```
1
2
3
4   // fifo
5
6   module fifo_16x9_router(clk,reset,data_in,write_enb,read_enb,soft_reset,lfd_state,full,
    empty,data_out);
7
8   // declaring the input and ouput ports
9
10        input clk,reset,write_enb,read_enb,lfd_state,soft_reset;
11        input [7:0]data_in;
12        output full,empty;
13        output reg [7:0]data_out;
14
15  // declaring the internal write addr read addr memory and fifi down count reg
16
17        reg [5:0]write_addr,read_addr;
18        reg [8:0]mem[0:15];
19        reg [6:0]fifo_down_counter;
20        reg lfd_state_temp;
21        integer i;
22
23  /* delaying the lfd state bcuz its arriving at one clock  from the fsm mean
24  while data header comes at 2 clock pulses  from register so we synxc the
25  two things so we using d flip flop to genrate 1 clk delay for lfd state */
26
27
28        always@(posedge clk)
29          begin
30           if(!reset)
31              lfd_state_temp<=0;
32           else
33                lfd_state_temp<=lfd_state;
34          end
35
36     // assign lfd_state = lfd_state_temp;
37
38        /*always@(posedge clk)
39         begin
40             if(write_enb && (!full))
41                mem[write_addr[3:0]][8] <= lfd_state_temp;
42                else
43                 mem[write_addr[3:0]][8] <=  mem[write_addr[3:0]][8] ;
44          end */
45
46  // increment the write and read address
47
48     /*  always@(posedge clk)
49         begin
50          if(!reset )
51                 {read_addr,write_addr}<=0;
52              else if(soft_reset)
53                 {write_addr,read_addr}<=0;
54              else
55                  begin
56                  if(write_enb && (!full))
57
58                  if (read_enb && (!empty))
59                 read_addr <= read_addr +1;
60               end
61          end */
62  // increment the read address
63
64       /*always@(posedge clk)
65          begin
66               if(!reset)
67                 read_addr<=0;
68               else if(soft_reset)
```

```
                read_addr<=0;
              else if(read_enb && (!empty))
                 read_addr <= read_addr +1;
          end
   */
   // writing opearion

       always@(posedge clk)
            begin
                if(!reset)
                  begin
                    for(i=0;i<16;i=i+1)
                           begin
                       mem[i]<=0;
                            end
                       write_addr<=0;
                end
                else if(soft_reset)
                    begin
                      for(i=0;i<16;i=i+1)
                             begin
                          mem[i]<=0;
                             end
                             write_addr<=0;
                end
                else
                   if(write_enb && (!full))
                begin
                 mem[write_addr[3:0]]<={lfd_state_temp,data_in};
                    write_addr <= write_addr+1;
                    //mem[write_addr[3:0]][8] <= lfd_state_temp;
                   end

         end

   // read opearion

        always@(posedge clk)
              begin
              if(!reset) begin
                    data_out<=0;
                       read_addr<=0; end
              else if(soft_reset) begin
                   data_out<=8'bzzzz_zzzz;
                       read_addr<=0; end
                 else if (fifo_down_counter ==0 && data_out !=0)
                    data_out<=8'bzzzz_zzzz;
                  else
                  if(read_enb && (!empty))
                        begin
               data_out<=mem[read_addr[3:0]];
                    read_addr <= read_addr +1;
                     end
              end
   /* fifo down count logic when hear byte recived it check that 8 bt of the byt
    it is one then fifo counte load with playload data then after ti decremented every clk
    pulse*/


        always@(posedge clk)
              begin
                    if(!reset)
                       fifo_down_counter<=0;
                   else if(soft_reset)
                       fifo_down_counter<=0;

                   else  if(read_enb && !empty)
                                    begin
                                    if(mem[read_addr[3:0]][8]==1'b1)
```

```
                    fifo_down_counter<=mem[read_addr[3:0]][7:2]+1;
                else if(fifo_down_counter != 0)
                    fifo_down_counter<=fifo_down_counter-1;
            end
                end
// fifo full signal logic and empty signal logic

     assign full = ( (write_addr[3:0] == read_addr[3:0]) && (write_addr[4] != read_addr[4
     ]))?1'b1:1'b0;
     assign empty =( (write_addr[3:0] == read_addr[3:0]) && (write_addr[4] == read_addr[4
     ]))?1'b1:1'b0;

endmodule


// fsmm

module fsm_router_controller(clk,reset,pkt_valid,busy,parity_done,data_in,soft_reset_0,
soft_reset_1,soft_reset_2,fifo_full,low_pkt_valid,fifo_empty_0,fifo_empty_1,fifo_empty_2,
detect_add,ld_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state);

// declaring the input and output ports

    input clk,reset,pkt_valid,parity_done,soft_reset_0,soft_reset_1,soft_reset_2,
    fifo_full,low_pkt_valid,fifo_empty_0,fifo_empty_1,fifo_empty_2;
    input [1:0]data_in;
    output detect_add,ld_state,laf_state,full_state,rst_int_reg,lfd_state;
    output busy,write_enb_reg;
// parameter and internal next_state,present state regs declarations

    parameter DECODE_ADDRESS=3'b000,LOAD_FIRST_DATA=3'b001,WAIT_TILL_EMPTY=3'b010,
    LOAD_DATA=3'b011,FIFO_FULL_STATE=3'b100,LOAD_AFTER_FULL=3'b101,LOAD_PARITY=3'b110,
    CHECK_PARITY_ERROR=3'B111;
    reg [1:0]addr_data;
    reg [2:0]present,next;

// storng detecting the destination_address_signal

    always@(*)
       begin
       if(!reset)
          addr_data<=0;
      else
        begin
             if(detect_add)
              addr_data <= data_in;
             end

       end

// sequntial logic for present state

    always@(posedge clk)
        begin
             if(!reset)
               present<=DECODE_ADDRESS;
                 else if(soft_reset_0 || soft_reset_1 || soft_reset_2)
               present<=DECODE_ADDRESS;
             else
          present<=next;
        end

// combinational logic for next state

    always@(*)
          begin
                case(present)
                DECODE_ADDRESS:
                              if((pkt_valid && addr_data == 2'd0 && fifo_empty_0) || (
```

```
                        pkt_valid && addr_data== 2'd1 && fifo_empty_1) || (
                        pkt_valid && addr_data == 2'd2 && fifo_empty_2))
199                   begin
200                         next=LOAD_FIRST_DATA;
201                   end
202                 else if((pkt_valid && addr_data == 2'd0 && (!fifo_empty_0)) || (
                    pkt_valid && addr_data == 2'd1 && (!fifo_empty_1)) || (pkt_valid
                    && addr_data == 2 && (!fifo_empty_2)))

203                   begin
204                   next=WAIT_TILL_EMPTY;
205                    end
206                 else
207                 begin
208                   next=DECODE_ADDRESS;
209               end
210             WAIT_TILL_EMPTY:
211                           if((addr_data == 2'd0 && fifo_empty_0) || (
                              addr_data == 2'd1 && fifo_empty_1) || (addr_data
                              == 2'd2 && fifo_empty_2))
212                   begin
213                         next=LOAD_FIRST_DATA;
214                   end
215                     else
216                         begin
217                           next=WAIT_TILL_EMPTY;
218                             end
219                 LOAD_FIRST_DATA:
220                               next=LOAD_DATA;
221                 LOAD_DATA      :
222                               if(fifo_full)
223                         begin
224                            next=FIFO_FULL_STATE;
225                   end
226                               else if ((!fifo_full) && (!pkt_valid))
227                   begin
228                      next=LOAD_PARITY;
229                                 end
230              else
231                 begin
232                   next=LOAD_DATA;
233                    end
234               FIFO_FULL_STATE :
235                    if(!fifo_full)
236                   begin
237                                 next=LOAD_AFTER_FULL;
238                   end
239                else
240                               begin
241                     next=FIFO_FULL_STATE;
242                   end
243              LOAD_AFTER_FULL:
244                           if((!parity_done) && (!low_pkt_valid))
245                     begin
246                        next=LOAD_DATA;
247                         end
248                          else if((!parity_done) && low_pkt_valid)
249                            begin
250                              next=LOAD_PARITY;
251                               end
252                else if(parity_done)
253                   begin
254                      next=DECODE_ADDRESS;
255                       end
256           LOAD_PARITY  :
257                       next=CHECK_PARITY_ERROR;
258         CHECK_PARITY_ERROR:
259                           if(fifo_full)
260                    begin
```

```
                                    next=FIFO_FULL_STATE;
                                  end
                                    else
                                      begin
                                    next=DECODE_ADDRESS;
                                  end
                                default:next=DECODE_ADDRESS;
                endcase

     end

  // combinational logic for output

  // busy is going to low at decode state and load stae other states  it was
  // high bcaz it is  not
  // allowing new data from the sourece
     assign  busy = (present == DECODE_ADDRESS || present == LOAD_DATA)?1'b0:1'b1;


     assign detect_add =(present == DECODE_ADDRESS)?1'b1:1'b0;
     assign ld_state   =(present == LOAD_DATA)?1'b1:1'b0;
     assign laf_state  =(present == LOAD_AFTER_FULL)? 1'b1:1'b0;
     assign full_state =(present == FIFO_FULL_STATE)?1'b1:1'b0;
     assign lfd_state  =(present == LOAD_FIRST_DATA)?1'b1:1'b0;
     assign rst_int_reg=(present == CHECK_PARITY_ERROR)?1'b1:1'b0;

  // write enb reg going tp high only in these state bcaz , these stae have
  // capabul for sending payload and parity data to fifo when write enb reg is
  // high then only we know which fifo destinastion.neccasary to send the data
  // at he particluar stae
  //

    assign write_enb_reg = (present == LOAD_DATA || present == LOAD_PARITY || present ==
    LOAD_AFTER_FULL)? 1'b1:1'b0;

  endmodule


  // register


   module register_router(clk,reset,pkt_valid,data_in,fifo_full,rst_int_reg,detect_add,
   ld_state,laf_state,full_state,lfd_state,parity_done,low_pkt_valid,error,dout);

  // declaring the regs and wires

        input clk,reset,pkt_valid,fifo_full,rst_int_reg,detect_add,ld_state,laf_state,
        full_state,lfd_state;
        input [7:0]data_in;
        output reg error,low_pkt_valid,parity_done;
        output reg [7:0]dout;

  // creating 4 internal register for header byt storing,internal parity
  // byte,packet parity byte,fifio full state byte each are 8 bits bcz all are bytes

       reg [7:0]header_byte,internal_parity_byte,packet_parity_byte,fifo_full_state_byte;

  //  writing logic for header byte sstoring
  //  header store add detect address and presetn state at load fikrst data and
  //  pkt valid is high and corect address destination

     always@(posedge clk)
          begin
               if(!reset)
                  begin
                       header_byte<=0;
                  end
               else
                   if(pkt_valid && detect_add && (data_in[1:0] != 2'd3))
```

```
                begin
                     header_byte <=data_in;
                end

    end
// logic for fifo sull state byte
// stroing the data in after full state

    always@(posedge clk)
        begin
                        if(!reset)
                   fifo_full_state_byte <= 0;
                               else if(ld_state && fifo_full)
                   fifo_full_state_byte <= data_in;
                else if(detect_add)
                   fifo_full_state_byte <= 0;
                else
                                    fifo_full_state_byte <=  fifo_full_state_byte ;

          end


// writing logic for dout
// data out is works only at  play load data in header data writhout error

   always@(posedge clk)
       begin
        if(!reset)
           begin
            dout<=0;
              end
           else if(detect_add && pkt_valid && (data_in[1:0] != 2'd3))
           begin
               dout <=dout;
               end
               else if(lfd_state)
                   begin
                      dout<=header_byte;
                   end
               else if(ld_state &&(!fifo_full))
              begin
                       dout <=data_in;
              end
           else if(full_state)
              begin
                dout <= dout;
              end
               else if(laf_state)
                  begin
                     dout<=fifo_full_state_byte;
                  end
               else
                   begin
                     dout <=dout;
                   end
         end

// writing logic for internal parity
// first it is ex or opeartion with header byte and then after contious ex or
// wuith each pay load data stored into internal parity


    always@(posedge clk)
           begin
        if(!reset)
           begin
               internal_parity_byte<=0;
               end
```

```
                else if(detect_add)
                   begin
                      internal_parity_byte<=0;
                   end
                else if(lfd_state)
                begin
                   internal_parity_byte <= internal_parity_byte ^ header_byte;
                    end
                else if(ld_state && !fifo_full)
                     begin
                       internal_parity_byte <= internal_parity_byte ^ data_in;
                     end
                else
                     begin
             internal_parity_byte <= internal_parity_byte;
                 end
        end

// writing logic for packet parity
// packet parity check wheather state uis parity_data only in this state
// only we are getting parity of packets

    always@(posedge clk)
        begin
          if(!reset)
             begin
                packet_parity_byte <= 0;
                 end
                  else if(detect_add)
                     begin
                       packet_parity_byte <=0;
                     end
                   else if(ld_state && (!pkt_valid) && (!fifo_full))
              begin
                packet_parity_byte <=data_in;
                  end
          else if(!pkt_valid && rst_int_reg)
              begin
               packet_parity_byte<=0;
              end
                 else
                     begin
                        packet_parity_byte <= packet_parity_byte;
                     end
           end

//parity done logic


   always@(posedge clk)
        begin
         if(!reset)
             begin
               parity_done <= 1'b0;
                 end
                else if(ld_state && (!pkt_valid) && (!fifo_full))
                     begin
                       parity_done <=1'b1;
                     end
                else if(laf_state && (!parity_done) && low_pkt_valid)
                     begin
                       parity_done <=1'b1;
                     end
                else
                parity_done <=1'b0;
           end


// error logic
```

```
464
465      always@(posedge clk)
466          begin
467               if(!reset)
468                    error <=1'b0;
469                else if((packet_parity_byte != internal_parity_byte) && parity_done)
470                 error <=1'b1;
471                else if((packet_parity_byte == internal_parity_byte) && parity_done)
472                     error <=1'b0;
473                else
474                error <= 0;
475
476
477               end
478
479    // low_packet_valid logic this means if ther is no packets packet goes low so
480    // this is negation of pkt valid
481
482       always@(*)
483              begin
484                   if(!reset)
485                low_pkt_valid <=0;
486               else if(parity_done)
487                low_pkt_valid <=1'b1;
488               else if(!pkt_valid)
489                         low_pkt_valid <=1'b1;
490               else
491                low_pkt_valid <=0;
492
493          end
494    endmodule
495
496
497    // synchronizer
498
499
500    module synchronizer_router(detect_add,data_in,write_enb_reg,clk,reset,vld_out_0,vld_out_1
       ,vld_out_2,read_enb_0,read_enb_1,read_enb_2,write_enb,fifo_full,empty_0,empty_1,empty_2,
       soft_reset_0,soft_reset_1,soft_reset_2,full_0,full_1,full_2);
501
502    // declaring the input and output ports
503
504       input detect_add,write_enb_reg,clk,reset,read_enb_0,read_enb_1,read_enb_2,empty_0,
       empty_1,empty_2,full_0,full_1,full_2;
505       input [1:0]data_in;
506       output vld_out_0,vld_out_1,vld_out_2;
507       output reg [2:0]write_enb;
508        output reg fifo_full;
509        output reg soft_reset_0,soft_reset_1,soft_reset_2;
510
511    // creating the internal register
512
513        reg [1:0]addr_data;
514        reg [6:0]out_fifo_0_counter;
515         reg [6:0]out_fifo_1_counter;
516         reg [6:0]out_fifo_2_counter;
517    // storng detecting the destination_address_signal
518
519        always@(posedge clk)
520           begin
521           if(!reset)
522              addr_data <=0;
523            else if(detect_add)
524              addr_data <= data_in;
525           end
526
527    // destinatoin addrss decodeing and sending write enb signal sending for
528    // respective fifo
529
```

```
     always@(*)
         begin
             if(!reset)
               write_enb=0;
             else
           begin
                 if(write_enb_reg)
                   begin
               case(addr_data)
                  2'b00:write_enb=3'b001;
                  2'b01:write_enb=3'b010;
                  2'b10:write_enb=3'b100;
                  default:write_enb=3'b000;
               endcase
                   end
                  else
                 write_enb=3'b000;
               end
         end

  // fifo full signal asserted based on full status of each fifo

      always@(*)
       begin
              if(!reset)
                fifo_full=0;
              else
             begin
                 case(addr_data)
                   2'b00:fifo_full=full_0;
                   2'b01:fifo_full=full_1;
                   2'b10:fifo_full=full_2;
                default:fifo_full=0;
                 endcase
             end
        end
  // valid out signal genration

     assign vld_out_0=~empty_0;
     assign vld_out_1=~empty_1;
     assign vld_out_2=~empty_2;

  // soft start signal generation foer each fifo
  // the respective     soft reset signal go high if read _ en  is not asserted
  // with in colk cycle so we take colkc counter to counte clks
  // vld out being assrent being assreted and taking the reset condiotn also

  wire count_0 = (out_fifo_0_counter == 6'd29) ? 1'b1:1'b0;
  wire count_1 = (out_fifo_1_counter == 6'd29) ? 1'b1:1'b0;
  wire count_2 = (out_fifo_2_counter == 6'd29) ? 1'b1:1'b0;

     always@(posedge clk)
        begin
              if(!reset)
                   begin
                    soft_reset_0<=0;
                     out_fifo_0_counter<=0;
                end
               else if(!vld_out_0)
                begin
                    soft_reset_0<=0;
                     out_fifo_0_counter<=0;
                end
                   else if (read_enb_0)
                     begin
                        soft_reset_0<=0;
                         out_fifo_0_counter <=0;
                       end
               else if(count_0 )
```

```
                begin
                 out_fifo_0_counter <=0;
