Lab7Part1

// Background image display

module background

    (

        CLOCK\_50,                        //    On Board 50 MHz

        KEY,                            //    Push Button[0:0]

        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]

    );

    input    CLOCK\_50;                //    50 MHz

    input    [0:0] KEY;                //    Button[0:0]

    output    VGA\_CLK;                   //    VGA Clock

    output    VGA\_HS;                    //    VGA H\_SYNC

    output    VGA\_VS;                    //    VGA V\_SYNC

    output    VGA\_BLANK;                //    VGA BLANK

    output    VGA\_SYNC;                //    VGA SYNC

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

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

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

    wire resetn, gnd;

    assign resetn = KEY[0];

    assign gnd = 1'b0;

    // Create the color, x, y and writeEn wires that are inputs to the controller.

    wire [2:0] color;

    wire [7:0] x;

    wire [6:0] y;

    assign x = 8'd0;

    assign y = 7'd0;

    assign color = 3'd0;

    // Create an Instance of a VGA controller - "There can be only one!"

    // Define the number of colours as well as the initial background

    // image file (.MIF) for the controller.

    vga\_adapter VGA(

            .resetn(resetn),

            .clock(CLOCK\_50),

            .colour(color),

            .x(x),

            .y(y),

            .plot(gnd),

            /\* Signals for the DAC to drive the monitor. \*/

            .VGA\_R(VGA\_R),

            .VGA\_G(VGA\_G),

            .VGA\_B(VGA\_B),

            .VGA\_HS(VGA\_HS),

            .VGA\_VS(VGA\_VS),

            .VGA\_BLANK(VGA\_BLANK),

            .VGA\_SYNC(VGA\_SYNC),

            .VGA\_CLK(VGA\_CLK));

        defparam VGA.RESOLUTION = "160x120";

        defparam VGA.MONOCHROME = "FALSE";

        defparam VGA.BITS\_PER\_COLOUR\_CHANNEL = 1;

        defparam VGA.BACKGROUND\_IMAGE = "display.mif";

endmodule

Lab7Part1b

// Background image display

module background

    (

        CLOCK\_50,                        //    On Board 50 MHz

        SW,                            //    Push Button[0:0]

        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,

        KEY

        //    VGA Blue[9:0]

    );

    input    CLOCK\_50;                //    50 MHz

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

    input    [17:0] SW;                //    Button[0:0]

    output    VGA\_CLK;                   //    VGA Clock

    output    VGA\_HS;                    //    VGA H\_SYNC

    output    VGA\_VS;                    //    VGA V\_SYNC

    output    VGA\_BLANK;                //    VGA BLANK

    output    VGA\_SYNC;                //    VGA SYNC

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

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

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

    wire resetn, plot;

    wire [2:0] color;

    wire [7:0] x;

    wire [6:0] y;

    // Further assignments go here...

    // Define the number of colours as well as the initial background

    // image file (.MIF) for the controller.

    vga\_adapter VGA(

            .resetn(KEY[0]),

            .clock(CLOCK\_50),

            .colour(SW[17:15]),

            .x(SW[7:0]),

            .y(SW[14:8]),

            .plot(~KEY[1]),

            /\* Signals for the DAC to drive the monitor. \*/

            .VGA\_R(VGA\_R),

            .VGA\_G(VGA\_G),

            .VGA\_B(VGA\_B),

            .VGA\_HS(VGA\_HS),

            .VGA\_VS(VGA\_VS),

            .VGA\_BLANK(VGA\_BLANK),

            .VGA\_SYNC(VGA\_SYNC),

            .VGA\_CLK(VGA\_CLK));

        defparam VGA.RESOLUTION = "160x120";

        defparam VGA.MONOCHROME = "FALSE";

        defparam VGA.BITS\_PER\_COLOUR\_CHANNEL = 1;

        defparam VGA.BACKGROUND\_IMAGE = "display.mif";

endmodule

Part2

// Etch-and-sketch

module sketch

(

CLOCK\_50, // On Board 50 MHz

KEY, // Push Button[3:0]

SW, // DPDT Switch[17:0]

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,

LEDR,//, VGA Blue[9:0]

y\_state

);

input CLOCK\_50; // 50 MHz

input [3:0] KEY; // KEY[0] for resetting, KEY[1] for drawing, KEY[2] for clearing

input [17:0] SW; // SW[7:0]X,SW[14:8]Y,SW[17:15]color

output VGA\_CLK; // VGA Clock

output VGA\_HS; // VGA H\_SYNC

output VGA\_VS; // VGA V\_SYNC

output VGA\_BLANK; // VGA BLANK

output VGA\_SYNC; // VGA SYNC

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

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

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

output [17:0]LEDR;

output [2:0]y\_state;

