

```

1  /*
2  Ben Davis
3  1/30/24
4  EE 371
5  Lab 3, Task 1
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 );
28
29 /*
30 * You'll need to create some registers to keep track of things
31 * such as error and direction
32 * Example: */
33 logic signed [11:0] error;
34 logic signed [10:0] dx;
35 logic signed [9:0] dy;
36
37 logic signed x_step;
38 logic signed y_step;
39
40 assign x_step = (x1 > x0) ? 1 : -1;
41 assign y_step = (y1 > y0) ? 1 : -1;
42
43 // effectively finding the absolute value of the
44 // change in x and change in y
45 always_comb begin
46     if (x1 > x0) begin
47         dx <= x1 - x0;
48     end else begin
49         dx <= x0 - x1;
50     end
51     if (y1 > y0) begin
52         dy <= y1 - y0;
53     end else begin
54         dy <= y0 - y1;
55     end
56 end
57
58 // the drawing of the line segment
59 always_ff @(posedge clk) begin
60     if (reset) begin
61         // sets error for a shallow slope
62         if (dx >= dy) begin
63             error <= -1 * (dx / 2);
64         // sets error for a steep slope
65         end else begin
66             error <= -1 * (dy / 2);
67         end
68         x <= x0;
69         y <= y0;
70     end else begin
71
72         // check to see if last pixel
73         if ((x < x1) || (y < y1)) begin

```

```

74
75 //horizontal line
76 if (dy == 0) begin
77     x <= x + x_step;
78
79 //vertical line
80 end else if (dx == 0) begin
81     y <= y + y_step;
82
83 //slope is less than 1
84 end else if (dx > dy) begin
85     x <= x + x_step;
86     error <= error + dy;
87     if ((error >= 0)) begin
88         y <= y + y_step;
89         error <= error - dx;
90     end
91
92 //slope is greater than 1
93 end else if (dy > dx) begin
94     y <= y + y_step;
95     error <= error + dx;
96     if ((error >= 0)) begin
97         x <= x + x_step;
98         error <= error - dy;
99     end // of steep
100
101 //slope equals 1
102 end else begin
103     x <= x + x_step;
104     y <= y + y_step;
105 end
106 end else begin
107     x <= x;
108     y <= y;
109 end
110 end //of not reset
111 end //of ff block
112
113 endmodule
114 //testbench
115 module line_drawer_testbench();
116 //reset logic variables
117 logic clk, reset;
118 logic [9:0] x0, x1, x;
119 logic [8:0] y0, y1, y;
120
121 //reinstantiate module
122 line_drawer dut (.clk, .reset, .x0, .x1, .y0, .y1, .x, .y);
123
124 //clock setup
125 parameter clk_period = 100;
126 initial begin
127     clk <= 0;
128     forever #(clk_period / 2) clk <= ~clk;
129 end //of clock setup
130
131 //testing an instance where there is an initial reset
132 //and the module needs to draw a line of slope 1.
133 initial begin
134     reset <= 1;
135     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
136     y1 <= 000000110; @(posedge clk);
137     reset <= 0;
138     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
139     y1 <= 000000110; @(posedge clk);
140     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
141     y1 <= 000000110; @(posedge clk);
142     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
143     y1 <= 000000110; @(posedge clk);
144     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
145     y1 <= 000000110; @(posedge clk);
146     x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;

```

```
147             y1 <= 000000110; @(posedge clk);
148         x0 <= 000000000; x1 <= 000000110; y0 <= 000000000;
149             y1 <= 000000110; @(posedge clk);
150
151         $stop; //simulation
152     end
153 endmodule //for testbench
```