```
module vga_demo (
   input          CLOCK_50,            // 50 MHz
   input   [17:0]  SW,
   input   [3:0]   KEY,
   output  [17:0]  LEDR,
   output  [7:0]   LEDG,
   output  [6:0]   HEX0,
   output  [6:0]   HEX1,
   output  [6:0]   HEX2,
   output  [6:0]   HEX3,
   output  [6:0]   HEX4,
   output  [6:0]   HEX5,
   output  [6:0]   HEX6,
   output  [6:0]   HEX7,
   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]
   );

   assign LEDR = SW;
   assign LEDG = 0;

//Turns off HEX1 and HEX0, which aren't used
   assign HEX1 = 7'h7f, HEX0 = 7'h7f;

//hexdigit modules simply assigned to display what values they represent
   hexdigit mode (
      .in   ({2'b00, SW[16:15]}),
      .out  (HEX7)
   );


   hexdigit speed (
      .in   ({2'b0, SW[14:13]}),
      .out  (HEX3)
   );

   hexdigit red (
      .in   ({2'b0, SW[6:5]}),
      .out  (HEX6)
   );

   hexdigit green (
      .in   ({2'b00, SW[4:3]}),
      .out  (HEX5)
   );

   hexdigit blue (
      .in   ({2'b00, SW[2:1]}),
      .out  (HEX4)
   );


   hexdigit size (
      .in   ({2'b00, SW[12:11]}),
      .out  (HEX2)
   );


   wire w3;
   wire w4;
   wire w5;
   wire w6;

//Four counters, whose enables are KEYS[0:3]
//Depending on the value of s, f will output a
//1 once the counter reaches a certain number.  This
//is done to allow the user to control the cursor at
//a reasonable speed.  Otherwise, the cursor would move
//across the screen 50 million times a second.

```

```

//KEY0 - cursor right
//KEY1 - cursor left
//KEY2 - cursor down
//KEY3 - cursor up

   counter c0 (
      .clk (CLOCK_50),
      .en (~KEY[0]),
      .s (SW[14:13]),
      .f (w3)
      );

   counter c1 (
      .clk (CLOCK_50),
      .en (~KEY[1]),
      .s (SW[14:13]),
      .f (w4)
      );

   counter c2 (
      .clk (CLOCK_50),
      .en (~KEY[2]),
      .s (SW[14:13]),
      .f (w5)
      );

   counter c3 (
      .clk (CLOCK_50),
      .en (~KEY[3]),
      .s (SW[14:13]),
      .f (w6)
      );


   wire [7:0] xLoc;
   wire [7:0] yLoc;

//Screen Resolution is 160 x 120, so max for
//x is 160, max for y is 120

   buttonLogic bX (
      .clk (CLOCK_50),
      .max (160),
      .addsignal (w3),
      .subsignal (w4),
      .k (xLoc[7:0])
   );

   buttonLogic bY (
      .clk (CLOCK_50),
      .max (120),
      .addsignal (w5),
      .subsignal (w6),
      .k (yLoc[7:0])
   );

   wire w7;
   wire [5:0] cw;

   counter c4 (
      .clk (CLOCK_50),
      .en (1),
      .s (SW[14:13]),
      .f (w7)
      );


//Each of these cycle color modules have a six bit
//output, which corresponds to the 6 bits of color data.
//This 6 bit output changes with the clock, so the color
//constantly changes.  The exact specifics of the inputs
//to these cycle color modules were simply tested to find
//colors that looked appealing.

   cycleColor cC0 (
```

```
      .clk (CLOCK_50),
      .max (6'b111111),
      .changeSig(w7),
      .col (cw)
   );


   wire [3:0] cw1;
      cycleColor4 cC1 (
      .clk (CLOCK_50),
      .max (4'b1111),
      .changeSig(w7),
      .col (cw1)
   );

   wire [3:0] cw2;
      cycleColor4 cC2 (
      .clk (CLOCK_50),
      .max (4'b1111),
      .changeSig(w7),
      .col (cw2)
   );

   wire [7:0] cr;

//Decides which color should be displayed on the screen.
//Either the color determined by the switches, or the color
//determined by one of the cycle color modules

   muxX4 m0 (
      .a ({2'b00, SW[6:1]}),
      .b ({2'b00, cw}),
      .c ({4'b0011, cw1}),
      .d ({2'b00, cw2[3:2], 2'b11, cw2[1:0]}),
      .s (SW[16:15]),
      .f (cr)
   );

   wire scw;


   //This counter is essentially used as a clock, which
   //slightly offset from the internal CLOCK50.
