FRONT END VLSI TRAINING - 2015

FINAL PROJECT - ASSIGNMENT 2

TOPIC - COLOR TO GRAYSCALE TRANSFORM OF GIVEN IMAGE

WORKING -

GIVEN IMAGE (16 X 16)

FF	FF	FF	FF	FF	FF	E0	E0	E0	E0	E0	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	E0	E0	E0	E0	E0	E0	E0	E0	FF	FF	FF
FF	FF	FF	FF	00	00	00	F6	F6	00	F6	FF	FF	FF	FF	FF
FF	FF	FF	00	F6	00	F6	F6	F6	00	F6	F6	F6	FF	FF	FF
FF	FF	FF	00	F6	00	00	F6	F6	F6	00	F6	F6	F6	FF	FF
FF	FF	FF	00	00	F6	F6	F6	F6	00	00	00	00	FF	FF	FF
FF	FF	FF	FF	FF	F6	F6	F6	F6	F6	F6	F6	FF	FF	FF	FF
FF	FF	FF	FF	E0	E0	13	E0	E0	13	E0	E0	FF	FF	FF	FF
FF	FF	FF	E0	E0	E0	13	E0	E0	13	E0	E0	E0	FF	FF	FF
FF	FF	E0	E0	E0	E0	13	13	13	13	E0	E0	E0	E0	FF	FF
FF	FF	F6	F6	E0	13	F9	13	13	F9	13	E0	F6	F6	FF	FF
FF	FF	F6	F6	F6	13	13	13	13	13	13	F6	F6	F6	FF	FF
FF	FF	F6	F6	13	13	13	13	13	13	13	13	F6	F6	FF	FF
FF	FF	FF	FF	13	13	13	FF	FF	13	13	13	FF	FF	FF	FF
FF	FF	FF	E0	E0	E0	FF	FF	FF	FF	E0	E0	E0	FF	FF	FF
FF	FF	E0	E0	E0	E0	FF	FF	FF	FF	E0	E0	E0	E0	FF	FF

- The grayscale color conversion has been carried out by using threshold technique.
- We see that there are total 6 different colors visible in the given image, as per the 3-3-2 RGB color format.
- Every pixel of the 16 X 16 color image has specified values RGB which are to be manipulated for conversion to Black or White color.
- The 3-3-2 RGB format is expressed in 8-bit format with first three pixels representing red color, next three for green and last two for blue color.

1	FF		WHITE
2	00		BLACK
3	E0		RED
4	F6		~ORANGE
5	13		BLUE
6	F9		~PINK

- As can be observed from the image, the pixels fall under one of the 6 specified colors.
- There are total 3 possible grey levels possible for the 3-3-2 RGB format when there is equal contribution of the R, G, and B values.

- The 6 different colors are then divided as per the thresholds they fall into.
- Apart from black and white, three shades of grey are obtained for the grayscale image.
- Thresholds with color and hexadecimal values -

00		BLACK
25		DARK GREY
4A		MEDIUM GREY
6F		LIGHT GREY
FF		WHITE

CODE -

[main module]

```
module MainModule(
                        input clk,
                        input RST,
                        output HS,VS,
                        output [7:0]Data
                                        );

parameter ImageWidth  = 16;
parameter ImageHeight = 16;
integer i,j;

reg  WEN;
reg  [7:0]Data_in,Counter;
wire [3:0]VAdd,HAdd;
reg  [7:0]ImageRom[(ImageWidth*ImageHeight)-1:0];

assign VAdd = {Counter[7:4]};
assign HAdd =  {Counter[3:0]};

VGADriver VGADriver0 (          .clk(clk),
                                                .WEN(WEN),
                                                .HS(HS),
                                                .VS(VS),
                                                .Data_in(Data_in),
                                                .VAdd(VAdd),
                                                .HAdd(HAdd),
                                                .Data(Data)
                                        );

initial
        begin
        for(i=0;i<ImageHeight;i=i+1)
                for(j=0;j<ImageWidth;j=j+1)
                        ImageRom[(i*ImageWidth)+j] = 8'b11111111;          //Dummy Data
        $readmemh("MarioClear.dat",ImageRom);                              // Load Image
        end
```

```
always@(Counter)
begin

          if(ImageRom[Counter] == 8'h00)   // If pixel is black
            begin
             Data_in = 8'h00;   //Black
            end
          if(ImageRom[Counter] > 8'h00 &&ImageRom[Counter] <= 8'h25)   //Threshold 1
           begin
                        Data_in = 8'h25;  //Threshold 1 - Dark gray
           end
          else if(ImageRom[Counter] > 8'h25 && ImageRom[Counter] < 8'hE1)  //Between threshold 1
and 2
            begin
                        Data_in = 8'h4A;  //Threshold 2  - Medium grey
                end
           else if(ImageRom[Counter] > 8'hE1 && ImageRom[Counter] <= 8'hFA) //Between threshold
2 and 3
                begin
                        Data_in = 8'h6F;  //Threshold 3 - light grey
                end
          else if(ImageRom[Counter] > 8'hFA)   // If greater than threshold 3
                begin
                        Data_in = 8'hFF;  // White
                end
end

end




// Run a counter to Read Each Pixel From main ROM
always@(posedge clk)
        begin
                WEN = 1;
                if (RST == 1)
                        begin
                                Counter = 8'b00000000;
                        end
                else if (!(Counter==8'b11111111))
                        begin
                                Counter = Counter+1;
                                WEN = 0;
                        end
        end


        endmodule
```