Design Verilog HDL Code -

The code for the top module, controller, and DataPath is given in the textbook. Comments included to follow and show understanding of functionality.

ate: July 06, 2024 UART_RCVR.v Project: UART

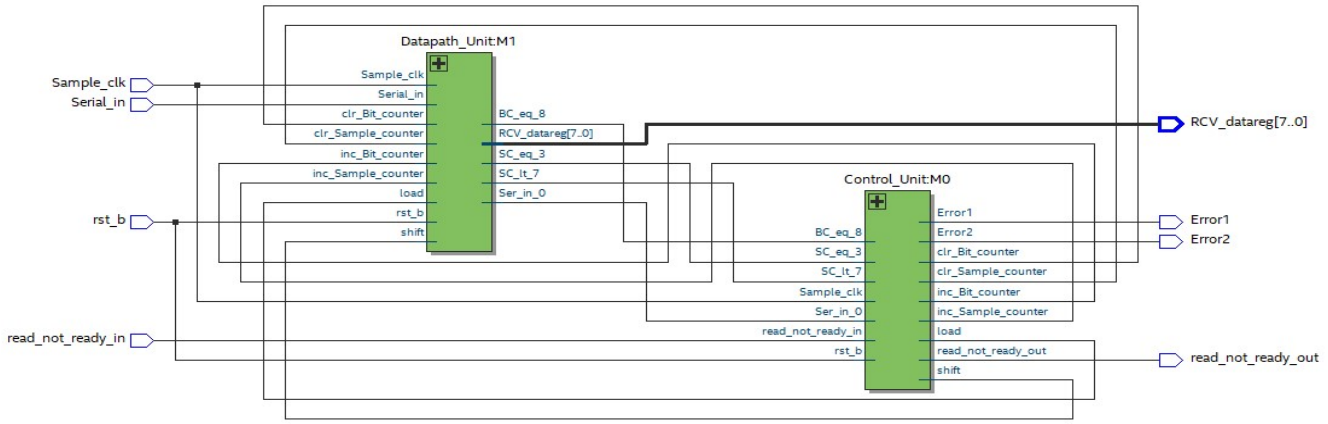
```
1   module UART_RCVR #(parameter word_size = 8, half_word = word_size /2)
2   (output [word_size -1: 0] RCV_datareg,
3    output read_not_ready_out, Error1, Error2,
4    input Serial_in,  read_not_ready_in,  Sample_clk,  rst_b );
5
6   Control_Unit M0 (
7     read_not_ready_out,
8     Error1,
9     Error2,
10    clr_Sample_counter,
11    inc_Sample_counter,
12    clr_Bit_counter,
13    inc_Bit_counter,
14    shift,
15    load,
16    read_not_ready_in,
17    Ser_in_0,
18    SC_eq_3,
19    SC_lt_7,
20    BC_eq_8,
21    Sample_clk,
22    rst_b  );
23
24  Datapath_Unit M1 (
25    RCV_datareg,
26    Ser_in_0,
27    SC_eq_3,
28    SC_lt_7,
29    BC_eq_8,
30    Serial_in,
31    clr_Sample_counter,
32    inc_Sample_counter,
33    clr_Bit_counter,
34    inc_Bit_counter,
35    shift,
36    load,
37    Sample_clk,
38    rst_b  );
39
40  endmodule
41
42  module Control_Unit #(
43    parameter  word_size = 8, half_word = word_size /2, Num_state_bits = 2,  idle = 2'b00,
    starting = 2'b01, receiving = 2'b10// one-hot assignment
44  )(
45    output reg read_not_ready_out,
46           Error1, Error2,
47           clr_Sample_counter,
48           inc_Sample_counter,
49           clr_Bit_counter,
50           inc_Bit_counter,
51           shift,
52           load,
53           input read_not_ready_in,
54           Ser_in_0,
55           SC_eq_3,
56           SC_lt_7,
57           BC_eq_8,
58           Sample_clk,
59           rst_b  );
60    reg [word_size-1: 0] RCV_shftreg;
61    reg [Num_state_bits -1: 0] state, next_state;
62    always @ (posedge Sample_clk)
63      if (rst_b == 1'b0) state <= idle; else state <= next_state;
64    always @ (state, Ser_in_0, SC_eq_3, SC_lt_7, read_not_ready_in) begin
65     read_not_ready_out = 0;
66     clr_Sample_counter = 0;
67     clr_Bit_counter = 0;
68     inc_Sample_counter = 0;
69     inc_Bit_counter = 0;
```

