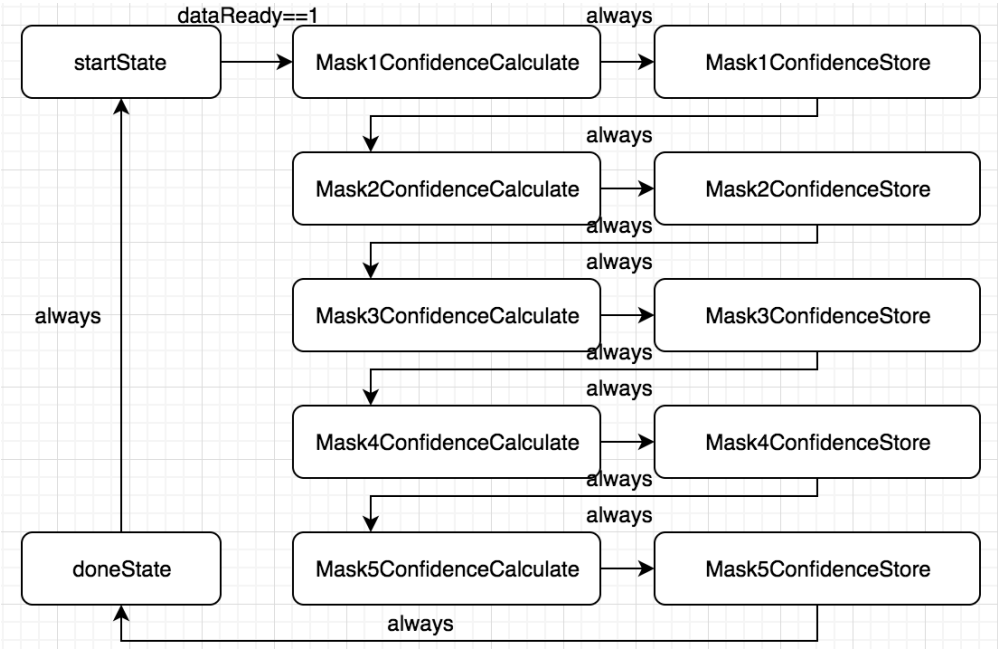
To accelerate the calculation of the confidence, we decide to utilize FPGA's parallel computation advantage. As stated in description of the on-chip register array, we are able to transmit the whole black and white picture(150*300) to the computation module in one clock cycle by a huge amount a parallel data transmission. This allows us to accomplish one step mask convolution in one clock cycle. These great amount of parallel computing also makes the face tracking and the multiple face detection possible. When a face is in the middle rectangular region, this machine can recognize it nicely, and when the face is moving, the confidence outcome of both white and black changes accordingly and only one monitor frame clock cycle delay. A face tracking can thus happen. Same



logic applies for multiple face detection. A demo picture for multiple recognition is provided above.

Finite State Machine

Unlike the VGA_controller module, the convolutional computation module uses the 50MHz clock. Notice that this finite state machine should cooperate well with the finite state machine which updates the on-chip register array. To



accomplish this, the doneState should transfer to startState automatically. Also, in both doneState and startState, the output doneSignal, here canSend signal, should be active. Only in this case two finite state machine can work properly.

VGA Output Description

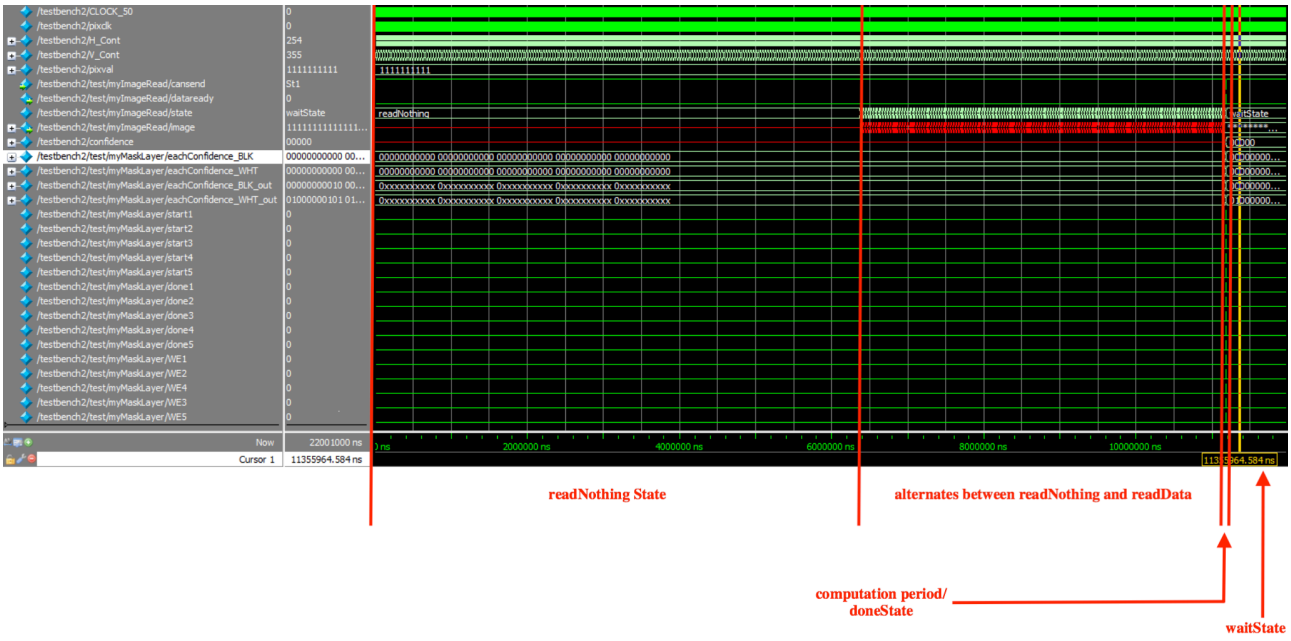
The VGA output is controlled by the VGA_Controller module. We designed the module to achieve display of two major types of information. One is the image frame data read in by the CCD camera and stored in the SDRAM, the other is the rectangular marker that shows where faces are located in the 300*150 region in the middle of the screen.

Since the data taken in by the computational modules are purely binary (black and white) and we want the VGA display to reflect the information processed by the computational core, we modified the original outputs to display black and white pixels by setting all color channels the same with black represented by 37.5% greyscale or lower and white by 37.5% greyscale or higher.

In order to display the rectangular markers, we enumerated all five possible frame positions and the position of the outer frame using the H_Cont and V_Cont signal listed in the original module. With a five bit input [0:4] drawrec that is processed as the final output of the computational core, the final output oVGA_R, oVGA_G and oVGA_B will make a choice between displaying the black and white data processed from the SDRAM or the rectangular markers based on the condition checks using “drawrec” signal.

The two type of data are processed simultaneously and in parallel, with a purely combinational check to determine the total current frame output. Therefore, the actual rectangular markers has a short delay with respect to the current frame.

Simulation



Conclusion

The functionality meets our expectation. This machine can successfully continuously track the face movement and detects multiple faces within the middle rectangle region. Also the convolutional computing speed is fast due to large number of parallel programming. After complete the project, we realize that improvements can be done to promote the accuracy of the detection. Also we realize that the hardcoded mask and weight sometimes is not good as expectation: a well trained convolutional neutral network can provide robust detection performance.

Through this semester's learning, ECE385 completely changes our understanding of hardware design. This project demonstrates our ability to integrate design and program in system verilog to create a fast, purely hardware processor.

