

```

1
2
3
4 // fifo
5
6 module fifo_16x9_router(clk,reset,data_in,write_enb,read_enb,soft_reset,lfd_state,full,
7 empty,data_out);
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)

```

```

69         read_addr<=0;
70     else if(read_enb && (!empty))
71         read_addr <= read_addr +1;
72     end
73 */
74 // writing opearion
75
76     always@(posedge clk)
77     begin
78         if(!reset)
79             begin
80                 for(i=0;i<16;i=i+1)
81                     begin
82                         mem[i]<=0;
83                     end
84                 write_addr<=0;
85             end
86         else if(soft_reset)
87             begin
88                 for(i=0;i<16;i=i+1)
89                     begin
90                         mem[i]<=0;
91                     end
92                 write_addr<=0;
93             end
94         else
95             if(write_enb && (!full))
96             begin
97                 mem[write_addr[3:0]]<={lfd_state_temp,data_in};
98                 write_addr <= write_addr+1;
99                 //mem[write_addr[3:0]][8] <= lfd_state_temp;
100             end
101
102     end
103
104 // read opearion
105
106     always@(posedge clk)
107     begin
108         if(!reset) begin
109             data_out<=0;
110             read_addr<=0; end
111         else if(soft_reset) begin
112             data_out<=8'bzzzz_zzzz;
113             read_addr<=0; end
114         else if (fifo_down_counter ==0 && data_out !=0)
115             data_out<=8'bzzzz_zzzz;
116         else
117             if(read_enb && (!empty))
118                 begin
119                     data_out<=mem[read_addr[3:0]];
120                     read_addr <= read_addr +1;
121                 end
122             end
123 /* fifo down count logic when hear byte recived it check that 8 bt of the byt
124 it is one then fifo counte load with payload data then after ti decremented every clk
pulse*/
125
126
127     always@(posedge clk)
128     begin
129         if(!reset)
130             fifo_down_counter<=0;
131         else if(soft_reset)
132             fifo_down_counter<=0;
133
134         else if(read_enb && !empty)
135             begin
136                 if(mem[read_addr[3:0]][8]==1'b1)

```

```

137         fifo_down_counter<=mem[read_addr[3:0]][7:2]+1;
138     else if(fifo_down_counter != 0)
139         fifo_down_counter<=fifo_down_counter-1;
140     end
141     end
142 // fifo full signal logic and empty signal logic
143
144     assign full = ( (write_addr[3:0] == read_addr[3:0]) && (write_addr[4] != read_addr[4
145 ]))?'b1:1'b0;
146     assign empty =( (write_addr[3:0] == read_addr[3:0]) && (write_addr[4] == read_addr[4
147 ]))?'b1:1'b0;
148
149 endmodule
150
151 // fsmm
152 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);
153
154 // declaring the input and output ports
155
156     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;
157     input [1:0]data_in;
158     output detect_add,ld_state,laf_state,full_state,rst_int_reg,lfd_state;
159     output busy,write_enb_reg;
160 // parameter and internal next_state,present state regs declarations
161
162     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;
163     reg [1:0]addr_data;
164     reg [2:0]present,next;
165
166 // storing detecting the destination_address_signal
167
168     always@(*)
169     begin
170         if(!reset)
171             addr_data<=0;
172         else
173             begin
174                 if(detect_add)
175                     addr_data <= data_in;
176                 end
177             end
178         end
179
180 // sequential logic for present state
181
182     always@(posedge clk)
183     begin
184         if(!reset)
185             present<=DECODE_ADDRESS;
186         else if(soft_reset_0 || soft_reset_1 || soft_reset_2)
187             present<=DECODE_ADDRESS;
188         else
189             present<=next;
190         end
191
192 // combinational logic for next state
193
194     always@(*)
195     begin
196         case(present)
197             DECODE_ADDRESS:
198                 if((pkt_valid && addr_data == 2'd0 && fifo_empty_0) || (

```

```

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

```

```

261         next=FIFO_FULL_STATE;
262     end
263     else
264         begin
265             next=DECODE_ADDRESS;
266         end
267     default:next=DECODE_ADDRESS;
268 endcase
269
270 end
271
272 // combinational logic for output
273
274 // busy is going to low at decode state and load state other states it was
275 // high bcaz it is not
276 // allowing new data from the source
277 assign busy = (present == DECODE_ADDRESS || present == LOAD_DATA)?1'b0:1'b1;
278
279
280 assign detect_add =(present == DECODE_ADDRESS)?1'b1:1'b0;
281 assign ld_state =(present == LOAD_DATA)?1'b1:1'b0;
282 assign laf_state =(present == LOAD_AFTER_FULL)? 1'b1:1'b0;
283 assign full_state =(present == FIFO_FULL_STATE)?1'b1:1'b0;
284 assign lfd_state =(present == LOAD_FIRST_DATA)?1'b1:1'b0;
285 assign rst_int_reg=(present == CHECK_PARITY_ERROR)?1'b1:1'b0;
286
287 // write enb reg going tp high only in these state bcaz , these state have
288 // capabul for sending payload and parity data to fifo when write enb reg is
289 // high then only we know which fifo destination.neccasary to send the data
290 // at he particluar state
291 //
292
293 assign write_enb_reg = (present == LOAD_DATA || present == LOAD_PARITY || present ==
LOAD_AFTER_FULL)? 1'b1:1'b0;
294
295 endmodule
296
297
298 // register
299
300
301 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);
302
303 // declaring the regs and wires
304
305     input clk,reset,pkt_valid,fifo_full,rst_int_reg,detect_add,ld_state,laf_state,
full_state,lfd_state;
306     input [7:0]data_in;
307     output reg error,low_pkt_valid,parity_done;
308     output reg [7:0]dout;
309
310 // creating 4 internal register for header byte storing,internal parity
311 // byte,packet parity byte,fifio full state byte each are 8 bits bcz all are bytes
312
313     reg [7:0]header_byte,internal_parity_byte,packet_parity_byte,fifo_full_state_byte;
314
315 // writing logic for header byte sstoring
316 // header store add detect address and presetn state at load fikrst data and
317 // pkt valid is high and corect address destination
318
319     always@(posedge clk)
320     begin
321         if(!reset)
322             begin
323                 header_byte<=0;
324             end
325         else
326             if(pkt_valid && detect_add && (data_in[1:0] != 2'd3))