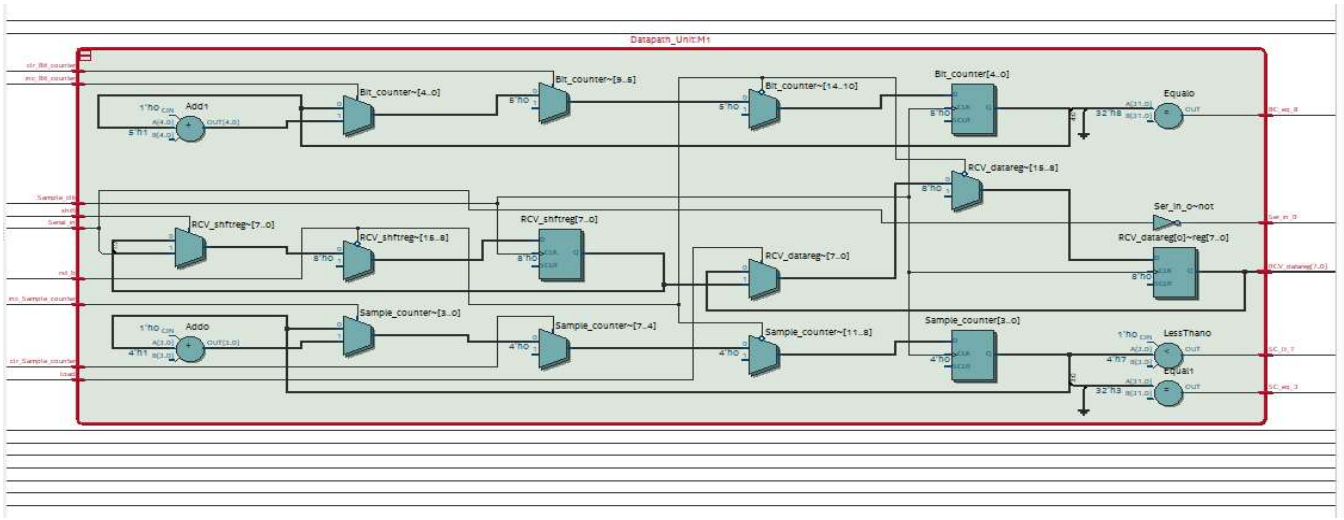
```
 70      shift = 0;
 71      Error1 = 0;
 72      Error2 = 0;
 73      load = 0;
 74      next_state = idle;
 75      case (state)
 76        idle: if (Ser_in_0 == 1'b1) next_state = starting;
 77              else next_state = idle;
 78        starting:   if (Ser_in_0 == 1'b0) begin
 79                     next_state = idle;
 80                     clr_Sample_counter = 1;
 81                    end
 82                    else if (SC_eq_3 == 1'b1) begin
 83                     next_state = receiving;
 84                     clr_Sample_counter = 1;
 85                    end else begin inc_Sample_counter = 1; next_state = starting; end
 86        receiving:  if (SC_lt_7 == 1'b1) begin
 87                      inc_Sample_counter = 1;
 88                      next_state = receiving;
 89                    end
 90                    else begin
 91                     clr_Sample_counter = 1;
 92                     if (!BC_eq_8) begin
 93                      shift = 1;
 94                      inc_Bit_counter = 1;
 95                      next_state = receiving;
 96                     end
 97                     else begin
 98                      next_state = idle;
 99                      read_not_ready_out = 1;
100                      clr_Bit_counter = 1;
101                      if (read_not_ready_in == 1'b1) Error1 = 1;
102                      else if (Ser_in_0 == 1'b1) Error2 = 1;
103                      else load = 1;
104                     end
105                    end
106        default:    next_state = idle;
107       endcase
108     end
109   endmodule
110
111   module Datapath_Unit #(parameter
112    word_size = 8, half_word = word_size /2, Num_counter_bits = 4
113   )(
114    output reg [word_size-1: 0] RCV_datareg,
115    output                      Ser_in_0,
116                                SC_eq_3,
117                                SC_lt_7,
118                                BC_eq_8,
119    input                       Serial_in,
120                                clr_Sample_counter,
121                                inc_Sample_counter,
122                                clr_Bit_counter,
123                                inc_Bit_counter,
124                                shift,
125                                load,
126                                Sample_clk,
127                                rst_b  );
128    reg [word_size-1: 0]        RCV_shftreg;
129    reg [Num_counter_bits-1: 0] Sample_counter;
130    reg [Num_counter_bits: 0]   Bit_counter;
131    assign Ser_in_0 = (Serial_in == 1'b0);
132    assign BC_eq_8 = (Bit_counter == word_size);
133    assign SC_lt_7 = (Sample_counter < word_size -1);
134    assign SC_eq_3 = (Sample_counter == half_word -1);
135
136    always @ (posedge Sample_clk)
137    if (rst_b == 1'b0) begin// synchronous rst_b
138      Sample_counter <= 0;
139      Bit_counter <= 0;
```

