

```

1 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2 //                                                                                                                //
3 //   File name : monitor.sv                                                                                       //
4 //   Author    : G. Andres Mancera                                                                                   //
5 //   License   : GNU Lesser General Public License                                                                //
6 //   Course    : Advanced Verification with SystemVerilog OOP                                                    //
7 //               Testbench - UCSC Silicon Valley Extension                                                         //
8 //                                                                                                                //
9 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
10
11 class monitor;
12
13     virtual xge_mac_interface      mon_vi;
14     packet                          xge_mac_pkt;
15     mailbox                        mon2sb;
16
17     // ===== Contructor =====
18     function new( input virtual xge_mac_interface vif,
19                  input mailbox mon2sb                );
20         $display("MONITOR :: inside new() function");
21         this.mon_vi = vif;
22         this.mon2sb = mon2sb;
23         xge_mac_pkt = new();
24     endfunction : new
25
26
27     // ===== Class methods =====
28     task collect_packet();
29         packet      mon_pkt;
30         bit          pkt_in_progress;
31         bit          err_in_packet;
32         bit [7:0]    rx_data_q[$];
33         int          idx;
34         bit          packet_captured;
35         mon_pkt      = new();
36         pkt_in_progress = 0;
37         packet_captured = 0;
38         err_in_packet  = 0;
39
40         forever begin
41             @(mon_vi.cb)
42             begin
43                 if ( mon_vi.cb.pkt_rx_avail ) begin
44                     mon_vi.cb.pkt_rx_ren <= 1'b1;
45                 end
46                 if ( mon_vi.cb.pkt_rx_val ) begin
47                     if ( mon_vi.cb.pkt_rx_sop && !mon_vi.cb.pkt_rx_eop && pkt_in_progress==0 )
48                         // ----- SOP cycle -----
49                         pkt_in_progress = 1;
50                         mon_vi.cb.pkt_rx_ren <= 1'b1;
51                         mon_pkt.sop_mark      = mon_vi.cb.pkt_rx_sop;
52                         mon_pkt.mac_dst_addr  = mon_vi.cb.pkt_rx_data[63:16];
53                         mon_pkt.mac_src_addr[47:32] = mon_vi.cb.pkt_rx_data[15:0];
54                         mon_pkt.mac_src_addr[31:0]  = 32'h0;
55                         mon_pkt.ethernet_type     = 16'h0;
56                         mon_pkt.payload = new[0];
57                         while ( rx_data_q.size()>0 ) begin
58                             rx_data_q.pop_front();
59                         end

```

```

60     end // ----- SOP cycle -----
61     if ( !mon_vi.cb.pkt_rx_sop && !mon_vi.cb.pkt_rx_eop && pkt_in_progress==1
) begin
62         // ----- MOP cycle -----
63         pkt_in_progress = 1;
64         mon_vi.cb.pkt_rx_ren <= 1'b1;
65         if ( rx_data_q.size()==0 ) begin
66             mon_pkt.mac_src_addr[31:0] = mon_vi.cb.pkt_rx_data[63:32];
67             mon_pkt.ether_type = mon_vi.cb.pkt_rx_data[31:16];
68             rx_data_q.push_back(mon_vi.cb.pkt_rx_data[15:8]);
69             rx_data_q.push_back(mon_vi.cb.pkt_rx_data[7:0]);
70         end
71         else begin
72             for ( int i=0; i<8; i++ ) begin
73                 rx_data_q.push_back( (mon_vi.cb.pkt_rx_data >> (64-8*(i+1))) & 8'hFF
);
74             end
75         end
76     end // ----- MOP cycle -----
77     if ( mon_vi.cb.pkt_rx_eop && pkt_in_progress==1 ) begin
78         // ----- EOP cycle -----
79         mon_pkt.eop_mark= mon_vi.cb.pkt_rx_eop;
80         pkt_in_progress = 0;
81         err_in_packet = mon_vi.cb.pkt_rx_err;
82         mon_vi.cb.pkt_rx_ren <= 1'b0;
83         if ( mon_vi.cb.pkt_rx_mod==0 ) begin
84             for ( int i=0; i<8; i++ ) begin
85                 rx_data_q.push_back( (mon_vi.cb.pkt_rx_data >> (64-8*(i+1))) & 8'hFF
);
86             end
87         end
88         else begin
89             for ( int i=0; i<mon_vi.cb.pkt_rx_mod; i++ ) begin
90                 rx_data_q.push_back( (mon_vi.cb.pkt_rx_data >> (64-8*(i+1))) & 8'hFF
);
91             end
92         end
93         //$display("MONITOR DEBUG :: mon_pkt.mac_dst_addr = %0x",
mon_pkt.mac_dst_addr);
94         //$display("MONITOR DEBUG :: mon_pkt.mac_src_addr = %0x",
mon_pkt.mac_src_addr);
95         //$display("MONITOR DEBUG :: mon_pkt.ether_type = %0x",
mon_pkt.ether_type);
96         //$display("MONITOR DEBUG :: rx_data_q size =%0d", rx_data_q.size());
97         mon_pkt.payload = new[rx_data_q.size()];
98         idx = 0;
99         while ( rx_data_q.size()>0 ) begin
100             mon_pkt.payload[idx] = rx_data_q.pop_front();
101             idx++;
102         end
103         packet_captured = 1;
104         // ----- EOP cycle -----
105     end
106     if ( mon_vi.cb.pkt_rx_sop && mon_vi.cb.pkt_rx_eop && pkt_in_progress==0)
begin
107         // ----- SOP/EOP cycle -----
108         err_in_packet = mon_vi.cb.pkt_rx_err;
109         mon_vi.cb.pkt_rx_ren <= 1'b1;
110         mon_pkt.sop_mark = mon_vi.cb.pkt_rx_sop;
111         mon_pkt.eop_mark = mon_vi.cb.pkt_rx_eop;

```

```

112     mon_pkt.mac_dst_addr      = mon_vi.cb.pkt_rx_data[63:16];
113     mon_pkt.mac_src_addr[47:32] = mon_vi.cb.pkt_rx_data[15:0];
114     mon_pkt.mac_src_addr[31:0]  = 32'h0;
115     mon_pkt.ether_type         = 16'h0;
116     mon_pkt.payload = new[0];
117     while ( rx_data_q.size()>0 ) begin
118         rx_data_q.pop_front();
119     end
120     packet_captured = 1;
121     // ----- SOP/EOP cycle -----
122 end
123 if ( packet_captured ) begin
124     mon_pkt.print("FROM MONITOR");
125     // Put the collected packet into the mailbox if the packet has no errors
126     if ( !err_in_packet && mon_pkt.sop_mark && mon_pkt.eop_mark ) begin
127         mon2sb.put(mon_pkt);
128     end
129     else begin
130         $display("MONITOR :: t=%2t, ERROR PACKET, WILL NOT SEND IT TO
SCOREBOARD", $time);
131     end
132     packet_captured = 0;
133 end
134 end
135 end
136 end
137 endtask : collect_packet
138
139 endclass
140

```