```
1   // Top-level module that defines the I/Os for the DE-1 SoC board
2   module DE1_SoC (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, SW, LEDR, GPIO_1, CLOCK_50);
3       output logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
4       output logic [9:0]  LEDR;
5       input  logic [3:0]  KEY;
6       input  logic [9:0]  SW;
7       output logic [35:0] GPIO_1;
8       input logic CLOCK_50;
9
10      // Turn off HEX displays
11      assign HEX0 = '1;
12      assign HEX1 = '1;
13      assign HEX2 = '1;
14      assign HEX3 = '1;
15      assign HEX4 = '1;
16      assign HEX5 = '1;
17
18
19      /* Set up system base clock to 1526 Hz (50 MHz / 2**(14+1))
20         ================================================================*/
21      logic [31:0] clk;
22      logic SYSTEM_CLOCK;
23
24      clock_divider divider (.clock(CLOCK_50), .divided_clocks(clk));
25
26      assign SYSTEM_CLOCK = clk[12]; // 1526 Hz clock signal
27
28      /* If you notice flickering, set SYSTEM_CLOCK faster.
29         However, this may reduce the brightness of the LED board. */
30
31
32      /* Set up LED board driver
33         ================================================================ */
34      logic [15:0][15:0]RedPixels; // 16 x 16 array representing red LEDs
35      logic [15:0][15:0]GrnPixels; // 16 x 16 array representing green LEDs
36      logic RST;                   // reset - toggle this on startup
37
38      assign RST = SW[1];
39
40      /* Standard LED Driver instantiation - set once and 'forget it'.
41         See LEDDriver.sv for more info. Do not modify unless you know what you are doing! */
42      LEDDriver Driver (.CLK(SYSTEM_CLOCK), .RST, .EnableCount(1'b1), .RedPixels, .GrnPixels,
    .GPIO_1);
43
44
45      /* LED board test submodule - paints the board with a static pattern.
46         Replace with your own code driving RedPixels and GrnPixels.
47
48         KEY0      : Reset
49         ================================================================ */
50      //LED_test test (.RST(~KEY[0]), .RedPixels, .GrnPixels);
51      // update_cars -> draw_game -> board just by ouputting to Red and Grn Pixels
52      logic [15:0] car2, car5, car7, car9, car11, car12;
53      logic [3:0] frog_x, frog_y;
54
55     // Set up player input
56     // UP:
57     logic up_first_dff;
58     logic up_second_dff;
59     logic up_to_game;
60     flipflop up_raw(.clk(SYSTEM_CLOCK), .reset(RST), .D(~KEY[3]), .Q(up_first_dff));
61     flipflop up_stable(.clk(SYSTEM_CLOCK), .reset(RST), .D(up_first_dff), .Q(up_second_dff));
62     button_press up_input(.clk(SYSTEM_CLOCK), .reset(RST), .in(up_second_dff), .out(
    up_to_game));
63
64     // DOWN:
65     logic down_first_dff;
66     logic down_second_dff;
67     logic down_to_game;
68     flipflop down_raw(.clk(SYSTEM_CLOCK), .reset(RST), .D(~KEY[2]), .Q(down_first_dff));
69     flipflop down_stable(.clk(SYSTEM_CLOCK), .reset(RST), .D(down_first_dff), .Q(
    down_second_dff));
70     button_press down_input(.clk(SYSTEM_CLOCK), .reset(RST), .in(down_second_dff), .out(
    down_to_game));
71
72     // LEFT:
```

```
73      logic left_first_dff;
74      logic left_second_dff;
75      logic left_to_game;
76      flipflop left_raw(.clk(SYSTEM_CLOCK), .reset(RST), .D(~KEY[1]), .Q(left_first_dff));
77      flipflop left_stable(.clk(SYSTEM_CLOCK), .reset(RST), .D(left_first_dff), .Q(
    left_second_dff));
78      button_press left_input(.clk(SYSTEM_CLOCK), .reset(RST), .in(left_second_dff), .out(
    left_to_game));
79
80      // RIGHT:
81      logic right_first_dff;
82      logic right_second_dff;
83      logic right_to_game;
84      flipflop right_raw(.clk(SYSTEM_CLOCK), .reset(RST), .D(~KEY[0]), .Q(right_first_dff));
85      flipflop right_stable(.clk(SYSTEM_CLOCK), .reset(RST), .D(right_first_dff), .Q(
    right_second_dff));
86      button_press right_input(.clk(SYSTEM_CLOCK), .reset(RST), .in(right_second_dff), .out(
    right_to_game));
87
88      logic [3:0] all_player_input;
89      assign all_player_input[1] = up_to_game;
90      assign all_player_input[0] = down_to_game;
91      assign all_player_input[3] = left_to_game;
92      assign all_player_input[2] = right_to_game;
93
94      // Set up game logic / state
95      logic gameover;
96      collision detect(.clk(SYSTEM_CLOCK), .reset(RST), .RedPixels, .frog_x, .frog_y, .gameover
    );
97      update_frog frog(.clk(SYSTEM_CLOCK), .new_game(SW[3]), .reset(RST), .gameover, .
