

```

1 // Handles user's config of the initial state before GoL begins to run
2
3 module userInput (clk, reset, KEY, SW0, start_game, row_select, col_select, set_initial,
4 cell_state, GrnPixels);
5     input logic clk, reset;
6     input logic [3:0] KEY;
7     input logic SW0, start_game;
8     output logic [7:0] row_select, col_select;
9     output logic set_initial, cell_state;
10    output logic [15:0][15:0] GrnPixels;
11
12    logic [7:0] row, col;
13    logic [7:0] saved_row, saved_col; // Save cursor position
14    logic [3:0] ps, ns; // present state, next state for each key
15    logic blink_state;
16    logic [31:0] blink_counter;
17    logic key_press_detected;
18    parameter BLINK_INTERVAL = 250000; // Adjust for desired blink speed
19
20    // State encoding
21    enum {off, press} state;
22
23    // Next state logic for each key
24    always_comb begin
25        ns[3] = (KEY[3] ? press : off);
26        ns[2] = (KEY[2] ? press : off);
27        ns[1] = (KEY[1] ? press : off);
28        ns[0] = (KEY[0] ? press : off);
29    end
30
31    // State registers and other logic
32    always_ff @(posedge clk or posedge reset) begin
33        if (reset) begin
34            ps <= 4'b0000;
35            row <= 8'd8; // Starting in the middle
36            col <= 8'd8;
37            saved_row <= 8'd8;
38            saved_col <= 8'd8;
39            cell_state <= 1'b0;
40            blink_counter <= 0;
41            blink_state <= 0;
42            key_press_detected <= 0;
43            set_initial <= 0;
44        end else begin
45            ps <= ns;
46            key_press_detected <= 0;
47
48            // Detect key press
49            if ((ps[3] == off && ns[3] == press) || (ps[2] == off && ns[2] == press) || (ps[
50            1] == off && ns[1] == press) || (ps[0] == off && ns[0] == press)) begin
51                key_press_detected <= 1;
52            end
53            if (key_press_detected) begin
54                set_initial <= 1;
55            end else begin
56                set_initial <= 0;
57            end
58
59            // Cursor Movement
60            if (!start_game) begin
61                if (ps[1] == off && ns[1] == press && row < 8'd15) row <= row + 8'd1; //
62                Move down
63                if (ps[2] == off && ns[2] == press && row > 8'd0) row <= row - 8'd1; //
64                Move up
65                if (ps[3] == off && ns[3] == press && col < 8'd15) col <= col + 8'd1; //
66                Move right
67                if (ps[0] == off && ns[0] == press && col > 8'd0) col <= col - 8'd1; //
68                Move left
69                saved_row <= row;
70                saved_col <= col;
71            end else begin
72                row <= saved_row;
73                col <= saved_col;
74            end
75        end
76    end

```

```

68         end
69
70         // Cell State Toggle
71         if (SW0 == 1) cell_state <= 1'b1; // Set cell on
72         else cell_state <= 1'b0; // Set cell off
73
74         // Blinking Cursor Logic
75         if (!start_game) begin
76             if (blink_counter == BLINK_INTERVAL) begin
77                 blink_counter <= 0;
78                 blink_state <= ~blink_state;
79             end else begin
80                 blink_counter <= blink_counter + 1;
81             end
82         end else begin
83             blink_state <= 0; // Turn off blinking when the game is running
84         end
85     end
86 end
87
88 // Set the current cursor position to blink green
89 always_comb begin
90     GrnPixels = '{default: 0};
91     if (blink_state) begin
92         GrnPixels[row][col] = 1'b1;
93     end
94 end
95
96 assign row_select = row;
97 assign col_select = col;
98 endmodule
99
100
101
102
103
104
105
106
107 module userInput_testbench();
108     logic clk, reset;
109     logic [3:0] KEY;
110     logic SW0;
111     logic [7:0] row_select;
112     logic [7:0] col_select;
113     logic set_initial;
114     logic cell_state;
115     logic [15:0][15:0] GrnPixels;
116
117     userInput dut (clk, reset, KEY, SW0, row_select, col_select, set_initial, cell_state,
118 GrnPixels);
119
120 // Set up a simulated clock.
121 parameter CLOCK_PERIOD = 100;
122 initial begin
123     clk <= 0;
124     forever #(CLOCK_PERIOD/2) clk <= ~clk; // Forever toggle the clock
125 end
126
127 // Test the design.
128 initial begin
129     reset <= 1; @(posedge clk); // reset every time we start
130     @(posedge clk);
131     reset <= 0; @(posedge clk);
132     @(posedge clk);
133
134 // Move the cursor and set cells
135 KEY <= 4'b1110; // Move right
136 SW0 <= 1; // Set cell
137 @(posedge clk); @(posedge clk);
138 KEY <= 4'b1101; // Move down
139 @(posedge clk); @(posedge clk);
140 SW0 <= 0; // Clear cell

```

```
140      @(posedge clk);      @(posedge clk);
141      KEY <= 4'b1111;      // No movement
142      @(posedge clk);      @(posedge clk);
143
144      // Additional test cases
145      KEY <= 4'b1110;      // Move right
146      @(posedge clk);      @(posedge clk);
147      KEY <= 4'b1111;      // No movement
148      @(posedge clk);      @(posedge clk);
149      KEY <= 4'b1101;      // Move down
150      SW0 <= 1;           // Set cell
151      @(posedge clk);      @(posedge clk);
152      KEY <= 4'b1111;      // No movement
153      @(posedge clk);      @(posedge clk);
154      KEY <= 4'b1110;      // Move right
155      @(posedge clk);      @(posedge clk);
156      KEY <= 4'b1111;      // No movement
157      SW0 <= 0;           // Clear cell
158      @(posedge clk);      @(posedge clk);
159      KEY <= 4'b1101;      // Move down
160      @(posedge clk);      @(posedge clk);
161      KEY <= 4'b1111;      // No movement
162      @(posedge clk);      @(posedge clk);
163      KEY <= 4'b1011;      // Move left
164      @(posedge clk);      @(posedge clk);
165      KEY <= 4'b1111;      // No movement
166      SW0 <= 1;           // Set cell
167      @(posedge clk);      @(posedge clk);
168      KEY <= 4'b0111;      // Move up
169      @(posedge clk);      @(posedge clk);
170      KEY <= 4'b1111;      // No movement
171      @(posedge clk);      @(posedge clk);
172
173      $stop;
174  end
175 endmodule
176
```