   Scounter sc0 (
      .clk (CLOCK_50),
      .f (scw)
   );


   wire [7:0]ox1w;
   wire [7:0]ox2w;
   wire [7:0]ox3w;

   wire [7:0]oy1w;
   wire [7:0]oy2w;
   wire [7:0]oy3w;


   //Oscillators are used to change the size of what is
   //being drawn.  For example, if the user's cursor is
   //2x2, the program must plot four pixels total.  So the
   //xLocation oscillates between its original position and the
   //position one to the left, and the yLocation oscialltes between
   //its original position and the position one below it.  Since the
   //clock is 50MHZ, even though all these pixels are not plotted at
   //the same time, to the user, their cursor appears to be the desired
   //size.

   OscillatorX1 ox1(
      .clk (CLOCK_50),
      .x (xLoc),
      .f (ox1w)
   );

   OscillatorX2 ox2(
      .clk (CLOCK_50),
```

```verilog
            .x (xLoc),
            .f (ox2w)
        );

        OscillatorX ox3(
            .clk (CLOCK_50),
            .x (xLoc),
            .f (ox3w)
        );

        OscillatorX1 oy1(
            .clk (scw),
            .x ({1'b0, yLoc[6:0]}),
            .f (oy1w)
        );

        OscillatorX2 oy2(
            .clk (scw),
            .x ({1'b0, yLoc[6:0]}),
            .f (oy2w)
        );


        OscillatorX oy3(
            .clk (scw),
            .x ({1'b0, yLoc[6:0]}),
            .f (oy3w)
        );


        wire [7:0] fxLoc;
        wire [7:0] fyLoc;


        //Muxes to select the size of the cursor based
        //on the values of SW[12:11]

        muxX4 mX4(
        .a (xLoc),
        .b (ox1w),
        .c (ox2w),
        .d (ox3w),
        .s (SW[12:11]),
        .f (fxLoc),
        );

        muxX4 mY4(
        .a ({1'b0, yLoc[6:0]}),
        .b (oy1w),
        .c (oy2w),
        .d (oy3w),
        .s (SW[12:11]),
        .f (fyLoc),
        );

        wire [7:0] finalX;
        wire [7:0] finalY;
        wire [7:0] finalColor;

        //These final four muxes are simply used for the erase feature.
        //The erase feature is toggled with SW[17].  When the feature
        //is toggled on, plot is set to 1, the color is set to black, and
        //the location sweeps across the screen.  In this way, the screen is
        //completely plotted black, which erases the screen.


        //Also, for these last four muxes, inputs c and d can never be output,
        //since the s input can only be 00 or 01.

        muxX4 fcm (
            .a (cr),
            .b (8'b00000000),
            .c (8'b00000000),
            .d (8'b00000000),
            .s ({1'b0, SW[17]}),
            .f (finalColor)
```

```
    );

    reg zX = 8'b00000000;
    reg zY = 7'b0000000;

    wire [7:0] runningX;
    wire [7:0] runningY;

    wire modY;

    //These RunX modules are used to determine the location
    //of the cursor when erase mode is toggled on.  rX sweeps
    //the cursor from left to right across the screen, and its
    //output next is 1 when it reaches the very right of the screen.
    //rY uses this next ouput as its clock.  So when the cursor reaches
    //the very right of the screen, the yLocation is incremented by 1.
    //In this way, every pixel of the screen is plotted black.  Because
    //of the speed of the 50MHZ clock, the user does not notice this sweeping
    //across the screen, and simply sees the entire screen immediately erased.

    RunX rX (
       .clk (CLOCK_50),
       .max (8'b11111111),
       .next (modY),
       .f (runningX)
    );

    wire empty;

    RunX rY (
       .clk (modY),
       .max (8'b01111111),
       .next (empty),
       .f (runningY)
    );

    muxX4 fxm (
       .a (fxLoc),
       .b (runningX),
       .c (8'b00000000),
       .d (8'b00000000),
       .s ({1'b0, SW[17]}),
       .f (finalX)
    );

    muxX4 fym (
       .a (fyLoc),
       .b (runningY),
       .c (8'b00000000),
       .d (8'b00000000),
       .s ({1'b0, SW[17]}),
       .f (finalY)
    );

    wire [7:0]finalPlot;


    //This mux ensures that plot is set to 1 when erase mode
    //is on.  If plot was off, then no black would be plotted
    //and the erase feature would not work.
    muxX4 m1x0 (
       .a ({7'b0000000, SW[0]}),
       .b (8'b00000001),
       .c (8'b00000000),
       .d (8'b00000000),
       .s ({1'b0, SW[17]}),
       .f (finalPlot)
    );

    //The vga_adapter is sent all the bits its should have based on the user input.

    vga_adapter VGA(
       .resetn     (1),
       .clock      (CLOCK_50),
       .colour     (finalColor[5:0]),
       .x          (finalX),
```

```
        .y          (finalY[6:0]),
        .plot       (finalPlot[0]),
        .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  = 2;

endmodule

```