```
1   /*-------------------------------------------------------------------------------
2   Name Ron Kalin
3   CLass: EE417 Summer 2024
4   Lesson 07 HW Question 01
5   Group: Ron Kalin/ Lamin Jammeh
6   Project Description: test-bench for sequential multiplier
7   -------------------------------------------------------------------------------*/
8   module Sequential_multiplier_tb ();
9
10  //set the parameters wires and registers
11  parameter      word_size = 4;  //bit length of word inputs
12  parameter      half_cycle = 5; //half cycle time of clock
13  parameter      full_cycle = 10;//full cycle time of clock
14  parameter      cycle_time = 160;//number of cycles before next cycle
15  /*top level module under test original declaration
16  module Sequential_multiplier (product, final_product,
17                                Ready, start,
18                               word1, word2,
19                               clk, rst);*/
20  //define outputs as wires, inputs as registers
21  wire [2*word_size-1: 0] product, final_product;
22  wire                    Ready;
23  wire [word_size-2:0]    stateProbe2; //internal probe wire for troubleshooting
24  wire [word_size-1:0]    multiplierProbe;
25  wire [2*word_size-1: 0] multiplicandProbe;
26  reg  [word_size -1: 0]  multiplier2;
27  reg  [2*word_size-1: 0] multiplicand1;
28  reg                     start, clk, rst;
29  integer cycles;
30  //define the unit under test UUT
31  Sequential_multiplier UUT (product, final_product,
32                             Ready, start,
33                             multiplicand1, multiplier2,
34                             clk, rst);
35  //internal probes to track logic and troubleshoot
36  //assign stateProbe = UUT.state;
37  assign stateProbe2 = UUT.M2.state; //UUT=top module, M2 =submodule instance name, state is
    register
38  assign multiplierProbe   = UUT.M1.multiplier;
39  assign multiplicandProbe = UUT.M1.multiplicand;
40
41  //instantiate clock
42  initial
43      begin: clock_loop
44        clk = 1'b1;
45        cycles = 0;
46        forever begin
47           #half_cycle    clk = ~clk;
48           cycles = cycles + 1;
49        end
50        if (cycles == cycle_time) disable clock_loop;
51      end
52
53  //define input words and observe the outputs
54  initial begin
55    start = 1;  //start to high means start process
56    rst = 1;    //reset high will set everything to zero and ready to high
57    multiplicand1 = 8'b0000_0101; //initialize both words random test values
58    multiplier2   = 4'b0101;
59    #10 rst = 0;//reset low
60    #cycle_time
61    rst =1;
62    multiplicand1 = 8'b0000_1111; //initialize both words random test values
63    multiplier2   = 4'b1101;
64    #10 rst = 0;
65    #cycle_time //disable clock_loop;
66  //end
67    multiplier2 = 4'b0000;
68    multiplicand1 = 8'b0000_0000;
```

```
69      forever begin: input_loop
70       multiplier2 = multiplier2 + 1'b1;  //increment multiplier
71       if (multiplier2 == 4'b1111) begin      //if multiplier reaches max value
72           multiplier2 = 4'b0001;               //reset multiplier to one
73           multiplicand1 = multiplicand1 + 1'b1;//incr multiplicand
74       end
75       if (multiplicand1 == 8'b1111_1111 && multiplier2 == 4'b1111) begin //both reach max value
76          disable input_loop; disable clock_loop; end
77       #cycle_time;
78      end
79    end
80    ////monitor and display the output
81    initial begin
82    $monitor("multiplicand1 = %b: multiplier2 = %b: stateProbe2=%d: product=%d:
      final_product=%d:",multiplicand1, multiplier2,
83       stateProbe2,product, final_product);
84       end
85    endmodule
```