                     soft_reset_0<=1;
                end
               else
                begin
                out_fifo_0_counter <= out_fifo_0_counter+1;
                soft_reset_0<=0;
            end
           end

               always@(posedge clk)
      begin
             if(!reset)
                 begin
                  soft_reset_1<=0;
                   out_fifo_1_counter<=0;
                end
               else if(!vld_out_1)
                begin
                    soft_reset_1<=0;
                     out_fifo_1_counter<=0;
                end
                   else if (read_enb_1)
                     begin
                        soft_reset_1 <=0;
                         out_fifo_1_counter <=0;
                       end
               else if(count_1)
               begin
                 out_fifo_1_counter <=0;
                     soft_reset_1<=1;
                end
               else
                begin
                out_fifo_1_counter <= out_fifo_1_counter+1;
                soft_reset_1<=0;
            end
           end

               always@(posedge clk)
      begin
             if(!reset)
                 begin
                  soft_reset_2<=0;
                   out_fifo_2_counter<=0;
                end
               else if(!vld_out_2)
                begin
                    soft_reset_2<=0;
                     out_fifo_2_counter<=0;
                end
                   else if (read_enb_2)
                     begin
                        soft_reset_2 <=0;
                         out_fifo_2_counter <=0;
                       end
               else if(count_2)
               begin
                 out_fifo_2_counter <=0;
                     soft_reset_2<=1;
                end
               else
                begin
                out_fifo_2_counter <= out_fifo_2_counter+1;
                soft_reset_2<=0;
            end
           end
endmodule
```

```
668
669
670
671   // top module
672
673   module router_top(clk,reset,read_enb_0,read_enb_1,read_enb_2,data_in,pkt_valid,data_out_0
      ,data_out_1,data_out_2,valid_out_0,valid_out_1,valid_out_2,error,busy);
674
675   // input and output declaration
676       input clk,reset,read_enb_0,read_enb_1,read_enb_2,pkt_valid;
677       input [7:0]data_in;
678       output  [7:0]data_out_0,data_out_1,data_out_2;
679       output valid_out_0,valid_out_1,valid_out_2,error,busy;
680   // declaring the internal wire for sub block connections
681   // fifo output wires
682
683      wire empty_0,full_0,empty_1,full_1,empty_2,full_2;
684
685   // synchronizer output wires
686
687      wire soft_reset_0,soft_reset_1,soft_reset_2;
688      wire [2:0]write_enb;
689
690   // fsm output wires
691
692     wire detect_add,ld_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state;
693
694   // register output internal connections
695
696     wire parity_done,low_pkt_valid;
697     wire [7:0]data_out;
698
699   /*
700   module
      fsm_router_controller(clk,reset,pkt_valid,busy,parity_done,data_in,soft_reset_0,soft_rese
      t_1,soft_reset_2,fifo_full,low_pkt_valid,fifo_empty_0,fifo_empty_1,fifo_empty_2,detect_ad
      d,ld_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state);
701   */
702
703   fsm_router_controller FSM(clk,reset,pkt_valid,busy,parity_done,data_in[1:0],soft_reset_0,
      soft_reset_1,soft_reset_2,fifo_full,low_pkt_valid,empty_0,empty_1,empty_2,detect_add,
      ld_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state);
704
705   /*
706   module
      synchronizer_router(detect_add,data_in,write_enb_reg,clk,reset,vld_out_0,vld_out_1,vld_ou
      t_2,read_enb_0,read_enb_1,read_enb_2,write_enb,fifo_full,empty_0,empty_1,empty_2,soft_res
      et_0,soft_reset_1,soft_reset_2,full_0,full_1,full_2);
707   */
708
709   synchronizer_router SYNCHRONIZER(
710       detect_add,data_in[1:0],write_enb_reg,clk,reset,valid_out_0,valid_out_1,valid_out_2,
       read_enb_0,read_enb_1,read_enb_2,write_enb,fifo_full,empty_0,empty_1,empty_2,
       soft_reset_0,soft_reset_1,soft_reset_2,full_0,full_1,full_2);
711
712   /*
713
714    module
       register_router(clk,reset,pkt_valid,data_in,fifo_full,rst_int_reg,detect_add,ld_state,la
       f_state,full_state,lfd_state,parity_done,low_pkt_valid,error,dout);
715   */
716
717   register_router REGISTER(clk,reset,pkt_valid,data_in,fifo_full,rst_int_reg,detect_add,
      ld_state,laf_state,full_state,lfd_state,parity_done,low_pkt_valid,error,data_out);
718
719   /*
720
721   module
      fifo_16x9_router(clk,reset,data_in,write_enb,read_enb,soft_reset,lfd_state,full,empty,dat
```

```
    a_out);
    */
722
723
724 fifo_16x9_router FIFO1(clk,reset,data_out,write_enb[0],read_enb_0,soft_reset_0,lfd_state,
    full_0,empty_0,data_out_0);
725 fifo_16x9_router FIFO2(clk,reset,data_out,write_enb[1],read_enb_1,soft_reset_1,lfd_state,
    full_1,empty_1,data_out_1);
726 fifo_16x9_router FIFO3(clk,reset,data_out,write_enb[2],read_enb_2,soft_reset_2,lfd_state,
    full_2,empty_2,data_out_2);
727
728
729 endmodule
730
731
732
733
734 // interface for source
735
736 interface router_source_if(input bit clk);
737
738
739 //  input and output declaration
740 //    input clk,reset,read_enb_0,read_enb_1,read_enb_2,pkt_valid;
741 //    input [7:0]data_in;
742 //    output  [7:0]data_out_0,data_out_1,data_out_2;
743 //    output valid_out_0,valid_out_1,valid_out_2,error,busy;
744
745
746
747        logic [7:0]data_in;
748        bit reset;
749        bit pkt_valid;
750        bit error;
751        bit busy;
752
753 // declaring the clocking block for source driver
754
755        clocking source_drv@(posedge clk);
756            default input#1 output#1;
757
758                output reset;
759                output pkt_valid;
760                output data_in;
761                input  busy;
762                input  error;
763
764        endclocking
765
766 // declaring the clocking  block for source monitor
767
768        clocking source_mon@(posedge clk);
769            default input#1 output#1;
770
771                input reset;
772                input pkt_valid;
773                input data_in;
774                input busy;
775                input error;
776
777        endclocking
778
779 // modports for mon and drv
780
781          modport S_MON(clocking source_mon);
782          modport S_DRV(clocking source_drv);
783
784 endinterface
785
786
```

```
787
788
789
790
791   // destination interface
792
793   interface router_destin_if(input bit clk);
794
795   //  input and output declaration
796   //    input clk,reset,read_enb_0,read_enb_1,read_enb_2,pkt_valid;
797   //    input [7:0]data_in;
798   //    output  [7:0]data_out_0,data_out_1,data_out_2;
799   //    output valid_out_0,valid_out_1,valid_out_2,error,busy;
800
801
802          logic [7:0]data_out;
803          bit read_enb;
804          bit valid_out;
805
806
807   // clocking block for destination driver
808
809        clocking destin_drv@(posedge clk);
810              default input#1 output#1;
811
812                  output read_enb;
813                  input  valid_out;
814                  input data_out;
815
816        endclocking
817
818   // clocking block for destination monitor
819
820        clocking destin_mon@(posedge clk);
821             default input #1 output #1;
822
823               input data_out;
824               input read_enb;
825               input valid_out;
826
827
828          endclocking
829
830   // modports for destin drv and mon
831
832         modport D_MON(clocking destin_mon);
833         modport D_DRV(clocking destin_drv);
834
835   endinterface
836
837
838
839   // Top module uvm
840
841
842   module top();
843
844   // including the  uvm file
845
846          import uvm_pkg :: *;
847         // include "uvm_macros.svh"
848
849   // importing  the tb files
850
851          import router_pkg :: *;
852
853
854   // generating clock
855
```

```
        bit clk = 1'b0;
        always #10 clk = ~clk;

// instantiating interace

         router_source_if VIF(clk);
         router_destin_if VIF0(clk);
         router_destin_if VIF1(clk);
         router_destin_if VIF2(clk);



 // instantiating DUT
// top module
//module
router_top(clk,reset,read_enb_0,read_enb_1,read_enb_2,data_in,pkt_valid,data_out_0,data_o
ut_1,data_out_2,valid_out_0,valid_out_1,valid_out_2,error,busy);


     router_top DUT(
                    .clk(clk),
                    .reset(VIF.reset),
                    .read_enb_0(VIF0.read_enb),
                    .read_enb_1(VIF1.read_enb),
                    .read_enb_2(VIF2.read_enb),
                    .data_in(VIF.data_in),
                    .pkt_valid(VIF.pkt_valid),
                    .data_out_0(VIF0.data_out),
                    .data_out_1(VIF1.data_out),
                    .data_out_2(VIF2.data_out),
                    .valid_out_0(VIF0.valid_out),
                    .valid_out_1(VIF1.valid_out),
                    .valid_out_2(VIF2.valid_out),
                    .error(VIF.error),
                    .busy(VIF.busy)

                     );


        initial begin

                  `ifdef VCS
                   $fsdbDumpvars(0,top);
                  `endif

// setting the virtual interface  for source agent and 3 destination agents
                  uvm_config_db #(virtual router_source_if) :: set(null,"*","vif",VIF);
                  uvm_config_db #(virtual router_destin_if) :: set(null,"*","vif0",VIF0);
                  uvm_config_db #(virtual router_destin_if) :: set(null,"*","vif1",VIF1);
                  uvm_config_db #(virtual router_destin_if) :: set(null,"*","vif2",VIF2);
// calling the runtest()

                   run_test();
             end


endmodule







// class for destinationj agetn configratoion data base object


class destin_agent_config extends uvm_object;


```

```
// factory registration

          `uvm_object_utils(destin_agent_config)

// declare the varible required

            uvm_active_passive_enum is_active = UVM_ACTIVE;
// declare the virtual inteface handle

                virtual router_destin_if  vif;


// fucntion new construction

           function new(string name = "destin_agent_config");

                        super.new(name);

           endfunction



endclass







// source  agent config

class source_agent_config extends uvm_object;

// factory registration

        `uvm_object_utils(source_agent_config)


     uvm_active_passive_enum is_active = UVM_ACTIVE;

// declare virtual interfae handle

        virtual router_source_if vif;

// function new constructor


       function new(string name = "source_agent_config" );

             super.new(name);

       endfunction




endclass



//destination driver class

class destin_driver extends uvm_driver #(destin_xtn);
```

```
992
993   // factory registration
994
995           `uvm_component_utils(destin_driver)
996
997   // declare the virtual interface handle
998
999           destin_agent_config m_cfg;
1000
1001          virtual router_destin_if.D_DRV vif;
1002
1003  // funcion new constructor
1004
1005
1006        function new (string name = "destin_driver", uvm_component parent = null );
1007
1008
1009                super.new(name,parent);
1010
1011         endfunction
1012
1013  // build phase
1014
1015          function void build_phase(uvm_phase phase);
1016
1017                  super.build_phase(phase);
1018                 // `uvm_info(get_full_name(),"this is driver destin",UVM_NONE)
1019                  if(!uvm_config_db #(destin_agent_config) :: get(this,"",
                    "destin_agent_config",m_cfg))
1020                        `uvm_fatal(get_full_name(),"cannot get  the destin agetn config
                       handle driver  m_cfg from desti agetn top")
1021
1022          endfunction
1023  // connect phase connect the virtual interface to local interface
1024
1025          function void connect_phase(uvm_phase phase);
1026
1027                 vif = m_cfg.vif;
1028
1029          endfunction
1030
1031
1032   // task run_phase here we send req and and then data req sended to new method for
      driving
1033
1034           task run_phase(uvm_phase phase);
1035
1036                forever
1037                      begin
1038                              super.run_phase(phase);
1039
1040                              seq_item_port.get_next_item(req);
1041
1042
1043                                send_to_dut();
1044                                req.print();
1045
1046                               seq_item_port.item_done();
1047
1048                      end
1049           endtask
1050
1051
1052  // task send to dut
1053
1054          task send_to_dut();
1055
1056                   while(vif.destin_drv.valid_out !== 1'b1)
1057                   begin
```

```
                        @(vif.destin_drv);
                      end

                    repeat(req.delay)
                          begin
                               @(vif.destin_drv);
                          end
                    vif.destin_drv.read_enb <= 1'b1;


                    while(vif.destin_drv.valid_out !== 1'b0)
                   begin
                               @(vif.destin_drv);
              end

                    vif.destin_drv.read_enb <= 1'b0;
            @(vif.destin_drv);


        endtask



endclass







// destination monitor class

class destin_monitor extends uvm_monitor;

// factory registration

        `uvm_component_utils(destin_monitor)

// declare virtual interface handle

            destin_agent_config m_cfg;
            virtual router_destin_if.D_MON vif;
// declaring the destin transaction

                  destin_xtn dmon;

// tlm port for write to score board

        uvm_analysis_port #(destin_xtn) dmon_port;

// function new constructor


        function new(string name = "destin_monitor" , uvm_component parent = null );

                    super.new(name,parent);
                    dmon_port = new ("dmon_port",this);

         endfunction

// build phase
         function void build_phase(uvm_phase phase);

                 super.build_phase(phase);
                // `uvm_info(get_full_name(),"this is monitor destin",UVM_NONE)
                  if(!uvm_config_db #(destin_agent_config) :: get(this,"",
```

```
                    "destin_agent_config",m_cfg))
1127                      `uvm_fatal(get_full_name(),"cannot get  the destin agetn config
                          handle monitor m_cfg from desti agetn top")
1128
1129          endfunction
1130
1131  // connect_phase connecting the virtual interfcce to local interface handle
1132
1133          function void connect_phase(uvm_phase phase);
1134
1135                   vif = m_cfg.vif;
1136
1137          endfunction
1138
1139  // run phase
1140
1141
1142           task run_phase(uvm_phase phase);
1143
1144                forever
1145                      begin
1146
1147                         collect_data();
1148                         dmon.print();
1149                         dmon_port.write(dmon);
1150                      end
1151
1152           endtask
1153
1154  // collect the data from interface
1155
1156          task collect_data();
1157
1158                dmon = destin_xtn :: type_id :: create("dmon");
1159
1160                while( vif.destin_mon.read_enb !== 1'b1)
1161                  begin
1162                              @(vif.destin_mon);
1163                       end
1164                 @(vif.destin_mon);
1165                 dmon.header_byte = vif.destin_mon.data_out;
1166
1167                 @(vif.destin_mon);
1168                 dmon.payload = new[dmon.header_byte[7:2]];
1169                foreach(dmon.payload[i])
1170                    begin
1171                         while( vif.destin_mon.valid_out !== 1'b1)
1172                              begin
1173                                   @(vif.destin_mon);
1174                              end
1175                         dmon.payload[i] = vif.destin_mon.data_out;
1176                         @(vif.destin_mon);
1177
1178                    end
1179                 dmon.parity_byte = vif.destin_mon.data_out;
1180                    @(vif.destin_mon);
1181
1182          endtask
1183
1184
1185  endclass
1186
1187
1188
1189
1190
1191
1192
1193
```

```
1194
1195  // destin seq class
1196
1197  class destin_seq extends uvm_sequence #(destin_xtn);
1198
1199  // factory registation
1200
1201            `uvm_object_utils(destin_seq)
1202
1203  // function new constructor
1204
1205          function new(string name = "destin_seq");
1206
1207                    super.new(name);
1208
1209          endfunction
1210  //build phase
1211
1212
1213
1214  // task body for ganerate stimulus
1215
1216         task body();
1217         req = destin_xtn :: type_id :: create ("req");
1218         start_item(req);
1219
1220
1221            assert(req.randomize() with {(req.delay >0) && (req.delay < 30);})
1222                 else
1223                     begin
1224                         `uvm_fatal(get_full_name()," randomization not  happend in
                            destinaion  sequence  check it")
1225                     end
1226          finish_item(req);
1227
1228         endtask
1229
1230
1231
1232  endclass
1233
1234
1235
1236  /destination sequencer class
1237
1238  class destin_seqr extends uvm_sequencer #(destin_xtn);
1239
1240  // factory registration
1241
1242        `uvm_component_utils(destin_seqr)
1243
1244  // fuinction new constructor
1245
1246        function new (string name="destin_seqr" , uvm_component parent= null);
1247
1248                  super.new(name,parent);
1249
1250
1251        endfunction
1252
1253  // build phase
1254
1255       /*    function void build_phase(uvm_phase phase);
1256
1257                    super.build_phase(phase);
1258
1259           endfunction
1260  */
1261
```

```
1262  endclass
1263
1264
1265
1266
1267
1268  // class deriver for source
1269
1270  class source_driver extends uvm_driver #(source_xtn);
1271
1272
1273  // factory registration
1274
1275        `uvm_component_utils(source_driver)
1276
1277
1278  // declare the virtual interface
1279
1280           virtual router_source_if.S_DRV vif;
1281            source_agent_config m_cfg;
1282
1283  // function new constructor
1284
1285         function new(string name = "source_driver" , uvm_component parent = null);
1286
1287
1288                super.new(name,parent);
1289
1290
1291         endfunction:new
1292
1293  // build phase
1294
1295           function void build_phase(uvm_phase phase);
1296
1297                   super.build_phase(phase);
1298                  //`uvm_info(get_full_name(),"this is driver soruce",UVM_NONE)
1299                  if(!uvm_config_db #(source_agent_config) :: get(this,"",
                    "source_agent_config",m_cfg))
1300                         `uvm_fatal(get_full_name(),"cannot get  the source agent config
                          handle driver m_cfg from source agent top")
1301
1302           endfunction
1303
1304  // connect phase
1305
1306            function void connect_phase(uvm_phase phase);
1307
1308                     vif = m_cfg.vif;
1309
1310            endfunction
1311
1312  // task run phase and send to dut method calling
1313
1314           task run_phase(uvm_phase phase);
1315                 super.run_phase(phase);
1316                       @(vif.source_drv);
1317                      vif.source_drv.reset <= 1'b0;
1318                      @(vif.source_drv);
1319                      vif.source_drv.reset <= 1'b1;
1320
1321
1322                 forever begin
1323
1324                 seq_item_port.get_next_item(req);
1325
1326                 send_to_dut(req);
1327                 seq_item_port.item_done();
1328                 req.print();
```

```
                end
         endtask

// send to dut

      task send_to_dut(source_xtn req);

           while(vif.source_drv.busy != 1'b0)
           begin
                 @(vif.source_drv);
           end
           vif.source_drv.pkt_valid <= 1'b1;
           vif.source_drv.data_in   <=  req.header_byte;
