

```

1  /*
2  Ben Davis
3  2/1/24
4  EE 371
5  Lab 3, Task 2
6
7  This is a module that uses Bresenham's algorithm to draw
8  the straightest line possible on a screen with pixels. It
9  takes in four inputs, x0 and x1 are both 10 bit inputs and
10 are the start and end x coordinates. y0 and y1 are 9 bit inputs,
11 and are the start and end y coordinates. The module also takes
12 in a clock and reset input. Its only outputs are a 10bit x
13 coordinate for the current x value of the pixel drawer, and the
14 9 bit y output for the current y coordinate.
15 */
16
17 module line_drawer(
18     input logic clk, reset,
19
20     // x and y coordinates for the start and end points of the line
21     input logic [9:0] x0, x1,
22     input logic [8:0] y0, y1,
23
24     // outputs corresponding to the coordinate pair (x, y)
25     output logic [9:0] x,
26     output logic [8:0] y,
27     output logic internal_rst // internal reset to check
28                               // for new coordinates when system is done
29 );
30
31 /*
32  * You'll need to create some registers to keep track of things
33  * such as error and direction
34  * Example: */
35 int error;
36 int dx;
37 int dy;
38
39 int x_step;
40 int y_step;
41
42 assign x_step = (x1 > x0) ? 1 : -1;
43 assign y_step = (y1 > y0) ? 1 : -1;
44
45 // state system to keep track of if line drawing is done
46 enum {work, done} ps, ns;
47
48 // effectively finding the absolute value of the
49 // change in x and change in y
50 always_comb begin
51     if (x1 > x0) begin
52         dx <= x1 - x0;
53     end else begin
54         dx <= x0 - x1;
55     end
56     if (y1 > y0) begin
57         dy <= y1 - y0;
58     end else begin
59         dy <= y0 - y1;
60     end
61 end
62
63 // the drawing of the line segment
64 always_ff @(posedge clk) begin
65     if (reset) begin
66         // sets error for a shallow slope
67         if (dx >= dy) begin
68             error <= -1 * (dx / 2);
69         // sets error for a steep slope
70         end else begin
71             error <= -1 * (dy / 2);
72         end
73     x <= x0;

```

```

74         y <= y0;
75         ps <= work;
76     end else begin
77         ps <= ns;
78
79         //check to see if last pixel
80         if ((x != x1) || (y != y1)) begin
81
82             //horizontal line
83             if (dy == 0) begin
84                 x <= x + x_step;
85
86             //vertical line
87             end else if (dx == 0) begin
88                 y <= y + y_step;
89
90             //slope is less than 1
91             end else if (dx > dy) begin
92                 x <= x + x_step;
93                 error <= error + dy;
94                 if ((error >= 0)) begin
95                     y <= y + y_step;
96                     error <= error - dx;
97                 end
98
99             //slope is greater than 1
100            end else if (dy > dx) begin
101                y <= y + y_step;
102                error <= error + dx;
103                if ((error >= 0)) begin
104                    x <= x + x_step;
105                    error <= error - dy;
106                end // of steep
107
108            //slope equals 1
109            end else begin
110                x <= x + x_step;
111                y <= y + y_step;
112            end
113        end else begin
114            x <= x;
115            y <= y;
116        end
117    end //of not reset
118 end //of ff block
119
120 always_comb begin
121     case(ps)
122         work: if((x==x1) && (y==y1)) ns <= done;
123             else ns <= work;
124
125         done: ns <= work;
126
127     endcase
128 end
129
130 assign internal_rst = (ps==done);
131
132
133 endmodule
134 //testbench
135 module line_drawer_testbench();
136     //reset logic variables
137     logic clk, reset, internal_rst;
138     logic [9:0] x0, x1, x;
139     logic [8:0] y0, y1, y;
140
141     //reinstantiate module
142     line_drawer dut (.clk, .reset, .x0, .x1, .y0, .y1, .x, .y, .internal_rst);
143
144     //clock setup
145     parameter clk_period = 100;
146     initial begin

```

```
147     clk <= 0;
148     forever #(clk_period / 2) clk <= ~clk;
149 end //of clock setup
150
151 //testing an instance where there is an initial reset
152 //and the module needs to draw a line of slope 1.
153 initial begin
154     reset <= 1;
155     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
156         y1 <= 000000110; @(posedge clk);
157     reset <= 0;
158     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
159         y1 <= 000000110; @(posedge clk);
160     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
161         y1 <= 000000110; @(posedge clk);
162     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
163         y1 <= 000000110; @(posedge clk);
164     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
165         y1 <= 000000110; @(posedge clk);
166     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
167         y1 <= 000000110; @(posedge clk);
168     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
169         y1 <= 000000110; @(posedge clk);
170
171     $stop; //simulation
172 end
173 endmodule //for testbench
```