```
1   module DE1_SoC (CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
2      input  logic CLOCK_50; // 50MHz clock.
3
4      output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
5      output logic [9:0] LEDR;
6      input  logic [3:0] KEY; // True when not pressed, False when pressed
7      input  logic [9:0] SW;
8
9      // Generate clk off of CLOCK_50, whichClock picks rate.
10     logic [31:0] clk;
11     parameter whichClock = 15;
12     clock_divider cdiv (.clock(CLOCK_50),.reset(reset),.divided_clocks(clk));
13
14     // Hook up FSM inputs and outputs.
15     logic  reset, game_reset, Compr_out, victory_l, victory_r;
16     logic [1:0] U_In_out, buttons1, buttons2;
17     logic [2:0] Computer_count, Player_count;
18     logic [9:0] LFSR_out;
19     logic [9:1] leds;
20
21     lfsr_10 LFSR (.clk(clk[whichClock]), .reset(reset), .out(LFSR_out));
22
23     Comparator_10b Compr (.A({1'b0, SW[8:0]}), .B(LFSR_out), .out(Compr_out));
24
25     UserInput U_In (.clk(clk[whichClock]), .reset(game_reset), .buttons(buttons2), .out(
    U_In_out));
26
27     genvar i;
28     generate
29        for (i = 1; i < 10; i++) begin : EachLed
30           if (i == 1)
31              Light2 Led(.clk(clk[whichClock]), .reset(game_reset), .is_center(1'b0), .in({
    U_In_out, leds[i+1],1'b0}),     .out(leds[i]));
32           else if (i == 5)
33              Light2 Led(.clk(clk[whichClock]), .reset(game_reset), .is_center(1'b1), .in({
    U_In_out, leds[i+1],leds[i-1]}),.out(leds[i]));
34           else if (i == 9)
35              Light2 Led(.clk(clk[whichClock]), .reset(game_reset), .is_center(1'b0), .in({
    U_In_out, 1'b0,    leds[i-1]}), .out(leds[i]));
36           else
37              Light2 Led(.clk(clk[whichClock]), .reset(game_reset), .is_center(1'b0), .in({
    U_In_out, leds[i+1],leds[i-1]}),.out(leds[i]));
38        end
39     endgenerate
40
41     Victory2 v  (.clk(clk[whichClock]), .reset, .in({U_In_out, leds[9], leds[1]}), .out({
    victory_l, victory_r}));
42
43     Counter_3b Computer (.clk(clk[whichClock]), .reset, .in(victory_l), .out(Computer_count));
44     Counter_3b Player (.clk(clk[whichClock]), .reset, .in(victory_r), .out(Player_count));
45
46     seg7 left  (.bcd({1'b0, Computer_count}), .leds(HEX5));
47     seg7 right (.bcd({1'b0, Player_count}),   .leds(HEX0));
48
49
50     // Show signals on LEDRs so we can see what is happening.
51     assign reset = SW[9];
52     assign LEDR[9:1] = leds[9:1];
53     assign LEDR[0] = reset;
54     assign game_reset = (victory_l | victory_r | reset);
55
56     // Filter the user input
57     always_ff @(posedge clk[whichClock]) begin
58        buttons1 <= {Compr_out, ~KEY[0]};
59        buttons2 <= buttons1;
60     end
61
62  endmodule
63
64  // divided_clocks[0] = 25MHz, [1] = 12.5Mhz, ... [23] = 3Hz, [24] = 1.5Hz, [25] = 0.75Hz, ...
65  module clock_divider (clock, reset, divided_clocks);
66     input logic reset, clock;
67     output logic [31:0] divided_clocks = 0;
68
69     always_ff @(posedge clock) begin
70        divided_clocks <= divided_clocks + 1;
```

```
71      end
72    endmodule
73
74    module Cyber_War_testbench();
75      logic CLOCK_50;
76      logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
77      logic [3:0] KEY;
78      logic [9:0] LEDR, SW;
79
80      DE1_SoC dut(CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
81
82      parameter CLOCK_PERIOD = 50;
83      initial begin
84        CLOCK_50 <= 0;
85        forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50;
86      end
87
88      initial begin                                                    @(posedge CLOCK_50);
89        SW[9] <= 1;                                                    @(posedge CLOCK_50);
90        SW<= 10'b0000010000;                                           @(posedge CLOCK_50);
91                                                                       @(posedge CLOCK_50);
92                                                                       @(posedge CLOCK_50);
93                                                                       @(posedge CLOCK_50);
94                                                                       @(posedge CLOCK_50);
95                                                                       @(posedge CLOCK_50);
96                                                                       @(posedge CLOCK_50);
97                                                                       @(posedge CLOCK_50);
98                                                                       @(posedge CLOCK_50);
99                                                                       @(posedge CLOCK_50);
100                                                                      @(posedge CLOCK_50);
101                                                                      @(posedge CLOCK_50);
102                                                                      @(posedge CLOCK_50);
103                                                                      @(posedge CLOCK_50);
104                                                                      @(posedge CLOCK_50);
105                                                                      @(posedge CLOCK_50);
106                                                                      @(posedge CLOCK_50);
107       $stop;
108     end
109   endmodule
110
111
112
```