Module Description

Module:	VGA_Controller.sv
Input:	iCLK, iRST_N, [4:0] drawrec, [9:0] ired, [9:0] iGreen, [9:0] iblue
Output:	oRequest, [9:0] oVGA_R, [9:0] oVGA_G, [9:0] oVGA_B, oVGA_H_SYNC, oVGA_V_SYNC, oVGA_SYNC, oVGA_BLANK, oVGA_CLOCK, [9:0] H_Cont, [9:0] V_Cont
Description:	The module takes in data from computational core and from mirror column to display frame on to the VGA monitor.
Purpose:	Interface of hardware and VGA monitor.

```
module      VGA_Controller(     //      Host Side
                                drawrec,            //5-bit input from
calculation module to determine which rectangle to draw
                                iRed,
                                iGreen,
                                iBlue,
                                oRequest,
                                //      VGA Side
                                oVGA_R, // the outmost signals
                                oVGA_G,
                                oVGA_B,
                                oVGA_H_SYNC,
                                oVGA_V_SYNC,
                                oVGA_SYNC,
                                oVGA_BLANK,
                                oVGA_CLOCK,
                                //      Control Signal
                                iCLK,
```

```
                                    iRST_N,
                                    H_Cont, //output
                                    V_Cont,     //output

                                    // for debugging
                                    //image,
                                    //cansend
                                    );

`include "VGA_Param.h"

input       [4:0]   drawrec;                    //5-bit input from calculation
module to determine which rectangle to draw

//    Host Side
input       [9:0]   iRed;
input       [9:0]   iGreen;
input       [9:0]   iBlue;
output      reg             oRequest;
//    VGA Side
output      reg     [9:0]   oVGA_R;
output      reg     [9:0]   oVGA_G;
output      reg     [9:0]   oVGA_B;
output      reg             oVGA_H_SYNC;
output      reg             oVGA_V_SYNC;
output                      oVGA_SYNC;
output                      oVGA_BLANK;
output                      oVGA_CLOCK;
//    Control Signal
input               iCLK;
input               iRST_N;

//    Internal Registers and Wires
output      reg     [9:0]       H_Cont;         //changed to output to the
frameinput module
output      reg     [9:0]       V_Cont;         //changed to output to the
frameinput module

// for debuging
//input [0:44999] image;
//input cansend;

reg         [9:0]       Cur_Color_R;
reg         [9:0]       Cur_Color_G;
reg         [9:0]       Cur_Color_B;
```

```
wire                    mCursor_EN;
wire                    mRed_EN;
wire                    mGreen_EN;
wire                    mBlue_EN;

wire    [9:0]   internalR;      //added not-middle rectangle output
signal
wire    [9:0]   internalG;
wire    [9:0]   internalB;

wire    outerframe;                     //bool logic for the fixed frame and
five possible rectangles
wire    frame1;
wire    frame2;
wire    frame3;
wire    frame4;
wire    frame5;

assign outerframe       =       (((H_Cont==X_START+170 ||
H_Cont==X_START+171 || H_Cont==X_START+468 || H_Cont==X_START+469) &&
(V_Cont>=Y_START+165 && V_Cont<=Y_START+314)) || ((V_Cont==Y_START+165 ||
V_Cont==Y_START+166 || V_Cont==Y_START+313 || V_Cont==Y_START+314) &&
(H_Cont>=X_START+170 && H_Cont<=X_START+469)))
                                        ?       1       :       0;
assign frame1           =       (((H_Cont==X_START+174 ||
H_Cont==X_START+269) && (V_Cont>=Y_START+169 && V_Cont<=Y_START+310)) ||
((V_Cont==Y_START+169 || V_Cont==Y_START+310) && (H_Cont>=X_START+174 &&
H_Cont<=X_START+269)))
                                        ?       1       :       0;
assign frame3           =       (((H_Cont==X_START+272 ||
H_Cont==X_START+367) && (V_Cont>=Y_START+169 && V_Cont<=Y_START+310)) ||
((V_Cont==Y_START+169 || V_Cont==Y_START+310) && (H_Cont>=X_START+272 &&
H_Cont<=X_START+367)))
                                        ?       1       :       0;
assign frame5           =       (((H_Cont==X_START+370 ||
H_Cont==X_START+465) && (V_Cont>=Y_START+169 && V_Cont<=Y_START+310)) ||
((V_Cont==Y_START+169 || V_Cont==Y_START+310) && (H_Cont>=X_START+370 &&
H_Cont<=X_START+465)))
                                        ?       1       :       0;
assign frame2           =       (((H_Cont==X_START+223 ||
H_Cont==X_START+318) && (V_Cont>=Y_START+169 && V_Cont<=Y_START+310)) ||
((V_Cont==Y_START+169 || V_Cont==Y_START+310) && (H_Cont>=X_START+223 &&
H_Cont<=X_START+318)))
                                        ?       1       :       0;
```

```
assign frame4 = (((H_Cont==X_START+321 ||
H_Cont==X_START+416) && (V_Cont>=Y_START+169 && V_Cont<=Y_START+310)) ||
((V_Cont==Y_START+169 || V_Cont==Y_START+310) && (H_Cont>=X_START+321 &&
H_Cont<=X_START+416)))
                                                    ?  1  :  0;

assign oVGA_BLANK  =  oVGA_H_SYNC & oVGA_V_SYNC;
assign oVGA_SYNC   =  1'b0;
assign oVGA_CLOCK  =  iCLK;

assign internalR  =  (  H_Cont>=X_START  &&
H_Cont<X_START+H_SYNC_ACT &&
                        V_Cont>=Y_START  &&
V_Cont<Y_START+V_SYNC_ACT )
                        ?  (iBlue > 10'b0110000000 ? 10'b1111111111 :
10'b0) :  0;
assign internalG  =  (  H_Cont>=X_START  &&
H_Cont<X_START+H_SYNC_ACT &&
                        V_Cont>=Y_START  &&
V_Cont<Y_START+V_SYNC_ACT ) / /(iBlue > 10'b0110000000 ? 10'b1111111111 : 10'b0)
                        ?  (iBlue > 10'b0110000000 ? 10'b1111111111 :
10'b0) :  0;
assign internalB  =  (  H_Cont>=X_START  &&
H_Cont<X_START+H_SYNC_ACT &&
                        V_Cont>=Y_START  &&
V_Cont<Y_START+V_SYNC_ACT )
                        ?  (iBlue > 10'b0110000000 ? 10'b1111111111 :
10'b0) :  0;

always@( * )
begin


    case (drawrec)
        5'b10000:
            begin
            oVGA_R  =  (outerframe || frame1)
            ?  10'b0111111111  :  internalR;
            oVGA_G  =  (outerframe || frame1)
            ?  10'b0111111111  :  internalG;
            oVGA_B  =  (outerframe || frame1)
            ?  10'b0111111111  :  internalB;
            end
        5'b01000:
            begin
```

```
        oVGA_R      =       (outerframe || frame2)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2)
        ?       10'b0111111111      :       internalB;
        end
5'b00100:
        begin
        oVGA_R      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalB;
        end
5'b00010:
        begin
        oVGA_R      =       (outerframe || frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b00001:
        begin
        oVGA_R      =       (outerframe || frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame5)
        ?       10'b0111111111      :       internalB;
        end

5'b11000:
        begin
        oVGA_R      =       (outerframe || frame2)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2)
        ?       10'b0111111111      :       internalB;
        end
5'b10100:
```

```
        begin
        oVGA_R      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalB;
        end
5'b10010:
        begin
        oVGA_R      =       (outerframe || frame1 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b10001:
        begin
        oVGA_R      =       (outerframe || frame1 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b01100:
        begin
        oVGA_R      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalB;
        end
5'b01010:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b01001:
```

```
        begin
        oVGA_R      =       (outerframe || frame2 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b00110:
        begin
        oVGA_R      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame3)
        ?       10'b0111111111      :       internalB;
        end
5'b00101:
        begin
        oVGA_R      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b00011:
        begin
        oVGA_R      =       (outerframe || frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame4)
        ?       10'b0111111111      :       internalB;
        end

5'b11100:
        begin
        oVGA_R      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalB;
        end
```

```
5'b11010:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b11001:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b10110:
        begin
        oVGA_R      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 ||frame3)
        ?       10'b0111111111      :       internalB;
        end
5'b10101:
        begin
        oVGA_R      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b10011:
        begin
        oVGA_R      =       (outerframe || frame1 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
```

```
5'b01110:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b01101:
        begin
        oVGA_R      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b01011:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b00111:
        begin
        oVGA_R      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame3 ||frame5)
        ?       10'b0111111111      :       internalB;
        end

5'b11110:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
```

```
        end
5'b11101:
        begin
        oVGA_R      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b11011:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
        end
5'b10111:
        begin
        oVGA_R      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalB;
        end
5'b01111:
        begin
        oVGA_R      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame2 ||frame4)
        ?       10'b0111111111      :       internalB;
        end

5'b11111:
        begin
        oVGA_R      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalR;
        oVGA_G      =       (outerframe || frame1 || frame3 ||frame5)
        ?       10'b0111111111      :       internalG;
        oVGA_B      =       (outerframe || frame1 || frame3 ||frame5)
```

```
                    ?       10'b0111111111      :       internalB;
                    end

                default:
                    begin
                    oVGA_R      =       (outerframe)
                    ?       10'b0111111111      :       internalR;
                    oVGA_G      =       (outerframe)
                    ?       10'b0111111111      :       internalG;
                    oVGA_B      =       (outerframe)
                    ?       10'b0111111111      :       internalB;
                    end
        endcase


//      if (H_Cont>=X_START     && H_Cont<=X_START+300 && V_Cont>=Y_START
        && V_Cont<=Y_START+150 )
//              begin
//                  oVGA_R=image[(H_Cont-X_START)+300*(V_Cont-Y_START)] ?
10'b1111111111 : 10'b0;
//                  oVGA_G=image[(H_Cont-X_START)+300*(V_Cont-Y_START)] ?
10'b1111111111 : 10'b0;
//                  oVGA_B=image[(H_Cont-X_START)+300*(V_Cont-Y_START)] ?
10'b1111111111 : 10'b0;
//                  //oVGA_R=(!cansend) ? 10'b1111111111 : 10'b0;
//                  //oVGA_G=(!cansend) ? 10'b1111111111 : 10'b0;
//                  //oVGA_B=(!cansend) ? 10'b1111111111 : 10'b0;
//              end

//      if (H_Cont>=X_START     && H_Cont<=X_START+300 && V_Cont>=Y_START
        && V_Cont<=Y_START+150 )
//      begin
//      end
end




// assign   oVGA_R      =       (outerframe || frame1 || frame2 || frame3 || frame4
|| frame5)
//                                  ?       10'b1111111111      :       internalR;
// assign   oVGA_G      =       (outerframe || frame1 || frame2 || frame3 || frame4
|| frame5)
//                                  ?       10'b1111111111      :       internalR;
```

```
// assign oVGA_B = (outerframe || frame1 || frame2 || frame3 || frame4
|| frame5)
//                                  ?  10'b1111111111  :  internalR;

// assign oVGA_R = ((((H_Cont>=X_START+170 &&
H_Cont<X_START+175) || (H_Cont>=X_START+465 && H_Cont<X_START+470)) &&
(V_Cont>=Y_START+165 && V_Cont<Y_START+315)) || (((V_Cont>=Y_START+165 &&
V_Cont<Y_START+170) || (V_Cont>=Y_START+310 && V_Cont<Y_START+315)) &&
(H_Cont>=X_START+170 && H_Cont<X_START+470)))  // change the center
rectangle to pure white
//                                  ?  10'b1111111111  :  internalR;
// assign oVGA_G = ((((H_Cont>=X_START+170 &&
H_Cont<X_START+175) || (H_Cont>=X_START+465 && H_Cont<X_START+470)) &&
(V_Cont>=Y_START+165 && V_Cont<Y_START+315)) || (((V_Cont>=Y_START+165 &&
V_Cont<Y_START+170) || (V_Cont>=Y_START+310 && V_Cont<Y_START+315)) &&
(H_Cont>=X_START+170 && H_Cont<X_START+470)))
//                                  ?  10'b1111111111  :  internalG;
// assign oVGA_B = ((((H_Cont>=X_START+170 &&
H_Cont<X_START+175) || (H_Cont>=X_START+465 && H_Cont<X_START+470)) &&
(V_Cont>=Y_START+165 && V_Cont<Y_START+315)) || (((V_Cont>=Y_START+165 &&
V_Cont<Y_START+170) || (V_Cont>=Y_START+310 && V_Cont<Y_START+315)) &&
(H_Cont>=X_START+170 && H_Cont<X_START+470)))
//                                  ?  10'b1111111111  :  internalB;

// assign oVGA_R = (  H_Cont>=X_START  &&
H_Cont<X_START+H_SYNC_ACT &&
//                  V_Cont>=Y_START  &&
V_Cont<Y_START+V_SYNC_ACT )
//                                  ?  iRed  :  0;
// assign oVGA_G = (  H_Cont>=X_START  &&
H_Cont<X_START+H_SYNC_ACT &&
//                  V_Cont>=Y_START  &&
V_Cont<Y_START+V_SYNC_ACT )
//                                  ?  iGreen:  0;
// assign oVGA_B = (  H_Cont>=X_START  &&
H_Cont<X_START+H_SYNC_ACT &&
//                  V_Cont>=Y_START  &&
V_Cont<Y_START+V_SYNC_ACT )
//                                  ?  iBlue  :  0;


//    Pixel LUT Address Generator
always@(posedge iCLK or negedge iRST_N)
begin
    if(!iRST_N)
```

```
        oRequest    <=    0;
        else
        begin
            if(    H_Cont>=X_START-2 && H_Cont<X_START+H_SYNC_ACT-2 &&
                   V_Cont>=Y_START && V_Cont<Y_START+V_SYNC_ACT )
            oRequest    <=    1;
            else
            oRequest    <=    0;
        end
end

//    H_Sync Generator, Ref. 25.175 MHz Clock
always@(posedge iCLK or negedge iRST_N)
begin
        if(!iRST_N)
        begin
            H_Cont          <=    0;
            oVGA_H_SYNC     <=    0;
        end
        else
        begin
            //    H_Sync Counter
            if( H_Cont < H_SYNC_TOTAL )
            H_Cont      <=    H_Cont+1;
            else
            H_Cont      <=    0;
            //    H_Sync Generator
            if( H_Cont < H_SYNC_CYC )
            oVGA_H_SYNC     <=    0;
            else
            oVGA_H_SYNC     <=    1;
        end
end

//    V_Sync Generator, Ref. H_Sync
always@(posedge iCLK or negedge iRST_N)
begin
        if(!iRST_N)
        begin
            V_Cont          <=    0;
            oVGA_V_SYNC     <=    0;
        end
        else
        begin
            //    When H_Sync Re-start
```

```
            if(H_Cont==0)
            begin
                //  V_Sync Counter
                if( V_Cont < V_SYNC_TOTAL )
                V_Cont      <=  V_Cont+1;
                else
                V_Cont      <=  0;
                //  V_Sync Generator
                if(     V_Cont < V_SYNC_CYC )
                oVGA_V_SYNC     <=  0;
                else
                oVGA_V_SYNC     <=  1;
            end
        end
end

endmodule
```

```
Module:         DE2_CCD.v
Input:          CLOCK_27, CLOCK_50, EXT_CLOCK, [3:0]  KEY, [17:0]  SW,
                UART_RXD, IRDA_RXD, OTG_INT0, OTG_INT1, OTG_DREQ0,
                OTG_DREQ1, PS2_DAT, PS2_CLK, TDI, TCK, TCS,
Output:         [6:0]  HEX0,[6:0]  HEX1, [6:0]  HEX2, [6:0]  HEX3, [6:0]  HEX4,
                [6:0]  HEX5, [6:0]  HEX6, [6:0]  HEX7, [8:0]  LEDG, [17:0] LEDR,
                UART_TXD, IRDA_TXD, [11:0]  DRAM_ADDR, DRAM_LDQM,
                DRAM_WE_N, DRAM_CAS_N, DRAM_RAS_N, DRAM_CS_N,
                DRAM_BA_0, DRAM_BA_1, DRAM_CLK, DRAM_CKE, [21:0]
                FL_ADDR, FL_WE_N, FL_RST_N, FL_OE_N, FL_CE_N, [17:0]
                SRAM_ADDR, SRAM_UB_N, SRAM_LB_N, SRAM_WE_N,
                SRAM_CE_N, SRAM_OE_N, [1:0]       OTG_ADDR, OTG_CS_N,
                        OTG_RD_N, OTG_WR_N, OTG_RST_N, OTG_FSPEED,
                        OTG_LSPEED, OTG_DACK0_N, OTG_DACK1_N, LCD_ON,
                                LCD_BLON, LCD_RW, LCD_EN, LCD_RS, SD_CLK,
I2C_SCLK,                       TDO, VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK,
VGA_SYNC,                       [7:0]  VGA_R, [7:0] VGA_G, [7:0]VGA_B,
Inout:          [15:0]  DRAM_DQ, [7:0]  FL_DQ, [15:0]       SRAM_DQ, [15:0]
                OTG_DATA, [7:0]  LCD_DATA, SD_DAT, SD_DAT3, SD_CMD,
                I2C_SDAT, [35:0]  GPIO_0, [35:0]       GPIO
Description:    The top level entity of the design.
Purpose:        Interface and route signals in top level design.
```

```
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
// --------------------------------------------------------------------
//
//                     Terasic Technologies Inc
//                     356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                     HsinChu County, Taiwan
//                     302
//
//                     web: http://www.terasic.com/
//                     email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions:DE2 CMOS Camera Demo
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//   Ver  :| Author           :| Mod. Date :| Changes Made:
//   V1.0 :| Johnny Chen      :| 06/01/06  :|      Initial Revision
//   V1.1 :| Johnny Chen      :| 06/02/06  :|      Modify Image Quality
//   V1.2 :| Johnny Chen      :| 06/03/22  :|      Change Pin Assignment For New Sensor
//   V1.3 :| Johnny Chen      :| 06/03/22  :|      Fixed to Compatible with Quartus II 6.0
//
//   last modified Yuan Ma          :| 17/11/24  :|                adapt for the code De2-115
// --------------------------------------------------------------------

module DE2_CCD
```

```
        (
            //////////////////// 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
                                        // NOT FOUND
            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                  // NOT FOUND CHECK DRAM_DQM[0]
PIN_U2
```

```
        DRAM_UDQM,                          //  SDRAM High-
byte Data Mask                  // NOT FOUND CHECK DRAM_DQM[1]
PIN_W4
        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                       // NOT FOUND CHECK DRAM_BA[0]
PIN_U7
        DRAM_BA_1,                          //  SDRAM Bank
Address 0                       // NOT FOUND CHECK DRAM_BA[1]
PIN_R4
        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
                                        // NOT FOUND
        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
                        // NOT FOUND CHECK
//      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
                        // NOT FOUND
//      OTG_INT1,                           //  ISP1362 Interrupt 1
                        // NOT FOUND
//      OTG_DREQ0,                          //  ISP1362 DMA
Request 0                   // NOT FOUND
//      OTG_DREQ1,                          //  ISP1362 DMA
Request 1                   // NOT FOUND
//      OTG_DACK0_N,                        //  ISP1362 DMA
Acknowledge 0               // NOT FOUND CHECK OTG_DACK_N[0] PIN_C4
//      OTG_DACK1_N,                        //  ISP1362 DMA
Acknowledge 1               // NOT FOUND CHECK OTG_DACK_N[1] PIN_D4
        ////////////////////    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
                                    // NOT FOUND
//      SD_DAT3,                            //  SD Card Data 3
                                    // NOT FOUND
```

```
//          SD_CMD,                                         //      SD Card
Command Signal
//          SD_CLK,                                         //      SD Card Clock
            ////////////////////        USB JTAG
link    ////////////////////
//          TDI,                                    // CPLD -> FPGA (data in)
                        // NOT FOUND
//          TCK,                                    // CPLD -> FPGA (clk)
                        // NOT FOUND
//          TCS,                                    // CPLD -> FPGA (CS)
                                // NOT FOUND
//      TDO,                                        // FPGA -> CPLD (data out)
                        // NOT FOUND
            ////////////////////
I2C         ////////////////////////////
            I2C_SDAT,                               //      I2C Data
            I2C_SCLK,                               //      I2C Clock
            ////////////////////
PS2         ////////////////////////////
//          PS2_DAT,                                //      PS2 Data
                                            // NOT FOUND
//          PS2_CLK,                                //      PS2 Clock
                                        // NOT FOUND
            ////////////////////
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                                                 //NOT
FOUND
//          ENET_CMD,                               //      DM9000A Command/
Data Select, 0 = Command, 1 = Data      //NOT FOUND
//          ENET_CS_N,                              //      DM9000A Chip Select
                                                        //NOT
FOUND
//          ENET_WR_N,                              //      DM9000A Write
                                                                /
/NOT FOUND
```

```
//        ENET_RD_N,                                    //    DM9000A Read
                                                                    /
/NOT FOUND
//        ENET_RST_N,                                   //    DM9000A Reset
                                                                    /
/NOT FOUND
//        ENET_INT,                                 //    DM9000A Interrupt
                                                          //NOT
FOUND
//        ENET_CLK,                                 //    DM9000A Clock 25
MHz                                                       //NOT
FOUND
          ////////////////    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
                              // NOT FOUND
          ////////////////////
GPIO  ////////////////////////////
          GPIO_0,                                       //    GPIO
Connection 0                          // NOT FOUND
          GPIO                                          //    GPIO
Connection 1                          // NOT FOUND chang from GPIO_1 to
GPIO
     );

//////////////////////  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  [7:0]  VGA_R;          //  VGA Red[9:0]
                  // turn to 8 bits
```

```
output      [7:0]   VGA_G;                          //      VGA Green[9:0]
                                    // turn to 8 bits
output      [7:0]   VGA_B;                          //      VGA Blue[9:0]
                                    // turn to 8 bits
//////////////////      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
                // NOT FOUND
inout   [35:0]  GPIO;                           //      GPIO Connection 1
                // NOT FOUND chang from GPIO_1 to GPIO

assign LCD_ON       =       1'b1;
assign LCD_BLON     =       1'b1;
assign TD_RESET     =       1'b1;
```

```
//      All inout port turn to tri-state
assign FL_DQ            =       8'hzz;
assign SRAM_DQ          =       16'hzzzz;
assign OTG_DATA  =      16'hzzzz;
assign LCD_DATA  =      8'hzz;
//assign        SD_DAT          =       1'bz;
assign I2C_SDAT     =       1'bz;
//assign        ENET_DATA=      16'hzzzz;
assign AUD_ADCLRCK      =       1'bz;
assign AUD_DACLRCK      =       1'bz;
assign AUD_BCLK  =      1'bz;

//      CCD
wire    [9:0]   CCD_DATA;
wire            CCD_SDAT;
wire            CCD_SCLK;
wire            CCD_FLASH;
wire            CCD_FVAL;
wire            CCD_LVAL;
wire            CCD_PIXCLK;
reg                     CCD_MCLK;//      CCD Master Clock

wire    [15:0]  Read_DATA1;
wire    [15:0]  Read_DATA2;
wire            VGA_CTRL_CLK;
wire            AUD_CTRL_CLK;
wire    [9:0]   mCCD_DATA;
wire            mCCD_DVAL;
wire            mCCD_DVAL_d;
wire    [10:0]  X_Cont;
wire    [10:0]  Y_Cont;
wire    [9:0]   X_ADDR;
wire    [31:0]  Frame_Cont;
wire    [9:0]   mCCD_R;
wire    [9:0]   mCCD_G;
wire    [9:0]   mCCD_B;
wire            DLY_RST_0;
wire            DLY_RST_1;
wire            DLY_RST_2;
wire            Read;
reg             [9:0]   rCCD_DATA;
reg                     rCCD_LVAL;
reg                     rCCD_FVAL;
```

```
wire   [9:0]   sCCD_R;
wire   [9:0]   sCCD_G;
wire   [9:0]   sCCD_B;
wire           sCCD_DVAL;


// VGA R/G/B
wire [9:0] VGA_R9;
wire [9:0] VGA_G9;
wire [9:0] VGA_B9;
//modified
wire           cansend;                //added internal logic
wire           [9:0]   H_Cont;
wire           [9:0]   V_Cont;
wire           dataready;
wire           [0:44999] midrec;
wire           [4:0]   drawrec;


assign VGA_R =
{VGA_B9[9],VGA_B9[8],VGA_B9[7],VGA_B9[6],VGA_B9[5],VGA_B9[4],VGA_B9[3],VGA_
B9[2]} ;
assign VGA_G =
{VGA_B9[9],VGA_B9[8],VGA_B9[7],VGA_B9[6],VGA_B9[5],VGA_B9[4],VGA_B9[3],VGA_
B9[2]} ;
assign VGA_B =
{VGA_B9[9],VGA_B9[8],VGA_B9[7],VGA_B9[6],VGA_B9[5],VGA_B9[4],VGA_B9[3],VGA_
B9[2]} ;

//      For Sensor 1
assign CCD_DATA[0]      =       GPIO[0];
assign CCD_DATA[1]      =       GPIO[1];
assign CCD_DATA[2]      =       GPIO[5];
assign CCD_DATA[3]      =       GPIO[3];
assign CCD_DATA[4]      =       GPIO[2];
assign CCD_DATA[5]      =       GPIO[4];
assign CCD_DATA[6]      =       GPIO[6];
assign CCD_DATA[7]      =       GPIO[7];
assign CCD_DATA[8]      =       GPIO[8];
assign CCD_DATA[9]      =       GPIO[9];
assign GPIO[11]     =       CCD_MCLK;               // change from GPIO[11] to
GPIO[13]
//assign      GPIO_1[15]  =       CCD_SDAT;
//assign      GPIO_1[14]  =       CCD_SCLK;
assign CCD_FVAL  =      GPIO[13];           // change from GPIO[13] to GPIO[15]
```

```
assign CCD_LVAL   =     GPIO[12];           // change from GPIO[12] to GPIO[14]
assign CCD_PIXCLK       =     GPIO[10]; // change from GPIO[10] to GPIO[12]
//      For Sensor 2
/*
assign CCD_DATA[0]      =     GPIO_1[0+20];
assign CCD_DATA[1]      =     GPIO_1[1+20];
assign CCD_DATA[2]      =     GPIO_1[5+20];
assign CCD_DATA[3]      =     GPIO_1[3+20];
assign CCD_DATA[4]      =     GPIO_1[2+20];
assign CCD_DATA[5]      =     GPIO_1[4+20];
assign CCD_DATA[6]      =     GPIO_1[6+20];
assign CCD_DATA[7]      =     GPIO_1[7+20];
assign CCD_DATA[8]      =     GPIO_1[8+20];
assign CCD_DATA[9]      =     GPIO_1[9+20];
assign GPIO_1[11+20]    =     CCD_MCLK;
assign GPIO_1[15+20]    =     CCD_SDAT;
assign GPIO_1[14+20]    =     CCD_SCLK;
assign CCD_FVAL   =     GPIO_1[13+20];
assign CCD_LVAL   =     GPIO_1[12+20];
assign CCD_PIXCLK       =     GPIO_1[10+20];
*/
assign LEDR       =     SW;
assign LEDG       =     Y_Cont;
assign VGA_CTRL_CLK=  CCD_MCLK;
assign VGA_CLK          =     ~CCD_MCLK;

always@(posedge CLOCK_50)     CCD_MCLK <=     ~CCD_MCLK;

always@(posedge CCD_PIXCLK)
begin
        rCCD_DATA<=     CCD_DATA;
        rCCD_LVAL <=    CCD_LVAL;
        rCCD_FVAL <=    CCD_FVAL;
end
/*modified*/
ImageRead myImageRead(
                                                .pixclk(VGA_CTRL_CLK),
                                                .RESET(1'b0),
                                                .cansend(cansend),
                                                .hsync(H_Cont),
                                                .vsync(V_Cont),
                                                .pixvalue(Read_DATA1[9:0]), //
read blue data
                                                .dataready(dataready),
                                                .image(midrec)
```

```
                                                );

MaskLayer myMaskLayer(
                                                        .CLK(CLOCK_50),
                                                        .RESET(1'b0),
                                                        .image(midrec),
                                                        .start(dataready),
                                                        .done(cansend),
                                                        .confidence(drawrec)
                                                );
VGA_Controller          u1      (       //      Host Side
                                                .oRequest(Read),
                                                .iRed(Read_DATA2[9:0]),
                                                .iGreen({Read_DATA1[14:10],Read_DATA
2[14:10]}),
                                                .iBlue(Read_DATA1[9:0]),
                                                //      VGA Side
                                                .oVGA_R(VGA_R9),
                                                .oVGA_G(VGA_G9),
                                                .oVGA_B(VGA_B9),
                                                .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),
                                                .H_Cont(H_Cont),
                                                .V_Cont(V_Cont),
                                                .drawrec(drawrec),
                                                // for debugging
                                                //.image(midrec),
                                                //.cansend(cansend)
                                                );

/*modified*/
//VGA_Controller        u1      (       //      Host Side
//                                              .oRequest(Read),
//                                              .iRed(Read_DATA2[9:0]),
//                                              .iGreen({Read_DATA1[14:10],Read_DATA
2[14:10]}),
//                                              .iBlue(Read_DATA1[9:0]),
//                                              //      VGA Side
//                                              .oVGA_R(VGA_R9),
//                                              .oVGA_G(VGA_G9),
```

```
//                                      .oVGA_B(VGA_B9),
//                                      .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_1)   );

RAW2RGB                 u4      (       .oRed(mCCD_R),
                                        .oGreen(mCCD_G),
                                        .oBlue(mCCD_B),
                                        .oDVAL(mCCD_DVAL_d),
                                        .iX_Cont(X_Cont),
                                        .iY_Cont(Y_Cont),
                                        .iDATA(mCCD_DATA),
                                        .iDVAL(mCCD_DVAL),
                                        .iCLK(CCD_PIXCLK),
                                        .iRST(DLY_RST_1)   );

SEG7_LUT_8              u5      (       .oSEG0(HEX0),.oSEG1(HEX1),
                                        .oSEG2(HEX2),.oSEG3(HEX3),
                                        .oSEG4(HEX4),.oSEG5(HEX5),
                                        .oSEG6(HEX6),.oSEG7(HEX7),
                                        .iDIG(Frame_Cont) );
```

```
Sdram_Control_4Port	u6	(	//	HOST Side
                    .REF_CLK(CLOCK_50),
                    .RESET_N(1'b1),
                        //	FIFO Write Side 1
                    .WR1_DATA(	{sCCD_G[9:5],
                                    sCCD_B[9:0]}),
                        .WR1(sCCD_DVAL),
                        .WR1_ADDR(0),
                        .WR1_MAX_ADDR(640*512),
                        .WR1_LENGTH(9'h100),
                        .WR1_LOAD(!DLY_RST_0),
                        .WR1_CLK(CCD_PIXCLK),
                        //	FIFO Write Side 2
                    .WR2_DATA(	{sCCD_G[4:0],
                                    sCCD_R[9:0]}),
                        .WR2(sCCD_DVAL),
                        .WR2_ADDR(22'h100000),
                        .WR2_MAX_ADDR(22'h100000+640*512),
                        .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(640*16),
                        .RD1_MAX_ADDR(640*496),
                        .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+640*16),
                        .RD2_MAX_ADDR(22'h100000+640*496),
                        .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}),
                                .SDR_CLK(DRAM_CLK)   );

I2C_CCD_Config          u7   (      //    Host Side
                                .iCLK(CLOCK_50),
                                .iRST_N(KEY[1]),
                                .iExposure(SW[15:0]),
                                //       I2C Side
                                .I2C_SCLK(GPIO[14]), // change from
GPIO[14] to GPIO[16]
                                .I2C_SDAT(GPIO[15])      );      //
change from GPIO[15] to GPIO[17]

Mirror_Col              u8   (      //    Input Side
                                .iCCD_R(mCCD_R),
                                .iCCD_G(mCCD_G),
                                .iCCD_B(mCCD_B),
                                .iCCD_DVAL(mCCD_DVAL_d),
                                .iCCD_PIXCLK(CCD_PIXCLK),
                                .iRST_N(DLY_RST_1),
                                //       Output Side
                                .oCCD_R(sCCD_R),
                                .oCCD_G(sCCD_G),
                                .oCCD_B(sCCD_B),
                                .oCCD_DVAL(sCCD_DVAL));

endmodule
```