           @(vif.source_drv);
           foreach(req.payload[i])
                    begin
                        while(vif.source_drv.busy != 1'b0)
            begin
                            @(vif.source_drv);
            end
                        vif.source_drv.data_in <= req.payload[i];
                        @(vif.source_drv);
                    end

          vif.source_drv.pkt_valid <= 1'b0;
          vif.source_drv.data_in <= req.parity_byte;
           repeat(2)
          @(vif.source_drv);
       endtask




endclass




// class source monitor

class source_monitor extends uvm_monitor;

// factory registration

     `uvm_component_utils(source_monitor)

// dfeclare  virtual interface

          virtual router_source_if.S_MON vif;
          source_agent_config m_cfg;
          source_xtn smon;

// declaring the anlysis port

          uvm_analysis_port #(source_xtn) smon_port;


// function new constructor

        function new (string name = "source_monitor",uvm_component parent = null );

              super.new(name,parent);
              smon_port = new("smon_port",this);

        endfunction

// build phase
```

```
1398
1399          function void build_phase(uvm_phase phase);
1400
1401                  super.build_phase(phase);
1402                // `uvm_info(get_full_name(),"this is monior soruce",UVM_NONE)
1403                 if(!uvm_config_db #(source_agent_config) :: get(this,"",
                    "source_agent_config",m_cfg))
1404                        `uvm_fatal(get_full_name(),"cannot get  the source agent config
                           handle monitor m_cfg from source agent top")
1405
1406
1407          endfunction
1408
1409   // connect phase
1410
1411           function void connect_phase(uvm_phase phase);
1412
1413                    vif = m_cfg.vif;
1414
1415           endfunction
1416
1417   // run phase
1418
1419          task run_phase(uvm_phase phase);
1420
1421               forever
1422                     begin
1423
1424                          super.run_phase(phase);
1425                          collect_data();
1426                           smon.print();
1427                           smon_port.write(smon);
1428
1429                     end
1430          endtask
1431
1432   // collect task for from interace to monnitor
1433
1434          task collect_data();
1435              smon = source_xtn :: type_id :: create("smon");
1436              while(vif.source_mon.busy != 1'b0)
1437                    @(vif.source_mon);
1438              while(vif.source_mon.pkt_valid != 1'b1)
1439                    @(vif.source_mon);
1440              smon.header_byte = vif.source_mon.data_in;
1441              @(vif.source_mon);
1442              smon.payload = new[smon.header_byte[7:2]];
1443              foreach(smon.payload[i])
1444                    begin
1445                          while(vif.source_mon.busy != 1'b0)
1446                          begin
1447                          @(vif.source_mon);
1448                          end
1449                          smon.payload[i] = vif.source_mon.data_in;
1450                          @(vif.source_mon);
1451                    end
1452              smon.parity_byte = vif.source_mon.data_in;
1453
1454                @(vif.source_mon);
1455                @(vif.source_mon);
1456                smon.error = vif.source_mon.error;
1457                $display("Signal Error From Source Monitor = %0b",smon.error);
1458
1459          endtask
1460
1461
1462   endclass
1463
1464
```

```
1465
1466
1467
1468
1469  // source sequence class
1470
1471  class source_seq extends uvm_sequence #(source_xtn);
1472
1473  // factory registration
1474
1475       `uvm_object_utils(source_seq)
1476
1477  // addres declaration
1478
1479        bit [1:0] addr;
1480
1481  // function new constructor
1482
1483        function new(string name = "source_seq");
1484
1485              super.new(name);
1486      //`uvm_info(get_full_name(),"this is sequence soruce",UVM_NONE)
1487
1488        endfunction
1489
1490  endclass
1491
1492  //---------------- Extended class  Small packets-----------//
1493
1494  class small_seq extends source_seq;
1495
1496  // factory registration
1497
1498       `uvm_object_utils(small_seq)
1499
1500  // function new constructor
1501
1502       function new (string name = "small_seq");
1503
1504           super.new(name);
1505
1506       endfunction
1507
1508  // body task =generate 1 to 20 packets payload generates
1509
1510        task body();
1511           if(!uvm_config_db #(bit[2])::get(null,get_full_name(),"bit",addr))
1512                     `uvm_fatal(get_full_name(),"cannot get the address from test class are
                          set it check")
1513
1514           req = source_xtn :: type_id :: create("req");
1515           start_item(req);
1516           assert(req.randomize() with {req.header_byte[1:0] == addr;req.header_byte[7:2]
              inside{[1:21]};} )
1517               else
1518                  begin
1519                        `uvm_info(get_full_name(),"!!!!!!!!!!!randomization is failed in
                          source seqs at small seq!!!!!!!!",UVM_LOW)
1520                  end
1521
1522
1523           finish_item(req);
1524
1525        endtask
1526
1527  endclass
1528
1529
1530  // -------------- Extended clas of Medium packets ---------//
```

```
1531
1532
1533  class medium_seq extends source_seq;
1534
1535  // factory registration
1536       `uvm_object_utils(medium_seq)
1537
1538  // function new constructor
1539
1540         function new(string name = "medium_seq");
1541
1542                 super.new(name);
1543
1544         endfunction
1545
1546  // task body for packets stimulus 22 to 41
1547
1548         task body();
1549           if(!uvm_config_db #(bit[2])::get(null,get_full_name(),"bit",addr))
1550                     `uvm_fatal(get_full_name(),"cannot get the address from test class are
                          set it check")
1551
1552             req = source_xtn :: type_id :: create("req");
1553             start_item(req);
1554             assert(req.randomize() with {req.header_byte[7:2] inside {[22:41]};
1555                                          req.header_byte[1:0] == addr; } )
1556                  else
1557                      begin
1558                           `uvm_info(get_full_name(),"!!!!!!!!randomization is failed in
                                source seqs at medium seq!!!!!!!!!!",UVM_LOW)
1559                       end
1560              finish_item(req);
1561
1562         endtask
1563
1564  endclass
1565
1566
1567  // large sequnce for generating the from 42 to 63
1568
1569  class large_seq extends source_seq;
1570
1571  // factory registration
1572
1573       `uvm_object_utils(large_seq)
1574
1575  // function new constructor
1576
1577          function new(string name = "large_seq" );
1578
1579                    super.new(name);
1580
1581           endfunction
1582  // task body for 42 to 63
1583
1584
1585         task body();
1586          if(!uvm_config_db #(bit[2])::get(null,get_full_name(),"bit",addr))
1587                     `uvm_fatal(get_full_name(),"cannot get the address from test class are
                          set it check")
1588
1589             req = source_xtn :: type_id :: create("req");
1590             start_item(req);
1591             assert(req.randomize() with {req.header_byte[7:2] inside {[42:63]};
1592                                          req.header_byte[1:0] == addr; } )
1593                  else
1594                      begin
1595                           `uvm_info(get_full_name(),"!!!!!!!!randomization is failed in
                                source seqs at medium seq!!!!!!!!!!",UVM_LOW)
```

```
                    end
            finish_item(req);
        endtask

endclass




// source  sequnce class

class source_seqr extends uvm_sequencer #(source_xtn);

// factory registration

     `uvm_component_utils(source_seqr)

// fcuntion new constructor

      function new(string name ="source_seqr", uvm_component parent = null);

              super.new(name,parent);

      endfunction:new




endclass




// virtual sequnce class

class virtual_seq extends uvm_sequence #(uvm_sequence_item);

// factory registration

      `uvm_object_utils(virtual_seq)

// declaring the handle for sequnces

           small_seq   s_seq;
           medium_seq  m_seq;
           large_seq   l_seq;

// declaring the handle for virtual sequncer

          virtual_seqr v_seqr;

// declare dynamic handle for physical seqr

           source_seqr ps_seqr[];

// declaring the hande for env_config to getting

          router_env_config m_cfg;

// function new constructor
```

```
1665
1666                 function new(string name = "virtual_seq");
1667
1668                     super.new(name);
1669
1670                  endfunction:new
1671
1672  // task body for assging the v sequncer to m_sequncer and assingn the these handle to
     env_config class
1673
1674                task body();
1675
1676                     if(! uvm_config_db #(router_env_config) :: get(null,get_full_name
                         (),"router_env_config",m_cfg) )
1677                                    `uvm_fatal(get_full_name(),"canot get the
                                       router_env_config handle m_cdfg from test in
                                       virtual sequence")
1678
1679
1680                      ps_seqr = new[m_cfg.no_of_source];
1681
1682  // assiging the m_sequncer to physical virtual sequncer through $cast
1683
1684                     assert($cast(v_seqr,m_sequencer))
1685                               else
1686                                   begin
1687                                         `uvm_error(get_full_name(),"error becuase of
                                            cast not working in the virtual sequence check
                                            that")
1688                                   end
1689                     foreach(ps_seqr[i])
1690                                begin
1691                                     ps_seqr[i] = v_seqr.s_seqr[i];
1692                                end
1693
1694                endtask:body
1695
1696  endclass
1697
1698
1699  // Extended class from virtual_seq to small seq sequnce class
1700
1701  class virtual_small_seq extends virtual_seq;
1702
1703
1704  // factory registration
1705
1706            `uvm_object_utils(virtual_small_seq)
1707
1708  // function new constructor
1709
1710          function new( string name  = "virtual_small_seq");
1711
1712                    super.new(name);
1713
1714          endfunction
1715
1716
1717  // task body
1718
1719           task body();
1720
1721              s_seq = small_seq :: type_id :: create("s_seq");
1722              super.body();
1723              for(int i=0; i < m_cfg.no_of_source ; i++)
1724              begin
1725
1726                    s_seq.start(ps_seqr[i]);
1727
```

```
1728                end
1729
1730             endtask
1731
1732
1733    endclass
1734
1735
1736    // Extended class for medium seq
1737
1738    class virtual_medium_seq extends virtual_seq;
1739
1740    // factory registration
1741
1742           `uvm_object_utils(virtual_medium_seq)
1743
1744    // function new constructor
1745
1746           function new (string name = "virtual_medium_seq" );
1747
1748                        super.new(name);
1749
1750           endfunction
1751    //  task body
1752
1753            task body();
1754                  m_seq = medium_seq :: type_id :: create("m_seq");
1755                  super.body();
1756                  for(int i=0; i<m_cfg.no_of_source ; i++)
1757                       begin
1758
1759                           m_seq.start(ps_seqr[i]);
1760
1761                       end
1762            endtask
1763
1764    endclass
1765
1766
1767    // Extended class for large seq
1768
1769    class virtual_large_seq extends virtual_seq;
1770
1771    // factory registration
1772
1773             `uvm_object_utils(virtual_large_seq)
1774
1775    // function new constructor
1776
1777             function new(string name ="virtual_large_seq");
1778
1779                    super.new(name);
1780
1781             endfunction
1782
1783
1784    //  task body
1785
1786
1787              task body();
1788                      l_seq = large_seq :: type_id :: create("l_seq");
1789                      super.body();
1790                       for(int i=0; i< m_cfg.no_of_source ; i++)
1791                            begin
1792
1793                                l_seq.start(ps_seqr[i]);
1794
1795                             end
1796              endtask
```

```
1797
1798  endclass
1799
1800
1801  // virtual sequncer class
1802
1803  class virtual_seqr extends uvm_sequencer #(uvm_sequence_item);
1804
1805  // factory registration
1806
1807          `uvm_component_utils(virtual_seqr)
1808
1809
1810  // declaring the handle for source and destination sequencer
1811           router_env_config m_cfg;
1812           source_seqr s_seqr[];
1813
1814
1815  // fuunction new construction
1816
1817           function new(string name= "virtual_seqr" , uvm_component parent = null);
1818
1819                           super.new(name,parent);
1820
1821          endfunction
1822
1823  // build phase
1824
1825           function void build_phase(uvm_phase phase);
1826
1827                   super.build_phase(phase);
1828                   if(!uvm_config_db #(router_env_config) :: get(this,"",
                       "router_env_config",m_cfg))
1829                         `uvm_fatal(get_full_name()," cannot get the router_env_config
                            handle mcfg from tesdt class in virtual sequncerr")
1830
1831                   s_seqr = new[m_cfg.no_of_source];
1832           endfunction
1833
1834
1835
1836
1837  endclass:virtual_seqr
1838
1839
1840
1841
1842
1843  class tb_scoreboard extends uvm_scoreboard;
1844
1845
1846  // factory registration
1847
1848            `uvm_component_utils(tb_scoreboard)
1849
1850  // declaring the  handle for env_config
1851
1852            router_env_config m_cfg;
1853
1854  // declaring the handle for tlm ports
1855
1856           uvm_tlm_analysis_fifo #(source_xtn) source_fifo[];
1857           uvm_tlm_analysis_fifo #(destin_xtn) destin_fifo[];
1858
1859  // declaring the source and destination transaction class
1860
1861          source_xtn s_xtn;
1862          source_xtn source_cov_xtn;
1863
```

```
1864          destin_xtn d_xtn;
1865          destin_xtn destin_cov_xtn;
1866
1867
1868    // write fucntion covrage cover group
1869
1870        covergroup source_cg;
1871
1872          ADDER:   coverpoint s_xtn.header_byte[1:0]{
1873
1874                                                                  bins b1 = {2'b00};
1875                                                                  bins b2 = {2'b01};
1876                                                                  bins b3 = {2'b10};



1877
1878                                                                }
1879          DATA_IN: coverpoint s_xtn.header_byte[7:2]{
1880
1881                                                                  bins b1 = {[1:21]};
1882                                                                  bins b2 = {[22:41]};
1883                                                                  bins b3 = {[42:63]};
1884
1885                                                                   }
1886          ERROR:   coverpoint s_xtn.error {
1887
1888                                                                  bins b1 = { 1'b1 };
1889                                                                  bins b2 = { 1'b0 };
1890
1891                                                    }
1892
1893         /* BUSY:     coverpoint s_xtn.busy {
1894
1895                                                                  bins b1 = { 1'b1 };
1896                                                                  bins b2 = { 1'b0
                                                                    };


1897
1898                                                   } */
1899
1900         // CROSS_ALL: cross
             ADDER,DATA_IN,ERROR,BUSY;
1901
1902        endgroup
1903
1904        covergroup destin_cg;
1905
1906            VALID_OUT: coverpoint d_xtn.valid_out{
1907
1908                                                                  bins b1 = {1'b1};
1909                                                                  bins b2 = {1'b0};
1910
1911                                                                }
1912            DATA_OUT: coverpoint d_xtn.header_byte[7:2]{
1913
1914                                                                  bins b1 = {[1:21]};
1915                                                                  bins b2 = {[22:41]};
1916                                                                  bins b3 = {[42:63]};
1917
1918                                                                   }



1919
1920        endgroup
1921
1922
1923    // we have to implement check method  for comparing the data
```

```
1924
1925
1926   // function new constructor
1927
1928            function new(string name = "tb_scoreboard",uvm_component parent= null );
1929
1930                  super.new(name,parent);
1931                  source_cg = new();
1932                  destin_cg = new();
1933
1934            endfunction:new
1935
1936
1937   // build_phase
1938
1939           function void build_phase(uvm_phase phase);
1940
1941                       super.build_phase(phase);
1942
1943                       if(!uvm_config_db #(router_env_config) :: get (this,"",
                           "router_env_config",m_cfg))
1944                            begin
1945                           `uvm_fatal(get_full_name(),"cannot get he router_env_config
                           handle m_cg in scoreboard ")
1946                            end
1947
1948
1949                       source_fifo = new [m_cfg.no_of_source];
1950
1951                       foreach(source_fifo[i])
1952                             begin
1953                                  source_fifo[i] = new ("source_fifo",this);
1954                              end
1955
1956                       destin_fifo = new[m_cfg.no_of_destin];
1957
1958                       foreach(destin_fifo[i])
1959                             begin
1960                                 destin_fifo[i] = new($sformatf("destin_fifo[%0d]",i),this);
1961                             end
1962           endfunction
1963
1964
1965          task run_phase(uvm_phase phase);
1966                super.run_phase(phase);
1967                forever
1968                       begin
1969                            fork
1970                              begin
1971
1972                                   source_fifo[0].get(s_xtn);
1973                                   source_cg.sample();
1974
1975                              end
1976                              fork
1977                                  begin
1978
1979                                       destin_fifo[0].get(d_xtn);
1980                                       destin_cg.sample();
1981
1982                                  end
1983                                  begin
1984
1985                                       destin_fifo[1].get(d_xtn);
1986                                       destin_cg.sample();
1987
1988                                  end
1989                                  begin
1990
```

```
1991                                    destin_fifo[2].get(d_xtn);
1992                                    destin_cg.sample();
1993
1994                                end
1995                            join_any
1996                            disable fork;
1997                        join
1998                        compare(s_xtn,d_xtn);
1999                    $display(
                        "\n=================================================================
                        ======\n");
2000                    $display("\n source  side functional coverage = %.3f \n",source_cg.