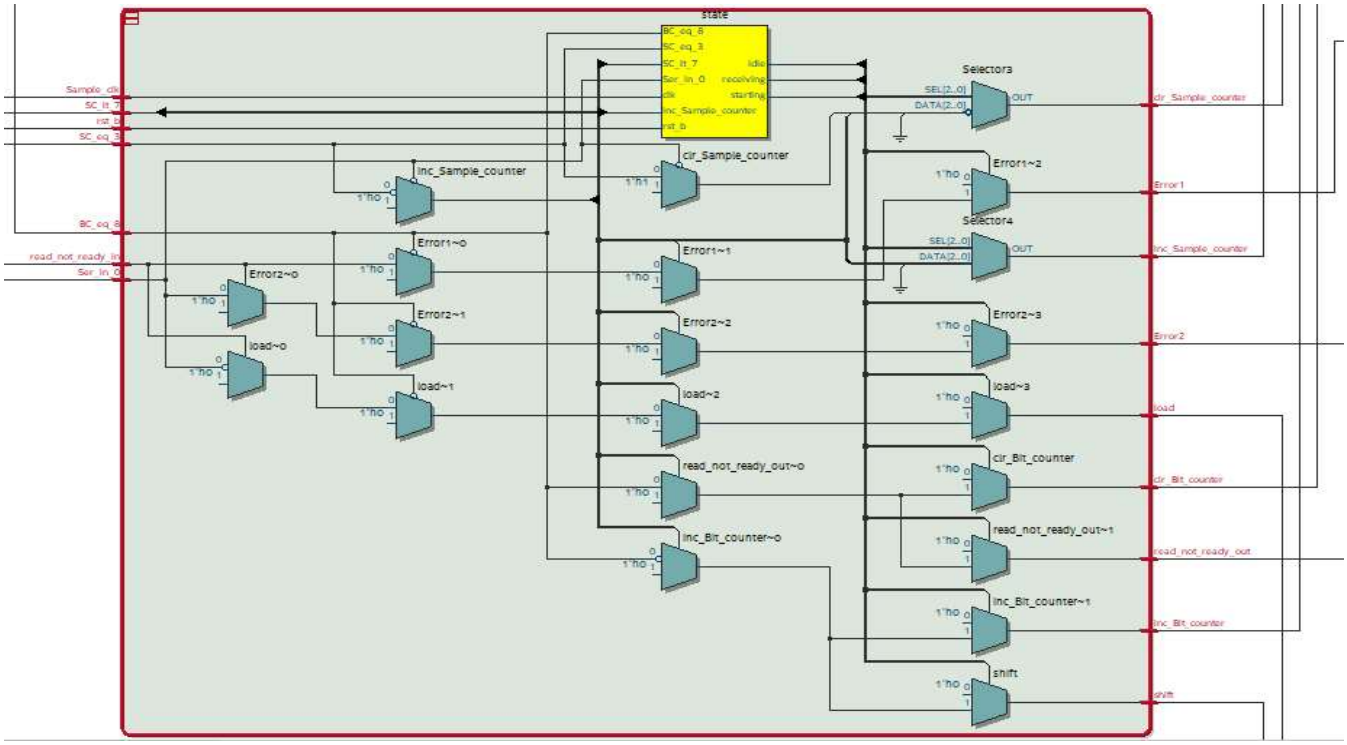


```
140       RCV_datareg <= 0;
141       RCV_shftreg <= 0;
142     end
143     else begin
144      if (clr_Sample_counter == 1) Sample_counter <= 0;
145      else if (inc_Sample_counter == 1) Sample_counter <= Sample_counter + 1;
146
147      if (clr_Bit_counter == 1) Bit_counter <= 0;
148      else if (inc_Bit_counter == 1) Bit_counter <= Bit_counter + 1;
149      if (shift == 1) RCV_shftreg <= {Serial_in, RCV_shftreg[word_size-1:1]};
150      if (load == 1) RCV_datareg <= RCV_shftreg;
151     end
152   endmodule
```