Module:	MaskOnePath.sv
Input:	[0:16949] image, CLK, RESET, start,
Output:	done, [9:0] confidence_BLK, [9:0] confidence_WHT
Description:	image processing module of one mask, receives data inside one mask and returns the confidence of both black and white.
Purpose:	used to calculate confidence inside a mask, basic component of MaskLayer module.

```
module MaskOnePath (
                                input logic [0:16949] image  ,
                                input logic CLK, RESET,
                                input logic start,          // active high
                                output logic done,          // active high
```

```
                                        output logic [9:0] confidence_BLK,
                                        output logic [9:0] confidence_WHT
                                    );
                                    enum logic [1:0] { waitState,

startAdd,

doneState
                                                                                    }
state, nextState;

        // basic flip flop
        always_ff @ (posedge CLK)
  begin
    if (RESET)
      state <= waitState;
    else
      state <= nextState;
  end


      always_comb
      begin
            // state transition
            nextState = state;
            unique case(state)
                  waitState :
                        if (start == 1'b1)
                              nextState = startAdd;
                  startAdd :
                        nextState = doneState;
                  doneState :
                        if (start == 1'b0)
                              nextState = waitState;
                  default:
                        nextState=waitState;
            endcase

            // when not to calculate, set confidence to zero
            confidence_BLK=10'b0;
            confidence_WHT=10'b0;

            done=1'b0;
            // state implementation
            unique case(state)
```

```
waitState : ; / / do nothing
startAdd:
begin
        confidence_WHT =
image[5839]+image[5840]+image[5841]+image[5842]+image[5843]+image[5844]+image[58
45]+image[5846]+image[5847]+image[5906]+image[5907]+image[5908]+image[5909]+imag
e[5910]+image[5911]+image[5912]+image[5913]+image[5914]+image[5952]+image[5953]+i
mage[5954]+image[5955]+image[5956]+image[5957]+image[5958]+image[5959]+image[596
0]+image[6019]+image[6020]+image[6021]+image[6022]+image[6023]+image[6024]+image
[6025]+image[6026]+image[6027]+image[6723]+image[6724]+image[6725]+image[6835]+i
mage[6836]+image[6837]+image[6838]+image[6839]+image[6948]+image[6949]+image[695
0]+image[6951]+image[6952]+image[7060]+image[7061]+image[7062]+image[7063]+image
[7064]+image[7065]+image[7066]+image[7173]+image[7174]+image[7175]+image[7176]+i
mage[7177]+image[7178]+image[7179]+image[7286]+image[7287]+image[7288]+image[728
9]+image[7290]+image[7291]+image[7292]+image[7399]+image[7400]+image[7401]+image
[7402]+image[7403]+image[7404]+image[7405]+image[7512]+image[7513]+image[7514]+i
mage[7515]+image[7516]+image[7517]+image[7518]+image[7625]+image[7626]+image[762
7]+image[7628]+image[7629]+image[7630]+image[7631]+image[7738]+image[7739]+image
[7740]+image[7741]+image[7742]+image[7743]+image[7744]+image[7851]+image[7852]+i
mage[7853]+image[7854]+image[7855]+image[7856]+image[7857]+image[7964]+image[796
5]+image[7966]+image[7967]+image[7968]+image[7969]+image[7970]+image[8077]+image
[8078]+image[8079]+image[8080]+image[8081]+image[8082]+image[8083]+image[8190]+i
mage[8191]+image[8192]+image[8193]+image[8194]+image[8195]+image[8196]+image[830
3]+image[8304]+image[8305]+image[8306]+image[8307]+image[8308]+image[8309]+image
[8416]+image[8417]+image[8418]+image[8419]+image[8420]+image[8421]+image[8422]+i
mage[8529]+image[8530]+image[8531]+image[8532]+image[8533]+image[8534]+image[853
5]+image[8618]+image[8619]+image[8620]+image[8621]+image[8622]+image[8623]+image
[8624]+image[8625]+image[8626]+image[8642]+image[8643]+image[8644]+image[8645]+i
mage[8646]+image[8647]+image[8648]+image[8664]+image[8665]+image[8666]+image[866
7]+image[8668]+image[8669]+image[8670]+image[8731]+image[8732]+image[8733]+image
[8734]+image[8735]+image[8736]+image[8737]+image[8738]+image[8739]+image[8740]+i
mage[8755]+image[8756]+image[8757]+image[8758]+image[8759]+image[8760]+image[876
1]+image[8776]+image[8777]+image[8778]+image[8779]+image[8780]+image[8781]+image
[8782]+image[8783]+image[8784]+image[8844]+image[8845]+image[8846]+image[8847]+i
mage[8848]+image[8849]+image[8850]+image[8851]+image[8852]+image[8853]+image[885
4]+image[8868]+image[8869]+image[8870]+image[8871]+image[8872]+image[8873]+image
[8874]+image[8888]+image[8889]+image[8890]+image[8891]+image[8892]+image[8893]+i
mage[8894]+image[8895]+image[8896]+image[8897]+image[8898]+image[8957]+image[895
8]+image[8959]+image[8960]+image[8961]+image[8962]+image[8963]+image[8964]+image
[8965]+image[8966]+image[8967]+image[8981]+image[8982]+image[8983]+image[8984]+i
mage[8985]+image[8986]+image[8987]+image[9001]+image[9002]+image[9003]+image[900
4]+image[9005]+image[9006]+image[9007]+image[9008]+image[9009]+image[9010]+image
[9011]+image[9070]+image[9071]+image[9072]+image[9073]+image[9074]+image[9075]+i
mage[9076]+image[9077]+image[9078]+image[9079]+image[9080]+image[9094]+image[909
5]+image[9096]+image[9097]+image[9098]+image[9099]+image[9100]+image[9114]+image
```

