```
1   // Eugene Ngo
2   // 1/20/2023
3   // EE 371
4   // Lab 2 Task 1
5
6   // the RAM module is a 32x4 RAM that implements synchronous read/write functionalities
7
8   module RAM #(parameter ADDR_WIDTH=5, parameter DATA_WIDTH=4) (addr_in, data_in, data_out,
    write, clk, reset);
9
10      input logic [ADDR_WIDTH-1:0] addr_in;
11      input logic [DATA_WIDTH-1:0] data_in;
12      input logic write, clk, reset;
13      output logic [DATA_WIDTH-1:0] data_out;
14
15      // Initiating the 2D array for the memory module
16      logic [2**ADDR_WIDTH-1:0][DATA_WIDTH-1:0] memory_array;
17
18      always_ff @(posedge clk) begin
19          // Implementing reset logic
20          if (reset)
21              memory_array <= '0;
22          // Implementing write logic
23          else if (write) begin
24              memory_array[addr_in] <= data_in;
25              data_out <= memory_array[addr_in];
26          end
27          // Implementing read logic
28          else
29              data_out <= memory_array[addr_in];
30      end
31
32  endmodule
33
34  // RAM_testbench tests all expected, unexpected, and edgecase behaviors
35  module RAM_testbench();
36      logic [4:0] addr_in;
37      logic [3:0] data_in;
38      logic write, reset;
39      logic [3:0] data_out;
40      logic CLOCK_50;
41
42      RAM dut (.addr_in, .data_in, .data_out, .write, .clk(CLOCK_50), .reset);
43
44      // Setting up a simulated clock.
45      parameter CLOCK_PERIOD = 100;
46      initial begin
47          CLOCK_50 <= 0;
48          forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // Forever toggle the clock
49      end
50
51      initial begin
52          // resetting the module
53          reset <= 1; repeat(5) @(posedge CLOCK_50);
54
55          // writing some random data into the RAM
56          reset <= 0; addr_in <= 5'b00001; data_in <= 4'b0001; write <= 1'b1; @(posedge CLOCK_50
    );
57                      addr_in <= 5'b00010; data_in <= 4'b0010; write <= 1'b1; @(posedge CLOCK_50
    );
58                      addr_in <= 5'b00011; data_in <= 4'b0011; write <= 1'b1; @(posedge CLOCK_50
    );
59                      addr_in <= 5'b11111; data_in <= 4'b1111; write <= 1'b1; @(posedge CLOCK_50
    );
60
61          // reading the previously written data from the RAM
62                      addr_in <= 5'b00001; data_in <= 4'b1010; write <= 1'b0; @(posedge CLOCK_50
    );
63                      addr_in <= 5'b00010; data_in <= 4'b1010; write <= 1'b0; @(posedge CLOCK_50
    );
64                      addr_in <= 5'b00011; data_in <= 4'b1010; write <= 1'b0; @(posedge CLOCK_50
    );
65                      addr_in <= 5'b11111; data_in <= 4'b1010; write <= 1'b0; @(posedge CLOCK_50
```

```
    );
66          $stop;
67      end
68  endmodule
```