```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//  File name : original_testcase/testcase.sv                         //
//  Author    : G. Andres Mancera                                     //
//  License   : GNU Lesser General Public License                     //
//  Course    : Advanced Verification with SystemVerilog OOP          //
//              Testbench - UCSC Silicon Valley Extension             //
//                                                                    //
////////////////////////////////////////////////////////////////////////

program testcase (  interface tcif_driver,
                    interface tcif_monitor  );

  reg [7:0]     tx_buffer[0:10000];
  integer       tx_length;
  integer       tx_count;
  integer       rx_count;


  initial begin
    tx_count = 0;
    rx_count = 0;

    tcif_driver.cb.wb_adr_i <= 8'b0;
    tcif_driver.cb.wb_cyc_i <= 1'b0;
    tcif_driver.cb.wb_dat_i <= 32'b0;
    tcif_driver.cb.wb_stb_i <= 1'b0;
    tcif_driver.cb.wb_we_i  <= 1'b0;
  end

  // Init signals
  initial begin
    for (tx_length = 0; tx_length <= 1000; tx_length = tx_length + 1) begin
      tx_buffer[tx_length] = 0;
    end
    tcif_driver.cb.pkt_rx_ren   <= 1'b0;
    tcif_driver.cb.pkt_tx_data  <= 64'b0;
    tcif_driver.cb.pkt_tx_val   <= 1'b0;
    tcif_driver.cb.pkt_tx_sop   <= 1'b0;
    tcif_driver.cb.pkt_tx_eop   <= 1'b0;
    tcif_driver.cb.pkt_tx_mod   <= 3'b0;
  end

  initial begin
    forever begin
      tcif_driver.cb.xgmii_rxc  <= tcif_driver.cb.xgmii_txc;
      tcif_driver.cb.xgmii_rxd  <= tcif_driver.cb.xgmii_txd;
      @(posedge tcif_monitor.clk_156m25);
    end
  end

  //=========================================================
  // Wishbone interface read/write
  initial begin
    WaitNS(1000);
    // Initial read to configuration register 0.
    tcif_driver.cb.wb_adr_i     <= 8'b0;
    tcif_driver.cb.wb_cyc_i     <= 1'b1;
    tcif_driver.cb.wb_stb_i     <= 1'b1;
    WaitNS(10);
```

```
61      tcif_driver.cb.wb_cyc_i     <= 1'b0;
62      tcif_driver.cb.wb_stb_i     <= 1'b0;
63      WaitNS(100);
64
65      // Write into configuration register 0 (Address 0x00).
66      // As long as wb_dat_i[0]=1'b1, transmission will be enabled.
67      // The remaining bits (wb_dat_i[31:1]) are don't care.
68      tcif_driver.cb.wb_adr_i     <= 8'b0;
69      tcif_driver.cb.wb_cyc_i     <= 1'b1;
70      tcif_driver.cb.wb_stb_i     <= 1'b1;
71      tcif_driver.cb.wb_we_i      <= 1'b1;
72      tcif_driver.cb.wb_dat_i     <= {$urandom_range(0, 31'h7FFF_FFF), 1'b1};
73      WaitNS(10);
74      tcif_driver.cb.wb_cyc_i     <= 1'b0;
75      tcif_driver.cb.wb_stb_i     <= 1'b0;
76      tcif_driver.cb.wb_we_i      <= 1'b0;
77      WaitNS(100);
78    end
79    //=============================================================
80
81    initial begin
82      WaitNS(5000);
83      ProcessCmdFile();
84    end
85
86    initial begin
87      forever begin
88        if (tcif_monitor.cb.pkt_rx_avail) begin
89          RxPacket();
90          if (rx_count == tx_count) begin
91            $display("All packets received. Simulation done!!!\n");
92          end
93        end
94        @(posedge tcif_monitor.clk_156m25);
95      end
96    end
97
98
99    // Other tasks
100   task WaitNS(input [31:0] delay);
101     begin
102       #(1000*delay);
103     end
104   endtask : WaitNS
105
106
107   task TxPacket;
108     integer     i;
109     begin
110       $display("Transmit packet with length: %d", tx_length);
111       @(posedge tcif_driver.clk_156m25);
112       WaitNS(1);