// Create the color, x, y and writeEn wires that are inputs to the controller.

reg [2:0] color;

reg [7:0] x\_;

reg [6:0] y\_;

reg writeEn;

wire Draw;

wire clear;

wire resetn;

wire [7:0]x\_start;

wire [6:0]y\_start;

assign resetn = KEY[0];

assign Draw=KEY[1];

assign clear=KEY[2];

assign x\_start=SW[7:0];

assign y\_start=SW[14:8];

assign y\_state=y;

// Create an Instance of a VGA controller - there can be only one!

// Define the number of colours as well as the initial background

// image file (.MIF) for the controller.

vga\_adapter VGA(

.resetn(resetn),

.clock(CLOCK\_50),

.colour(color),

.x(x\_),

.y(y\_),

.plot(writeEn),

/\* Signals for the DAC to drive the monitor. \*/

.VGA\_R(VGA\_R),

.VGA\_G(VGA\_G),

.VGA\_B(VGA\_B),

.VGA\_HS(VGA\_HS),

.VGA\_VS(VGA\_VS),

.VGA\_BLANK(VGA\_BLANK),

.VGA\_SYNC(VGA\_SYNC),

.VGA\_CLK(VGA\_CLK));

defparam VGA.RESOLUTION = "160x120";

defparam VGA.MONOCHROME = "FALSE";

defparam VGA.BITS\_PER\_COLOUR\_CHANNEL = 1;

defparam VGA.BACKGROUND\_IMAGE = "display.mif";//WHAT'S THIS BACKGROUND IMAGE\*\*\*\*\*\*\*\*\*\*\*

// Put your code here. Your code should produce signals x,y,color and writeEn

// for the VGA controller, in addition to any other functionality your design may require.

reg [2:0]y;//current state

reg [2:0]Y;//next state

reg [7:0]countx1;

reg [6:0]county1;

reg [7:0]countx2;

reg [6:0]county2;

parameter Idle=3'b000,iter\_x1=3'b001,iter\_y1=3'b010,iter\_x2=3'b011,iter\_y2=3'b100;

always@(\*)

begin

case(y)

Idle:

begin //recall wire writeEn; wire clear;

if(resetn==0)

Y=Idle;

if(Draw==0)//because these are all key buttons

Y=iter\_x2;

if(clear==0)

Y=iter\_x1;

end

iter\_x1:

begin

if(countx1<8'd159)

Y=iter\_x1;

if(resetn==0)

Y=Idle;

else Y=iter\_y1;

end

iter\_y1:

begin

if(resetn==0)

Y=Idle;

if(county1<8'd119)

Y=iter\_x1;

end

iter\_x2:

begin

if(resetn==0)

Y=Idle;

if(countx2<8'd5)

Y=iter\_x2;

else Y=iter\_y2;

end

iter\_y2:

begin

if(resetn==0)

Y=Idle;

if(county2<8'd5)

Y=iter\_x2;

end

endcase

end

always@(posedge CLOCK\_50)

begin

y<=Y;

if(y==Idle)

begin

countx1<=0;//If the current state is idle, we set every count to 0

county1<=0;

countx2<=0;

county2<=0;

end

if(y==iter\_x1)

countx1<=countx1+1;

if(y==iter\_y1)

begin

county1<=county1+1;

countx1<=0;//Because we are moving to a new line, countx1 will be "reset" to 0

end

if(y==iter\_x2)

countx2<=countx2+1;

if(y==iter\_y2)

begin

county2<=county2+1;

countx2<=0;//Because we are moving to a new line, countx2 will be "reset" to 0

end

end

//directly below will be related to the output, i.e. the VGA monitor

always@(\*) //here we are going to plot things depended on the current state

begin

writeEn=(y==iter\_x1)|(y==iter\_x2);

case(y)

iter\_x1: begin

x\_=countx1;//countx1 will increment by 1

y\_=county1;

color=3'b000; //this is straight forward, when we clean up the screen we use 000

end

iter\_x2:

begin

x\_=x\_start+countx2;

y\_=y\_start+county2;

if(county2<=2)color=SW[17:15];

if(county2>2) color=~SW[17:15];

end

endcase

end

assign LEDR[17:15]=color;

assign LEDR[14:8]=y\_;

assign LEDR[7:0]=x\_;

endmodule

