```
 1  module DE1_SoC (CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW, GPIO_0);
 2
 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     input logic CLOCK_50;
 8     inout logic [35:0] GPIO_0;
 9
10     assign GPIO_0[33:0] = 34'd0;
11     assign GPIO_0[35] = enter;
12     assign GPIO_0[34] = exit;
13
14     logic a, b, enter, exit, reset;
15     logic [4:0] counter;
16
17     assign a = SW[1];
18     assign b = SW[0];
19     assign reset = SW[9];
20
21     // doing the clock
22     logic [31:0] clk;
23     parameter whichClock = 10;
24     clock_divider cdiv (CLOCK_50, clk);
25
26     FSM fsm (.clk(clk[whichClock]), .reset, .in({a, b}), .enter, .exit);
27
28     up_counter count (.clk(clk[whichClock]), .reset, .enter, .exit, .counter);
29
30     display disp (.clk(clk[whichClock]), .z(counter), .HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .
   HEX5);
31
32  endmodule
33
34  // divided_clocks[0] = 25MHz, [1] = 12.5Mhz, ... [23] = 3Hz, [24] = 1.5Hz, [25] = 0.75Hz, ...
35  module clock_divider (clock, divided_clocks);
36     input  logic          clock;
37     output logic  [31:0]  divided_clocks = 0;
38
39     always_ff @(posedge clock) begin
40        divided_clocks <= divided_clocks + 1;
41     end
42
43  endmodule
44
45  module DE1_SoC_testbench();
46
47     logic       CLOCK_50; // 50MHz clock.
48     logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
49     logic [9:0] LEDR;
50     logic [3:0] KEY; // True when not pressed, False when pressed
51     logic [9:0] SW;
52     logic [35:0] GPIO_0;
53
54     DE1_SoC dut (.CLOCK_50, .HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .LEDR, .SW, .
   GPIO_0);
55
56     // Set up the clock.
57     parameter CLOCK_PERIOD=100;
58     initial begin
59        CLOCK_50 <= 0;
60        forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50;
61     end
62
63     initial begin
64                                    @(posedge CLOCK_50);
65        SW[9] <= 1;                 @(posedge CLOCK_50);
66        SW[9] <= 0;                 @(posedge CLOCK_50);
67                                    @(posedge CLOCK_50);
68                    SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
69                    SW[1] <= 0; SW[0] <= 1; @(posedge CLOCK_50);
70                    SW[1] <= 1; SW[0] <= 1; @(posedge CLOCK_50);
71                    SW[1] <= 1; SW[0] <= 0; @(posedge CLOCK_50);
72                    SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
73                    SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
```

```
74                      SW[1] <= 0; SW[0] <= 1; @(posedge CLOCK_50);
75                      SW[1] <= 1; SW[0] <= 1; @(posedge CLOCK_50);
76                      SW[1] <= 1; SW[0] <= 0; @(posedge CLOCK_50);
77                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
78                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
79                      SW[1] <= 0; SW[0] <= 1; @(posedge CLOCK_50);
80                      SW[1] <= 1; SW[0] <= 1; @(posedge CLOCK_50);
81                      SW[1] <= 1; SW[0] <= 0; @(posedge CLOCK_50);
82                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
83                                              @(posedge CLOCK_50);
84                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
85                      SW[1] <= 1; SW[0] <= 0; @(posedge CLOCK_50);
86                      SW[1] <= 1; SW[0] <= 1; @(posedge CLOCK_50);
87                      SW[1] <= 0; SW[0] <= 1; @(posedge CLOCK_50);
88                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
89                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
90                      SW[1] <= 1; SW[0] <= 0; @(posedge CLOCK_50);
91                      SW[1] <= 1; SW[0] <= 1; @(posedge CLOCK_50);
92                      SW[1] <= 0; SW[0] <= 1; @(posedge CLOCK_50);
93                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
94                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
95                      SW[1] <= 1; SW[0] <= 0; @(posedge CLOCK_50);
96                      SW[1] <= 1; SW[0] <= 1; @(posedge CLOCK_50);
97                      SW[1] <= 0; SW[0] <= 1; @(posedge CLOCK_50);
98                      SW[1] <= 0; SW[0] <= 0; @(posedge CLOCK_50);
99                                              @(posedge CLOCK_50);
100                                             @(posedge CLOCK_50);
101
102         $stop; // End the simulation.
103     end
104 endmodule
```