```
1   // Grid module with pattern detection
2
3   module grid (clk, reset, row_select, col_select, set_initial, new_state, enable_update, grid
    , grid_next, pattern_count);
4       input logic clk;
5       input logic reset;
6       input logic [7:0] row_select;
7       input logic [7:0] col_select;
8       input logic set_initial;
9       input logic new_state;
10      input logic enable_update;
11      output logic [15:0][15:0] grid;
12      input logic [15:0][15:0] grid_next;
13      output logic [3:0] pattern_count;
14
15      logic [15:0][15:0] grid_prev;
16      logic [3:0] temp_pattern_count;
17
18      // State registers for grid and pattern count
19      always_ff @(posedge clk or posedge reset) begin
20          if (reset) begin
21              grid <= '{default: 0};
22              grid_prev <= '{default: 0};
23              pattern_count <= 0;
24          end else if (set_initial) begin
25              grid[row_select][col_select] <= new_state;
26          end else if (enable_update) begin
27              grid <= grid_next;
28              grid_prev <= grid;
29              pattern_count <= temp_pattern_count;
30          end
31      end
32
33      // Pattern detection logic for blinkers
34      always_comb begin
35          temp_pattern_count = pattern_count;
36          for (int row = 1; row < 15; row++) begin
37              for (int col = 1; col < 15; col++) begin
38                  // Horizontal blinker check in current state
39                  if (grid[row][col-1] == 1 && grid[row][col] == 1 && grid[row][col+1] == 1)
    begin
40                      // Check for vertical blinker in previous state
41                      if (grid_prev[row-1][col] == 1 && grid_prev[row][col] == 1 && grid_prev[
    row+1][col] == 1) begin
42                          temp_pattern_count = temp_pattern_count + 1;
43                      end
44                  end
45              end
46          end
47      end
48  endmodule
49
50
51  module grid_testbench();
52      logic clk, reset;
53      logic [7:0] row_select, col_select;
54      logic set_initial, new_state, enable_update;
55      logic [15:0][15:0] grid, grid_next;
56      logic [3:0] pattern_count;
57
58      grid dut (clk, reset, row_select, col_select, set_initial, new_state, enable_update,
    grid, grid_next, pattern_count);
59
60      // Set up a simulated clock.
61      parameter CLOCK_PERIOD = 100;
62      initial begin
63          clk <= 0;
64          forever #(CLOCK_PERIOD/2) clk <= ~clk; // Forever toggle the clock
65      end
66
67      // Test the design.
68      initial begin
69          // Reset and configure the grid
```

```
70          reset <= 1;          @(posedge clk); // reset every time we start
71          @(posedge clk);
72          reset <= 0;          @(posedge clk);
73          @(posedge clk);
74
75          // Initial configuration for vertical blinker
76          set_initial <= 1;
77          row_select <= 8'd7; col_select <= 8'd8; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
78          row_select <= 8'd8; col_select <= 8'd8; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
79          row_select <= 8'd9; col_select <= 8'd8; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
80          set_initial <= 0; enable_update <= 1; @(posedge clk); @(posedge clk); enable_update
    <= 0;
81
82          // Let the game run for a cycle
83          repeat (1) @(posedge clk);
84
85          // Update grid to form a horizontal blinker
86          grid_next[7][8] = 0;
87          grid_next[8][7] = 1;
88          grid_next[8][8] = 1;
89          grid_next[8][9] = 1;
90          grid_next[9][8] = 0;
91          enable_update <= 1; @(posedge clk); @(posedge clk); enable_update <= 0;
92
93          // Let the game run for a cycle
94          repeat (1) @(posedge clk);
95
96
97
98          // Reset and configure the grid for another blinker
99          reset <= 1; @(posedge clk); reset <= 0; @(posedge clk);
100
101         // Initial configuration for another vertical blinker
102         set_initial <= 1;
103         row_select <= 8'd4; col_select <= 8'd4; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
104         row_select <= 8'd5; col_select <= 8'd4; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
105         row_select <= 8'd6; col_select <= 8'd4; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
106         set_initial <= 0; enable_update <= 1; @(posedge clk); @(posedge clk); enable_update
    <= 0;
107
108         // Let the game run for a cycle
109         repeat (1) @(posedge clk);
110
111         // Update grid to form a horizontal blinker
112         grid_next[4][4] = 0;
113         grid_next[5][3] = 1;
114         grid_next[5][4] = 1;
115         grid_next[5][5] = 1;
116         grid_next[6][4] = 0;
117         enable_update <= 1; @(posedge clk); @(posedge clk); enable_update <= 0;
118
119         // Let the game run for a cycle
120         repeat (1) @(posedge clk);
121
122
123         // Reset and configure the grid for a glider (not a blinker, should not affect
    pattern_count)
124         reset <= 1; @(posedge clk); reset <= 0; @(posedge clk);
125
126         // Initial configuration for a glider
127         set_initial <= 1;
128         row_select <= 8'd1; col_select <= 8'd1; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
129         row_select <= 8'd2; col_select <= 8'd2; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
130         row_select <= 8'd2; col_select <= 8'd3; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
```

```
131             row_select <= 8'd3; col_select <= 8'd1; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
132             row_select <= 8'd3; col_select <= 8'd2; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
133             set_initial <= 0; enable_update <= 1; @(posedge clk); @(posedge clk); enable_update
    <= 0;
134
135             // Let the game run for a few cycles to see if the pattern count is unaffected
136             repeat (5) @(posedge clk);
137
138
139             // Reset for the next pattern
140             reset <= 1; @(posedge clk); reset <= 0; @(posedge clk);
141
142             // Initial configuration for a horizontal blinker that becomes vertical
143             set_initial <= 1;
144             row_select <= 8'd10; col_select <= 8'd10; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
145             row_select <= 8'd10; col_select <= 8'd11; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
146             row_select <= 8'd10; col_select <= 8'd12; new_state <= 1; @(posedge clk); @(posedge
    clk); // Set cell
147             set_initial <= 0; enable_update <= 1; @(posedge clk); @(posedge clk); enable_update
    <= 0;
148
149             // Let the game run for a cycle
150             repeat (1) @(posedge clk);
151
152             // Update grid to form a vertical blinker
153             grid_next[9][11] = 1;
154             grid_next[10][10] = 0;
155             grid_next[10][11] = 1;
156             grid_next[10][12] = 0;
157             grid_next[11][11] = 1;
158             enable_update <= 1; @(posedge clk); @(posedge clk); enable_update <= 0;
159
160             // Let the game run for a cycle
161             repeat (1) @(posedge clk);
162
163             $stop;
164         end
165     endmodule
166
```