[9115]+image[9116]+image[9117]+image[9118]+image[9119]+image[9120]+image[9121]+im
age[9122]+image[9123]+image[9124]+image[9184]+image[9185]+image[9186]+image[9187]
+image[9188]+image[9189]+image[9190]+image[9191]+image[9192]+image[9193]+image[9
207]+image[9208]+image[9209]+image[9210]+image[9211]+image[9212]+image[9213]+ima
ge[9227]+image[9228]+image[9229]+image[9230]+image[9231]+image[9232]+image[9233]
+image[9234]+image[9235]+image[9236]+image[9298]+image[9299]+image[9300]+image[9
301]+image[9302]+image[9303]+image[9304]+image[9305]+image[9306]+image[9320]+ima
ge[9321]+image[9322]+image[9323]+image[9324]+image[9325]+image[9326]+image[9340]
+image[9341]+image[9342]+image[9343]+image[9344]+image[9345]+image[9346]+image[9
347]+image[9348]+image[9412]+image[9413]+image[9414]+image[9415]+image[9416]+ima
ge[9417]+image[9418]+image[9419]+image[9433]+image[9434]+image[9435]+image[9436]
+image[9437]+image[9438]+image[9439]+image[9453]+image[9454]+image[9455]+image[9
456]+image[9457]+image[9458]+image[9459]+image[9460]+image[9546]+image[9547]+ima
ge[9548]+image[9549]+image[9550]+image[9551]+image[9552]+image[9659]+image[9660]
+image[9661]+image[9662]+image[9663]+image[9664]+image[9665]+image[10541]+image[
10542]+image[10543]+image[10544]+image[10545]+image[10546]+image[10547]+image[10
585]+image[10586]+image[10587]+image[10588]+image[10589]+image[10590]+image[1059
1]+image[10655]+image[10656]+image[10657]+image[10658]+image[10659]+image[10660]
+image[10661]+image[10697]+image[10698]+image[10699]+image[10700]+image[10701]+i
mage[10702]+image[10703]+image[10769]+image[10770]+image[10771]+image[10772]+im
age[10773]+image[10774]+image[10775]+image[10809]+image[10810]+image[10811]+imag
e[10812]+image[10813]+image[10814]+image[10815]+image[10883]+image[10884]+image[
10885]+image[10886]+image[10887]+image[10888]+image[10922]+image[10923]+image[10
924]+image[10925]+image[10926]+image[10927]+image[11569]+image[11570]+image[1159
6]+image[11597]+image[11682]+image[11683]+image[11684]+image[11708]+image[11709]+
image[11710]+image[11795]+image[11796]+image[11797]+image[11798]+image[11820]+im
age[11821]+image[11822]+image[11823]+image[11908]+image[11909]+image[11910]+image
[11911]+image[11933]+image[11934]+image[11935]+image[11936]+image[12021]+image[12
022]+image[12023]+image[12024]+image[12046]+image[12047]+image[12048]+image[1204
9]+image[12135]+image[12136]+image[12137]+image[12159]+image[12160]+image[12161]
+image[12249]+image[12250]+image[12272]+image[12273]+image[13609]+image[13610]+i
mage[13611]+image[13612]+image[13613]+image[13614]+image[13615]+image[13616]+im
age[13617]+image[13618]+image[13619]+image[13620]+image[13621]+image[13622]+imag
e[13623]+image[13624]+image[13625]+image[13722]+image[13723]+image[13724]+image[
13725]+image[13726]+image[13727]+image[13728]+image[13729]+image[13730]+image[13
731]+image[13732]+image[13733]+image[13734]+image[13735]+image[13736]+image[1373
7]+image[13738]+image[13837]+image[13838]+image[13839]+image[13840]+image[13841]
+image[13842]+image[13843]+image[13844]+image[13845]+image[13846]+image[13847]+i
mage[13848]+image[13849];

confidence_BLK = (~image[4777])+(~image[4778])+
(~image[4779])+(~image[4780])+(~image[4781])+(~image[4782])+(~image[4783])+
(~image[4784])+(~image[4785])+(~image[4821])+(~image[4822])+(~image[4823])+
(~image[4824])+(~image[4825])+(~image[4826])+(~image[4827])+(~image[4828])+
(~image[4829])+(~image[4889])+(~image[4890])+(~image[4891])+(~image[4892])+
(~image[4893])+(~image[4894])+(~image[4895])+(~image[4896])+(~image[4897])+

(~image[4898])+(~image[4934])+(~image[4935])+(~image[4936])+(~image[4937])+
(~image[4938])+(~image[4939])+(~image[4940])+(~image[4941])+(~image[4942])+
(~image[4943])+(~image[5001])+(~image[5002])+(~image[5003])+(~image[5004])+
(~image[5005])+(~image[5006])+(~image[5007])+(~image[5008])+(~image[5009])+
(~image[5010])+(~image[5011])+(~image[5047])+(~image[5048])+(~image[5049])+
(~image[5050])+(~image[5051])+(~image[5052])+(~image[5053])+(~image[5054])+
(~image[5055])+(~image[5056])+(~image[5057])+(~image[5114])+(~image[5115])+
(~image[5116])+(~image[5117])+(~image[5118])+(~image[5119])+(~image[5120])+
(~image[5121])+(~image[5122])+(~image[5123])+(~image[5124])+(~image[5160])+
(~image[5161])+(~image[5162])+(~image[5163])+(~image[5164])+(~image[5165])+
(~image[5166])+(~image[5167])+(~image[5168])+(~image[5169])+(~image[5170])+
(~image[5228])+(~image[5229])+(~image[5230])+(~image[5231])+(~image[5232])+
(~image[5233])+(~image[5234])+(~image[5235])+(~image[5236])+(~image[5237])+
(~image[5273])+(~image[5274])+(~image[5275])+(~image[5276])+(~image[5277])+
(~image[5278])+(~image[5279])+(~image[5280])+(~image[5281])+(~image[5282])+
(~image[5342])+(~image[5343])+(~image[5344])+(~image[5345])+(~image[5346])+
(~image[5347])+(~image[5348])+(~image[5349])+(~image[5350])+(~image[5386])+
(~image[5387])+(~image[5388])+(~image[5389])+(~image[5390])+(~image[5391])+
(~image[5392])+(~image[5393])+(~image[5394])+(~image[6468])+(~image[6469])+
(~image[6470])+(~image[6471])+(~image[6472])+(~image[6473])+(~image[6474])+
(~image[6475])+(~image[6476])+(~image[6477])+(~image[6478])+(~image[6479])+
(~image[6480])+(~image[6481])+(~image[6482])+(~image[6514])+(~image[6515])+
(~image[6516])+(~image[6517])+(~image[6518])+(~image[6519])+(~image[6520])+
(~image[6521])+(~image[6522])+(~image[6523])+(~image[6524])+(~image[6525])+
(~image[6526])+(~image[6527])+(~image[6528])+(~image[6581])+(~image[6582])+
(~image[6583])+(~image[6584])+(~image[6585])+(~image[6586])+(~image[6587])+
(~image[6588])+(~image[6589])+(~image[6590])+(~image[6591])+(~image[6592])+
(~image[6593])+(~image[6594])+(~image[6595])+(~image[6627])+(~image[6628])+
(~image[6629])+(~image[6630])+(~image[6631])+(~image[6632])+(~image[6633])+
(~image[6634])+(~image[6635])+(~image[6636])+(~image[6637])+(~image[6638])+
(~image[6639])+(~image[6640])+(~image[6641])+(~image[6694])+(~image[6695])+
(~image[6696])+(~image[6697])+(~image[6698])+(~image[6699])+(~image[6700])+
(~image[6701])+(~image[6702])+(~image[6703])+(~image[6704])+(~image[6705])+
(~image[6706])+(~image[6707])+(~image[6708])+(~image[6740])+(~image[6741])+
(~image[6742])+(~image[6743])+(~image[6744])+(~image[6745])+(~image[6746])+
(~image[6747])+(~image[6748])+(~image[6749])+(~image[6750])+(~image[6751])+
(~image[6752])+(~image[6753])+(~image[6754])+(~image[6807])+(~image[6808])+
(~image[6809])+(~image[6810])+(~image[6811])+(~image[6812])+(~image[6813])+
(~image[6814])+(~image[6815])+(~image[6816])+(~image[6817])+(~image[6818])+
(~image[6819])+(~image[6820])+(~image[6821])+(~image[6853])+(~image[6854])+
(~image[6855])+(~image[6856])+(~image[6857])+(~image[6858])+(~image[6859])+
(~image[6860])+(~image[6861])+(~image[6862])+(~image[6863])+(~image[6864])+
(~image[6865])+(~image[6866])+(~image[6867])+(~image[6921])+(~image[6922])+
(~image[6923])+(~image[6924])+(~image[6925])+(~image[6926])+(~image[6927])+
(~image[6928])+(~image[6929])+(~image[6930])+(~image[6931])+(~image[6932])+

(~image[6933])+(~image[6967])+(~image[6968])+(~image[6969])+(~image[6970])+
(~image[6971])+(~image[6972])+(~image[6973])+(~image[6974])+(~image[6975])+
(~image[6976])+(~image[6977])+(~image[6978])+(~image[6979])+(~image[7035])+
(~image[7036])+(~image[7037])+(~image[7038])+(~image[7039])+(~image[7040])+
(~image[7041])+(~image[7042])+(~image[7043])+(~image[7044])+(~image[7045])+
(~image[7081])+(~image[7082])+(~image[7083])+(~image[7084])+(~image[7085])+
(~image[7086])+(~image[7087])+(~image[7088])+(~image[7089])+(~image[7090])+
(~image[7091])+(~image[10334])+(~image[10335])+(~image[10336])+(~image[10337])+
(~image[10338])+(~image[10339])+(~image[10340])+(~image[10341])+(~image[10342])+
(~image[10343])+(~image[10344])+(~image[10345])+(~image[10346])+(~image[10446])+
(~image[10447])+(~image[10448])+(~image[10449])+(~image[10450])+(~image[10451])+
(~image[10452])+(~image[10453])+(~image[10454])+(~image[10455])+(~image[10456])+
(~image[10457])+(~image[10458])+(~image[10459])+(~image[10460])+(~image[10559])+
(~image[10560])+(~image[10561])+(~image[10562])+(~image[10563])+(~image[10564])+
(~image[10565])+(~image[10566])+(~image[10567])+(~image[10568])+(~image[10569])+
(~image[10570])+(~image[10571])+(~image[10572])+(~image[10573])+(~image[10673])+
(~image[10674])+(~image[10675])+(~image[10676])+(~image[10677])+(~image[10678])+
(~image[10679])+(~image[10680])+(~image[10681])+(~image[10682])+(~image[10683])+
(~image[10684])+(~image[10685])+(~image[12816])+(~image[12817])+(~image[12818])+
(~image[12819])+(~image[12820])+(~image[12821])+(~image[12822])+(~image[12823])+
(~image[12824])+(~image[12825])+(~image[12826])+(~image[12827])+(~image[12828])+
(~image[12829])+(~image[12830])+(~image[12831])+(~image[12832])+(~image[12833])+
(~image[12834])+(~image[12835])+(~image[12836])+(~image[12929])+(~image[12930])+
(~image[12931])+(~image[12932])+(~image[12933])+(~image[12934])+(~image[12935])+
(~image[12936])+(~image[12937])+(~image[12938])+(~image[12939])+(~image[12940])+
(~image[12941])+(~image[12942])+(~image[12943])+(~image[12944])+(~image[12945])+
(~image[12946])+(~image[12947])+(~image[12948])+(~image[12949])+(~image[13035])+
(~image[13036])+(~image[13037])+(~image[13038])+(~image[13039])+(~image[13040])+
(~image[13041])+(~image[13042])+(~image[13043])+(~image[13044])+(~image[13045])+
(~image[13046])+(~image[13047])+(~image[13048])+(~image[13049])+(~image[13050])+
(~image[13051])+(~image[13052])+(~image[13053])+(~image[13054])+(~image[13055])+
(~image[13056])+(~image[13057])+(~image[13058])+(~image[13059])+(~image[13060])+
(~image[13061])+(~image[13062])+(~image[13063])+(~image[13064])+(~image[13065])+
(~image[13066])+(~image[13067])+(~image[13068])+(~image[13069])+(~image[13148])+
(~image[13149])+(~image[13150])+(~image[13151])+(~image[13152])+(~image[13153])+
(~image[13154])+(~image[13155])+(~image[13156])+(~image[13157])+(~image[13158])+
(~image[13159])+(~image[13160])+(~image[13161])+(~image[13162])+(~image[13163])+
(~image[13164])+(~image[13165])+(~image[13166])+(~image[13167])+(~image[13168])+
(~image[13169])+(~image[13170])+(~image[13171])+(~image[13172])+(~image[13173])+
(~image[13174])+(~image[13175])+(~image[13176])+(~image[13177])+(~image[13178])+
(~image[13179])+(~image[13180])+(~image[13181])+(~image[13182])+(~image[14177])+
(~image[14178])+(~image[14179])+(~image[14180])+(~image[14181])+(~image[14182])+
(~image[14183])+(~image[14184])+(~image[14185])+(~image[14186])+(~image[14187])+
(~image[14290])+(~image[14291])+(~image[14292])+(~image[14293])+(~image[14294])+
(~image[14295])+(~image[14296])+(~image[14297])+(~image[14298])+(~image[14299])+