113       tcif_driver.cb.pkt_tx_val <= 1'b1;
114       for (i = 0; i < tx_length; i = i + 8) begin
115         tcif_driver.cb.pkt_tx_sop <= 1'b0;
116         tcif_driver.cb.pkt_tx_eop <= 1'b0;
117         tcif_driver.cb.pkt_tx_mod <= 2'b0;
118         if (i == 0) tcif_driver.cb.pkt_tx_sop <= 1'b1;
119         if (i + 8 >= tx_length) begin
120           tcif_driver.cb.pkt_tx_eop <= 1'b1;
```

```
121           tcif_driver.cb.pkt_tx_mod <= tx_length % 8;
122         end
123         tcif_driver.cb.pkt_tx_data[`LANE7] <= tx_buffer[i];
124         tcif_driver.cb.pkt_tx_data[`LANE6] <= tx_buffer[i+1];
125         tcif_driver.cb.pkt_tx_data[`LANE5] <= tx_buffer[i+2];
126         tcif_driver.cb.pkt_tx_data[`LANE4] <= tx_buffer[i+3];
127         tcif_driver.cb.pkt_tx_data[`LANE3] <= tx_buffer[i+4];
128         tcif_driver.cb.pkt_tx_data[`LANE2] <= tx_buffer[i+5];
129         tcif_driver.cb.pkt_tx_data[`LANE1] <= tx_buffer[i+6];
130         tcif_driver.cb.pkt_tx_data[`LANE0] <= tx_buffer[i+7];
131         @(posedge tcif_driver.clk_156m25);
132         WaitNS(1);
133       end
134       tcif_driver.cb.pkt_tx_val <= 1'b0;
135       tcif_driver.cb.pkt_tx_eop <= 1'b0;
136       tcif_driver.cb.pkt_tx_mod <= 3'b0;
137       tx_count = tx_count + 1;
138     end
139   endtask : TxPacket
140
141   task CmdTxPacket(input [31:0] file);
142     integer     count;
143     integer     data;
144     integer     i;
145     begin
146       count = $fscanf(file, "%2d", tx_length);
147       if (count == 1) begin
148         for (i = 0; i < tx_length; i = i + 1) begin
149           count = $fscanf(file, "%2X", data);
150           if (count) begin
151             tx_buffer[i] = data;
152           end
153         end
154         TxPacket();
155       end
156     end
157   endtask : CmdTxPacket
158
159   task ProcessCmdFile;
160     integer         file_cmd;
161     integer         count;
162     reg [8*8-1:0]   str;
163     begin
164       file_cmd = $fopen("../../testbench/verilog/packets_tx.txt", "r");
165       if (!file_cmd) $stop;
166       while (!$feof(file_cmd)) begin
167         count = $fscanf(file_cmd, "%s", str);
168         if (count != 1) continue;
169         $display("CMD %s", str);
170         case (str)
171           "SEND_PKT": begin
172                         CmdTxPacket(file_cmd);
173                       end
174         endcase
175       end
176       $fclose(file_cmd);
177       WaitNS(50000);
178       $finish;
179     end
180   endtask : ProcessCmdFile
```

```
181
182   // Task to read a single packet from receive interface and display
183   task RxPacket;
184     reg     done;
185     begin
186       done = 0;
187       tcif_monitor.cb.pkt_rx_ren <= 1'b1;
188       @(posedge tcif_monitor.clk_156m25);
189       while (!done) begin
190         if (tcif_monitor.cb.pkt_rx_val) begin
191           if (tcif_monitor.cb.pkt_rx_sop) begin
192             $display("\n\n-----------------------");
193             $display("Received Packet");
194             $display("-----------------------");
195           end
196           $display("%x", tcif_monitor.cb.pkt_rx_data);
197           if (tcif_monitor.cb.pkt_rx_eop) begin
198             done <= 1;
199             tcif_monitor.cb.pkt_rx_ren <= 1'b0;
200           end
201           if (tcif_monitor.cb.pkt_rx_eop) begin
202             $display("-----------------------\n\n");
203           end
204         end
205         @(posedge tcif_monitor.clk_156m25);
206       end
207       rx_count = rx_count + 1;
208     end
209   endtask : RxPacket
210
211 endprogram
212
```