```
1   // Eugene Ngo
2   // 1/20/2023
3   // EE 371
4   // Lab 2 Task 1
5
6   // DE1_SoC is the top-level module that defines the I/Os for the DE-1 SoC board.
7   // DE1_SoC uses switches SW 3:0 to provide input data for the RAM and switches SW 8:4 to
    specify the address
8   // for the RAM module. It uses SW9 as the Write signal and KEY0 as the clk input. Using
    hexadecimal values,
9   // it show the address value on the 7-segment displays HEX5:4, shows the data being input
    to the memory on HEX2,
10  // and shows the data read out of the memory on HEX0.
11
12  module DE1_SoC_task1 (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, SW, CLOCK_50);
13      output logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
14      input  logic [3:0]  KEY;
15      input  logic [9:0]  SW;
16      input logic CLOCK_50;
17
18     // Assiging default off value to HEX3 and HEX1
19     assign HEX3 = 7'b1111111;
20     assign HEX1 = 7'b1111111;
21
22     // Assigning clk and reset
23     logic clk, reset;
24     assign clk = ~KEY[0];
25     assign reset = ~KEY[3];
26
27     // Establishing data_in, data_out, and addr_in as intermediate connections
28     logic [3:0] data_in, data_out;
29     logic [4:0] addr_in;
30     assign data_in = SW[3:0];
31     assign addr_in = SW[8:4];
32
33     // ram32x4 takes addr_in, write, and data_in, and serves as a synchronous read/write
    32x4 RAM module.
34     // It sets all the stored data to 0 on reset.
35     RAM ram32x4 (.addr_in, .data_in, .data_out, .write(SW[9]), .clk, .reset);
36
37     // hd1 takes addr_in, data_in, and data_out, and displays those values in hexadecimal to
    the
38     // corresponding hex display.
39     seg7 display (.addr_in, .data_in, .data_out, .HEX5, .HEX4, .HEX2, .HEX0);
40
41  endmodule
42
43  // DE1_SoC_task1_testbench tests all expected, unexpected, and edgecase behaviors
44  module DE1_SoC_task1_testbench();
45     logic [6:0]  HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
46     logic [3:0]  KEY;
47     logic [9:0]  SW;
48     logic CLOCK_50;
49
50     DE1_SoC_task1 dut (.HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .SW, .CLOCK_50);
51
52     // Setting up logic to make testbench cleaner
53     logic [4:0] addr_in;
54     logic [3:0] data_in;
55     logic write, clk, reset;
56     assign SW[3:0] = data_in;
57     assign SW[8:4] = addr_in;
58     assign SW[9] = write;
59     assign KEY[0] = ~clk;
60     assign KEY[3] = ~reset;
61
62     // Setting up a simulated clk.
63     parameter CLOCK_PERIOD = 100;
64     initial begin
65        CLOCK_50 = 0;
66        forever #(CLOCK_PERIOD/2) CLOCK_50 = ~CLOCK_50; // Forever toggle the clk
67     end
68
```

```
69      // Trying all combinations of inputs (x and y).
70      initial begin
71         // resetting the module
72         reset = 1; #10;
73
74         // writing some random data into the RAM
75         reset = 0; addr_in = 5'b00001; data_in = 4'b0001; write = 1'b1; #10;
76                    clk = 1; #10; clk = 0; #10; // simulated clk pulse
77                    addr_in = 5'b00010; data_in = 4'b0010; write = 1'b1; #10;
78                    clk = 1; #10; clk = 0; #10;
79                    addr_in = 5'b00011; data_in = 4'b0011; write = 1'b1; #10;
80                    clk = 1; #10; clk = 0; #10;
81                    addr_in = 5'b11111; data_in = 4'b1111; write = 1'b1; #10;
82                    clk = 1; #10; clk = 0; #10;
83
84         // reading the previously written data from the RAM
85                    addr_in = 5'b00001; data_in = 4'b1010; write = 1'b0; #10;
86                    clk = 1; #10; clk = 0; #10;
87                    addr_in = 5'b00010; data_in = 4'b1010; write = 1'b0; #10;
88                    clk = 1; #10; clk = 0; #10;
89                    addr_in = 5'b00011; data_in = 4'b1010; write = 1'b0; #10;
90                    clk = 1; #10; clk = 0; #10;
91                    addr_in = 5'b11111; data_in = 4'b1010; write = 1'b0; #10;
92                    clk = 1; #10; clk = 0; #10;
93         $stop;
94      end
95   endmodule
```