(~image[14300])+(~image[14403])+(~image[14404])+(~image[14405])+(~image[14406])+
(~image[14407])+(~image[14408])+(~image[14409])+(~image[14410])+(~image[14411])+
(~image[14412])+(~image[14413])+(~image[14516])+(~image[14517])+(~image[14518])+
(~image[14519])+(~image[14520])+(~image[14521])+(~image[14522])+(~image[14523])+
(~image[14524])+(~image[14525])+(~image[14526])+(~image[14629])+(~image[14630])+
(~image[14631])+(~image[14632])+(~image[14633])+(~image[14634])+(~image[14635])+
(~image[14636])+(~image[14637])+(~image[14638])+(~image[14639]);

end

doneState:

begin

confidence_WHT =

image[5839]+image[5840]+image[5841]+image[5842]+image[5843]+image[5844]+image[58
45]+image[5846]+image[5847]+image[5906]+image[5907]+image[5908]+image[5909]+imag
e[5910]+image[5911]+image[5912]+image[5913]+image[5914]+image[5952]+image[5953]+i
mage[5954]+image[5955]+image[5956]+image[5957]+image[5958]+image[5959]+image[596
0]+image[6019]+image[6020]+image[6021]+image[6022]+image[6023]+image[6024]+image
[6025]+image[6026]+image[6027]+image[6723]+image[6724]+image[6725]+image[6835]+i
mage[6836]+image[6837]+image[6838]+image[6839]+image[6948]+image[6949]+image[695
0]+image[6951]+image[6952]+image[7060]+image[7061]+image[7062]+image[7063]+image
[7064]+image[7065]+image[7066]+image[7173]+image[7174]+image[7175]+image[7176]+i
mage[7177]+image[7178]+image[7179]+image[7286]+image[7287]+image[7288]+image[728
9]+image[7290]+image[7291]+image[7292]+image[7399]+image[7400]+image[7401]+image
[7402]+image[7403]+image[7404]+image[7405]+image[7512]+image[7513]+image[7514]+i
mage[7515]+image[7516]+image[7517]+image[7518]+image[7625]+image[7626]+image[762
7]+image[7628]+image[7629]+image[7630]+image[7631]+image[7738]+image[7739]+image
[7740]+image[7741]+image[7742]+image[7743]+image[7744]+image[7851]+image[7852]+i
mage[7853]+image[7854]+image[7855]+image[7856]+image[7857]+image[7964]+image[796
5]+image[7966]+image[7967]+image[7968]+image[7969]+image[7970]+image[8077]+image
[8078]+image[8079]+image[8080]+image[8081]+image[8082]+image[8083]+image[8190]+i
mage[8191]+image[8192]+image[8193]+image[8194]+image[8195]+image[8196]+image[830
3]+image[8304]+image[8305]+image[8306]+image[8307]+image[8308]+image[8309]+image
[8416]+image[8417]+image[8418]+image[8419]+image[8420]+image[8421]+image[8422]+i
mage[8529]+image[8530]+image[8531]+image[8532]+image[8533]+image[8534]+image[853
5]+image[8618]+image[8619]+image[8620]+image[8621]+image[8622]+image[8623]+image
[8624]+image[8625]+image[8626]+image[8642]+image[8643]+image[8644]+image[8645]+i
mage[8646]+image[8647]+image[8648]+image[8664]+image[8665]+image[8666]+image[866
7]+image[8668]+image[8669]+image[8670]+image[8731]+image[8732]+image[8733]+image
[8734]+image[8735]+image[8736]+image[8737]+image[8738]+image[8739]+image[8740]+i
mage[8755]+image[8756]+image[8757]+image[8758]+image[8759]+image[8760]+image[876
1]+image[8776]+image[8777]+image[8778]+image[8779]+image[8780]+image[8781]+image
[8782]+image[8783]+image[8784]+image[8844]+image[8845]+image[8846]+image[8847]+i
mage[8848]+image[8849]+image[8850]+image[8851]+image[8852]+image[8853]+image[885
4]+image[8868]+image[8869]+image[8870]+image[8871]+image[8872]+image[8873]+image
[8874]+image[8888]+image[8889]+image[8890]+image[8891]+image[8892]+image[8893]+i
mage[8894]+image[8895]+image[8896]+image[8897]+image[8898]+image[8957]+image[895

8]+image[8959]+image[8960]+image[8961]+image[8962]+image[8963]+image[8964]+image[8965]+image[8966]+image[8967]+image[8981]+image[8982]+image[8983]+image[8984]+image[8985]+image[8986]+image[8987]+image[9001]+image[9002]+image[9003]+image[9004]+image[9005]+image[9006]+image[9007]+image[9008]+image[9009]+image[9010]+image[9011]+image[9070]+image[9071]+image[9072]+image[9073]+image[9074]+image[9075]+image[9076]+image[9077]+image[9078]+image[9079]+image[9080]+image[9094]+image[9095]+image[9096]+image[9097]+image[9098]+image[9099]+image[9100]+image[9114]+image[9115]+image[9116]+image[9117]+image[9118]+image[9119]+image[9120]+image[9121]+image[9122]+image[9123]+image[9124]+image[9184]+image[9185]+image[9186]+image[9187]+image[9188]+image[9189]+image[9190]+image[9191]+image[9192]+image[9193]+image[9207]+image[9208]+image[9209]+image[9210]+image[9211]+image[9212]+image[9213]+image[9227]+image[9228]+image[9229]+image[9230]+image[9231]+image[9232]+image[9233]+image[9234]+image[9235]+image[9236]+image[9298]+image[9299]+image[9300]+image[9301]+image[9302]+image[9303]+image[9304]+image[9305]+image[9306]+image[9320]+image[9321]+image[9322]+image[9323]+image[9324]+image[9325]+image[9326]+image[9340]+image[9341]+image[9342]+image[9343]+image[9344]+image[9345]+image[9346]+image[9347]+image[9348]+image[9412]+image[9413]+image[9414]+image[9415]+image[9416]+image[9417]+image[9418]+image[9419]+image[9433]+image[9434]+image[9435]+image[9436]+image[9437]+image[9438]+image[9439]+image[9453]+image[9454]+image[9455]+image[9456]+image[9457]+image[9458]+image[9459]+image[9460]+image[9546]+image[9547]+image[9548]+image[9549]+image[9550]+image[9551]+image[9552]+image[9659]+image[9660]+image[9661]+image[9662]+image[9663]+image[9664]+image[9665]+image[10541]+image[10542]+image[10543]+image[10544]+image[10545]+image[10546]+image[10547]+image[10585]+image[10586]+image[10587]+image[10588]+image[10589]+image[10590]+image[10591]+image[10655]+image[10656]+image[10657]+image[10658]+image[10659]+image[10660]+image[10661]+image[10697]+image[10698]+image[10699]+image[10700]+image[10701]+image[10702]+image[10703]+image[10769]+image[10770]+image[10771]+image[10772]+image[10773]+image[10774]+image[10775]+image[10809]+image[10810]+image[10811]+image[10812]+image[10813]+image[10814]+image[10815]+image[10883]+image[10884]+image[10885]+image[10886]+image[10887]+image[10888]+image[10922]+image[10923]+image[10924]+image[10925]+image[10926]+image[10927]+image[11569]+image[11570]+image[11596]+image[11597]+image[11682]+image[11683]+image[11684]+image[11708]+image[11709]+image[11710]+image[11795]+image[11796]+image[11797]+image[11798]+image[11820]+image[11821]+image[11822]+image[11823]+image[11908]+image[11909]+image[11910]+image[11911]+image[11933]+image[11934]+image[11935]+image[11936]+image[12021]+image[12022]+image[12023]+image[12024]+image[12046]+image[12047]+image[12048]+image[12049]+image[12135]+image[12136]+image[12137]+image[12159]+image[12160]+image[12161]+image[12249]+image[12250]+image[12272]+image[12273]+image[13609]+image[13610]+image[13611]+image[13612]+image[13613]+image[13614]+image[13615]+image[13616]+image[13617]+image[13618]+image[13619]+image[13620]+image[13621]+image[13622]+image[13623]+image[13624]+image[13625]+image[13722]+image[13723]+image[13724]+image[13725]+image[13726]+image[13727]+image[13728]+image[13729]+image[13730]+image[13731]+image[13732]+image[13733]+image[13734]+image[13735]+image[13736]+image[13737]+image[13738]+image[13837]+image[13838]+image[13839]+image[13840]+image[13841]

+image[13842]+image[13843]+image[13844]+image[13845]+image[13846]+image[13847]+image[13848]+image[13849];

confidence_BLK = (~image[4777])+(~image[4778])+(~image[4779])+(~image[4780])+(~image[4781])+(~image[4782])+(~image[4783])+(~image[4784])+(~image[4785])+(~image[4821])+(~image[4822])+(~image[4823])+(~image[4824])+(~image[4825])+(~image[4826])+(~image[4827])+(~image[4828])+(~image[4829])+(~image[4889])+(~image[4890])+(~image[4891])+(~image[4892])+(~image[4893])+(~image[4894])+(~image[4895])+(~image[4896])+(~image[4897])+(~image[4898])+(~image[4934])+(~image[4935])+(~image[4936])+(~image[4937])+(~image[4938])+(~image[4939])+(~image[4940])+(~image[4941])+(~image[4942])+(~image[4943])+(~image[5001])+(~image[5002])+(~image[5003])+(~image[5004])+(~image[5005])+(~image[5006])+(~image[5007])+(~image[5008])+(~image[5009])+(~image[5010])+(~image[5011])+(~image[5047])+(~image[5048])+(~image[5049])+(~image[5050])+(~image[5051])+(~image[5052])+(~image[5053])+(~image[5054])+(~image[5055])+(~image[5056])+(~image[5057])+(~image[5114])+(~image[5115])+(~image[5116])+(~image[5117])+(~image[5118])+(~image[5119])+(~image[5120])+(~image[5121])+(~image[5122])+(~image[5123])+(~image[5124])+(~image[5160])+(~image[5161])+(~image[5162])+(~image[5163])+(~image[5164])+(~image[5165])+(~image[5166])+(~image[5167])+(~image[5168])+(~image[5169])+(~image[5170])+(~image[5228])+(~image[5229])+(~image[5230])+(~image[5231])+(~image[5232])+(~image[5233])+(~image[5234])+(~image[5235])+(~image[5236])+(~image[5237])+(~image[5273])+(~image[5274])+(~image[5275])+(~image[5276])+(~image[5277])+(~image[5278])+(~image[5279])+(~image[5280])+(~image[5281])+(~image[5282])+(~image[5342])+(~image[5343])+(~image[5344])+(~image[5345])+(~image[5346])+(~image[5347])+(~image[5348])+(~image[5349])+(~image[5350])+(~image[5386])+(~image[5387])+(~image[5388])+(~image[5389])+(~image[5390])+(~image[5391])+(~image[5392])+(~image[5393])+(~image[5394])+(~image[6468])+(~image[6469])+(~image[6470])+(~image[6471])+(~image[6472])+(~image[6473])+(~image[6474])+(~image[6475])+(~image[6476])+(~image[6477])+(~image[6478])+(~image[6479])+(~image[6480])+(~image[6481])+(~image[6482])+(~image[6514])+(~image[6515])+(~image[6516])+(~image[6517])+(~image[6518])+(~image[6519])+(~image[6520])+(~image[6521])+(~image[6522])+(~image[6523])+(~image[6524])+(~image[6525])+(~image[6526])+(~image[6527])+(~image[6528])+(~image[6581])+(~image[6582])+(~image[6583])+(~image[6584])+(~image[6585])+(~image[6586])+(~image[6587])+(~image[6588])+(~image[6589])+(~image[6590])+(~image[6591])+(~image[6592])+(~image[6593])+(~image[6594])+(~image[6595])+(~image[6627])+(~image[6628])+(~image[6629])+(~image[6630])+(~image[6631])+(~image[6632])+(~image[6633])+(~image[6634])+(~image[6635])+(~image[6636])+(~image[6637])+(~image[6638])+(~image[6639])+(~image[6640])+(~image[6641])+(~image[6694])+(~image[6695])+(~image[6696])+(~image[6697])+(~image[6698])+(~image[6699])+(~image[6700])+(~image[6701])+(~image[6702])+(~image[6703])+(~image[6704])+(~image[6705])+(~image[6706])+(~image[6707])+(~image[6708])+(~image[6740])+(~image[6741])+(~image[6742])+(~image[6743])+(~image[6744])+(~image[6745])+(~image[6746])+(~image[6747])+(~image[6748])+(~image[6749])+(~image[6750])+(~image[6751])+(~image[6752])+(~image[6753])+(~image[6754])+(~image[6807])+(~image[6808])+