                        get_coverage);
2001                    $display("\n destin  side functional coverage = %.3f \n",destin_cg.
                        get_coverage);
2002                    $display(
                        "\n=================================================================
                        ======\n");
2003                      end
2004
2005        endtask:run_phase
2006
2007
2008  // task for comparing the source and destination packets
2009
2010        task compare(source_xtn s_xtn,destin_xtn d_xtn);
2011
2012              bit check;
2013
2014              if(s_xtn.header_byte == d_xtn.header_byte)
2015                   begin
2016                        $display("\n==================== Header_Byte Matched
                            SuccessFull ====================\n");
2017                        check  = 1'b1;
2018                   end
2019              else
2020                   begin
2021                        $display("\n==================== Header_Byte Not Matched
                            ====================\n");
2022                        check  = 1'b0;
2023                   end
2024
2025               if(s_xtn.payload == d_xtn.payload)
2026                   begin
2027                        $display("\n==================== Payload Matched SuccessFull
                            ====================\n");
2028                        check  = 1'b1;
2029                   end
2030              else
2031                   begin
2032                        $display("\n==================== Payload Not Matched
                            ====================\n");
2033                        check  = 1'b0;
2034                   end
2035               if(s_xtn.parity_byte == d_xtn.parity_byte)
2036                   begin
2037                        $display("\n==================== Parity_Byte Matched
                            SuccessFull ====================\n");
2038                        check  = 1'b1;
2039                   end
2040              else
2041                   begin
2042                        $display("\n==================== Parity_Byte Not Matched
                            ====================\n");
2043                        check  = 1'b0;
2044                   end
2045
2046               if(check == 1'b1)
2047                    begin
```

```
2048
2049                        $display(
                            "\n================================================================
                            ==========\n");
2050                        $display("\n     SUCCESSFULLY MATCHED    \n");
2051                        $display(
                            "\n================================================================
                            ==========\n");
2052
2053                     end
2054                else
2055                    begin
2056
2057                        $display(
                            "\n================================================================
                            ==========\n");
2058                        $display("\n     PACKETS NOT MATCHED    \n");
2059                        $display(
                            "\n================================================================
                            ==========\n");
2060                    end
2061
2062
2063
2064         endtask
2065
2066
2067
2068
2069   endclass:tb_scoreboard
2070
2071
2072
2073
2074   // environment class
2075
2076   class tb_env extends uvm_env;
2077
2078   // factory registration
2079
2080            `uvm_component_utils(tb_env)
2081
2082   // declaring the handle for vritual seqr and source agent top and destin agent top i
     dont know
2083
2084            source_agent_top  source_agth;
2085            destin_agent_top  destin_agth;
2086            tb_scoreboard     sb;
2087
2088
2089   // declaring the int varialbles for no of wr agent and  no of read agent - idont know
2090
2091
2092   // declaring for the scoreboard and virtual sequencer no  i dont know
2093
2094            virtual_seqr  v_seqr;
2095
2096   // declaring the  handle for the env_config class to get and
2097
2098          router_env_config m_cfg;
2099
2100   // construction new function
2101
2102            function new(string name = "tb_env", uvm_component parent = null);
2103
2104                        super.new(name,parent);
2105
2106            endfunction:new
2107
```

```
2108  // build phase
2109
2110          function void build_phase(uvm_phase phase);
2111                super.build_phase(phase);
2112                 if(! uvm_config_db #(router_env_config) :: get(this,"","router_env_config"
                     ,m_cfg))
2113                       `uvm_fatal(get_full_name(),"cannot get the router_env_config handle
                           m_cfg from test in env_tb ")
2114                 source_agth = source_agent_top :: type_id :: create("source_agth",this);
2115                 destin_agth = destin_agent_top :: type_id :: create("destin_agth",this);
2116                 sb          = tb_scoreboard    :: type_id :: create("sb",this);
2117                 v_seqr      = virtual_seqr     :: type_id :: create("v_seqr",this);
2118
2119          endfunction
2120
2121
2122  // connect phase
2123
2124         function void connect_phase(uvm_phase phase);
2125
2126                foreach(v_seqr.s_seqr[i])
2127                      begin
2128                         v_seqr.s_seqr[i] = source_agth.agth[i].seqr;
2129                      end
2130
2131                for(int j = 0; j < m_cfg.no_of_source;j++)
2132                      begin
2133                           source_agth.agth[j].monh.smon_port.connect(sb.source_fifo[j].
                             analysis_export);
2134                      end
2135
2136                for(int i = 0; i < m_cfg.no_of_destin ; i++)
2137                      begin
2138                           destin_agth.agth[i].monh.dmon_port.connect(sb.destin_fifo[i].
                             analysis_export);
2139                      end
2140
2141         endfunction:connect_phase
2142
2143
2144
2145  endclass:tb_env
2146
2147
2148
2149
2150
2151  // test class
2152
2153  class base_test extends uvm_test;
2154
2155  // factory registration
2156
2157           `uvm_component_utils(base_test)
2158
2159  // declaring the test bench env class handle and env_config handles
2160
2161             tb_env envh;
2162             router_env_config m_cfg;
2163
2164             source_agent_config  m_source_agth[];
2165             destin_agent_config  m_destin_agth[];
2166
2167              int no_of_source = 1;
2168              int no_of_destin = 3;
2169
2170  // header byte destination address setting
2171
2172             bit [2]addr;
```

```
2173
2174   // funciton new constructor
2175
2176            function new(string name = "base_test",uvm_component parent = null );
2177
2178                    super.new(name,parent);
2179
2180            endfunction
2181   // config function
2182
2183                 function void config_data();
2184
2185                              m_source_agth = new[no_of_source];
2186                              m_destin_agth = new[no_of_destin];
2187
2188                              foreach(m_source_agth[i])
2189                                     begin
2190                                          m_source_agth[i] = source_agent_config :: type_id
                                              :: create($sformatf("m_source_agth[%0d]",i));
2191                                          m_source_agth[i].is_active = UVM_ACTIVE;
2192                                          if(!uvm_config_db #(virtual router_source_if) ::
                                              get(this,"","vif",m_source_agth[i].vif))
2193                                               `uvm_fatal(get_full_name(),"cannot get the
                                                  interface handle from source are set in top"
                                                  )
2194
2195                                     end
2196                              foreach(m_destin_agth[i])
2197                                     begin
2198                                          m_destin_agth[i] = destin_agent_config :: type_id
                                              :: create($sformatf("m_destin_agth[%0d]",i));
2199                                          m_destin_agth[i].is_active = UVM_ACTIVE;
2200                                          if(!uvm_config_db #(virtual router_destin_if) ::
                                              get(this,"",$sformatf("vif%0d",i),m_destin_agth[i
                                              ].vif))
2201                                               `uvm_fatal(get_full_name(),"cannot get the
                                                  interface handle from destination are set
                                                  in top")
2202
2203                                     end
2204                              m_cfg.no_of_source = no_of_source;
2205                              m_cfg.no_of_destin = no_of_destin;
2206
2207
2208                 endfunction
2209
2210   // fucniton build_phase
2211
2212
2213           function void build_phase(uvm_phase phase);
2214
2215                 super.build_phase(phase);
2216                 m_cfg = router_env_config :: type_id :: create("m_cfg");
2217
2218                 m_cfg.m_source_agth = new[no_of_source];
2219                 m_cfg.m_destin_agth = new[no_of_destin];
2220
2221                 config_data();
2222
2223                 foreach(m_source_agth[i])
2224                           m_cfg.m_source_agth[i] = m_source_agth[i];
2225
2226                 foreach(m_destin_agth[i])
2227                           m_cfg.m_destin_agth[i] = m_destin_agth[i];
2228
2229                  uvm_config_db #(router_env_config) :: set(this,"*","router_env_config",
                     m_cfg);
2230                  envh = tb_env :: type_id :: create("envh",this);
2231
```

```
2232
2233          endfunction:build_phase
2234
2235   // function void end o elaboration
2236
2237          function void end_of_elaboration_phase(uvm_phase phase);
2238
2239
2240                uvm_top.print_topology();
2241
2242
2243
2244         endfunction
2245
2246
2247
2248   endclass
2249
2250   //-------------Extended base test class to small test class ---------------//
2251
2252   class small_seq_test extends base_test;
2253
2254   // factory regisdtration
2255
2256            `uvm_component_utils(small_seq_test)
2257
2258   // declaring the virtual smal_seq class handle
2259
2260                //small_seq seqh;
2261         virtual_small_seq s_seqh;
2262         destin_seq  dseq;
2263   // function new constructor
2264
2265         function new(string name = "small_seq_test",uvm_component parent);
2266
2267                super.new(name,parent);
2268                addr = 2'd0;
2269
2270                uvm_config_db #(bit[2])::set(this,"*","bit",addr);
2271
2272         endfunction
2273
2274   // build phase
2275
2276           function void build_phase(uvm_phase phase);
2277
2278                  super.build_phase(phase);
2279
2280           endfunction
2281
2282   // run phase of small test seq
2283
2284   // task run phase
2285
2286             task run_phase(uvm_phase phase);
2287                    phase.raise_objection(this);
2288                           super.run_phase(phase);
2289                          s_seqh = virtual_small_seq :: type_id :: create("s_seqh");
2290                          dseq  = destin_seq        :: type_id :: create("dseq");
2291                          repeat(5)
2292                          begin
2293                          fork
2294                          s_seqh.start(envh.v_seqr);
2295                          dseq.start(envh.destin_agth.agth[addr].seqr);
2296                          join
2297                          end
2298            phase.drop_objection(this);
2299
2300             endtask
```

```
2301
2302  endclass
2303
2304  //----------Extended base test class to medium test class -------------//
2305
2306  class medium_seq_test extends base_test;
2307
2308  // factory registration
2309
2310        `uvm_component_utils(medium_seq_test)
2311
2312
2313  // declaring the handle for medum sdeq generater
2314
2315           // medium_seq seqh;
2316
2317        virtual_medium_seq m_seqh;
2318        destin_seq  dseq;
2319
2320
2321  // build phase
2322
2323           function void build_phase(uvm_phase phase);
2324
2325                 super.build_phase(phase);
2326
2327           endfunction
2328
2329  // function new constructor
2330
2331        function new(string name = "medium_seq_test", uvm_component parent );
2332
2333                   super.new(name,parent);
2334              addr = 2'd1;
2335
2336              uvm_config_db #(bit[2])::set(this,"*","bit",addr);
2337
2338
2339        endfunction
2340
2341  // run phase of medium test seq
2342
2343
2344           task run_phase(uvm_phase phase);
2345                  phase.raise_objection(this);
2346                         super.run_phase(phase);
2347                         m_seqh = virtual_medium_seq :: type_id :: create("m_seqh");
2348                         dseq  = destin_seq        :: type_id :: create("dseq");
2349                         repeat(5)
2350                         begin
2351                        fork
2352                        m_seqh.start(envh.v_seqr);
2353                        dseq.start(envh.destin_agth.agth[addr].seqr);
2354                        join
2355                        end
2356                  phase.drop_objection(this);
2357           endtask
2358
2359  endclass
2360
2361
2362  // --------------- Extended base test class to Large test class ---------//
2363
2364  class large_seq_test extends base_test;
2365
2366  // factory registration
2367
2368        `uvm_component_utils(large_seq_test)
2369
```

```
// declare the handle of large seq

         //large_seq seqh;

          virtual_large_seq l_seqh;
          destin_seq  dseq;


// build phase

         function void build_phase(uvm_phase phase);

                super.build_phase(phase);

         endfunction


// function new constructor

         function new(string name = "large_seq_test" , uvm_component parent);

                 super.new(name,parent);
              addr = 2'd2;

              uvm_config_db #(bit[2])::set(this,"*","bit",addr);


         endfunction

// run_phase of large test seq


         task run_phase(uvm_phase phase);
                     phase.raise_objection(this);
                          super.run_phase(phase);
                          l_seqh = virtual_large_seq :: type_id :: create("l_seqh");
                          dseq  = destin_seq        :: type_id :: create("dseq");
                         repeat(5)
                          begin
                        fork
                        l_seqh.start(envh.v_seqr);
                        dseq.start(envh.destin_agth.agth[addr].seqr);
                        join
                         end

                    phase.drop_objection(this);
         endtask



endclass


// packaGES files


package router_pkg;

// including the all ruoter file

import uvm_pkg ::*;

`include "uvm_macros.svh"

`include "destin_agent_config.sv"
`include "source_agent_config.sv"
`include "router_env_config.sv"
```

```
2439  `include "destin_xtn.sv"
2440
2441  `include "source_xtn.sv"
2442
2443  `include "source_seq.sv"
2444  `include "source_seqr.sv"
2445  `include "source_driver.sv"
2446  `include "source_monitor.sv"
2447  `include "source_agent.sv"
2448  `include "source_agent_top.sv"
2449
2450  `include "destin_seq.sv"
2451  `include "destin_driver.sv"
2452  `include "destin_monitor.sv"
2453  `include "destin_seqr.sv"
2454  `include "destin_agent.sv"
2455  `include "destin_agent_top.sv"
2456
2457
2458  `include "virtual_seqr.sv"
2459  `include "virtual_seq.sv"
2460  `include "tb_scoreboard.sv"
2461  `include "tb_env.sv"
2462  `include "base_test.sv"
2463
2464  //`include "top.sv"
2465
2466  endpackage
2467
2468
2469  // MAKE files
2470
2471  /*
2472
2473  #Makefile for UVM Testbench - Lab 10
2474
2475  # SIMULATOR = Questa for Mentor's Questasim
2476  # SIMULATOR = VCS for Synopsys's VCS
2477
2478  SIMULATOR = VCS
2479
2480
2481  FSDB_PATH=/home/cad/eda/SYNOPSYS/VERDI_2022/verdi/T-2022.06-SP1/share/PLI/VCS/LINUX64
2482
2483
2484  RTL= ../rtl/*
2485  work= work #library name
2486  SVTB1= ../tb/top.sv
2487  INC = +incdir+../tb +incdir+../test +incdir+../source_agent_top
      +incdir+../destin_agent_top
2488  SVTB2 = ../test/router_pkg.sv
2489  VSIMOPT= -vopt -voptargs=+acc
2490  VSIMCOV= -coverage -sva
2491  VSIMBATCH1= -c -do  " log -r /* ;coverage save -onexit mem_cov1;run -all; exit"
2492  VSIMBATCH2= -c -do  " log -r /* ;coverage save -onexit mem_cov2;run -all; exit"
2493  VSIMBATCH3= -c -do  " log -r /* ;coverage save -onexit mem_cov3;run -all; exit"
2494  VSIMBATCH4= -c -do  " log -r /* ;coverage save -onexit mem_cov4;run -all; exit"
2495
2496
2497  help:
2498      @echo
      ==========================================================================================
      ========================
2499      @echo "! USAGE      --  make target
      !"
2500      @echo "! clean      =>  clean the earlier log and intermediate
      files.                    !"
2501      @echo "! sv_cmp     =>  Create library and compile the
      code.                              !"
```

```
2502        @echo "! run_test   =>  clean, compile & run the simulation for small packets in
            batch mode.        !"
2503        @echo "! run_test1  =>  clean, compile & run the simulation for medium packets in
            batch mode.          !"
2504        @echo "! run_test2  =>  clean, compile & run the simulation for large packets in
            batch mode.          !"
2505        @echo "! run_test3  =>  clean, compile & run the simulation for ram_even_addr_test
            in batch mode.          !"
2506        @echo "! view_wave1 =>  To view the waveform of small
            packets                              !"
2507        @echo "! view_wave2 =>  To view the waveform of medium
            packets                             !"
2508        @echo "! view_wave3 =>  To view the waveform of large
            packets                          !"
2509        @echo "! view_wave4 =>  To view the waveform of
            ram_even_addr_test                                !"
2510        @echo "! regress    =>  clean, compile and run all testcases in batch
            mode.                         !"
2511        @echo "! report     =>  To merge coverage reports for all testcases and  convert to
            html format.            !"
2512        @echo "! cov        =>  To open merged coverage report in html
            format.                          !"