RTL Viewer


Datapath_Unit:M1
Sample_clk
Serial_in
clr_Bit_counter
clr_Sample_counter
inc_Bit_counter
inc_Sample_counter
load
rst_b
shift
BC_eq_8
RCV_datareg[7..0]
SC_eq_3
SC_lt_7
Ser_in_0
Control_Unit:M0
BC_eq_8
SC_eq_3
SC_lt_7
Sample_clk
Ser_in_0
read_not_ready_in
rst_b
Error1
Error2
clr_Bit_counter
clr_Sample_counter
inc_Bit_counter
inc_Sample_counter
load
read_not_ready_out
shift
RCV_datareg[7..0]
Error1
Error2
read_not_ready_out
read_not_ready_in


Datapath_Unit:M1
Add1
Bit_counter~[4..0]
Bit_counter~[9..5]
Bit_counter~[14..10]
Bit_counter[4..0]
Equal0
RCV_datareg~[15..8]
RCV_shftreg~[7..0]
RCV_shftreg~[15..8]
RCV_shftreg[7..0]
RCV_datareg~[7..0]
Ser_in_0~not
RCV_datareg[0]~reg[7..0]
Add0
Sample_counter~[3..0]
Sample_counter~[7..4]
Sample_counter~[11..8]
Sample_counter[3..0]
LessThan0
Equal1


state
Selector3
Selector4
inc_Sample_counter
clr_Sample_counter
Error1~0
Error1~1
Error1~2
Error2~0
Error2~1
Error2~2
Error2~3
load~0
load~1
load~2
load~3
read_not_ready_out~0
read_not_ready_out~1
inc_Bit_counter~0
inc_Bit_counter~1
clr_Bit_counter
shift
clr_Sample_counter
Error1
inc_Sample_counter
Error2
load
clr_Bit_counter
read_not_ready_out
inc_Bit_counter
shift

Testbench Verilog HDL Code

create the testbench code to obtain the correct simulation results given in the book.

ate: July 06, 2024 UART_RCVR_tb.v Project: L

```
/*-------------------------------------------------------------------------------------
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 the 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;
/*wire [word_size-2:0]    stateProbe2; //internal probe wire for troubleshooting
wire [word_size-1:0]    multiplierProbe;
wire [2*word_size-1: 0] multiplicandProbe;
reg  [word_size -1: 0]  multiplier2;
reg  [2*word_size-1: 0] multiplicand1;
reg                     start, clk, rst;
*/
//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
//assign stateProbe = UUT.state;
//assign Ser_in_0             = UUT.Datapath_Unit.Ser_in_0;
//assign clr_Sample_counter   = UUT.Datapath_Unit.clr_Sample_counter;
//assign inc_Sample_counter   = UUT.Datapath_Unit.inc_Sample_counter;
//assign clr_Bit_counter      = UUT.Datapath_Unit.clr_Bit_counter;
//assign inc_Bit_counter      = UUT.Datapath_Unit.inc_Bit_counter;
//assign RCV_shftreg          = UUT.Datapath_Unit.RCV_shftreg;
//assign Sample_counter       = UUT.Datapath_Unit.Sample_counter;
//assign Bit_counter          = UUT.Datapath_Unit.Bit_counter;
//assign SC_lt_7              = UUT.Control_Unit.SC_lt_7;
//assign SC_eq_3              = UUT.Control_Unit.SC_eq_3;
//assign state                = UUT.Control_Unit.state;
//assign shift                = UUT.Control_Unit.shift;
//assign load                 = UUT.Control_Unit.load;

//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
```

```
71        Serial_in = 3'b000;
72        read_not_ready_in = 0;
73        Sample_clk = 0;
74        rst_b = 0;
75
76        // Apply reset
77        rst_b = 1;
78        #10;
79        rst_b = 0;
80        #10;
81        rst_b = 1;  //reset high means system active
82
83        // Test Case 1: Transmit byte (ex: 8'b10101010)
84        Serial_in = 1; // Start bit
85        #104160; // Wait for one bit time (assuming 9600 baud rate)
86
87        Serial_in = 1; // Bit 0
88        #104160;
89
90        Serial_in = 0; // Bit 1
91        #104160;
92
93        Serial_in = 1; // Bit 2
94        #104160;
95
96        Serial_in = 0; // Bit 3
97        #104160;
98
99        Serial_in = 1; // Bit 4
100       #104160;
101
102       Serial_in = 0; // Bit 5
103       #104160;
104
105       Serial_in = 1; // Bit 6
106       #104160;
107
108       Serial_in = 0; // Bit 7
109       #104160;
110
111       Serial_in = 1; // Stop bit
112       #104160;
113
114       // Wait a few cycles
115       #500000;
116
117       #1000;
118       $stop;
119     end
120 endmodule
121
```