(~image[6809])+(~image[6810])+(~image[6811])+(~image[6812])+(~image[6813])+
(~image[6814])+(~image[6815])+(~image[6816])+(~image[6817])+(~image[6818])+
(~image[6819])+(~image[6820])+(~image[6821])+(~image[6853])+(~image[6854])+
(~image[6855])+(~image[6856])+(~image[6857])+(~image[6858])+(~image[6859])+
(~image[6860])+(~image[6861])+(~image[6862])+(~image[6863])+(~image[6864])+
(~image[6865])+(~image[6866])+(~image[6867])+(~image[6921])+(~image[6922])+
(~image[6923])+(~image[6924])+(~image[6925])+(~image[6926])+(~image[6927])+
(~image[6928])+(~image[6929])+(~image[6930])+(~image[6931])+(~image[6932])+
(~image[6933])+(~image[6967])+(~image[6968])+(~image[6969])+(~image[6970])+
(~image[6971])+(~image[6972])+(~image[6973])+(~image[6974])+(~image[6975])+
(~image[6976])+(~image[6977])+(~image[6978])+(~image[6979])+(~image[7035])+
(~image[7036])+(~image[7037])+(~image[7038])+(~image[7039])+(~image[7040])+
(~image[7041])+(~image[7042])+(~image[7043])+(~image[7044])+(~image[7045])+
(~image[7081])+(~image[7082])+(~image[7083])+(~image[7084])+(~image[7085])+
(~image[7086])+(~image[7087])+(~image[7088])+(~image[7089])+(~image[7090])+
(~image[7091])+(~image[10334])+(~image[10335])+(~image[10336])+(~image[10337])+
(~image[10338])+(~image[10339])+(~image[10340])+(~image[10341])+(~image[10342])+
(~image[10343])+(~image[10344])+(~image[10345])+(~image[10346])+(~image[10446])+
(~image[10447])+(~image[10448])+(~image[10449])+(~image[10450])+(~image[10451])+
(~image[10452])+(~image[10453])+(~image[10454])+(~image[10455])+(~image[10456])+
(~image[10457])+(~image[10458])+(~image[10459])+(~image[10460])+(~image[10559])+
(~image[10560])+(~image[10561])+(~image[10562])+(~image[10563])+(~image[10564])+
(~image[10565])+(~image[10566])+(~image[10567])+(~image[10568])+(~image[10569])+
(~image[10570])+(~image[10571])+(~image[10572])+(~image[10573])+(~image[10673])+
(~image[10674])+(~image[10675])+(~image[10676])+(~image[10677])+(~image[10678])+
(~image[10679])+(~image[10680])+(~image[10681])+(~image[10682])+(~image[10683])+
(~image[10684])+(~image[10685])+(~image[12816])+(~image[12817])+(~image[12818])+
(~image[12819])+(~image[12820])+(~image[12821])+(~image[12822])+(~image[12823])+
(~image[12824])+(~image[12825])+(~image[12826])+(~image[12827])+(~image[12828])+
(~image[12829])+(~image[12830])+(~image[12831])+(~image[12832])+(~image[12833])+
(~image[12834])+(~image[12835])+(~image[12836])+(~image[12929])+(~image[12930])+
(~image[12931])+(~image[12932])+(~image[12933])+(~image[12934])+(~image[12935])+
(~image[12936])+(~image[12937])+(~image[12938])+(~image[12939])+(~image[12940])+
(~image[12941])+(~image[12942])+(~image[12943])+(~image[12944])+(~image[12945])+
(~image[12946])+(~image[12947])+(~image[12948])+(~image[12949])+(~image[13035])+
(~image[13036])+(~image[13037])+(~image[13038])+(~image[13039])+(~image[13040])+
(~image[13041])+(~image[13042])+(~image[13043])+(~image[13044])+(~image[13045])+
(~image[13046])+(~image[13047])+(~image[13048])+(~image[13049])+(~image[13050])+
(~image[13051])+(~image[13052])+(~image[13053])+(~image[13054])+(~image[13055])+
(~image[13056])+(~image[13057])+(~image[13058])+(~image[13059])+(~image[13060])+
(~image[13061])+(~image[13062])+(~image[13063])+(~image[13064])+(~image[13065])+
(~image[13066])+(~image[13067])+(~image[13068])+(~image[13069])+(~image[13148])+
(~image[13149])+(~image[13150])+(~image[13151])+(~image[13152])+(~image[13153])+
(~image[13154])+(~image[13155])+(~image[13156])+(~image[13157])+(~image[13158])+
(~image[13159])+(~image[13160])+(~image[13161])+(~image[13162])+(~image[13163])+

```
(~image[13164])+(~image[13165])+(~image[13166])+(~image[13167])+(~image[13168])+
(~image[13169])+(~image[13170])+(~image[13171])+(~image[13172])+(~image[13173])+
(~image[13174])+(~image[13175])+(~image[13176])+(~image[13177])+(~image[13178])+
(~image[13179])+(~image[13180])+(~image[13181])+(~image[13182])+(~image[14177])+
(~image[14178])+(~image[14179])+(~image[14180])+(~image[14181])+(~image[14182])+
(~image[14183])+(~image[14184])+(~image[14185])+(~image[14186])+(~image[14187])+
(~image[14290])+(~image[14291])+(~image[14292])+(~image[14293])+(~image[14294])+
(~image[14295])+(~image[14296])+(~image[14297])+(~image[14298])+(~image[14299])+
(~image[14300])+(~image[14403])+(~image[14404])+(~image[14405])+(~image[14406])+
(~image[14407])+(~image[14408])+(~image[14409])+(~image[14410])+(~image[14411])+
(~image[14412])+(~image[14413])+(~image[14516])+(~image[14517])+(~image[14518])+
(~image[14519])+(~image[14520])+(~image[14521])+(~image[14522])+(~image[14523])+
(~image[14524])+(~image[14525])+(~image[14526])+(~image[14629])+(~image[14630])+
(~image[14631])+(~image[14632])+(~image[14633])+(~image[14634])+(~image[14635])+
(~image[14636])+(~image[14637])+(~image[14638])+(~image[14639]);
                                done=1'b1;
                                end
                        default: ;
                endcase
        end


endmodule
```

```
Module:         MaskLayer.sv
Input:          CLK, RESET, [0:44999]image, start
Output:         done, [0:4]  confidence
Description:    image processing module of five masks, receives data of the whole
                middle rectangular region and returns the position of the face.
Purpose:        computational core of face detection, instantiates five
                MaskOnePath modules.

/*****************************************
@ brainstorm how to load each mask for calculation:

        1. stored in SRAM at the first place
        2. hard code in FPGA

*****************************************/

module MaskLayer(// control signals
                                        input logic CLK,
                                        input logic RESET,  // active high
                                                // input image
```

```
                                        input logic [0:44999]image ,
                                                // output image
                                        //notice the output pictures are stored in the
fixed range in the SRAM
                                                // process control signal
                                        input logic start,      // active high
                                                // process control output
                                        output logic done,  // active high
                                                // SRAM control signals
                                        // need modified to adapt for the SRAM
                                        output logic [0:4]  confidence
                                        );

                                        enum logic [4:0] {  startFirstMask,//5'h00

waitState,

doneState,

storeFirstConf,

startSecondMask,

storeSecondConf,

startThirdMask,

storeThirdConf,

startFourthMask,

storeFourthConf,

startFifthMask,

storeFifthConf
                                                                                        }
state, nextState;

                                        logic [10:0] eachConfidence_BLK [0:4];          //
each confidence register input
                                        logic [10:0] eachConfidence_WHT [0:4];
                                        logic [10:0] eachConfidence_BLK_out [0:4];
        // each confidence register output
                                        logic [10:0] eachConfidence_WHT_out [0:4];
```

```
                                        logic start1, start2, start3, start4, start5;// five
start signals to five mask paths
                                        logic done1, done2, done3, done4, done5;
        // five done signals to five mask paths
                                        logic WE1, WE2, WE3, WE4, WE5;
                        // five write enable signals to five confidence register
                                        assign confidence[0]=
(eachConfidence_BLK_out[0]>250 && eachConfidence_WHT_out[0]>250) ? 1'b1 :
1'b0;    // confidence bound is 700
                                        assign confidence[1]=
(eachConfidence_BLK_out[1]>250 && eachConfidence_WHT_out[1]>250) ? 1'b1 : 1'b0;
                                        assign confidence[2]=
(eachConfidence_BLK_out[2]>250 && eachConfidence_WHT_out[2]>250) ? 1'b1 : 1'b0;
                                        assign confidence[3]=
(eachConfidence_BLK_out[3]>250 && eachConfidence_WHT_out[3]>250) ? 1'b1 : 1'b0;
                                        assign confidence[4]=
(eachConfidence_BLK_out[4]>250 && eachConfidence_WHT_out[4]>250) ? 1'b1 : 1'b0;



        // module declaration
        MaskOnePath
Mask1( .CLK(CLK), .RESET(RESET), .start(start1), .done(done1), .image({image[0:112],ima
ge[300:412],image[600:712],image[900:1012],image[1200:1312],image[1500:1612],image[180
0:1912],image[2100:2212],image[2400:2512],image[2700:2812],image[3000:3112],image[3300:
3412],image[3600:3712],image[3900:4012],image[4200:4312],image[4500:4612],image[4800:4
912],image[5100:5212],image[5400:5512],image[5700:5812],image[6000:6112],image[6300:64
12],image[6600:6712],image[6900:7012],image[7200:7312],image[7500:7612],image[7800:791
2],image[8100:8212],image[8400:8512],image[8700:8812],image[9000:9112],image[9300:9412]
,image[9600:9712],image[9900:10012],image[10200:10312],image[10500:10612],image[10800:
10912],image[11100:11212],image[11400:11512],image[11700:11812],image[12000:12112],ima
ge[12300:12412],image[12600:12712],image[12900:13012],image[13200:13312],image[13500:1
3612],image[13800:13912],image[14100:14212],image[14400:14512],image[14700:14812],ima
ge[15000:15112],image[15300:15412],image[15600:15712],image[15900:16012],image[16200:1
6312],image[16500:16612],image[16800:16912],image[17100:17212],image[17400:17512],ima
ge[17700:17812],image[18000:18112],image[18300:18412],image[18600:18712],image[18900:1
9012],image[19200:19312],image[19500:19612],image[19800:19912],image[20100:20212],ima
ge[20400:20512],image[20700:20812],image[21000:21112],image[21300:21412],image[21600:2
1712],image[21900:22012],image[22200:22312],image[22500:22612],image[22800:22912],ima
ge[23100:23212],image[23400:23512],image[23700:23812],image[24000:24112],image[24300:2
4412],image[24600:24712],image[24900:25012],image[25200:25312],image[25500:25612],ima
ge[25800:25912],image[26100:26212],image[26400:26512],image[26700:26812],image[27000:2
7112],image[27300:27412],image[27600:27712],image[27900:28012],image[28200:28312],ima
ge[28500:28612],image[28800:28912],image[29100:29212],image[29400:29512],image[29700:2
9812],image[30000:30112],image[30300:30412],image[30600:30712],image[30900:31012],ima
```

ge[31200:31312],image[31500:31612],image[31800:31912],image[32100:32212],image[32400:32512],image[32700:32812],image[33000:33112],image[33300:33412],image[33600:33712],image[33900:34012],image[34200:34312],image[34500:34612],image[34800:34912],image[35100:35212],image[35400:35512],image[35700:35812],image[36000:36112],image[36300:36412],image[36600:36712],image[36900:37012],image[37200:37312],image[37500:37612],image[37800:37912],image[38100:38212],image[38400:38512],image[38700:38812],image[39000:39112],image[39300:39412],image[39600:39712],image[39900:40012],image[40200:40312],image[40500:40612],image[40800:40912],image[41100:41212],image[41400:41512],image[41700:41812],image[42000:42112],image[42300:42412],image[42600:42712],image[42900:43012],image[43200:43312],image[43500:43612],image[43800:43912],image[44100:44212],image[44400:44512],image[44700:44812]}), .confidence_BLK(eachConfidence_BLK[0]) , .confidence_WHT(eachConfidence_WHT[0]));

MaskOnePath Mask2(.CLK(CLK), .RESET(RESET), .start(start2), .done(done2), .image({image[49:161],image[349:461],image[649:761],image[949:1061],image[1249:1361],image[1549:1661],image[1849:1961],image[2149:2261],image[2449:2561],image[2749:2861],image[3049:3161],image[3349:3461],image[3649:3761],image[3949:4061],image[4249:4361],image[4549:4661],image[4849:4961],image[5149:5261],image[5449:5561],image[5749:5861],image[6049:6161],image[6349:6461],image[6649:6761],image[6949:7061],image[7249:7361],image[7549:7661],image[7849:7961],image[8149:8261],image[8449:8561],image[8749:8861],image[9049:9161],image[9349:9461],image[9649:9761],image[9949:10061],image[10249:10361],image[10549:10661],image[10849:10961],image[11149:11261],image[11449:11561],image[11749:11861],image[12049:12161],image[12349:12461],image[12649:12761],image[12949:13061],image[13249:13361],image[13549:13661],image[13849:13961],image[14149:14261],image[14449:14561],image[14749:14861],image[15049:15161],image[15349:15461],image[15649:15761],image[15949:16061],image[16249:16361],image[16549:16661],image[16849:16961],image[17149:17261],image[17449:17561],image[17749:17861],image[18049:18161],image[18349:18461],image[18649:18761],image[18949:19061],image[19249:19361],image[19549:19661],image[19849:19961],image[20149:20261],image[20449:20561],image[20749:20861],image[21049:21161],image[21349:21461],image[21649:21761],image[21949:22061],image[22249:22361],image[22549:22661],image[22849:22961],image[23149:23261],image[23449:23561],image[23749:23861],image[24049:24161],image[24349:24461],image[24649:24761],image[24949:25061],image[25249:25361],image[25549:25661],image[25849:25961],image[26149:26261],image[26449:26561],image[26749:26861],image[27049:27161],image[27349:27461],image[27649:27761],image[27949:28061],image[28249:28361],image[28549:28661],image[28849:28961],image[29149:29261],image[29449:29561],image[29749:29861],image[30049:30161],image[30349:30461],image[30649:30761],image[30949:31061],image[31249:31361],image[31549:31661],image[31849:31961],image[32149:32261],image[32449:32561],image[32749:32861],image[33049:33161],image[33349:33461],image[33649:33761],image[33949:34061],image[34249:34361],image[34549:34661],image[34849:34961],image[35149:35261],image[35449:35561],image[35749:35861],image[36049:36161],image[36349:36461],image[36649:36761],image[36949:37061],image[37249:37361],image[37549:37661],image[37849:37961],image[38149:38261],image[38449:38561],image[38749:38861],image[39049:39161],image[39349:39461],image[39649:39761],image[39949:40061],image[40249:40361],image[40549:40661],image[40849:40961],image[41149:41261],image[41449:41561],image[41749:41861],image[42049:42161],image[42349:42461],image[42649:42761],image[42949:43061],image[43249:

43361],image[43549:43661],image[43849:43961],image[44149:44261],image[44449:44561],image[44749:44861]}), .confidence_BLK(eachConfidence_BLK[1]) , .confidence_WHT(eachConfidence_WHT[1]));

