```
1   // Eugene Ngo
2   // 1/20/2023
3   // EE 371
4   // Lab 2 Task 3
5
6   // FIFO module implements a 16x8 queue-like FIFO structure.
7   // When a read signal is received, FIFO stores the data on the input bus if the FIFO is not
    full.
8   // when a write signal is received, FIFO outputs the least recent stored value onto the
    output bus,
9   // and removes that value from the FIFO.
10
11  module FIFO #(
12                parameter depth = 4,
13                parameter width = 8
14                )(
15                  input logic clk, reset,
16                  input logic read, write,
17                  input logic [width-1:0] inputBus,
18                 output logic empty, full,
19                 output logic [width-1:0] outputBus
20                 );
21
22     // variables for interconnections and communication between the RAM and the controller
23     logic [3:0] addr_r, addr_w;
24     logic wr_en;
25
26
27     // Instantiation of the 16x8 RAm used for the FIFO
28     ram16x8 RAM (.clock(clk), .data(inputBus), .rdaddress(addr_r), .wraddress(addr_w), .wren(
    wr_en), .q(outputBus));
29
30     // Instantiation of the controller for the FIFO
31     FIFO_Control #(depth) FC (.clk, .reset,
32                               .read,
33                               .write,
34                               .wr_en,
35                               .empty,
36                               .full,
37                               .readAddr(addr_r),
38                               .writeAddr(addr_w)
39                              );
40
41  endmodule
42
43
44  // FIFO_testbench tests all expected, unexpected, and edgecase behaviors
45  `timescale 1 ps / 1 ps
46  module FIFO_testbench();
47
48     parameter depth = 4, width = 8;
49
50     logic clk, reset;
51     logic read, write;
52     logic [width-1:0] inputBus;
53     logic empty, full;
54     logic [width-1:0] outputBus;
55
56     // Instantiation of dut
57     FIFO #(depth, width) dut (.*);
58
59     // Generate a 50MHz clock
60     parameter CLK_Period = 100;
61     initial begin
62        clk <= 1'b0;
63        forever #(CLK_Period/2) clk <= ~clk;
64     end
65
66     initial begin
67        // reset the FIFO module
68        reset <= 1;                                          repeat(2) @(posedge clk);
69        reset <= 0;                                          repeat(2) @(posedge clk);
70           write <= 1'b1; read <= 1'b0;                               @(posedge clk);
```

```
71          // fill up the entire memory w/ 8'h00 through 8'h0f
72                                          inputBus <= 8'h00;          @(posedge clk);
73                                          inputBus <= 8'h01;          @(posedge clk);
74                                          inputBus <= 8'h02;          @(posedge clk);
75                                          inputBus <= 8'h03;          @(posedge clk);
76                                          inputBus <= 8'h04;          @(posedge clk);
77                                          inputBus <= 8'h05;          @(posedge clk);
78                                          inputBus <= 8'h06;          @(posedge clk);
79                                          inputBus <= 8'h07;          @(posedge clk);
80                                          inputBus <= 8'h08;          @(posedge clk);
81                                          inputBus <= 8'h09;          @(posedge clk);
82                                          inputBus <= 8'h0a;          @(posedge clk);
83                                          inputBus <= 8'h0b;          @(posedge clk);
84                                          inputBus <= 8'h0c;          @(posedge clk);
85                                          inputBus <= 8'h0d;          @(posedge clk);
86                                          inputBus <= 8'h0e;          @(posedge clk);
87                                          inputBus <= 8'h0f;          @(posedge clk);
88          // attempt to put in more data after the memory is full
89                                          inputBus <= 8'h10;          @(posedge clk);
90                                          inputBus <= 8'h11;          @(posedge clk);
91                                          inputBus <= 8'h12;          @(posedge clk);
92             write <= 1'b0;                                           @(posedge clk);
93          // clear out the entire memory & attempt to read more data
94                           read <= 1'b1;                  repeat(20) @(posedge clk);
95          // simultaneous read and write operations after writing
96                                           inputBus <= 8'h00;          @(posedge clk);
97             write <= 1'b1; read <= 1'b0;  inputBus <= 8'h00;          @(posedge clk);
98                                           inputBus <= 8'h01;          @(posedge clk);
99                                           inputBus <= 8'h02;          @(posedge clk);
100                                          inputBus <= 8'h03;          @(posedge clk);
101                                          inputBus <= 8'h04;          @(posedge clk);
102                                          inputBus <= 8'h05;          @(posedge clk);
103            write <= 1'b1; read <= 1'b1;  inputBus <= 8'h06;          @(posedge clk);
104                                          inputBus <= 8'h07;          @(posedge clk);
105                                          inputBus <= 8'h08;          @(posedge clk);
106                                          inputBus <= 8'h09;          @(posedge clk);
107                                          inputBus <= 8'h0a;          @(posedge clk);
108                                          inputBus <= 8'h0b;          @(posedge clk);
109                                          inputBus <= 8'h0c;          @(posedge clk);
110                                          inputBus <= 8'h0d;          @(posedge clk);
111                                          inputBus <= 8'h0e;          @(posedge clk);
112                                          inputBus <= 8'h0f;          @(posedge clk);
113         $stop;
114      end
115   endmodule
```