2513        @echo
            ================================================================================
            ================================

2514
2515    clean : clean_$(SIMULATOR)
2516    sv_cmp : sv_cmp_$(SIMULATOR)
2517    run_test : run_test_$(SIMULATOR)
2518    run_test1 : run_test1_$(SIMULATOR)
2519    run_test2 : run_test2_$(SIMULATOR)
2520    run_test3 : run_test3_$(SIMULATOR)
2521    view_wave1 : view_wave1_$(SIMULATOR)
2522    view_wave2 : view_wave2_$(SIMULATOR)
2523    view_wave3 : view_wave3_$(SIMULATOR)
2524    view_wave4 : view_wave4_$(SIMULATOR)
2525    regress : regress_$(SIMULATOR)
2526    report : report_$(SIMULATOR)
2527    cov : cov_$(SIMULATOR)
2528
2529    # ----------------------------- Start of Definitions for Mentor's Questa Specific
        Targets ------------------------------#
2530
2531    sv_cmp_Questa:
2532        vlib $(work)
2533        vmap work $(work)
2534        vlog -work $(work) $(RTL) $(INC) $(SVTB2) $(SVTB1)
2535
2536    run_test_Questa: sv_cmp
2537        vsim -cvgperinstance $(VSIMOPT) $(VSIMCOV) $(VSIMBATCH1)  -wlf wave_file1.wlf -l
            test1.log  -sv_seed random  work.top +UVM_TESTNAME=small_seq_test
2538        vcover report  -cvg  -details -nocompactcrossbins -codeAll -assert -directive -html
            mem_cov1
2539
2540    run_test1_Questa:
2541        vsim -cvgperinstance $(VSIMOPT) $(VSIMCOV) $(VSIMBATCH2)  -wlf wave_file2.wlf -l
            test2.log  -sv_seed random  work.top +UVM_TESTNAME=medium_seq_test
2542        vcover report  -cvg  -details -nocompactcrossbins -codeAll -assert -directive -html
            mem_cov2
2543
2544    run_test2_Questa:
2545        vsim -cvgperinstance $(VSIMOPT) $(VSIMCOV) $(VSIMBATCH3)  -wlf wave_file3.wlf -l
            test3.log  -sv_seed random  work.top +UVM_TESTNAME=large_seq_test
2546        vcover report  -cvg  -details -nocompactcrossbins -codeAll -assert -directive -html
            mem_cov3
2547
2548    run_test3_Questa:
2549        vsim -cvgperinstance $(VSIMOPT) $(VSIMCOV) $(VSIMBATCH4)  -wlf wave_file4.wlf -l
            test4.log  -sv_seed random  work.top +UVM_TESTNAME=ram_even_addr_test
```

```
2550        vcover report  -cvg  -details -nocompactcrossbins -codeAll -assert -directive -html
            mem_cov4
2551
2552    view_wave1_Questa:
2553        vsim -view wave_file1.wlf
2554
2555    view_wave2_Questa:
2556        vsim -view wave_file2.wlf
2557
2558    view_wave3_Questa:
2559        vsim -view wave_file3.wlf
2560
2561    view_wave4_Questa:
2562        vsim -view wave_file4.wlf
2563
2564    report_Questa:
2565        vcover merge mem_cov mem_cov1 mem_cov2 mem_cov3 mem_cov4
2566        vcover report -cvg -details -nocompactcrossbins -codeAll -assert -directive -html
            mem_cov
2567
2568    regress_Questa: clean_Questa run_test_Questa run_test1_Questa run_test2_Questa
        run_test3_Questa report_Questa cov_Questa
2569
2570    cov_Questa:
2571        firefox covhtmlreport/index.html&
2572
2573    clean_Questa:
2574        rm -rf transcript* *log* fcover* covhtml* mem_cov* *.wlf modelsim.ini work
2575        clear
2576
2577    # ------------------------------ End of Definitions for Mentor's Questa Specific Targets
        -------------------------------#
2578
2579    # ------------------------------ Start of Definitions for Synopsys's VCS Specific Targets
        -------------------------------#
2580
2581    sv_cmp_VCS:
2582        vcs -l vcs.log -timescale=1ns/1ps -sverilog -ntb_opts uvm -debug_access+all -full64
            -kdb  -lca -P $(FSDB_PATH)/novas.tab $(FSDB_PATH)/pli.a $(RTL) $(INC) $(SVTB2)
            $(SVTB1)
2583
2584    run_test_VCS:
2585        ./simv -a vcs.log +fsdbfile+wave1.fsdb -cm_dir ./mem_cov1 +ntb_random_seed_automatic
            +UVM_TESTNAME=small_seq_test
2586        urg -dir mem_cov1.vdb -format both -report urgReport1
2587
2588    run_test1_VCS:
2589        ./simv -a vcs.log +fsdbfile+wave2.fsdb -cm_dir ./mem_cov2 +ntb_random_seed_automatic
            +UVM_TESTNAME=medium_seq_test
2590        urg -dir mem_cov2.vdb -format both -report urgReport2
2591
2592    run_test2_VCS:
2593        ./simv -a vcs.log +fsdbfile+wave3.fsdb -cm_dir ./mem_cov3 +ntb_random_seed_automatic
            +UVM_TESTNAME=large_seq_test
2594        urg -dir mem_cov3.vdb -format both -report urgReport3
2595
2596    run_test3_VCS:
2597        ./simv -a vcs.log +fsdbfile+wave4.fsdb -cm_dir ./mem_cov4 +ntb_random_seed_automatic
            +UVM_TESTNAME=ram_even_addr_test
2598        urg -dir mem_cov4.vdb -format both -report urgReport4
2599
2600    view_wave1_VCS:
2601        verdi -ssf wave1.fsdb
2602
2603    view_wave2_VCS:
2604        verdi -ssf wave2.fsdb
2605
2606    view_wave3_VCS:
2607        verdi -ssf wave3.fsdb
```

```
2608
2609   view_wave4_VCS:
2610       verdi -ssf wave4.fsdb
2611
2612   report_VCS:
2613       urg -dir mem_cov1.vdb mem_cov2.vdb mem_cov3.vdb mem_cov4.vdb -dbname
       merged_dir/merged_test -format both -report urgReport
2614
2615   regress_VCS: clean_VCS sv_cmp_VCS run_test_VCS run_test1_VCS run_test2_VCS run_test3_VCS
       report_VCS
2616
2617   cov_VCS:
2618       verdi -cov -covdir merged_dir.vdb
2619
2620   clean_VCS:
2621       rm -rf simv* csrc* *.tmp *.vpd *.vdb *.key *.log *hdrs.h urgReport* *.fsdb novas*
       verdi*
2622       clear
2623
2624   # ---------------------------- END of Definitions for Synopsys's VCS Specific Targets
       ------------------------------#
2625
2626   */
2627
2628
2629   // OUTPUT ::: FOR 3 TEST CASES Small,Medium,Large
2630
2631   // Small PACKETS
2632
2633
2634   UVM_INFO @ 0: reporter [RNTST] Running test small_seq_test...
2635   UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
2636   ----------------------------------------------------------------------
2637   Name                          Type                          Size  Value
2638   ----------------------------------------------------------------------
2639   uvm_test_top                  small_seq_test                -     @474
2640     envh                        tb_env                        -     @499
2641       destin_agth               destin_agent_top              -     @516
2642         agth[0]                 destin_agent                  -     @660
2643           drvh                  destin_driver                 -     @708
2644             rsp_port            uvm_analysis_port             -     @725
2645             seq_item_port       uvm_seq_item_pull_port        -     @716
2646           monh                  destin_monitor                -     @691
2647             dmon_port           uvm_analysis_port             -     @699
2648           seqr                  destin_seqr                   -     @734
2649             rsp_export          uvm_analysis_export           -     @742
2650             seq_item_export     uvm_seq_item_pull_imp         -     @848
2651             arbitration_queue   array                         0     -
2652             lock_queue          array                         0     -
2653             num_last_reqs       integral                      32    'd1
2654             num_last_rsps       integral                      32    'd1
2655         agth[1]                 destin_agent                  -     @669
2656           drvh                  destin_driver                 -     @886
2657             rsp_port            uvm_analysis_port             -     @903
2658             seq_item_port       uvm_seq_item_pull_port        -     @894
2659           monh                  destin_monitor                -     @869
2660             dmon_port           uvm_analysis_port             -     @877
2661           seqr                  destin_seqr                   -     @912
2662             rsp_export          uvm_analysis_export           -     @920
2663             seq_item_export     uvm_seq_item_pull_imp         -     @1026
2664             arbitration_queue   array                         0     -
2665             lock_queue          array                         0     -
2666             num_last_reqs       integral                      32    'd1
2667             num_last_rsps       integral                      32    'd1
2668         agth[2]                 destin_agent                  -     @678
2669           drvh                  destin_driver                 -     @1064
2670             rsp_port            uvm_analysis_port             -     @1081
2671             seq_item_port       uvm_seq_item_pull_port        -     @1072
2672           monh                  destin_monitor                -     @1047
```

```
2673            dmon_port           uvm_analysis_port            -     @1055
2674          seqr                  destin_seqr                  -     @1090
2675            rsp_export          uvm_analysis_export          -     @1098
2676            seq_item_export     uvm_seq_item_pull_imp        -     @1204
2677            arbitration_queue   array                        0     -
2678            lock_queue          array                        0     -
2679            num_last_reqs       integral                     32    'd1
2680            num_last_rsps       integral                     32    'd1
2681      sb                        tb_scoreboard                -     @524
2682        destin_fifo[0]          uvm_tlm_analysis_fifo #(T)   -     @1274
2683          analysis_export       uvm_analysis_imp             -     @1318
2684          get_ap                uvm_analysis_port            -     @1309
2685          get_peek_export       uvm_get_peek_imp             -     @1291
2686          put_ap                uvm_analysis_port            -     @1300
2687          put_export            uvm_put_imp                  -     @1282
2688        destin_fifo[1]          uvm_tlm_analysis_fifo #(T)   -     @1327
2689          analysis_export       uvm_analysis_imp             -     @1371
2690          get_ap                uvm_analysis_port            -     @1362
2691          get_peek_export       uvm_get_peek_imp             -     @1344
2692          put_ap                uvm_analysis_port            -     @1353
2693          put_export            uvm_put_imp                  -     @1335
2694        destin_fifo[2]          uvm_tlm_analysis_fifo #(T)   -     @1380
2695          analysis_export       uvm_analysis_imp             -     @1424
2696          get_ap                uvm_analysis_port            -     @1415
2697          get_peek_export       uvm_get_peek_imp             -     @1397
2698          put_ap                uvm_analysis_port            -     @1406
2699          put_export            uvm_put_imp                  -     @1388
2700        source_fifo             uvm_tlm_analysis_fifo #(T)   -     @1221
2701          analysis_export       uvm_analysis_imp             -     @1265
2702          get_ap                uvm_analysis_port            -     @1256
2703          get_peek_export       uvm_get_peek_imp             -     @1238
2704          put_ap                uvm_analysis_port            -     @1247
2705          put_export            uvm_put_imp                  -     @1229
2706      source_agth               source_agent_top             -     @508
2707        agth[0]                 source_agent                 -     @1438
2708          drvh                  source_driver                -     @1468
2709            rsp_port            uvm_analysis_port            -     @1485
2710            seq_item_port       uvm_seq_item_pull_port       -     @1476
2711          monh                  source_monitor               -     @1451
2712            smon_port           uvm_analysis_port            -     @1459
2713          seqr                  source_seqr                  -     @1494
2714            rsp_export          uvm_analysis_export          -     @1502
2715            seq_item_export     uvm_seq_item_pull_imp        -     @1608
2716            arbitration_queue   array                        0     -
2717            lock_queue          array                        0     -
2718            num_last_reqs       integral                     32    'd1
2719            num_last_rsps       integral                     32    'd1
2720      v_seqr                    virtual_seqr                 -     @532
2721        rsp_export              uvm_analysis_export          -     @540
2722        seq_item_export         uvm_seq_item_pull_imp        -     @646
2723        arbitration_queue       array                        0     -
2724        lock_queue              array                        0     -
2725        num_last_reqs           integral                     32    'd1
2726        num_last_rsps           integral                     32    'd1
2727    ----------------------------------------------------------------------
2728
2729    Signal Error From Source Monitor = 1
2730    ----------------------------------------
2731    Name                Type         Size  Value
2732    ----------------------------------------
2733    source_mon          source_xtn   -     @1648
2734      destin_address    integral     2     'd0
2735      pay_lenth         integral     6     'd20
2736      header_byte       integral     8     'd80
2737      payload[0]        integral     8     157
2738      payload[1]        integral     8     'd15
2739      payload[2]        integral     8     'd66
2740      payload[3]        integral     8     'd39
2741      payload[4]        integral     8     'd13
```

```
  payload[5]      integral    8     207
  payload[6]      integral    8     173
  payload[7]      integral    8     158
  payload[8]      integral    8     'd91
  payload[9]      integral    8     'd0
  payload[10]     integral    8     220
  payload[11]     integral    8     230
  payload[12]     integral    8     'd113
  payload[13]     integral    8     'd40
  payload[14]     integral    8     130
  payload[15]     integral    8     'd93
  payload[16]     integral    8     'd55
  payload[17]     integral    8     'd18
  payload[18]     integral    8     'd78
  payload[19]     integral    8     188
  parity_byte     integral    8     'd102
-----------------------------------------
-----------------------------------
Name           Type        Size  Value
-----------------------------------
destin_mon     destin_xtn  -     @1630
  des_address  integral    2     'd0
  pay_lenth    integral    6     'd20
  header_byte  integral    8     'd80
  payload[0]   integral    8     157
  payload[1]   integral    8     'd15
  payload[2]   integral    8     'd66
  payload[3]   integral    8     'd39
  payload[4]   integral    8     'd13
  payload[5]   integral    8     207
  payload[6]   integral    8     173
  payload[7]   integral    8     158
  payload[8]   integral    8     'd91
  payload[9]   integral    8     'd0
  payload[10]  integral    8     220
  payload[11]  integral    8     230
  payload[12]  integral    8     'd113
  payload[13]  integral    8     'd40
  payload[14]  integral    8     130
  payload[15]  integral    8     'd93
  payload[16]  integral    8     'd55
  payload[17]  integral    8     'd18
  payload[18]  integral    8     'd78
  payload[19]  integral    8     188
  parity_byte  integral    8     'd102
  delay        integral    6     'd0
-----------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


     SUCCESSFULLY MATCHED


================================================================================


================================================================================
```

```
 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name               Type        Size  Value
----------------------------------------
source_mon         source_xtn  -     @1803
  destin_address   integral    2     'd0
  pay_lenth        integral    6     'd8
  header_byte      integral    8     'd32
  payload[0]       integral    8     205
  payload[1]       integral    8     'd57
  payload[2]       integral    8     251
  payload[3]       integral    8     251
  payload[4]       integral    8     232
  payload[5]       integral    8     147
  payload[6]       integral    8     'd78
  payload[7]       integral    8     188
  parity_byte      integral    8     'd93
----------------------------------------
-------------------------------------
Name            Type        Size  Value
-------------------------------------
destin_mon      destin_xtn  -     @1807
  des_address   integral    2     'd0
  pay_lenth     integral    6     'd8
  header_byte   integral    8     'd32
  payload[0]    integral    8     205
  payload[1]    integral    8     'd57
  payload[2]    integral    8     251
  payload[3]    integral    8     251
  payload[4]    integral    8     232
  payload[5]    integral    8     147
  payload[6]    integral    8     'd78
  payload[7]    integral    8     188
  parity_byte   integral    8     'd93
  delay         integral    6     'd0
-------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


      SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


```

```
 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name               Type         Size  Value
----------------------------------------
source_mon         source_xtn   -     @1826
  destin_address   integral     2     'd0
  pay_lenth        integral     6     'd2
  header_byte      integral     8     'd8
  payload[0]       integral     8     129
  payload[1]       integral     8     186
  parity_byte      integral     8     'd51
----------------------------------------
-------------------------------------
Name            Type         Size  Value
-------------------------------------
destin_mon      destin_xtn   -     @1830
  des_address   integral     2     'd0
  pay_lenth     integral     6     'd2
  header_byte   integral     8     'd8
  payload[0]    integral     8     129
  payload[1]    integral     8     186
  parity_byte   integral     8     'd51
  delay         integral     6     'd0
-------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


     SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name               Type         Size  Value
----------------------------------------
source_mon         source_xtn   -     @1848
  destin_address   integral     2     'd0
  pay_lenth        integral     6     'd18
  header_byte      integral     8     'd72
  payload[0]       integral     8     203
  payload[1]       integral     8     191
```

```
  payload[2]      integral    8     'd123
  payload[3]      integral    8     182
  payload[4]      integral    8     174
  payload[5]      integral    8     158
  payload[6]      integral    8     'd92
  payload[7]      integral    8     138
  payload[8]      integral    8     'd116
  payload[9]      integral    8     'd90
  payload[10]     integral    8     'd95
  payload[11]     integral    8     159
  payload[12]     integral    8     190
  payload[13]     integral    8     189
  payload[14]     integral    8     'd66
  payload[15]     integral    8     'd101
  payload[16]     integral    8     'd6
  payload[17]     integral    8     136
  parity_byte     integral    8     'd83
----------------------------------------
----------------------------------------
Name            Type        Size  Value
----------------------------------------
destin_mon      destin_xtn  -     @1852
  des_address   integral    2     'd0
  pay_lenth     integral    6     'd18
  header_byte   integral    8     'd72
  payload[0]    integral    8     203
  payload[1]    integral    8     191
  payload[2]    integral    8     'd123
  payload[3]    integral    8     182
  payload[4]    integral    8     174
  payload[5]    integral    8     158
  payload[6]    integral    8     'd92
  payload[7]    integral    8     138
  payload[8]    integral    8     'd116
  payload[9]    integral    8     'd90
  payload[10]   integral    8     'd95
  payload[11]   integral    8     159
  payload[12]   integral    8     190
  payload[13]   integral    8     189
  payload[14]   integral    8     'd66
  payload[15]   integral    8     'd101
  payload[16]   integral    8     'd6
  payload[17]   integral    8     136
  parity_byte   integral    8     'd83
  delay         integral    6     'd0
----------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


     SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889
```

```
 destin  side functional coverage = 41.667

================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name               Type        Size  Value
----------------------------------------
source_mon         source_xtn  -     @1870
  destin_address   integral    2     'd0
  pay_lenth        integral    6     'd11
  header_byte      integral    8     'd44
  payload[0]       integral    8     128
  payload[1]       integral    8     'd61
  payload[2]       integral    8     'd12
  payload[3]       integral    8     'd90
  payload[4]       integral    8     193
  payload[5]       integral    8     'd56
  payload[6]       integral    8     131
  payload[7]       integral    8     255
  payload[8]       integral    8     229
  payload[9]       integral    8     139
  payload[10]      integral    8     'd14
  parity_byte      integral    8     'd34
----------------------------------------
-------------------------------------
Name            Type        Size  Value
-------------------------------------
destin_mon      destin_xtn  -     @1874
  des_address   integral    2     'd0
  pay_lenth     integral    6     'd11
  header_byte   integral    8     'd44
  payload[0]    integral    8     128
  payload[1]    integral    8     'd61
  payload[2]    integral    8     'd12
  payload[3]    integral    8     'd90
  payload[4]    integral    8     193
  payload[5]    integral    8     'd56
  payload[6]    integral    8     131
  payload[7]    integral    8     255
  payload[8]    integral    8     229
  payload[9]    integral    8     139
  payload[10]   integral    8     'd14
  parity_byte   integral    8     'd34
  delay         integral    6     'd0
-------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


     SUCCESSFULLY MATCHED


================================================================================


================================================================================
```

```
 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================
UVM_INFO /home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_objection.svh(1274
) @ 3650000: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :    3
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[UVMTOP]     1
$finish called from file
"/home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_root.svh", line 437.