MaskOnePath

Mask3(.CLK(CLK), .RESET(RESET), .start(start3), .done(done3), .image({image[98:210],image[398:510],image[698:810],image[998:1110],image[1298:1410],image[1598:1710],image[1898:2010],image[2198:2310],image[2498:2610],image[2798:2910],image[3098:3210],image[3398:3510],image[3698:3810],image[3998:4110],image[4298:4410],image[4598:4710],image[4898:5010],image[5198:5310],image[5498:5610],image[5798:5910],image[6098:6210],image[6398:6510],image[6698:6810],image[6998:7110],image[7298:7410],image[7598:7710],image[7898:8010],image[8198:8310],image[8498:8610],image[8798:8910],image[9098:9210],image[9398:9510],image[9698:9810],image[9998:10110],image[10298:10410],image[10598:10710],image[10898:11010],image[11198:11310],image[11498:11610],image[11798:11910],image[12098:12210],image[12398:12510],image[12698:12810],image[12998:13110],image[13298:13410],image[13598:13710],image[13898:14010],image[14198:14310],image[14498:14610],image[14798:14910],image[15098:15210],image[15398:15510],image[15698:15810],image[15998:16110],image[16298:16410],image[16598:16710],image[16898:17010],image[17198:17310],image[17498:17610],image[17798:17910],image[18098:18210],image[18398:18510],image[18698:18810],image[18998:19110],image[19298:19410],image[19598:19710],image[19898:20010],image[20198:20310],image[20498:20610],image[20798:20910],image[21098:21210],image[21398:21510],image[21698:21810],image[21998:22110],image[22298:22410],image[22598:22710],image[22898:23010],image[23198:23310],image[23498:23610],image[23798:23910],image[24098:24210],image[24398:24510],image[24698:24810],image[24998:25110],image[25298:25410],image[25598:25710],image[25898:26010],image[26198:26310],image[26498:26610],image[26798:26910],image[27098:27210],image[27398:27510],image[27698:27810],image[27998:28110],image[28298:28410],image[28598:28710],image[28898:29010],image[29198:29310],image[29498:29610],image[29798:29910],image[30098:30210],image[30398:30510],image[30698:30810],image[30998:31110],image[31298:31410],image[31598:31710],image[31898:32010],image[32198:32310],image[32498:32610],image[32798:32910],image[33098:33210],image[33398:33510],image[33698:33810],image[33998:34110],image[34298:34410],image[34598:34710],image[34898:35010],image[35198:35310],image[35498:35610],image[35798:35910],image[36098:36210],image[36398:36510],image[36698:36810],image[36998:37110],image[37298:37410],image[37598:37710],image[37898:38010],image[38198:38310],image[38498:38610],image[38798:38910],image[39098:39210],image[39398:39510],image[39698:39810],image[39998:40110],image[40298:40410],image[40598:40710],image[40898:41010],image[41198:41310],image[41498:41610],image[41798:41910],image[42098:42210],image[42398:42510],image[42698:42810],image[42998:43110],image[43298:43410],image[43598:43710],image[43898:44010],image[44198:44310],image[44498:44610],image[44798:44910]}), .confidence_BLK(eachConfidence_BLK[2]) , .confidence_WHT(eachConfidence_WHT[2]));

MaskOnePath

Mask4(.CLK(CLK), .RESET(RESET), .start(start4), .done(done4), .image({image[143:255],image[443:555],image[743:855],image[1043:1155],image[1343:1455],image[1643:1755],image[1943:2055],image[2243:2355],image[2543:2655],image[2843:2955],image[3143:3255],image[3443:3555],image[3743:3855],image[4043:4155],image[4343:4455],image[4643:4755],image[4943:5055],image[5243:5355],image[5543:5655],image[5843:5955],image[6143:6255],image[644

3:6555],image[6743:6855],image[7043:7155],image[7343:7455],image[7643:7755],image[7943:8055],image[8243:8355],image[8543:8655],image[8843:8955],image[9143:9255],image[9443:9555],image[9743:9855],image[10043:10155],image[10343:10455],image[10643:10755],image[10943:11055],image[11243:11355],image[11543:11655],image[11843:11955],image[12143:12255],image[12443:12555],image[12743:12855],image[13043:13155],image[13343:13455],image[13643:13755],image[13943:14055],image[14243:14355],image[14543:14655],image[14843:14955],image[15143:15255],image[15443:15555],image[15743:15855],image[16043:16155],image[16343:16455],image[16643:16755],image[16943:17055],image[17243:17355],image[17543:17655],image[17843:17955],image[18143:18255],image[18443:18555],image[18743:18855],image[19043:19155],image[19343:19455],image[19643:19755],image[19943:20055],image[20243:20355],image[20543:20655],image[20843:20955],image[21143:21255],image[21443:21555],image[21743:21855],image[22043:22155],image[22343:22455],image[22643:22755],image[22943:23055],image[23243:23355],image[23543:23655],image[23843:23955],image[24143:24255],image[24443:24555],image[24743:24855],image[25043:25155],image[25343:25455],image[25643:25755],image[25943:26055],image[26243:26355],image[26543:26655],image[26843:26955],image[27143:27255],image[27443:27555],image[27743:27855],image[28043:28155],image[28343:28455],image[28643:28755],image[28943:29055],image[29243:29355],image[29543:29655],image[29843:29955],image[30143:30255],image[30443:30555],image[30743:30855],image[31043:31155],image[31343:31455],image[31643:31755],image[31943:32055],image[32243:32355],image[32543:32655],image[32843:32955],image[33143:33255],image[33443:33555],image[33743:33855],image[34043:34155],image[34343:34455],image[34643:34755],image[34943:35055],image[35243:35355],image[35543:35655],image[35843:35955],image[36143:36255],image[36443:36555],image[36743:36855],image[37043:37155],image[37343:37455],image[37643:37755],image[37943:38055],image[38243:38355],image[38543:38655],image[38843:38955],image[39143:39255],image[39443:39555],image[39743:39855],image[40043:40155],image[40343:40455],image[40643:40755],image[40943:41055],image[41243:41355],image[41543:41655],image[41843:41955],image[42143:42255],image[42443:42555],image[42743:42855],image[43043:43155],image[43343:43455],image[43643:43755],image[43943:44055],image[44243:44355],image[44543:44655],image[44843:44955]}), .confidence_BLK(eachConfidence_BLK[3]) , .confidence_WHT(eachConfidence_WHT[3]));

MaskOnePath

Mask5(.CLK(CLK), .RESET(RESET), .start(start5), .done(done5), .image({image[187:299],image[487:599],image[787:899],image[1087:1199],image[1387:1499],image[1687:1799],image[1987:2099],image[2287:2399],image[2587:2699],image[2887:2999],image[3187:3299],image[3487:3599],image[3787:3899],image[4087:4199],image[4387:4499],image[4687:4799],image[4987:5099],image[5287:5399],image[5587:5699],image[5887:5999],image[6187:6299],image[6487:6599],image[6787:6899],image[7087:7199],image[7387:7499],image[7687:7799],image[7987:8099],image[8287:8399],image[8587:8699],image[8887:8999],image[9187:9299],image[9487:9599],image[9787:9899],image[10087:10199],image[10387:10499],image[10687:10799],image[10987:11099],image[11287:11399],image[11587:11699],image[11887:11999],image[12187:12299],image[12487:12599],image[12787:12899],image[13087:13199],image[13387:13499],image[13687:13799],image[13987:14099],image[14287:14399],image[14587:14699],image[14887:14999],image[15187:15299],image[15487:15599],image[15787:15899],image[16087:16199],image[16387:16499],image[16687:16799],image[16987:17099],image[17287:17399],image[17587:17699],image[17887:17999],image[18187:18299],image[18487:18599],image[18787:18899],image[19

```
087:19199],image[19387:19499],image[19687:19799],image[19987:20099],image[20287:20399]
,image[20587:20699],image[20887:20999],image[21187:21299],image[21487:21599],image[21
787:21899],image[22087:22199],image[22387:22499],image[22687:22799],image[22987:23099]
,image[23287:23399],image[23587:23699],image[23887:23999],image[24187:24299],image[24
487:24599],image[24787:24899],image[25087:25199],image[25387:25499],image[25687:25799]
,image[25987:26099],image[26287:26399],image[26587:26699],image[26887:26999],image[27
187:27299],image[27487:27599],image[27787:27899],image[28087:28199],image[28387:28499]
,image[28687:28799],image[28987:29099],image[29287:29399],image[29587:29699],image[29
887:29999],image[30187:30299],image[30487:30599],image[30787:30899],image[31087:31199]
,image[31387:31499],image[31687:31799],image[31987:32099],image[32287:32399],image[32
587:32699],image[32887:32999],image[33187:33299],image[33487:33599],image[33787:33899]
,image[34087:34199],image[34387:34499],image[34687:34799],image[34987:35099],image[35
287:35399],image[35587:35699],image[35887:35999],image[36187:36299],image[36487:36599]
,image[36787:36899],image[37087:37199],image[37387:37499],image[37687:37799],image[37
987:38099],image[38287:38399],image[38587:38699],image[38887:38999],image[39187:39299]
,image[39487:39599],image[39787:39899],image[40087:40199],image[40387:40499],image[40
687:40799],image[40987:41099],image[41287:41399],image[41587:41699],image[41887:41999]
,image[42187:42299],image[42487:42599],image[42787:42899],image[43087:43199],image[43
387:43499],image[43687:43799],image[43987:44099],image[44287:44399],image[44587:44699]
,image[44887:44999]}), .confidence_BLK(eachConfidence_BLK[4]) , .confidence_WHT(each
Confidence_WHT[4]));
        ConfidenceReg ConfReg1
( .CLK(CLK), .RESET(RESET), .WE(WE1), .confidence_BLK(eachConfidence_BLK[0]) , .con
fidence_WHT(eachConfidence_WHT[0]), .confidence_BLK_out(eachConfidence_BLK_out[
0]), .confidence_WHT_out(eachConfidence_WHT_out[0]));
        ConfidenceReg ConfReg2
( .CLK(CLK), .RESET(RESET), .WE(WE2), .confidence_BLK(eachConfidence_BLK[1]) , .con
fidence_WHT(eachConfidence_WHT[1]), .confidence_BLK_out(eachConfidence_BLK_out[
1]), .confidence_WHT_out(eachConfidence_WHT_out[1]));
        ConfidenceReg ConfReg3
( .CLK(CLK), .RESET(RESET), .WE(WE3), .confidence_BLK(eachConfidence_BLK[2]) , .con
fidence_WHT(eachConfidence_WHT[2]), .confidence_BLK_out(eachConfidence_BLK_out[
2]), .confidence_WHT_out(eachConfidence_WHT_out[2]));
        ConfidenceReg ConfReg4
( .CLK(CLK), .RESET(RESET), .WE(WE4), .confidence_BLK(eachConfidence_BLK[3]) , .con
fidence_WHT(eachConfidence_WHT[3]), .confidence_BLK_out(eachConfidence_BLK_out[
3]), .confidence_WHT_out(eachConfidence_WHT_out[3]));
        ConfidenceReg ConfReg5
( .CLK(CLK), .RESET(RESET), .WE(WE5), .confidence_BLK(eachConfidence_BLK[4]) , .con
fidence_WHT(eachConfidence_WHT[4]), .confidence_BLK_out(eachConfidence_BLK_out[
4]), .confidence_WHT_out(eachConfidence_WHT_out[4]));
        // basic flip flop
        always_ff @ (posedge CLK)
  begin
    if (RESET)
```

```
        state <= waitState;
    else
        state <= nextState;
end

        // state transition
        always_comb
        begin
                nextState = state;
                unique case (state)
                        waitState:
                                begin
                                if (start == 1'b1)
                                        nextState = startFirstMask;
                                end
                        startFirstMask:
                                begin
                                if (done1 == 1'b1)
                                        nextState = storeFirstConf;
                                end

                        storeFirstConf:
                                nextState = startSecondMask;

                        startSecondMask:
                                if (done2 == 1'b1)
                                        nextState = storeSecondConf;
                        storeSecondConf:
                                nextState = startThirdMask;

                        startThirdMask:
                                if (done3 == 1'b1)
                                        nextState = storeThirdConf;
                        storeThirdConf:
                                nextState = startFourthMask;

                        startFourthMask:
                                if (done4 == 1'b1)
                                        nextState = storeFourthConf;
                        storeFourthConf:
                                nextState = startFifthMask;

                        startFifthMask:
                                if (done5 == 1'b1)
                                        nextState = storeFifthConf;
```

```
        storeFifthConf:
                nextState = doneState;

        doneState:
                nextState = waitState;
        default: nextState = waitState;
endcase

// state implementation
/*initialize the control signals to inactive at the first place*/
start1=1'b0;
start2=1'b0;
start3=1'b0;
start4=1'b0;
start5=1'b0;
WE1 = 1'b0;
WE2 = 1'b0;
WE3 = 1'b0;
WE4 = 1'b0;
WE5 = 1'b0;
done = 1'b0;
unique case(state)
        waitState :
                done = 1'b1;
        startFirstMask:
                start1=1'b1;
        storeFirstConf :
                begin
                        start1=1'b1;
                        WE1=1'b1;
                end
        startSecondMask :
                start2=1'b1;
        storeSecondConf :
                begin
                        start2=1'b1;
                        WE2=1'b1;
                end
        startThirdMask :
                start3=1'b1;
        storeThirdConf :
                begin
                        start3=1'b1;
                        WE3=1'b1;
                end
```

```
                startFourthMask :
                        start4=1'b1;
                storeFourthConf :
                        begin
                                start4=1'b1;
                                WE4=1'b1;
                        end
                startFifthMask :
                        start5=1'b1;
                storeFifthConf :
                        begin
                                start5=1'b1;
                                WE5=1'b1;
                        end
                doneState :
                        done = 1'b1;
        endcase
    end


endmodule
```

Module:	ImageReg.sv
Input:	CLK , WE, [9:0] X, [9:0] Y, eachConf
Output:	[0:149][0:299] image
Description:	receives the black or white confidence and stores that data into the corresponding position of the image register array.
Purpose:	The large register file used to store the black and white data of the middle rectangle.

```
module ImageReg ( input logic WE,
                                        input logic [9:0] X, Y,
                                        input logic eachConf,
                                        input CLK,
                                        output logic [0:149][0:299] image
                                        );


        always_ff @ (posedge CLK)
        begin
                        if(WE)
```

```
                        begin
                        image[Y][X]<=eachConf;
                        end
                end
endmodule
// ImageReg
myImageArray(.WE(loadConf), .eachConf(blackorwhite), .CLK(pixclk), .RESET(RESET), .
CLEAR(clear), .image(image));
```

