

```

1  /*
2   Ben Davis
3   1/22/24
4   EE 371
5   Lab 2, Task 1
6
7   This module implements a 32x4 ram and uses the hexadecimal
8   module. It takes in CLOCK_50, KEY0 as a reset, SW9 as a write
9   enable, SW3-0 as the input data, and SW8-4 as the data address.
10  It also outputs the data output on hex 0, the input data on
11  hex 2, and the data address on hex 4 and 5. All hex data is
12  displayed in hexadecimal format.
13
14  */
15
16  module DE1_SoC_RAM_32x4 (SW, KEY, HEX0, HEX2, HEX4, HEX5);
17      //setup logic variables for FPGA
18      input logic [9:0] SW;
19      input logic [3:0] KEY;
20      output logic [6:0] HEX0;
21      output logic [6:0] HEX2;
22      output logic [6:0] HEX4;
23      output logic [6:0] HEX5;
24      //intermediate logic
25      logic [3:0] out;
26      //instantiate the ram
27      RAM_32x4 task1_ram (.clk(KEY[0]), .wen(SW[9]), .data_in(SW[3:0]),
28                          .data_addr(SW[8:4]), .data_out(out));
29      //conversion of binary data to hexadecimal format
30      //on 7 segment active low hex displays
31      //also use of more intermediate data
32      logic [6:0] z_out;
33      hexadecimal zero (.in(out), .out(z_out));
34
35      logic [6:0] two_out;
36      hexadecimal two (.in(SW[3:0]), .out(two_out));
37
38      logic [6:0] four_out;
39      hexadecimal four (.in(SW[7:4]), .out(four_out));
40
41      logic [6:0] five_out;
42      hexadecimal five (.in(SW[8]), .out(five_out));
43      //assignment of hex displays
44      assign HEX0 = z_out;
45      assign HEX2 = two_out;
46      assign HEX4 = four_out;
47      assign HEX5 = five_out;
48
49  endmodule
50  //testbench
51  module DE1_SoC_RAM_32x4_testbench();
52      //reset logic variables
53      logic [9:0] SW;
54      logic [3:0] KEY;
55      logic [6:0] HEX0;
56      logic [6:0] HEX2;
57      logic [6:0] HEX4;
58      logic [6:0] HEX5;
59      //instantiate a case of this module
60      DE1_SoC_RAM_32x4 dut (.SW, .KEY, .HEX0, .HEX2, .HEX4, .HEX5);
61
62      // clock setup with KEY 0
63      parameter clock_period = 100;
64
65      initial begin
66          KEY[0] <= 0;
67          forever #(clock_period / 2) KEY[0] = ~KEY[0];
68      end // of clock setup
69
70      //this instance tests two pieces of data being written into the
71      //ram and then being read immediately afterwards
72      initial begin
73          SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 1; @(posedge KEY[0]);

```

```
74      SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
75      SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
76      SW[8:4] <= 5'b00010; SW[3:0] <= 4'b0011; SW[9] <= 1; @(posedge KEY[0]);
77      SW[8:4] <= 5'b00010; SW[3:0] <= 4'b0011; SW[9] <= 0; @(posedge KEY[0]);
78      SW[8:4] <= 5'b00010; SW[3:0] <= 4'b0011; SW[9] <= 0; @(posedge KEY[0]);
79      SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
80      SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
81      $stop;
82      end
83      endmodule
```