$finish at simulation time              3650000
           V C S   S i m u l a t i o n   R e p o r t
Time: 3650000 ps


// Medium PACKETS

UVM_INFO @ 0: reporter [RNTST] Running test medium_seq_test...
UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
----------------------------------------------------------------------
Name                          Type                            Size  Value
----------------------------------------------------------------------
uvm_test_top                  medium_seq_test                 -     @474
  envh                        tb_env                          -     @499
    destin_agth               destin_agent_top                -     @516
      agth[0]                 destin_agent                    -     @660
        drvh                  destin_driver                   -     @708
          rsp_port            uvm_analysis_port               -     @725
          seq_item_port       uvm_seq_item_pull_port          -     @716
        monh                  destin_monitor                  -     @691
          dmon_port           uvm_analysis_port               -     @699
        seqr                  destin_seqr                     -     @734
          rsp_export          uvm_analysis_export             -     @742
          seq_item_export     uvm_seq_item_pull_imp           -     @848
          arbitration_queue   array                           0     -
          lock_queue          array                           0     -
          num_last_reqs       integral                        32    'd1
          num_last_rsps       integral                        32    'd1
      agth[1]                 destin_agent                    -     @669
        drvh                  destin_driver                   -     @886
          rsp_port            uvm_analysis_port               -     @903
          seq_item_port       uvm_seq_item_pull_port          -     @894
        monh                  destin_monitor                  -     @869
          dmon_port           uvm_analysis_port               -     @877
        seqr                  destin_seqr                     -     @912
          rsp_export          uvm_analysis_export             -     @920
          seq_item_export     uvm_seq_item_pull_imp           -     @1026
          arbitration_queue   array                           0     -
          lock_queue          array                           0     -
          num_last_reqs       integral                        32    'd1
          num_last_rsps       integral                        32    'd1
      agth[2]                 destin_agent                    -     @678
        drvh                  destin_driver                   -     @1064
```

```
3154              rsp_port           uvm_analysis_port           -      @1081
3155              seq_item_port      uvm_seq_item_pull_port      -      @1072
3156            monh                 destin_monitor              -      @1047
3157              dmon_port          uvm_analysis_port           -      @1055
3158            seqr                 destin_seqr                 -      @1090
3159              rsp_export         uvm_analysis_export         -      @1098
3160              seq_item_export    uvm_seq_item_pull_imp       -      @1204
3161              arbitration_queue  array                       0      -
3162              lock_queue         array                       0      -
3163              num_last_reqs      integral                    32     'd1
3164              num_last_rsps      integral                    32     'd1
3165        sb                       tb_scoreboard               -      @524
3166          destin_fifo[0]         uvm_tlm_analysis_fifo #(T)  -      @1274
3167            analysis_export      uvm_analysis_imp            -      @1318
3168            get_ap               uvm_analysis_port           -      @1309
3169            get_peek_export      uvm_get_peek_imp            -      @1291
3170            put_ap               uvm_analysis_port           -      @1300
3171            put_export           uvm_put_imp                 -      @1282
3172          destin_fifo[1]         uvm_tlm_analysis_fifo #(T)  -      @1327
3173            analysis_export      uvm_analysis_imp            -      @1371
3174            get_ap               uvm_analysis_port           -      @1362
3175            get_peek_export      uvm_get_peek_imp            -      @1344
3176            put_ap               uvm_analysis_port           -      @1353
3177            put_export           uvm_put_imp                 -      @1335
3178          destin_fifo[2]         uvm_tlm_analysis_fifo #(T)  -      @1380
3179            analysis_export      uvm_analysis_imp            -      @1424
3180            get_ap               uvm_analysis_port           -      @1415
3181            get_peek_export      uvm_get_peek_imp            -      @1397
3182            put_ap               uvm_analysis_port           -      @1406
3183            put_export           uvm_put_imp                 -      @1388
3184          source_fifo            uvm_tlm_analysis_fifo #(T)  -      @1221
3185            analysis_export      uvm_analysis_imp            -      @1265
3186            get_ap               uvm_analysis_port           -      @1256
3187            get_peek_export      uvm_get_peek_imp            -      @1238
3188            put_ap               uvm_analysis_port           -      @1247
3189            put_export           uvm_put_imp                 -      @1229
3190        source_agth              source_agent_top            -      @508
3191          agth[0]                source_agent                -      @1438
3192            drvh                 source_driver               -      @1468
3193              rsp_port           uvm_analysis_port           -      @1485
3194              seq_item_port      uvm_seq_item_pull_port      -      @1476
3195            monh                 source_monitor              -      @1451
3196              smon_port          uvm_analysis_port           -      @1459
3197            seqr                 source_seqr                 -      @1494
3198              rsp_export         uvm_analysis_export         -      @1502
3199              seq_item_export    uvm_seq_item_pull_imp       -      @1608
3200              arbitration_queue  array                       0      -
3201              lock_queue         array                       0      -
3202              num_last_reqs      integral                    32     'd1
3203              num_last_rsps      integral                    32     'd1
3204        v_seqr                   virtual_seqr                -      @532
3205          rsp_export             uvm_analysis_export         -      @540
3206          seq_item_export        uvm_seq_item_pull_imp       -      @646
3207          arbitration_queue      array                       0      -
3208          lock_queue             array                       0      -
3209          num_last_reqs          integral                    32     'd1
3210          num_last_rsps          integral                    32     'd1
3211    ----------------------------------------------------------------------
3212
3213    Signal Error From Source Monitor = 1
3214    ----------------------------------------
3215    Name               Type        Size  Value
3216    ----------------------------------------
3217    source_mon         source_xtn  -     @1648
3218      destin_address   integral    2     'd1
3219      pay_lenth        integral    6     'd27
3220      header_byte      integral    8     'd109
3221      payload[0]       integral    8     195
3222      payload[1]       integral    8     218
```

```
  payload[2]      integral    8     227
  payload[3]      integral    8     'd122
  payload[4]      integral    8     'd64
  payload[5]      integral    8     191
  payload[6]      integral    8     'd24
  payload[7]      integral    8     138
  payload[8]      integral    8     'd22
  payload[9]      integral    8     235
  payload[10]     integral    8     139
  payload[11]     integral    8     'd42
  payload[12]     integral    8     197
  payload[13]     integral    8     'd44
  payload[14]     integral    8     249
  payload[15]     integral    8     141
  payload[16]     integral    8     198
  payload[17]     integral    8     220
  payload[18]     integral    8     'd93
  payload[19]     integral    8     224
  payload[20]     integral    8     235
  payload[21]     integral    8     'd45
  payload[22]     integral    8     'd42
  payload[23]     integral    8     214
  payload[24]     integral    8     149
  payload[25]     integral    8     196
  payload[26]     integral    8     'd31
  parity_byte     integral    8     146
----------------------------------------
----------------------------------------
Name            Type         Size  Value
----------------------------------------
destin_mon      destin_xtn   -     @1636
  des_address   integral     2     'd1
  pay_lenth     integral     6     'd27
  header_byte   integral     8     'd109
  payload[0]    integral     8     195
  payload[1]    integral     8     218
  payload[2]    integral     8     227
  payload[3]    integral     8     'd122
  payload[4]    integral     8     'd64
  payload[5]    integral     8     191
  payload[6]    integral     8     'd24
  payload[7]    integral     8     138
  payload[8]    integral     8     'd22
  payload[9]    integral     8     235
  payload[10]   integral     8     139
  payload[11]   integral     8     'd42
  payload[12]   integral     8     197
  payload[13]   integral     8     'd44
  payload[14]   integral     8     249
  payload[15]   integral     8     141
  payload[16]   integral     8     198
  payload[17]   integral     8     220
  payload[18]   integral     8     'd93
  payload[19]   integral     8     224
  payload[20]   integral     8     235
  payload[21]   integral     8     'd45
  payload[22]   integral     8     'd42
  payload[23]   integral     8     214
  payload[24]   integral     8     149
  payload[25]   integral     8     196
  payload[26]   integral     8     'd31
  parity_byte   integral     8     146
  delay         integral     6     'd0
----------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================
```

```
3292
3293
3294  ====================== Parity_Byte Matched SuccessFull ======================
3295
3296
3297  ================================================================================
3298
3299
3300       SUCCESSFULLY MATCHED
3301
3302
3303  ================================================================================
3304
3305
3306  ================================================================================
3307
3308
3309   source  side functional coverage = 38.889
3310
3311
3312   destin  side functional coverage = 41.667
3313
3314
3315  ================================================================================
3316
3317  Signal Error From Source Monitor = 1
3318  ----------------------------------------
3319  Name               Type         Size  Value
3320  ----------------------------------------
3321  source_mon         source_xtn   -     @1803
3322    destin_address   integral     2     'd1
3323    pay_lenth        integral     6     'd29
3324    header_byte      integral     8     'd117
3325    payload[0]       integral     8     'd107
3326    payload[1]       integral     8     'd22
3327    payload[2]       integral     8     250
3328    payload[3]       integral     8     133
3329    payload[4]       integral     8     'd106
3330    payload[5]       integral     8     'd88
3331    payload[6]       integral     8     'd61
3332    payload[7]       integral     8     'd16
3333    payload[8]       integral     8     194
3334    payload[9]       integral     8     'd56
3335    payload[10]      integral     8     237
3336    payload[11]      integral     8     'd37
3337    payload[12]      integral     8     156
3338    payload[13]      integral     8     'd34
3339    payload[14]      integral     8     194
3340    payload[15]      integral     8     210
3341    payload[16]      integral     8     'd115
3342    payload[17]      integral     8     152
3343    payload[18]      integral     8     242
3344    payload[19]      integral     8     'd105
3345    payload[20]      integral     8     237
3346    payload[21]      integral     8     246
3347    payload[22]      integral     8     175
3348    payload[23]      integral     8     'd14
3349    payload[24]      integral     8     249
3350    payload[25]      integral     8     'd44
3351    payload[26]      integral     8     'd104
3352    payload[27]      integral     8     186
3353    payload[28]      integral     8     162
3354    parity_byte      integral     8     155
3355  ----------------------------------------
3356  ----------------------------------------
3357  Name             Type         Size  Value
3358  ----------------------------------------
3359  destin_mon       destin_xtn   -     @1807
3360    des_address    integral     2     'd1
```

```
  pay_lenth    integral    6     'd29
  header_byte  integral    8     'd117
  payload[0]   integral    8     'd107
  payload[1]   integral    8     'd22
  payload[2]   integral    8     250
  payload[3]   integral    8     133
  payload[4]   integral    8     'd106
  payload[5]   integral    8     'd88
  payload[6]   integral    8     'd61
  payload[7]   integral    8     'd16
  payload[8]   integral    8     194
  payload[9]   integral    8     'd56
  payload[10]  integral    8     237
  payload[11]  integral    8     'd37
  payload[12]  integral    8     156
  payload[13]  integral    8     'd34
  payload[14]  integral    8     194
  payload[15]  integral    8     210
  payload[16]  integral    8     'd115
  payload[17]  integral    8     152
  payload[18]  integral    8     242
  payload[19]  integral    8     'd105
  payload[20]  integral    8     237
  payload[21]  integral    8     246
  payload[22]  integral    8     175
  payload[23]  integral    8     'd14
  payload[24]  integral    8     249
  payload[25]  integral    8     'd44
  payload[26]  integral    8     'd104
  payload[27]  integral    8     186
  payload[28]  integral    8     162
  parity_byte  integral    8     155
  delay        integral    6     'd0
-------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


     SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name              Type        Size  Value
----------------------------------------
source_mon        source_xtn  -     @1826
```

```
  destin_address  integral    2     'd1
  pay_lenth       integral    6     'd25
  header_byte     integral    8     'd101
  payload[0]      integral    8     'd76
  payload[1]      integral    8     238
  payload[2]      integral    8     216
  payload[3]      integral    8     'd123
  payload[4]      integral    8     'd86
  payload[5]      integral    8     'd7
  payload[6]      integral    8     243
  payload[7]      integral    8     220
  payload[8]      integral    8     130
  payload[9]      integral    8     156
  payload[10]     integral    8     142
  payload[11]     integral    8     132
  payload[12]     integral    8     150
  payload[13]     integral    8     'd13
  payload[14]     integral    8     142
  payload[15]     integral    8     195
  payload[16]     integral    8     'd5
  payload[17]     integral    8     239
  payload[18]     integral    8     'd119
  payload[19]     integral    8     159
  payload[20]     integral    8     'd69
  payload[21]     integral    8     162
  payload[22]     integral    8     235
  payload[23]     integral    8     'd74
  payload[24]     integral    8     202
  parity_byte     integral    8     'd86
-----------------------------------------
-----------------------------------
Name           Type        Size  Value
-----------------------------------
destin_mon     destin_xtn  -     @1830
  des_address  integral    2     'd1
  pay_lenth    integral    6     'd25
  header_byte  integral    8     'd101
  payload[0]   integral    8     'd76
  payload[1]   integral    8     238
  payload[2]   integral    8     216
  payload[3]   integral    8     'd123
  payload[4]   integral    8     'd86
  payload[5]   integral    8     'd7
  payload[6]   integral    8     243
  payload[7]   integral    8     220
  payload[8]   integral    8     130
  payload[9]   integral    8     156
  payload[10]  integral    8     142
  payload[11]  integral    8     132
  payload[12]  integral    8     150
  payload[13]  integral    8     'd13
  payload[14]  integral    8     142
  payload[15]  integral    8     195
  payload[16]  integral    8     'd5
  payload[17]  integral    8     239
  payload[18]  integral    8     'd119
  payload[19]  integral    8     159
  payload[20]  integral    8     'd69
  payload[21]  integral    8     162
  payload[22]  integral    8     235
  payload[23]  integral    8     'd74
  payload[24]  integral    8     202
  parity_byte  integral    8     'd86
  delay        integral    6     'd0
-----------------------------------

====================== Header_Byte Matched SuccessFull ======================
```

```
3499  ====================== Payload Matched SuccessFull ======================
3500
3501
3502  ====================== Parity_Byte Matched SuccessFull ======================
3503
3504
3505  ================================================================================
3506
3507
3508       SUCCESSFULLY MATCHED
3509
3510
3511  ================================================================================
3512
3513
3514  ================================================================================
3515
3516
3517   source  side functional coverage = 38.889
3518
3519
3520   destin  side functional coverage = 41.667
3521
3522
3523  ================================================================================
3524
3525  Signal Error From Source Monitor = 1
3526  ----------------------------------------
3527  Name               Type        Size  Value
3528  ----------------------------------------
3529  source_mon         source_xtn  -     @1848
3530    destin_address   integral    2     'd1
3531    pay_lenth        integral    6     'd25
3532    header_byte      integral    8     'd101
3533    payload[0]       integral    8     205
3534    payload[1]       integral    8     215
3535    payload[2]       integral    8     185
3536    payload[3]       integral    8     'd37
3537    payload[4]       integral    8     229
3538    payload[5]       integral    8     'd89
3539    payload[6]       integral    8     148
3540    payload[7]       integral    8     'd118
3541    payload[8]       integral    8     151
3542    payload[9]       integral    8     182
3543    payload[10]      integral    8     'd76
3544    payload[11]      integral    8     'd54
3545    payload[12]      integral    8     'd48
3546    payload[13]      integral    8     177
3547    payload[14]      integral    8     'd26
3548    payload[15]      integral    8     211
3549    payload[16]      integral    8     'd39
3550    payload[17]      integral    8     'd43
3551    payload[18]      integral    8     'd26
3552    payload[19]      integral    8     'd5
3553    payload[20]      integral    8     'd17
3554    payload[21]      integral    8     181
3555    payload[22]      integral    8     'd99
3556    payload[23]      integral    8     196
3557    payload[24]      integral    8     244
3558    parity_byte      integral    8     'd74
3559  ----------------------------------------
3560  ---------------------------------------
3561  Name            Type        Size  Value
3562  ---------------------------------------
3563  destin_mon      destin_xtn  -     @1852
3564    des_address   integral    2     'd1
3565    pay_lenth     integral    6     'd25
3566    header_byte   integral    8     'd101
3567    payload[0]    integral    8     205
```

```
  payload[1]   integral    8     215
  payload[2]   integral    8     185
  payload[3]   integral    8     'd37
  payload[4]   integral    8     229
  payload[5]   integral    8     'd89
  payload[6]   integral    8     148
  payload[7]   integral    8     'd118
  payload[8]   integral    8     151
  payload[9]   integral    8     182
  payload[10]  integral    8     'd76
  payload[11]  integral    8     'd54
  payload[12]  integral    8     'd48
  payload[13]  integral    8     177
  payload[14]  integral    8     'd26
  payload[15]  integral    8     211
  payload[16]  integral    8     'd39
  payload[17]  integral    8     'd43
  payload[18]  integral    8     'd26
  payload[19]  integral    8     'd5
  payload[20]  integral    8     'd17
  payload[21]  integral    8     181
  payload[22]  integral    8     'd99
  payload[23]  integral    8     196
  payload[24]  integral    8     244
  parity_byte  integral    8     'd74
  delay        integral    6     'd0
--------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


      SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name               Type        Size  Value
----------------------------------------
source_mon         source_xtn  -     @1870
  destin_address   integral    2     'd1
  pay_lenth        integral    6     'd22
  header_byte      integral    8     'd89
  payload[0]       integral    8     254
  payload[1]       integral    8     184
  payload[2]       integral    8     226
  payload[3]       integral    8     'd56
```

```
  payload[4]      integral    8     254
  payload[5]      integral    8     'd69
  payload[6]      integral    8     'd109
  payload[7]      integral    8     232
  payload[8]      integral    8     232
  payload[9]      integral    8     'd57
  payload[10]     integral    8     139
  payload[11]     integral    8     185
  payload[12]     integral    8     230
  payload[13]     integral    8     175
  payload[14]     integral    8     'd85
  payload[15]     integral    8     'd123
  payload[16]     integral    8     240
  payload[17]     integral    8     166
  payload[18]     integral    8     'd104
  payload[19]     integral    8     167
  payload[20]     integral    8     240
  payload[21]     integral    8     'd27
  parity_byte     integral    8     'd13
----------------------------------------
----------------------------------------
Name            Type        Size  Value
----------------------------------------
destin_mon      destin_xtn  -     @1874
  des_address   integral    2     'd1
  pay_lenth     integral    6     'd22
  header_byte   integral    8     'd89
  payload[0]    integral    8     254
  payload[1]    integral    8     184
  payload[2]    integral    8     226
  payload[3]    integral    8     'd56
  payload[4]    integral    8     254
  payload[5]    integral    8     'd69
  payload[6]    integral    8     'd109
  payload[7]    integral    8     232
  payload[8]    integral    8     232
  payload[9]    integral    8     'd57
  payload[10]   integral    8     139
  payload[11]   integral    8     185
  payload[12]   integral    8     230
  payload[13]   integral    8     175
  payload[14]   integral    8     'd85
  payload[15]   integral    8     'd123
  payload[16]   integral    8     240
  payload[17]   integral    8     166
  payload[18]   integral    8     'd104
  payload[19]   integral    8     167
  payload[20]   integral    8     240
  payload[21]   integral    8     'd27
  parity_byte   integral    8     'd13
  delay         integral    6     'd0
----------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


     SUCCESSFULLY MATCHED


================================================================================
```

```
3706
3707
3708  ================================================================================
3709
3710
3711   source  side functional coverage = 38.889
3712
3713
3714   destin  side functional coverage = 41.667
3715
3716
3717  ================================================================================
3718
3719  UVM_INFO /home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_objection.svh(1274
     ) @ 4850000: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
3720
3721  --- UVM Report Summary ---
3722
3723  ** Report counts by severity
3724  UVM_INFO :    3
3725  UVM_WARNING :    0
3726  UVM_ERROR :    0
3727  UVM_FATAL :    0
3728  ** Report counts by id
3729  [RNTST]     1
3730  [TEST_DONE]     1
3731  [UVMTOP]     1
3732  $finish called from file
     "/home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_root.svh", line 437.