Module:	ImageRead.sv
Input:	pixclk, pixclk, cansend, [9:0] hsync, [9:0] vsync, [9:0] pixvalue
Output:	dataready, [0:149][0:299] image
Description:	receives pixel value input and frame position counter from VGA_controller and calls the Image Reg to output the black and white image data of the middle rectangle.
Purpose:	state machine of organizing data from VGA input to form a large image register file.

```
module ImageRead (      input logic pixclk,
                                input logic pixclk, cansend, // active high
                                input logic [9:0] hsync, vsync,
                                input logic [9:0] pixvalue,
                                output logic dataready,         //
active high
                                output logic [0:149][0:299] image
                                );

                        logic   blackorwhite;
                        logic [9:0] x, y;
                        logic loadConf;
                        assign y=vsync-(34+165);
                        assign x=hsync-(144+170);
                        assign blackorwhite = (pixvalue >= 10'b0110000000) ? 1'b1 :
1'b0; // each confidence bit
                        // state declaration
                        enum logic [2:0] {waitState ,readNothing, readData, doneState }
state, nextState;
                        // module declaration
                        ImageReg
myImageArray(.WE(loadConf), .X(x), .Y(y), .eachConf(blackorwhite), .CLK(pixclk), .image
(image));

                        always_ff @ (posedge pixclk)
                        begin
```

```
		if (RESET)
			state <= waitState;
		else
			state <= nextState;
	end

	// state transition
	always_comb
	begin
		nextState = state;
		unique case(state)
		waitState:
			if(cansend && vsync==10'b0)
				nextState = readNothing;
		readNothing:
			begin
				if(vsync== 34+315 && hsync==144+470)
					nextState = doneState;
				else if (hsync == 144+170-2 && vsync >= 34+165 && vsync < 34+315)
					nextState = readData;
			end
		readData:
			begin
				if (hsync == 144+470-1 && vsync >= 34+165 && vsync < 34+315)
					nextState = readNothing;
			end
		doneState:
			begin
				nextState = waitState;
			end
		default:
			nextState = waitState;
		endcase

	// state implementation
		// control signal implementation
		loadConf=1'b0;
		dataready=1'b0;
		unique case(state)
			waitState:
				; // do nothing
			readNothing:
				; // do nothing
```

```
                                        readData:
                                                loadConf=1'b1;
                                        doneState:
                                                dataready=1'b1;
                                endcase
                        end
endmodule
```

Module: ConfidenceReg.sv
Input: CLK, WE, RESET, [9:0] confidence_BLK, [9:0] confidence_WHT
Output: [9:0] confidence_BLK_out, [9:0] confidence_WHT_out)
Description: cast input to output at write enable.
Purpose: register file to store confidence data.

```
module ConfidenceReg (input logic CLK, WE, RESET,
                                                input logic [9:0] confidence_BLK,
                                                input logic [9:0] confidence_WHT,
                                                output logic [9:0] confidence_BLK_out,
                                                output logic [9:0] confidence_WHT_out);

        always_ff @ (posedge CLK)
        begin
                if(RESET)
                        begin
                        confidence_WHT_out<=10'b0;
                        confidence_BLK_out<=10'b0;
                        end
                else if(WE)
                        begin
                        confidence_BLK_out<=confidence_BLK;
                        confidence_WHT_out<=confidence_WHT;
                        end
        end

endmodule
```

Module: I2C_Controller.sv
Input: CLOCK, [23:0]I2C_DATA, GO, RESET, W_R
Output: I2C_SCLK, END, ACK, [5:0] SD_COUNTER, SDO
Inout: I2C_SDAT
Description: Interface between camera and DE2 board
Purpose: Controls exposure

```
// --------------------------------------------------------------------
// Copyright (c) 2005 by Terasic Technologies Inc.
// --------------------------------------------------------------------
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altrea Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL or Verilog source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
// --------------------------------------------------------------------
//
//                     Terasic Technologies Inc
//                     356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                     HsinChu County, Taiwan
//                     302
//
//                     web: http://www.terasic.com/
//                     email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions:i2c controller
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//   Ver  :| Author          :| Mod. Date :| Changes Made:
//   V1.0 :| Joe Yang        :| 05/07/10  :|      Initial Revision
// --------------------------------------------------------------------
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
        W_R,     //W_R
        ACK,     //ACK
        RESET,
        //TEST
        SD_COUNTER,
        SDO
);
        input  CLOCK;
        input  [23:0]I2C_DATA;
        input  GO;
        input  RESET;
        input  W_R;
        inout  I2C_SDAT;
        output I2C_SCLK;
        output END;
        output ACK;

//TEST
        output [5:0] SD_COUNTER;
        output SDO;


reg SDO;
reg SCLK;
reg END;
reg [23:0]SD;
reg [5:0]SD_COUNTER;

wire I2C_SCLK=SCLK | ( ((SD_COUNTER >= 4) & (SD_COUNTER <=30))? ~CLOCK :0 );
wire I2C_SDAT=SDO?1'bz:0 ;

reg ACK1,ACK2,ACK3;
wire ACK=ACK1 | ACK2 |ACK3;

//--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 < 6'b111111) 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; END=1; end
else
case (SD_COUNTER)
        6'd0  : begin ACK1=0 ;ACK2=0 ;ACK3=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[23];
        6'd4  : SDO=SD[22];
        6'd5  : SDO=SD[21];
        6'd6  : SDO=SD[20];
        6'd7  : SDO=SD[19];
        6'd8  : SDO=SD[18];
        6'd9  : SDO=SD[17];
        6'd10 : SDO=SD[16];
        6'd11 : SDO=1'b1;//ACK

        //SUB ADDR
        6'd12  : begin SDO=SD[15]; ACK1=I2C_SDAT; end
        6'd13  : SDO=SD[14];
        6'd14  : SDO=SD[13];
        6'd15  : SDO=SD[12];
        6'd16  : SDO=SD[11];
        6'd17  : SDO=SD[10];
        6'd18  : SDO=SD[9];
        6'd19  : SDO=SD[8];
        6'd20  : SDO=1'b1;//ACK

        //DATA
        6'd21  : begin SDO=SD[7]; ACK2=I2C_SDAT; end
        6'd22  : SDO=SD[6];
        6'd23  : SDO=SD[5];
        6'd24  : SDO=SD[4];
        6'd25  : SDO=SD[3];
        6'd26  : SDO=SD[2];
        6'd27  : SDO=SD[1];
        6'd28  : SDO=SD[0];
        6'd29  : SDO=1'b1;//ACK
```

```
        //stop
    6'd30 : begin SDO=1'b0;  SCLK=1'b0; ACK3=I2C_SDAT; end
    6'd31 : SCLK=1'b1;
    6'd32 : begin SDO=1'b1; END=1; end

endcase
end



endmodule
```

Module: I2C_CCD_Config
Input: iCLK, iRST_N, [15:0] iExposure
Output: I2C_SCLK
Inout: I2C_SDAT
Description: lookup tables for I2C parameters.
Purpose: controls exposure data.

```
module I2C_CCD_Config (//       Host Side
                                    iCLK,
                                    iRST_N,
                                    iExposure,
                                    //       I2C Side
                                    I2C_SCLK,
                                    I2C_SDAT     );
//      Host Side
input                   iCLK;
input                   iRST_N;
input   [15:0]  iExposure;
//      I2C Side
output                  I2C_SCLK;
inout           I2C_SDAT;
//      Internal Registers/Wires
reg     [15:0]  mI2C_CLK_DIV;
reg     [23:0]  mI2C_DATA;
reg                     mI2C_CTRL_CLK;
reg                     mI2C_GO;
wire            mI2C_END;
wire            mI2C_ACK;
reg     [15:0]  LUT_DATA;
```

```
reg     [5:0]   LUT_INDEX;
reg     [3:0]   mSetup_ST;

//      Clock Setting
parameter       CLK_Freq    =       50000000;       //      50      MHz
parameter       I2C_Freq    =       20000;          //      20      KHz
//      LUT Data Number
parameter       LUT_SIZE    =       17;

//////////////////////      I2C Control Clock       ////////////////////////
always@(posedge iCLK or negedge iRST_N)
begin
        if(!iRST_N)
        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(iRST_N)      );
```

```
////////////////////////////////////////////////////////////////////
//
//////////////////////     Config Control      //////////////////////////
always@(posedge mI2C_CTRL_CLK or negedge iRST_N)
begin
	if(!iRST_N)
	begin
		LUT_INDEX <=	0;
		mSetup_ST <=	0;
		mI2C_GO		<=	0;
	end
	else
	begin
		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
end
////////////////////////////////////////////////////////////////////
//
//////////////////////     Config Data LUT     //////////////////////////
```

```
always
begin
	case(LUT_INDEX)
	0	:	LUT_DATA  <=	16'h0000;
	1	:	LUT_DATA  <=	16'h2000;
	2	:	LUT_DATA  <=	16'hF101;	//	Mirror Row and Columns
	3	:	LUT_DATA  <=	{8'h09,iExposure[15:8]};//  Exposure
	4	:	LUT_DATA  <=	{8'hF1,iExposure[7:0]};
	5	:	LUT_DATA  <=	16'h2B00;	//	Green 1 Gain
	6	:	LUT_DATA  <=	16'hF1B0;
	7	:	LUT_DATA  <=	16'h2C00;	//	Blue Gain
	8	:	LUT_DATA  <=	16'hF1CF;
	9	:	LUT_DATA  <=	16'h2D00;	//	Red Gain
	10	:	LUT_DATA  <=	16'hF1CF;
	11	:	LUT_DATA  <=	16'h2E00;	//	Green 2 Gain
	12	:	LUT_DATA  <=	16'hF1B0;
	13	:	LUT_DATA  <=	16'h0500;	//	H_Blanking
	14	:	LUT_DATA  <=	16'hF188;
	15	:	LUT_DATA  <=	16'h0600;	//	V_Blanking
	16	:	LUT_DATA  <=	16'hF119;
	default:LUT_DATA <=	16'h0000;
	endcase
end
////////////////////////////////////////////////////////////////////
//
endmodule
```

Module: RAW2RGB.v
Input: [10:0] iX_Cont, [10:0] iY_Cont, [9:0] iDATA, iDVAL, iCLK, iRST
Output: [9:0] oRed, [9:0] oGreen, [9:0] oBlue, oDVAL
Description: takes in original data from GPIO pins and transfer them to RGB data.
Purpose: data transfer from GPIO structure to RGB structure.

```
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   [9:0]   iDATA;
input           iDVAL;
input           iCLK;
input           iRST;
output  [9:0]   oRed;
output  [9:0]   oGreen;
output  [9:0]   oBlue;
output          oDVAL;
wire    [9:0]   mDATA_0;
wire    [9:0]   mDATA_1;
reg     [9:0]   mDATAd_0;
reg     [9:0]   mDATAd_1;
reg     [9:0]   mCCD_R;
reg     [10:0]  mCCD_G;
reg     [9:0]   mCCD_B;
reg             mDVAL;

assign oRed   =     mCCD_R[9:0];
assign oGreen =     mCCD_G[10:1];
assign oBlue  =     mCCD_B[9:0];
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'b01)
		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'b00)
		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'b11)
		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'b10)
		begin
			mCCD_R <= mDATAd_1;
			mCCD_G <= mDATAd_0+mDATA_1;
			mCCD_B <= mDATA_0;
		end
	end
end

endmodule
```

Module:	CCD_Capture.v
Input:	[9:0] iDATA, iFVAL, iLVAL, iSTART, iEND, iCLK, iRST
Output:	[9:0] oDATA, [10:0] oX_Cont, [10:0] oY_Cont, [31:0] oFrame_Cont, oDVAL
Description:	takes in key input and GPIO input and organize into frame counted data.
Purpose:	organizes GPIO raw data to organized frame counted data for further processing.

```
module CCD_Capture(     oDATA,
                        oDVAL,
                        oX_Cont,
                        oY_Cont,
                        oFrame_Cont,
                        iDATA,
                        iFVAL,
                        iLVAL,
                        iSTART,
                        iEND,
                        iCLK,
                        iRST    );

input  [9:0]    iDATA;
input           iFVAL;
input           iLVAL;
input           iSTART;
input           iEND;
input           iCLK;
input           iRST;
output  [9:0]   oDATA;
output  [10:0]  oX_Cont;
output  [10:0]  oY_Cont;
output  [31:0]  oFrame_Cont;
output          oDVAL;
reg             Pre_FVAL;
reg             mCCD_FVAL;
reg             mCCD_LVAL;
reg     [9:0]   mCCD_DATA;
reg     [10:0]  X_Cont;
reg     [10:0]  Y_Cont;
reg     [31:0]  Frame_Cont;
reg             mSTART;

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;
                mCCD_DATA       <=      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;
                mCCD_DATA       <=      iDATA;
                if(mCCD_FVAL)
                begin
                        if(mCCD_LVAL)
                        begin
                                if(X_Cont<1279)
                                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

endmodule
```

Module: Mirror_Col.v
Input: [9:0] iCCD_R, [9:0] iCCD_G, [9:0] iCCD_B, iCCD_DVAL, iCCD_PIXCLK, iRST_N
Output: [9:0] oCCD_R, [9:0] oCCD_G, [9:0] oCCD_B, oCCD_DVAL
Description: takes in image data and outputs the inverse.
Purpose: Mirror/Invert the image to output on VGA monitor.

```
module Mirror_Col(//      Input Side
                        iCCD_R,
                        iCCD_G,
                        iCCD_B,
                        iCCD_DVAL,
                        iCCD_PIXCLK,
                        iRST_N,
                        //      Output Side
                        oCCD_R,
                        oCCD_G,
                        oCCD_B,
                        oCCD_DVAL       );
//      Input Side
input   [9:0]   iCCD_R;
input   [9:0]   iCCD_G;
input   [9:0]   iCCD_B;
input           iCCD_DVAL;
```

```
input           iCCD_PIXCLK;
input           iRST_N;
//      Output Side
output  [9:0]   oCCD_R;
output  [9:0]   oCCD_G;
output  [9:0]   oCCD_B;
output          oCCD_DVAL;
//      Internal Registers
reg     [9:0]   Z_Cont;
reg             mCCD_DVAL;

assign oCCD_DVAL        =       mCCD_DVAL;

always@(posedge iCCD_PIXCLK or negedge iRST_N)
begin
        if(!iRST_N)
        begin
                mCCD_DVAL       <=      0;
                Z_Cont          <=      0;
        end
        else
        begin
                mCCD_DVAL       <=      iCCD_DVAL;
                if(Z_Cont<640)
                begin
                        if(iCCD_DVAL)
                        Z_Cont  <=      Z_Cont+1'b1;
                end
                else
                Z_Cont  <=      0;
        end
end

Stack_RAM (
                        .clock(iCCD_PIXCLK),
                        .data(iCCD_R),
                        .rdaddress(639-Z_Cont),
                        .wraddress(Z_Cont),
                        .wren(iCCD_DVAL),
                        .q(oCCD_R));

Stack_RAM (
                        .clock(iCCD_PIXCLK),
                        .data(iCCD_G),
                        .rdaddress(639-Z_Cont),
```

```
                .wraddress(Z_Cont),
                .wren(iCCD_DVAL),
                .q(oCCD_G));

Stack_RAM (
                .clock(iCCD_PIXCLK),
                .data(iCCD_B),
                .rdaddress(639-Z_Cont),
                .wraddress(Z_Cont),
                .wren(iCCD_DVAL),
                .q(oCCD_B));

endmodule
```