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

class packet;

  // Signals to be driven into the RTL
  rand bit [47:0]       mac_dst_addr;       // 6 Bytes
  rand bit [47:0]       mac_src_addr;       // 6 Bytes
  rand bit [15:0]       ether_type;         // 2 Bytes
  rand bit [7:0]        payload [];
  rand bit [31:0]       ipg;                // interpacket gap

  // Signals unrelated to the RTL
  rand bit              sop_mark;
  rand bit              eop_mark;
  static bit [31:0]     pkt_id;

  // ======== Constraints ========
  constraint C_proper_sop_eop_marks {
    sop_mark == 1;  // SOP mark should be driven
    eop_mark == 1;  // EOP mark should be driven
  }

  constraint C_payload_size {
    payload.size() inside {[46:1500]};
  }

  constraint C_ipg {
    ipg inside {[10:50]};
  }


  // ======== Constructor ========
  function new(input packet myself=null);
  endfunction : new


  // ======== Class methods ========
  function void print(string calling_class);
    int unsigned    Byte8_words;
    $display("PACKET %s :: t=%2t, mac_dst_addr=%h, mac_src_addr=%h, ether_type=%h,
payload_size=%0d, sop=%0d, eop=%0d",
             calling_class, $time, mac_dst_addr, mac_src_addr, ether_type,
             payload.size(), sop_mark, eop_mark);
    if ( payload.size()>0 ) begin
      Byte8_words = payload.size()%8 ? payload.size()/8+1 : payload.size()/8;
      for ( int i=0; i<Byte8_words; i++ ) begin
        if ( i!=Byte8_words-1 ) begin
          $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h%h%h_%h%h%h%h",
calling_class,
                   $time, 8*i, 8*i+7, payload[8*i], payload[8*i+1], payload[8*i+2],
payload[8*i+3],
                   payload[8*i+4], payload[8*i+5], payload[8*i+6], payload[8*i+7]);
```

```
58          end
59          else begin
60            case (payload.size()%8)
61              0: begin
62                $display("PACKET %s :: t=%2t,
payloadBytes[%0d:%0d]=%h%h%h%h_%h%h%h%h", calling_class,
63                         $time, 8*i, 8*i+7, payload[8*i], payload[8*i+1],
payload[8*i+2],
64                         payload[8*i+3], payload[8*i+4], payload[8*i+5],
payload[8*i+6],
65                         payload[8*i+7]);
66              end
67              1: begin
68                $display("PACKET %s :: t=%2t, payloadBytes[%0d]=%h", calling_class,
$time,
69                         8*i, payload[8*i]);
70              end
71              2: begin
72                $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h",
calling_class, $time,
73                         8*i, 8*i+1, payload[8*i], payload[8*i+1]);
74              end
75              3: begin
76                $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h%h",
calling_class, $time,
77                         8*i, 8*i+2, payload[8*i], payload[8*i+1], payload[8*i+2]);
78              end
79              4: begin
80                $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h%h%h",
calling_class, $time,
81                         8*i, 8*i+3, payload[8*i], payload[8*i+1], payload[8*i+2],
payload[8*i+3]);
82              end
83              5: begin
84                $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h%h%h_%h",
calling_class, $time,
85                         8*i, 8*i+4, payload[8*i], payload[8*i+1], payload[8*i+2],
payload[8*i+3],
86                         payload[8*i+4]);
87              end
88              6: begin
89                $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h%h%h_%h%h",
calling_class,
90                         $time, 8*i, 8*i+5, payload[8*i], payload[8*i+1],
payload[8*i+2],
91                         payload[8*i+3], payload[8*i+4], payload[8*i+5]);
92              end
93              7: begin
94                $display("PACKET %s :: t=%2t, payloadBytes[%0d:%0d]=%h%h%h%h_%h%h%h",
calling_class,
95                         $time, 8*i, 8*i+6, payload[8*i], payload[8*i+1],
payload[8*i+2],
96                         payload[8*i+3], payload[8*i+4], payload[8*i+5],
payload[8*i+6]);
97              end
98            endcase
99          end
100       end
101     end
102   endfunction : print
```

```
103
104   function set_mac_dst_addr(bit [47:0] mac_dst_addr);
105     this.mac_dst_addr = mac_dst_addr;
106   endfunction : set_mac_dst_addr
107
108   function bit [47:0] get_mac_dst_addr();
109     return this.mac_dst_addr;
110   endfunction : get_mac_dst_addr
111
112   function set_mac_src_addr(bit [47:0] mac_src_addr);
113     this.mac_src_addr = mac_src_addr;
114   endfunction : set_mac_src_addr
115
116   function bit [47:0] get_mac_src_addr();
117     return this.mac_src_addr;
118   endfunction : get_mac_src_addr
119
120   function bit set_ether_type (bit [15:0] ether_type);
121     this.ether_type = ether_type;
122   endfunction : set_ether_type
123
124   function bit [15:0] get_ether_type();
125     return this.ether_type;
126   endfunction : get_ether_type
127
128   function set_ipg(bit [31:0] ipg);
129     this.ipg = ipg;
130   endfunction : set_ipg
131
132   function bit [31:0] get_ipg();
133     return this.ipg;
134   endfunction : get_ipg
135
136   static function increase_pktid();
137     pkt_id++;
138   endfunction : increase_pktid
139
140   static function bit [15:0] get_pktid();
141     return pkt_id;
142   endfunction : get_pktid
143
144 endclass
145
```