3733  $finish at simulation time                4850000
3734             V C S   S i m u l a t i o n   R e p o r t
3735  Time: 4850000 ps
3736
3737
3738
3739  // Large PACKETS
3740
3741
3742  UVM_INFO @ 0: reporter [RNTST] Running test large_seq_test...
3743  UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
3744  ----------------------------------------------------------------------
3745  Name                           Type                           Size  Value
3746  ----------------------------------------------------------------------
3747  uvm_test_top                   large_seq_test                 -     @474
3748    envh                         tb_env                         -     @499
3749      destin_agth                destin_agent_top               -     @516
3750        agth[0]                  destin_agent                   -     @660
3751          drvh                   destin_driver                  -     @708
3752            rsp_port             uvm_analysis_port              -     @725
3753            seq_item_port        uvm_seq_item_pull_port         -     @716
3754          monh                   destin_monitor                 -     @691
3755            dmon_port            uvm_analysis_port              -     @699
3756          seqr                   destin_seqr                    -     @734
3757            rsp_export           uvm_analysis_export            -     @742
3758            seq_item_export      uvm_seq_item_pull_imp          -     @848
3759            arbitration_queue    array                          0     -
3760            lock_queue           array                          0     -
3761            num_last_reqs        integral                       32    'd1
3762            num_last_rsps        integral                       32    'd1
3763        agth[1]                  destin_agent                   -     @669
3764          drvh                   destin_driver                  -     @886
3765            rsp_port             uvm_analysis_port              -     @903
3766            seq_item_port        uvm_seq_item_pull_port         -     @894
3767          monh                   destin_monitor                 -     @869
3768            dmon_port            uvm_analysis_port              -     @877
3769          seqr                   destin_seqr                    -     @912
3770            rsp_export           uvm_analysis_export            -     @920
3771            seq_item_export      uvm_seq_item_pull_imp          -     @1026
3772            arbitration_queue    array                          0     -
```

```
            lock_queue          array                           0     -
            num_last_reqs       integral                        32    'd1
            num_last_rsps       integral                        32    'd1
        agth[2]                 destin_agent                    -     @678
          drvh                  destin_driver                   -     @1064
            rsp_port            uvm_analysis_port               -     @1081
            seq_item_port       uvm_seq_item_pull_port          -     @1072
          monh                  destin_monitor                  -     @1047
            dmon_port           uvm_analysis_port               -     @1055
          seqr                  destin_seqr                     -     @1090
            rsp_export          uvm_analysis_export             -     @1098
            seq_item_export     uvm_seq_item_pull_imp           -     @1204
            arbitration_queue   array                           0     -
            lock_queue          array                           0     -
            num_last_reqs       integral                        32    'd1
            num_last_rsps       integral                        32    'd1
      sb                        tb_scoreboard                   -     @524
        destin_fifo[0]          uvm_tlm_analysis_fifo #(T)      -     @1274
          analysis_export       uvm_analysis_imp                -     @1318
          get_ap                uvm_analysis_port               -     @1309
          get_peek_export       uvm_get_peek_imp                -     @1291
          put_ap                uvm_analysis_port               -     @1300
          put_export            uvm_put_imp                     -     @1282
        destin_fifo[1]          uvm_tlm_analysis_fifo #(T)      -     @1327
          analysis_export       uvm_analysis_imp                -     @1371
          get_ap                uvm_analysis_port               -     @1362
          get_peek_export       uvm_get_peek_imp                -     @1344
          put_ap                uvm_analysis_port               -     @1353
          put_export            uvm_put_imp                     -     @1335
        destin_fifo[2]          uvm_tlm_analysis_fifo #(T)      -     @1380
          analysis_export       uvm_analysis_imp                -     @1424
          get_ap                uvm_analysis_port               -     @1415
          get_peek_export       uvm_get_peek_imp                -     @1397
          put_ap                uvm_analysis_port               -     @1406
          put_export            uvm_put_imp                     -     @1388
        source_fifo             uvm_tlm_analysis_fifo #(T)      -     @1221
          analysis_export       uvm_analysis_imp                -     @1265
          get_ap                uvm_analysis_port               -     @1256
          get_peek_export       uvm_get_peek_imp                -     @1238
          put_ap                uvm_analysis_port               -     @1247
          put_export            uvm_put_imp                     -     @1229
      source_agth               source_agent_top                -     @508
        agth[0]                 source_agent                    -     @1438
          drvh                  source_driver                   -     @1468
            rsp_port            uvm_analysis_port               -     @1485
            seq_item_port       uvm_seq_item_pull_port          -     @1476
          monh                  source_monitor                  -     @1451
            smon_port           uvm_analysis_port               -     @1459
          seqr                  source_seqr                     -     @1494
            rsp_export          uvm_analysis_export             -     @1502
            seq_item_export     uvm_seq_item_pull_imp           -     @1608
            arbitration_queue   array                           0     -
            lock_queue          array                           0     -
            num_last_reqs       integral                        32    'd1
            num_last_rsps       integral                        32    'd1
      v_seqr                    virtual_seqr                    -     @532
        rsp_export              uvm_analysis_export             -     @540
        seq_item_export         uvm_seq_item_pull_imp           -     @646
        arbitration_queue       array                           0     -
        lock_queue              array                           0     -
        num_last_reqs           integral                        32    'd1
        num_last_rsps           integral                        32    'd1
  ----------------------------------------------------------------------

  Signal Error From Source Monitor = 1
  ------------------------------------------
  Name               Type         Size  Value
  ------------------------------------------
  source_mon         source_xtn   -     @1648
```

```
  destin_address  integral    2     2
  pay_lenth       integral    6     57
  header_byte     integral    8     230
  payload[0]      integral    8     'd16
  payload[1]      integral    8     161
  payload[2]      integral    8     'd50
  payload[3]      integral    8     245
  payload[4]      integral    8     'd74
  payload[5]      integral    8     'd71
  payload[6]      integral    8     202
  payload[7]      integral    8     'd23
  payload[8]      integral    8     'd73
  payload[9]      integral    8     241
  payload[10]     integral    8     'd80
  payload[11]     integral    8     'd8
  payload[12]     integral    8     'd79
  payload[13]     integral    8     199
  payload[14]     integral    8     'd54
  payload[15]     integral    8     'd103
  payload[16]     integral    8     192
  payload[17]     integral    8     138
  payload[18]     integral    8     'd125
  payload[19]     integral    8     'd104
  payload[20]     integral    8     157
  payload[21]     integral    8     'd86
  payload[22]     integral    8     'd112
  payload[23]     integral    8     'd11
  payload[24]     integral    8     'd118
  payload[25]     integral    8     133
  payload[26]     integral    8     'd24
  payload[27]     integral    8     171
  payload[28]     integral    8     133
  payload[29]     integral    8     'd120
  payload[30]     integral    8     'd9
  payload[31]     integral    8     'd0
  payload[32]     integral    8     'd83
  payload[33]     integral    8     210
  payload[34]     integral    8     224
  payload[35]     integral    8     204
  payload[36]     integral    8     'd14
  payload[37]     integral    8     'd106
  payload[38]     integral    8     'd122
  payload[39]     integral    8     'd5
  payload[40]     integral    8     'd119
  payload[41]     integral    8     'd116
  payload[42]     integral    8     'd23
  payload[43]     integral    8     'd43
  payload[44]     integral    8     'd91
  payload[45]     integral    8     'd60
  payload[46]     integral    8     'd91
  payload[47]     integral    8     252
  payload[48]     integral    8     'd42
  payload[49]     integral    8     'd94
  payload[50]     integral    8     154
  payload[51]     integral    8     'd71
  payload[52]     integral    8     'd67
  payload[53]     integral    8     251
  payload[54]     integral    8     'd75
  payload[55]     integral    8     214
  payload[56]     integral    8     'd117
  parity_byte     integral    8     146
-----------------------------------------
-----------------------------------------
Name            Type         Size  Value
-----------------------------------------
destin_mon      destin_xtn   -     @1642
  des_address   integral     2     2
  pay_lenth     integral     6     57
  header_byte   integral     8     230
```

```
3911     payload[0]   integral    8     'd16
3912     payload[1]   integral    8     161
3913     payload[2]   integral    8     'd50
3914     payload[3]   integral    8     245
3915     payload[4]   integral    8     'd74
3916     payload[5]   integral    8     'd71
3917     payload[6]   integral    8     202
3918     payload[7]   integral    8     'd23
3919     payload[8]   integral    8     'd73
3920     payload[9]   integral    8     241
3921     payload[10]  integral    8     'd80
3922     payload[11]  integral    8     'd8
3923     payload[12]  integral    8     'd79
3924     payload[13]  integral    8     199
3925     payload[14]  integral    8     'd54
3926     payload[15]  integral    8     'd103
3927     payload[16]  integral    8     192
3928     payload[17]  integral    8     138
3929     payload[18]  integral    8     'd125
3930     payload[19]  integral    8     'd104
3931     payload[20]  integral    8     157
3932     payload[21]  integral    8     'd86
3933     payload[22]  integral    8     'd112
3934     payload[23]  integral    8     'd11
3935     payload[24]  integral    8     'd118
3936     payload[25]  integral    8     133
3937     payload[26]  integral    8     'd24
3938     payload[27]  integral    8     171
3939     payload[28]  integral    8     133
3940     payload[29]  integral    8     'd120
3941     payload[30]  integral    8     'd9
3942     payload[31]  integral    8     'd0
3943     payload[32]  integral    8     'd83
3944     payload[33]  integral    8     210
3945     payload[34]  integral    8     224
3946     payload[35]  integral    8     204
3947     payload[36]  integral    8     'd14
3948     payload[37]  integral    8     'd106
3949     payload[38]  integral    8     'd122
3950     payload[39]  integral    8     'd5
3951     payload[40]  integral    8     'd119
3952     payload[41]  integral    8     'd116
3953     payload[42]  integral    8     'd23
3954     payload[43]  integral    8     'd43
3955     payload[44]  integral    8     'd91
3956     payload[45]  integral    8     'd60
3957     payload[46]  integral    8     'd91
3958     payload[47]  integral    8     252
3959     payload[48]  integral    8     'd42
3960     payload[49]  integral    8     'd94
3961     payload[50]  integral    8     154
3962     payload[51]  integral    8     'd71
3963     payload[52]  integral    8     'd67
3964     payload[53]  integral    8     251
3965     payload[54]  integral    8     'd75
3966     payload[55]  integral    8     214
3967     payload[56]  integral    8     'd117
3968     parity_byte  integral    8     146
3969     delay        integral    6     'd0
3970   ---------------------------------------
3971
3972   ====================== Header_Byte Matched SuccessFull ======================
3973
3974
3975   ====================== Payload Matched SuccessFull ======================
3976
3977
3978   ====================== Parity_Byte Matched SuccessFull ======================
3979
```

```
================================================================================


     SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
----------------------------------------
Name               Type        Size  Value
----------------------------------------
source_mon         source_xtn  -     @1803
  destin_address   integral    2     2
  pay_lenth        integral    6     56
  header_byte      integral    8     226
  payload[0]       integral    8     167
  payload[1]       integral    8     212
  payload[2]       integral    8     211
  payload[3]       integral    8     'd71
  payload[4]       integral    8     227
  payload[5]       integral    8     'd91
  payload[6]       integral    8     188
  payload[7]       integral    8     207
  payload[8]       integral    8     'd47
  payload[9]       integral    8     'd0
  payload[10]      integral    8     'd117
  payload[11]      integral    8     148
  payload[12]      integral    8     206
  payload[13]      integral    8     190
  payload[14]      integral    8     'd39
  payload[15]      integral    8     173
  payload[16]      integral    8     206
  payload[17]      integral    8     205
  payload[18]      integral    8     'd76
  payload[19]      integral    8     'd58
  payload[20]      integral    8     143
  payload[21]      integral    8     'd20
  payload[22]      integral    8     128
  payload[23]      integral    8     195
  payload[24]      integral    8     'd50
  payload[25]      integral    8     'd79
  payload[26]      integral    8     'd29
  payload[27]      integral    8     'd31
  payload[28]      integral    8     236
  payload[29]      integral    8     137
  payload[30]      integral    8     245
  payload[31]      integral    8     185
  payload[32]      integral    8     152
  payload[33]      integral    8     'd110
  payload[34]      integral    8     239
  payload[35]      integral    8     205
  payload[36]      integral    8     234
  payload[37]      integral    8     'd69
  payload[38]      integral    8     171
  payload[39]      integral    8     172
```

```
4049     payload[40]     integral    8     212
4050     payload[41]     integral    8     134
4051     payload[42]     integral    8     194
4052     payload[43]     integral    8     'd121
4053     payload[44]     integral    8     231
4054     payload[45]     integral    8     231
4055     payload[46]     integral    8     'd21
4056     payload[47]     integral    8     'd50
4057     payload[48]     integral    8     'd41
4058     payload[49]     integral    8     'd89
4059     payload[50]     integral    8     'd24
4060     payload[51]     integral    8     'd6
4061     payload[52]     integral    8     'd103
4062     payload[53]     integral    8     143
4063     payload[54]     integral    8     208
4064     payload[55]     integral    8     'd91
4065     parity_byte     integral    8     190
4066   ----------------------------------------
4067   ----------------------------------------
4068   Name            Type        Size  Value
4069   ----------------------------------------
4070   destin_mon      destin_xtn  -     @1807
4071     des_address   integral    2     2
4072     pay_lenth     integral    6     56
4073     header_byte   integral    8     226
4074     payload[0]    integral    8     167
4075     payload[1]    integral    8     212
4076     payload[2]    integral    8     211
4077     payload[3]    integral    8     'd71
4078     payload[4]    integral    8     227
4079     payload[5]    integral    8     'd91
4080     payload[6]    integral    8     188
4081     payload[7]    integral    8     207
4082     payload[8]    integral    8     'd47
4083     payload[9]    integral    8     'd0
4084     payload[10]   integral    8     'd117
4085     payload[11]   integral    8     148
4086     payload[12]   integral    8     206
4087     payload[13]   integral    8     190
4088     payload[14]   integral    8     'd39
4089     payload[15]   integral    8     173
4090     payload[16]   integral    8     206
4091     payload[17]   integral    8     205
4092     payload[18]   integral    8     'd76
4093     payload[19]   integral    8     'd58
4094     payload[20]   integral    8     143
4095     payload[21]   integral    8     'd20
4096     payload[22]   integral    8     128
4097     payload[23]   integral    8     195
4098     payload[24]   integral    8     'd50
4099     payload[25]   integral    8     'd79
4100     payload[26]   integral    8     'd29
4101     payload[27]   integral    8     'd31
4102     payload[28]   integral    8     236
4103     payload[29]   integral    8     137
4104     payload[30]   integral    8     245
4105     payload[31]   integral    8     185
4106     payload[32]   integral    8     152
4107     payload[33]   integral    8     'd110
4108     payload[34]   integral    8     239
4109     payload[35]   integral    8     205
4110     payload[36]   integral    8     234
4111     payload[37]   integral    8     'd69
4112     payload[38]   integral    8     171
4113     payload[39]   integral    8     172
4114     payload[40]   integral    8     212
4115     payload[41]   integral    8     134
4116     payload[42]   integral    8     194
4117     payload[43]   integral    8     'd121
```

```
  payload[44]  integral    8     231
  payload[45]  integral    8     231
  payload[46]  integral    8     'd21
  payload[47]  integral    8     'd50
  payload[48]  integral    8     'd41
  payload[49]  integral    8     'd89
  payload[50]  integral    8     'd24
  payload[51]  integral    8     'd6
  payload[52]  integral    8     'd103
  payload[53]  integral    8     143
  payload[54]  integral    8     208
  payload[55]  integral    8     'd91
  parity_byte  integral    8     190
  delay        integral    6     'd0
-------------------------------------

====================== Header_Byte Matched SuccessFull ======================


