```
/*-------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 08 HW Question 2
Group: Ron Kalin/ Lamin Jammeh
Project Description: Control_Unit for Reciever (Rx) will control the datapath for the
UART Rx. Once the Data is Transmitted by the Host processor, the Control_unit will
perform several checks before allowing the Datapath to load the recieved data
-------------------------------------------------------------------------------*/

module Control_Unit_Rx  #(parameter word_size = 8,
                          parameter half_word = word_size/2,
                          parameter Num_state_bits = 2
                         )
                        //state the outputs of the COntroller Unit
                        (output reg read_not_ready_out,
                         output reg Error1,
                         output reg Error2,
                         output reg clr_Sample_counter,
                         output reg inc_Sample_counter,
                         output reg clr_Bit_counter,
                         output reg inc_Bit_counter,
                         output reg shift,
                         output reg load,
                       //define the inputs of the Controller Unit
                         input read_not_ready_in,
                         input Ser_in_0,
                         input SC_eq_3,
                         input SC_lt_7,
                         input BC_eq_8,
                         input Sample_clk,
                         input rst_b
                         );

//Parameterize the different states of the Controller Unit as a 2-bit one-hot counter
parameter idle      = 2'b00;
parameter starting  = 2'b01;
parameter receiving = 2'b10;

//define the internal registers for state, next_stae and shift_reg
reg   [word_size-1:0]   Rx_shftreg;             //creates a temp register of 8-bits to
store the received data
reg   [Num_state_bits-1:0]  state, next_state;     //creates temp register of 2-bits for
the stae transition

//state Transition logic
always @(posedge Sample_clk)
   if   (rst_b == 1'b0)
        state <= idle;
   else state <= next_state;
   /*---------------   for the State Transition--------------------------
   ---if reset is low (0) the controller unit will remain in idle
   ---if reset is high (1) the controller will move to the next-state----*/

//create a initial or default condition for the controller
always @(state, Ser_in_0, SC_eq_3, SC_lt_7, read_not_ready_in)
   begin
      read_not_ready_out = 0;
      clr_Sample_counter = 0;
      clr_Bit_counter    = 0;
      inc_Sample_counter = 0;
      inc_Bit_counter    = 0;
      shift              = 0;
      Error1             = 0;
      Error2             = 0;
      load               = 0;
      next_state         = idle;
      //create the Next State logic
      case (state)
```

```
68            idle: if   (Ser_in_0 == 1'b1)
69                         next_state = starting;
70                  else next_state = idle;
71          /*-----------------@ idle------------------------------
72          --- if Ser_in_0 is high (1) move to starting state------
73          --- if Ser_in_0 is low (0) remain idle-----------------*/
74            starting: if (Ser_in_0 ==1'b0)
75                        begin
76                          next_state = idle;
77                          clr_Sample_counter = 1;
78                        end                          //Keep receiving the Ser_in_0 = 1'b0 until
    the next else if condition is met
79                      else if (SC_eq_3 ==1'b1)
80                        begin
81                          next_state = receiving;
82                          clr_Sample_counter = 1;
83                        end                          //once Sample_counter = 3 then move to the
    receiving state and clr_Sample_counter
84                       else
85                          begin
86                             inc_Sample_counter = 1;
87                             next_state = starting;
88                          end                  //  when the if conditons are false remain @
    starting and increment the Sample_counter
89            receiving: if (SC_lt_7 == 1'b1)
90                          begin
91                             inc_Sample_counter = 1;
92                             next_state = receiving;
93                          end                  //Keep receiving data as long as SC less than
    7 is high (1) and remain the receiving state
94                        else                   //once SC_lt_7 drops low the current data
    package is completed
95                          begin
96                             clr_Sample_counter = 1;    //clr the Sample_counter and check
    the next if condition
97                             if (!BC_eq_8)              //if BC_eq_8 is not true or low
98                                begin
99                                   shift = 1;           //send shift high to the Datapath
100                                  inc_Bit_counter = 1;
101                                  next_state = receiving;
102                               end             //keep shift as ong as BC_eq_8 is false and
    remain in receiving state
103                            else
104                               begin
105                                  next_state = idle;      //if BC_eq_8 id high move to idle
106                                  read_not_ready_out = 1; //and send read_not_ready_out to
    the Datapath
107                                  clr_Bit_counter = 1;
108                            //check the integrity of the data received and status of the
    host
109                                  if (read_not_ready_in == 1'b1)   //host not ready to
    receive the data
110                                     Error1 = 1;
111                                  else if (Ser_in_0 ==1'b1)        //stop bit not received
112                                     Error2 = 1;
113                                  else              // the both Error1 and 2 are false the
    data integrity is correct and send load signal
114                                     load = 1;         //send load signal to the Datapath
115                               end
116                            end
117               default: next_state = idle;
118           endcase
119      end
120   endmodule
121
122
123
124
125
```

126