```
/*
   Ben Davis
   2/1/24
   EE 371
   Lab 3, Task 2

   This is a module that uses Bresenhams algorithm to draw
   the straightest line possible on a screen with pixels. It
   takes in four inputs, x0 and x1 are both 10 bit inputs and
   are the start and end x coordinates. y0 and y1 are 9 bit inputs,
   and are the start and end y coordinates. The module also takes
   in a clock and reset input. Its only outputs are a 10bit x
   coordinate for the current x value of the pixel drawer, and the
   9 bit y output for the current y coordinate.
*/

module line_drawer(
   input logic clk, reset,

   // x and y coordinates for the start and end points of the line
   input logic [9:0] x0, x1,
   input logic [8:0] y0, y1,

   //outputs cooresponding to the coordinate pair (x, y)
   output logic [9:0] x,
   output logic [8:0] y,
   output logic internal_rst //internal reset to check
                             //for new coordinates when system is done
   );

   /*
    * You'll need to create some registers to keep track of things
    * such as error and direction
    * Example: */
   int error;
   int dx;
   int dy;

   int x_step;
   int y_step;

   assign x_step = (x1 > x0) ? 1 : -1;
   assign y_step = (y1 > y0) ? 1 : -1;

   //state system to keep track of if line drawing is done
   enum {work, done} ps, ns;

   //effectively finding the absolute value of the
   // change in x and change in y
   always_comb begin
      if (x1 > x0) begin
         dx <= x1 - x0;
      end else begin
         dx <= x0 - x1;
      end
      if (y1 > y0) begin
         dy <= y1 - y0;
      end else begin
         dy <= y0 - y1;
      end
   end

   //the drawing of the line segment
   always_ff @(posedge clk) begin
      if(reset) begin
         //sets error for a shallow slope
         if (dx >= dy) begin
            error <= -1 * (dx /2);
         //sets error for a steep slope
         end else begin
            error <= -1 * (dy /2);
         end
         x <= x0;
```

```
            y <= y0;
            ps <= work;
         end else begin
            ps <= ns;

            //check to see if last pixel
            if ((x != x1) || (y != y1)) begin

               //horizontal line
               if (dy == 0) begin
                  x <= x + x_step;

               //vertical line
               end else if (dx == 0) begin
                  y <= y + y_step;

               //slope is less than 1
               end else if (dx > dy) begin
                  x <= x + x_step;
                  error <= error + dy;
                  if ((error >= 0)) begin
                     y <= y + y_step;
                     error <= error - dx;
                  end

               //slope is greater than 1
               end else if(dy > dx) begin
                  y <= y + y_step;
                  error <= error + dx;
                  if ((error >= 0)) begin
                     x <= x + x_step;
                     error <= error - dy;
                  end// of steep

               //slope equals 1
               end else begin
                  x <= x + x_step;
                  y <= y + y_step;
               end
            end else begin
               x <= x;
               y <= y;
            end
         end //of not reset
      end //of ff block

      always_comb begin
         case(ps)
            work: if((x==x1) && (y==y1)) ns <= done;
                     else ns <= work;

            done: ns <= work;

         endcase
      end

      assign internal_rst = (ps==done);


   endmodule
   //testbench
   module line_drawer_testbench();
      //reset logic variables
      logic clk, reset, internal_rst;
      logic [9:0] x0, x1, x;
      logic [8:0] y0, y1, y;

      //reinstantiate module
      line_drawer dut (.clk, .reset, .x0, .x1, .y0, .y1, .x, .y, .internal_rst);

      //clock setup
      parameter clk_period = 100;
      initial begin
```

```
        clk <= 0;
        forever #(clk_period /2) clk <= ~clk;
    end //of clock setup

    //testing an instance where there is an initial reset
    //and the module needs to draw a line of slope 1.
    initial begin
        reset <= 1;
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);
        reset <= 0;
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);
        x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
                             y1 <= 000000110; @(posedge clk);

        $stop; //simulation
    end
endmodule //for testbench
```