RTL Simulation -from book

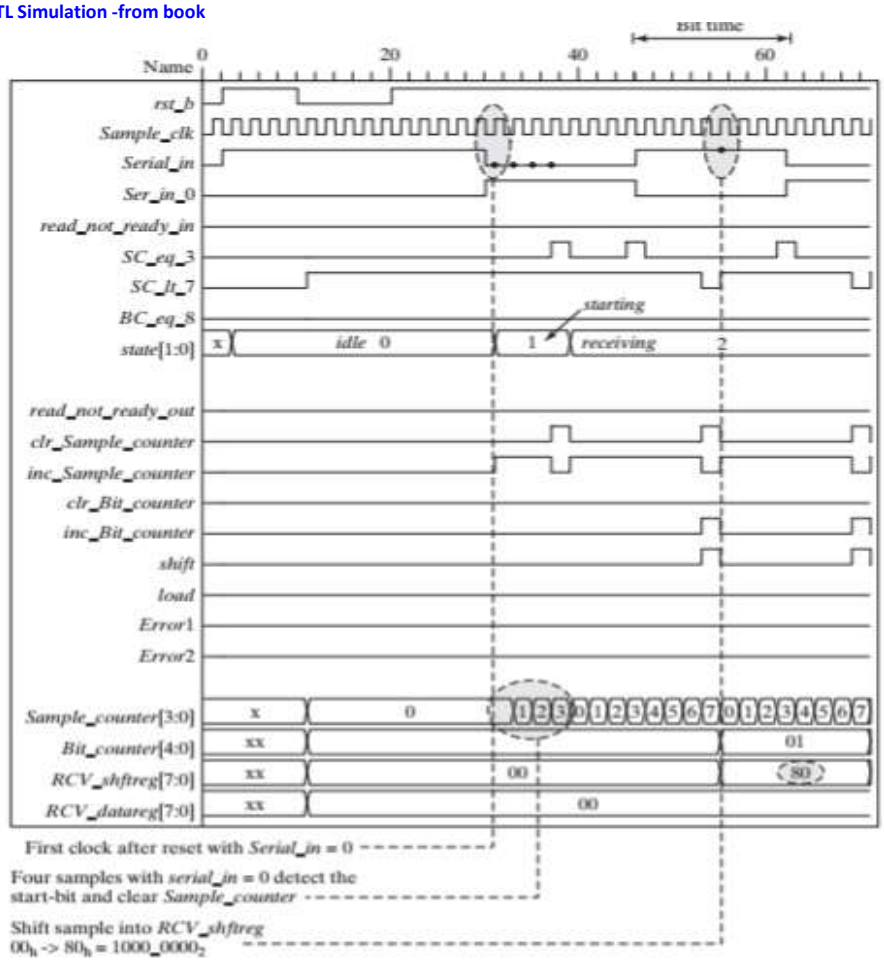


FIGURE 7-27 Annotated simulation results for *UART_receiver*.

RTL Simulation -from Quartus

Conclusion

When reset is low, this simulation shows that Receiver takes a serial input and creates a word that can be a parallel output.
When the reset goes low the process resets.