```
/*-----------------------------------------------------------------------------
Name Ron Kalin
Class: EE417 Summer 2024
Lesson 08 HW Question 02
Group: Ron Kalin/ Lamin Jammeh
Project Description: test-bench for UART receiver
-------------------------------------------------------------------------------*/
module UART_RCVR_tb ();

//set parameters
parameter  half_cycle = 5; //half cycle time of clock
parameter  full_cycle = 10;//full cycle time of clock
parameter  cycle_time = 160;//number of cycles before next cycle
parameter  word_size = 8, half_word = word_size /2; //bit length of word inputs
/*top level module under test original declaration
module UART_RCVR #(parameter word_size = 8, half_word = word_size /2)
(output [word_size -1: 0] RCV_datareg,
 output read_not_ready_out, Error1, Error2,
 input Serial_in,  read_not_ready_in,  Sample_clk,  rst_b );*/
//define outputs as wires, inputs as registers
wire [word_size -1: 0] RCV_datareg;
wire read_not_ready_out , Error1, Error2;
reg  Serial_in,  read_not_ready_in,  Sample_clk,  rst_b;

//input/output wire submodwires to monitor the inputs and outputs of submodules
wire        Ser_in_0;
wire        clr_Sample_counter;
wire        inc_Sample_counter;
wire        clr_Bit_counter;
wire        inc_Bit_counter;
wire [word_size-1:0]  RCV_shftreg;
wire [3:0]  Sample_counter;
wire [4:0]  Bit_counter;
wire        SC_eq_3;
wire        SC_lt_7;
wire [1:0]  state;
wire        shift;
wire        load;
wire        BC_eq_8;

//define the unit under test UUT
UART_RCVR #(word_size, half_word) UUT (
                      .RCV_datareg(RCV_datareg),
                      .read_not_ready_out(read_not_ready_out),
                      .Error1(Error1),
                      .Error2(Error2),

                      .Serial_in(Serial_in),
                      .read_not_ready_in(read_not_ready_in),
                      .Sample_clk(Sample_clk),
                      .rst_b(rst_b)
                      );

//internal probe monitors (M0=Control Unit, M1=Datapath Unit)
assign Ser_in_0             = UUT.M1.Ser_in_0;
assign clr_Sample_counter   = UUT.M1.clr_Sample_counter;
assign inc_Sample_counter   = UUT.M1.inc_Sample_counter;
assign clr_Bit_counter      = UUT.M1.clr_Bit_counter;
assign inc_Bit_counter      = UUT.M1.inc_Bit_counter;
assign RCV_shftreg          = UUT.M1.RCV_shftreg;
assign Sample_counter       = UUT.M1.Sample_counter;
assign Bit_counter          = UUT.M1.Bit_counter;
assign SC_lt_7              = UUT.M0.SC_lt_7;
assign SC_eq_3              = UUT.M0.SC_eq_3;
assign state                = UUT.M0.state;
assign shift                = UUT.M0.shift;
assign load                 = UUT.M0.load;
assign BC_eq_8              = UUT.M0.BC_eq_8;

//clock cycle
```

```
always
   begin
      Sample_clk = 0;
      forever #half_cycle Sample_clk = ~Sample_clk;
   end

//initialize reset, run sufficent clk cycles to get all desired counts
initial
   begin
      // Initialize Inputs
     Serial_in = 0;
     read_not_ready_in = 0;
     Sample_clk = 0;
     rst_b = 0;

     // Apply reset
     rst_b = 1;
     #10;
     rst_b = 0;
     #10;
     rst_b = 1;  //reset high means system active

     // Test Case 1: Transmit byte (ex: 8'b01010101)
     Serial_in = 1; // Start bit
     #100;

     Serial_in = 1; // Bit 0
     #100;

     Serial_in = 0; // Bit 1
     #100;

     Serial_in = 1; // Bit 2
     #100;

     Serial_in = 0; // Bit 3
     #100;

     Serial_in = 1; // Bit 4
     #100;

     Serial_in = 0; // Bit 5
     #100;

     Serial_in = 1; // Bit 6
     #100;

     Serial_in = 0; // Bit 7
     #100;

     Serial_in = 1; // Stop bit
     #100;

      // Force load signal to load the data
     force UUT.M0.load = 1;
     #100;
     force UUT.M0.load = 0;
     release UUT.M0.load; //release load signal
     #(60 * full_cycle);  //wait 60 cycles
     $stop;  //opens debug window, close debug window to see waveform
       begin
          $monitor ($time ,, "Serial_in = %h  RCV_datareg = %h" , Serial_in, RCV_datareg);
       end
   end
endmodule
```