====================== Payload Matched SuccessFull ======================


====================== Parity_Byte Matched SuccessFull ======================


================================================================================


      SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
-----------------------------------------
Name               Type         Size  Value
-----------------------------------------
source_mon         source_xtn   -     @1826
  destin_address   integral     2     2
  pay_lenth        integral     6     44
  header_byte      integral     8     178
  payload[0]       integral     8     'd39
  payload[1]       integral     8     'd125
  payload[2]       integral     8     216
  payload[3]       integral     8     215
  payload[4]       integral     8     'd0
  payload[5]       integral     8     'd125
  payload[6]       integral     8     'd42
  payload[7]       integral     8     173
  payload[8]       integral     8     'd0
  payload[9]       integral     8     'd1
  payload[10]      integral     8     'd63
  payload[11]      integral     8     'd89
  payload[12]      integral     8     149
  payload[13]      integral     8     200
  payload[14]      integral     8     203
  payload[15]      integral     8     'd43
```

```
  payload[16]     integral    8     212
  payload[17]     integral    8     'd113
  payload[18]     integral    8     215
  payload[19]     integral    8     134
  payload[20]     integral    8     'd106
  payload[21]     integral    8     215
  payload[22]     integral    8     241
  payload[23]     integral    8     'd2
  payload[24]     integral    8     'd113
  payload[25]     integral    8     135
  payload[26]     integral    8     'd115
  payload[27]     integral    8     225
  payload[28]     integral    8     'd27
  payload[29]     integral    8     190
  payload[30]     integral    8     255
  payload[31]     integral    8     171
  payload[32]     integral    8     181
  payload[33]     integral    8     247
  payload[34]     integral    8     208
  payload[35]     integral    8     'd120
  payload[36]     integral    8     'd56
  payload[37]     integral    8     147
  payload[38]     integral    8     191
  payload[39]     integral    8     225
  payload[40]     integral    8     'd14
  payload[41]     integral    8     'd43
  payload[42]     integral    8     158
  payload[43]     integral    8     221
  parity_byte     integral    8     145
-----------------------------------------
-----------------------------------
Name            Type        Size  Value
-----------------------------------
destin_mon      destin_xtn  -     @1830
  des_address   integral    2     2
  pay_lenth     integral    6     44
  header_byte   integral    8     178
  payload[0]    integral    8     'd39
  payload[1]    integral    8     'd125
  payload[2]    integral    8     216
  payload[3]    integral    8     215
  payload[4]    integral    8     'd0
  payload[5]    integral    8     'd125
  payload[6]    integral    8     'd42
  payload[7]    integral    8     173
  payload[8]    integral    8     'd0
  payload[9]    integral    8     'd1
  payload[10]   integral    8     'd63
  payload[11]   integral    8     'd89
  payload[12]   integral    8     149
  payload[13]   integral    8     200
  payload[14]   integral    8     203
  payload[15]   integral    8     'd43
  payload[16]   integral    8     212
  payload[17]   integral    8     'd113
  payload[18]   integral    8     215
  payload[19]   integral    8     134
  payload[20]   integral    8     'd106
  payload[21]   integral    8     215
  payload[22]   integral    8     241
  payload[23]   integral    8     'd2
  payload[24]   integral    8     'd113
  payload[25]   integral    8     135
  payload[26]   integral    8     'd115
  payload[27]   integral    8     225
  payload[28]   integral    8     'd27
  payload[29]   integral    8     190
  payload[30]   integral    8     255
  payload[31]   integral    8     171
```

```
  payload[32]  integral    8     181
  payload[33]  integral    8     247
  payload[34]  integral    8     208
  payload[35]  integral    8     'd120
  payload[36]  integral    8     'd56
  payload[37]  integral    8     147
  payload[38]  integral    8     191
  payload[39]  integral    8     225
  payload[40]  integral    8     'd14
  payload[41]  integral    8     'd43
  payload[42]  integral    8     158
  payload[43]  integral    8     221
  parity_byte  integral    8     145
  delay        integral    6     'd0
-------------------------------------

===================== Header_Byte Matched SuccessFull =====================


===================== Payload Matched SuccessFull =====================


===================== Parity_Byte Matched SuccessFull =====================


================================================================================


      SUCCESSFULLY MATCHED


================================================================================


================================================================================


 source  side functional coverage = 38.889


 destin  side functional coverage = 41.667


================================================================================

Signal Error From Source Monitor = 1
-----------------------------------------
Name               Type         Size  Value
-----------------------------------------
source_mon         source_xtn   -     @1848
  destin_address   integral     2     2
  pay_lenth        integral     6     61
  header_byte      integral     8     246
  payload[0]       integral     8     'd81
  payload[1]       integral     8     'd100
  payload[2]       integral     8     'd82
  payload[3]       integral     8     169
  payload[4]       integral     8     203
  payload[5]       integral     8     'd49
  payload[6]       integral     8     201
  payload[7]       integral     8     244
  payload[8]       integral     8     'd13
  payload[9]       integral     8     'd5
  payload[10]      integral     8     'd112
  payload[11]      integral     8     'd105
  payload[12]      integral     8     'd32
  payload[13]      integral     8     216
  payload[14]      integral     8     158
  payload[15]      integral     8     'd68
```

```
  payload[16]     integral    8     156
  payload[17]     integral    8     179
  payload[18]     integral    8     'd69
  payload[19]     integral    8     'd64
  payload[20]     integral    8     'd109
  payload[21]     integral    8     130
  payload[22]     integral    8     'd13
  payload[23]     integral    8     'd14
  payload[24]     integral    8     172
  payload[25]     integral    8     153
  payload[26]     integral    8     'd65
  payload[27]     integral    8     207
  payload[28]     integral    8     'd57
  payload[29]     integral    8     'd71
  payload[30]     integral    8     251
  payload[31]     integral    8     'd59
  payload[32]     integral    8     'd21
  payload[33]     integral    8     'd88
  payload[34]     integral    8     255
  payload[35]     integral    8     'd55
  payload[36]     integral    8     'd53
  payload[37]     integral    8     135
  payload[38]     integral    8     203
  payload[39]     integral    8     189
  payload[40]     integral    8     170
  payload[41]     integral    8     226
  payload[42]     integral    8     243
  payload[43]     integral    8     'd111
  payload[44]     integral    8     180
  payload[45]     integral    8     168
  payload[46]     integral    8     'd2
  payload[47]     integral    8     141
  payload[48]     integral    8     'd92
  payload[49]     integral    8     'd90
  payload[50]     integral    8     'd57
  payload[51]     integral    8     'd4
  payload[52]     integral    8     'd18
  payload[53]     integral    8     'd76
  payload[54]     integral    8     176
  payload[55]     integral    8     'd108
  payload[56]     integral    8     215
  payload[57]     integral    8     'd86
  payload[58]     integral    8     235
  payload[59]     integral    8     'd98
  payload[60]     integral    8     175
  parity_byte     integral    8     'd23
-----------------------------------------
-----------------------------------
Name            Type         Size  Value
-----------------------------------
destin_mon      destin_xtn   -     @1852
  des_address   integral     2     2
  pay_lenth     integral     6     61
  header_byte   integral     8     246
  payload[0]    integral     8     'd81
  payload[1]    integral     8     'd100
  payload[2]    integral     8     'd82
  payload[3]    integral     8     169
  payload[4]    integral     8     203
  payload[5]    integral     8     'd49
  payload[6]    integral     8     201
  payload[7]    integral     8     244
  payload[8]    integral     8     'd13
  payload[9]    integral     8     'd5
  payload[10]   integral     8     'd112
  payload[11]   integral     8     'd105
  payload[12]   integral     8     'd32
  payload[13]   integral     8     216
  payload[14]   integral     8     158
```

```
4394      payload[15]  integral    8     'd68
4395      payload[16]  integral    8     156
4396      payload[17]  integral    8     179
4397      payload[18]  integral    8     'd69
4398      payload[19]  integral    8     'd64
4399      payload[20]  integral    8     'd109
4400      payload[21]  integral    8     130
4401      payload[22]  integral    8     'd13
4402      payload[23]  integral    8     'd14
4403      payload[24]  integral    8     172
4404      payload[25]  integral    8     153
4405      payload[26]  integral    8     'd65
4406      payload[27]  integral    8     207
4407      payload[28]  integral    8     'd57
4408      payload[29]  integral    8     'd71
4409      payload[30]  integral    8     251
4410      payload[31]  integral    8     'd59
4411      payload[32]  integral    8     'd21
4412      payload[33]  integral    8     'd88
4413      payload[34]  integral    8     255
4414      payload[35]  integral    8     'd55
4415      payload[36]  integral    8     'd53
4416      payload[37]  integral    8     135
4417      payload[38]  integral    8     203
4418      payload[39]  integral    8     189
4419      payload[40]  integral    8     170
4420      payload[41]  integral    8     226
4421      payload[42]  integral    8     243
4422      payload[43]  integral    8     'd111
4423      payload[44]  integral    8     180
4424      payload[45]  integral    8     168
4425      payload[46]  integral    8     'd2
4426      payload[47]  integral    8     141
4427      payload[48]  integral    8     'd92
4428      payload[49]  integral    8     'd90
4429      payload[50]  integral    8     'd57
4430      payload[51]  integral    8     'd4
4431      payload[52]  integral    8     'd18
4432      payload[53]  integral    8     'd76
4433      payload[54]  integral    8     176
4434      payload[55]  integral    8     'd108
4435      payload[56]  integral    8     215
4436      payload[57]  integral    8     'd86
4437      payload[58]  integral    8     235
4438      payload[59]  integral    8     'd98
4439      payload[60]  integral    8     175
4440      parity_byte  integral    8     'd23
4441      delay        integral    6     'd0
4442    ------------------------------------

4443
4444    ====================== Header_Byte Matched SuccessFull ======================
4445
4446
4447    ====================== Payload Matched SuccessFull ======================
4448
4449
4450    ====================== Parity_Byte Matched SuccessFull ======================
4451
4452
4453    ================================================================================
4454
4455
4456         SUCCESSFULLY MATCHED
4457
4458
4459    ================================================================================
4460
4461
4462    ================================================================================
```

```
 source  side functional coverage = 38.889

 destin  side functional coverage = 41.667

================================================================================

Signal Error From Source Monitor = 1
-----------------------------------------
Name               Type        Size  Value
-----------------------------------------
source_mon         source_xtn  -     @1870
  destin_address   integral    2     2
  pay_lenth        integral    6     53
  header_byte      integral    8     214
  payload[0]       integral    8     'd87
  payload[1]       integral    8     247
  payload[2]       integral    8     228
  payload[3]       integral    8     210
  payload[4]       integral    8     248
  payload[5]       integral    8     168
  payload[6]       integral    8     'd18
  payload[7]       integral    8     'd71
  payload[8]       integral    8     'd83
  payload[9]       integral    8     'd104
  payload[10]      integral    8     184
  payload[11]      integral    8     243
  payload[12]      integral    8     'd25
  payload[13]      integral    8     247
  payload[14]      integral    8     'd24
  payload[15]      integral    8     'd19
  payload[16]      integral    8     'd57
  payload[17]      integral    8     'd52
  payload[18]      integral    8     136
  payload[19]      integral    8     182
  payload[20]      integral    8     'd42
  payload[21]      integral    8     191
  payload[22]      integral    8     150
  payload[23]      integral    8     'd94
  payload[24]      integral    8     186
  payload[25]      integral    8     'd126
  payload[26]      integral    8     'd78
  payload[27]      integral    8     'd126
  payload[28]      integral    8     252
  payload[29]      integral    8     'd28
  payload[30]      integral    8     'd8
  payload[31]      integral    8     'd113
  payload[32]      integral    8     242
  payload[33]      integral    8     'd74
  payload[34]      integral    8     251
  payload[35]      integral    8     'd39
  payload[36]      integral    8     'd92
  payload[37]      integral    8     201
  payload[38]      integral    8     151
  payload[39]      integral    8     195
  payload[40]      integral    8     192
  payload[41]      integral    8     132
  payload[42]      integral    8     198
  payload[43]      integral    8     'd52
  payload[44]      integral    8     252
  payload[45]      integral    8     'd116
  payload[46]      integral    8     233
  payload[47]      integral    8     'd46
  payload[48]      integral    8     'd109
  payload[49]      integral    8     'd57
  payload[50]      integral    8     'd46
```

```
4532      payload[51]     integral    8     'd86
4533      payload[52]     integral    8     189
4534      parity_byte     integral    8     'd30
4535    ----------------------------------------
4536    ---------------------------------------
4537    Name            Type        Size  Value
4538    ---------------------------------------
4539    destin_mon      destin_xtn  -     @1874
4540      des_address   integral    2     2
4541      pay_lenth     integral    6     53
4542      header_byte   integral    8     214
4543      payload[0]    integral    8     'd87
4544      payload[1]    integral    8     247
4545      payload[2]    integral    8     228
4546      payload[3]    integral    8     210
4547      payload[4]    integral    8     248
4548      payload[5]    integral    8     168
4549      payload[6]    integral    8     'd18
4550      payload[7]    integral    8     'd71
4551      payload[8]    integral    8     'd83
4552      payload[9]    integral    8     'd104
4553      payload[10]   integral    8     184
4554      payload[11]   integral    8     243
4555      payload[12]   integral    8     'd25
4556      payload[13]   integral    8     247
4557      payload[14]   integral    8     'd24
4558      payload[15]   integral    8     'd19
4559      payload[16]   integral    8     'd57
4560      payload[17]   integral    8     'd52
4561      payload[18]   integral    8     136
4562      payload[19]   integral    8     182
4563      payload[20]   integral    8     'd42
4564      payload[21]   integral    8     191
4565      payload[22]   integral    8     150
4566      payload[23]   integral    8     'd94
4567      payload[24]   integral    8     186
4568      payload[25]   integral    8     'd126
4569      payload[26]   integral    8     'd78
4570      payload[27]   integral    8     'd126
4571      payload[28]   integral    8     252
4572      payload[29]   integral    8     'd28
4573      payload[30]   integral    8     'd8
4574      payload[31]   integral    8     'd113
4575      payload[32]   integral    8     242
4576      payload[33]   integral    8     'd74
4577      payload[34]   integral    8     251
4578      payload[35]   integral    8     'd39
4579      payload[36]   integral    8     'd92
4580      payload[37]   integral    8     201
4581      payload[38]   integral    8     151
4582      payload[39]   integral    8     195
4583      payload[40]   integral    8     192
4584      payload[41]   integral    8     132
4585      payload[42]   integral    8     198
4586      payload[43]   integral    8     'd52
4587      payload[44]   integral    8     252
4588      payload[45]   integral    8     'd116
4589      payload[46]   integral    8     233
4590      payload[47]   integral    8     'd46
4591      payload[48]   integral    8     'd109
4592      payload[49]   integral    8     'd57
4593      payload[50]   integral    8     'd46
4594      payload[51]   integral    8     'd86
4595      payload[52]   integral    8     189
4596      parity_byte   integral    8     'd30
4597      delay         integral    6     'd0
4598    ---------------------------------------
4599
4600    ==================== Header_Byte Matched SuccessFull ====================
```

```
4601
4602
4603    ====================== Payload Matched SuccessFull ======================
4604
4605
4606    ====================== Parity_Byte Matched SuccessFull ======================
4607
4608
4609    ================================================================================
4610
4611
4612         SUCCESSFULLY MATCHED
4613
4614
4615    ================================================================================
4616
4617
4618    ================================================================================
4619
4620
4621     source  side functional coverage = 38.889
4622
4623
4624     destin  side functional coverage = 41.667
4625
4626
4627    ================================================================================
4628
4629    UVM_INFO /home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_objection.svh(1274
       ) @ 8110000: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
4630
4631    --- UVM Report Summary ---
4632
4633    ** Report counts by severity
4634    UVM_INFO :    3
4635    UVM_WARNING :    0
4636    UVM_ERROR :    0
4637    UVM_FATAL :    0
4638    ** Report counts by id
4639    [RNTST]     1
4640    [TEST_DONE]     1
4641    [UVMTOP]     1
4642    $finish called from file
       "/home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_root.svh", line 437.
4643    $finish at simulation time                8110000
4644                V C S   S i m u l a t i o n   R e p o r t
4645    Time: 8110000 ps
4646
```