    player_input(all_player_input), .frog_x, .frog_y);
98      update_cars cars(.clk(SYSTEM_CLOCK), .reset(RST), .car2, .car5, .car7, .car9, .car11, .
    car12);
99
100     draw_game   game(.clk(SYSTEM_CLOCK), .reset(RST), .gameover, .car2, .car5, .car7, .car9,
    .car11, .car12, .frog_x, .frog_y, .RedPixels, .GrnPixels);
101
102
103
104   endmodule
105
106   module DE1_SoC_testbench();
107     logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
108     logic [9:0]  LEDR;
109     logic [3:0]  KEY;
110     logic [9:0]  SW;
111     logic [35:0] GPIO_1;
112     logic CLOCK_50;
113
114     DE1_SoC dut(HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, SW, LEDR, GPIO_1, CLOCK_50);
115
116     // Set up the clock.
117     parameter CLOCK_PERIOD=100;
118     initial begin
119     CLOCK_50 <= 0;
120     forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50;
121     end
122
123     // Set up the inputs to the design. Each line is a clock cycle.
124     initial begin
125                                       @(posedge CLOCK_50);
126                                       @(posedge CLOCK_50);
127                                       @(posedge CLOCK_50);
128                                       @(posedge CLOCK_50);
129     // just sending some basic player inputs. first set all keys = 0
130     KEY[0] <= 0; KEY[1] <= 0; KEY[2] <= 0; KEY[3] <= 0;   @(posedge CLOCK_50);
131                                       @(posedge CLOCK_50);
132                                       @(posedge CLOCK_50);
133     KEY[0] <= 1;                                @(posedge CLOCK_50);
134     KEY[0] <= 0;                                @(posedge CLOCK_50);
135     KEY[0] <= 1;                                @(posedge CLOCK_50);
136     KEY[0] <= 0;                                @(posedge CLOCK_50);
137                                       @(posedge CLOCK_50);
138                                       @(posedge CLOCK_50);
139     KEY[1] <= 1;                                @(posedge CLOCK_50);
140     KEY[1] <= 0;                                @(posedge CLOCK_50);
```

```
141       KEY[1] <= 1;                                  @(posedge CLOCK_50);
142       KEY[1] <= 0;                                  @(posedge CLOCK_50);
143                                                 @(posedge CLOCK_50);
144                                                 @(posedge CLOCK_50);
145       KEY[2] <= 1;                                  @(posedge CLOCK_50);
146       KEY[2] <= 0;                                  @(posedge CLOCK_50);
147       KEY[2] <= 1;                                  @(posedge CLOCK_50);
148       KEY[2] <= 0;                                  @(posedge CLOCK_50);
149                                                 @(posedge CLOCK_50);
150                                                 @(posedge CLOCK_50);
151       KEY[3] <= 1;                                  @(posedge CLOCK_50);
152       KEY[3] <= 0;                                  @(posedge CLOCK_50);
153       KEY[3] <= 1;                                  @(posedge CLOCK_50);
154       KEY[3] <= 0;                                  @(posedge CLOCK_50);
155                                                 @(posedge CLOCK_50);
156                                                 @(posedge CLOCK_50);
157                                                 @(posedge CLOCK_50);
158                                                 @(posedge CLOCK_50);
159                                                 @(posedge CLOCK_50);
160                                                 @(posedge CLOCK_50);
161                                                 @(posedge CLOCK_50);
162                                                 @(posedge CLOCK_50);
163
164          $stop; // End the simulation.
165       end
166
167    endmodule
168
```