```
1   // Eugene Ngo
2   // 1/20/2023
3   // EE 371
4   // Lab 2 Task 3
5
6   // FIFO_Control module controls the FIFO. It sends out empty and full signals based on the
    state of the FIFO.
7   // It sends out an enable signal to the RAM when a data is to be written into the FIFO. It
    remembers the location
8   // of the least recent data and the location where the new data is supposed to be stored,
    and it passes those
9   // information to the RAM when they are needed.
10
11  module FIFO_Control #(
12                        parameter depth = 4
13                        )(
14                          input logic clk, reset,
15                          input logic read, write,
16                         output logic wr_en,
17                         output logic empty, full,
18                         output logic [depth-1:0] readAddr, writeAddr
19                         );
20
21     // number of elements in the FIFO structure
22     integer n;
23     // address of the least recent element
24     integer f;
25
26     // Implementing the main logic of the FIFO controller
27     always_ff @(posedge clk) begin
28        // reset
29        if (reset) begin
30            readAddr <= '0;
31           writeAddr <= '0;
32               empty <= 1'b1;
33                full <= 1'b0;
34                wr_en <= 1'b0;
35                n <= 0;
36                f <= 0;
37        end
38        else if (read | write) begin
39           // simultaneous operation
40           if (read & write) begin
41              readAddr <= f;
42              wr_en <= 1'b1;
43              f <= (f+1)%(2**depth);
44              writeAddr <= (f+n)%(2**depth);
45           end
46           else begin
47              // read
48              if (read & ~empty) begin
49                 if (n == 1) begin
50                    full <= 1'b0;
51                    empty <= 1'b1;
52                 end
53                 wr_en <= 1'b0;
54                 full <= 1'b0;
55                 readAddr <= f;
56                 n <= n - 1;
57                 f <= (f+1)%(2**depth);
58              end
59              // write
60              else if (write & ~full) begin
61                 if (n == (2**depth - 1)) begin
62                    full <= 1'b1;
63                    empty <= 1'b0;
64                 end
65                 wr_en <= 1'b1;
66                 empty <= 1'b0;
67                 writeAddr <= (f+n)%(2**depth);
68                 n <= n + 1;
69              end
70              else
```

```
71                  wr_en <= 1'b0;
72              end
73          end
74          else
75              wr_en <= 1'b0;
76      end
77  endmodule
78
79  // FIFO_Control_testbench tests all expected, unexpected, and edgecase behaviors
80  module FIFO_Control_testbench();
81
82      parameter depth = 4;
83
84      logic clk, reset;
85      logic read, write;
86      logic wr_en;
87      logic [depth-1:0] readAddr, writeAddr;
88      logic empty, full;
89
90      // Instantiation of dut
91      FIFO_Control #(depth) dut (.*);
92
93      // Generate a 50MHz clock
94      parameter CLK_Period = 100;
95      initial begin
96          clk <= 1'b0;
97          forever #(CLK_Period/2) clk <= ~clk;
98      end
99
100     initial begin
101         // reset the FIFO module
102         reset <= 1;                                         repeat(2) @(posedge clk);
103         reset <= 0;                                         repeat(2) @(posedge clk);
104             // write 20 times (over the size of the RAM)
105             write <= 1'b1; read <= 1'b0;                    repeat(20) @(posedge clk);
106             // read 20 times (over the size of the RAM)
107             write <= 1'b0; read <= 1'b1;                    repeat(20) @(posedge clk);
108             // write 5 times then read and write simultaneously for 10 cycles
109             write <= 1'b1; read <= 1'b0;                    repeat(5) @(posedge clk);
110             write <= 1'b1; read <= 1'b1;                    repeat(10) @(posedge clk);
111         $stop;
112     end
113 endmodule
```