```

```

327         begin
328             header_byte <=data_in;
329         end
330
331     end
332 // logic for fifo sull state byte
333 // stroing the data in after full state
334
335     always@(posedge clk)
336     begin
337         if(!reset)
338             fifo_full_state_byte <= 0;
339         else if(ld_state && fifo_full)
340             fifo_full_state_byte <= data_in;
341         else if(detect_add)
342             fifo_full_state_byte <= 0;
343         else
344             fifo_full_state_byte <= fifo_full_state_byte ;
345
346     end
347
348
349 // writing logic for dout
350 // data out is works only at play load data in header data writhout error
351
352     always@(posedge clk)
353     begin
354         if(!reset)
355             begin
356                 dout<=0;
357             end
358         else if(detect_add && pkt_valid && (data_in[1:0] != 2'd3))
359             begin
360                 dout <=dout;
361             end
362             else if(lfd_state)
363                 begin
364                     dout<=header_byte;
365                 end
366             else if(ld_state &&(!fifo_full))
367                 begin
368                     dout <=data_in;
369                 end
370             else if(full_state)
371                 begin
372                     dout <= dout;
373                 end
374             else if(laf_state)
375                 begin
376                     dout<=fifo_full_state_byte;
377                 end
378             else
379                 begin
380                     dout <=dout;
381                 end
382         end
383
384 // writing logic for internal parity
385 // first it is ex or opeartion with header byte and then after contious ex or
386 // wuith each pay load data stored into internal parity
387
388
389     always@(posedge clk)
390     begin
391         if(!reset)
392             begin
393                 internal_parity_byte<=0;
394             end

```

```

395         else if(detect_add)
396             begin
397                 internal_parity_byte<=0;
398             end
399         else if(lfd_state)
400             begin
401                 internal_parity_byte <= internal_parity_byte ^ header_byte;
402             end
403         else if(ld_state && !fifo_full)
404             begin
405                 internal_parity_byte <= internal_parity_byte ^ data_in;
406             end
407         else
408             begin
409                 internal_parity_byte <= internal_parity_byte;
410             end
411     end
412
413     // writing logic for packet parity
414     // packet parity check wheather state uis parity_data only in this state
415     // only we are getting parity of packets
416
417     always@(posedge clk)
418     begin
419         if(!reset)
420             begin
421                 packet_parity_byte <= 0;
422             end
423         else if(detect_add)
424             begin
425                 packet_parity_byte <=0;
426             end
427         else if(ld_state && (!pkt_valid) && (!fifo_full))
428             begin
429                 packet_parity_byte <=data_in;
430             end
431         else if(!pkt_valid && rst_int_reg)
432             begin
433                 packet_parity_byte<=0;
434             end
435         else
436             begin
437                 packet_parity_byte <= packet_parity_byte;
438             end
439     end
440
441     //parity done logic
442
443
444     always@(posedge clk)
445     begin
446         if(!reset)
447             begin
448                 parity_done <= 1'b0;
449             end
450         else if(ld_state && (!pkt_valid) && (!fifo_full))
451             begin
452                 parity_done <=1'b1;
453             end
454         else if(laf_state && (!parity_done) && low_pkt_valid)
455             begin
456                 parity_done <=1'b1;
457             end
458         else
459             parity_done <=1'b0;
460     end
461
462
463     // 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 there 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 // storing 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 // destination address decoding and sending write enable signal sending for
528 // respective fifo
529

```



```

530     always@(*)
531     begin
532         if(!reset)
533             write_enb=0;
534         else
535             begin
536                 if(write_enb_reg)
537                     begin
538                         case(addr_data)
539                             2'b00:write_enb=3'b001;
540                             2'b01:write_enb=3'b010;
541                             2'b10:write_enb=3'b100;
542                             default:write_enb=3'b000;
543                         endcase
544                     end
545                 else
546                     write_enb=3'b000;
547             end
548     end

549
550 // fifo full signal asserted based on full status of each fifo
551
552     always@(*)
553     begin
554         if(!reset)
555             fifo_full=0;
556         else
557             begin
558                 case(addr_data)
559                     2'b00:fifo_full=full_0;
560                     2'b01:fifo_full=full_1;
561                     2'b10:fifo_full=full_2;
562                     default:fifo_full=0;
563                 endcase
564             end
565     end

566 // valid out signal generation
567
568     assign vld_out_0=~empty_0;
569     assign vld_out_1=~empty_1;
570     assign vld_out_2=~empty_2;
571
572 // soft start signal generation foer each fifo
573 // the respective soft reset signal go high if read_en is not asserted
574 // with in colk cycle so we take colkc counter to counte clks
575 // vld out being assrent being assrted and taking the reset condiotn also
576
577 wire count_0 = (out_fifo_0_counter == 6'd29) ? 1'b1:1'b0;
578 wire count_1 = (out_fifo_1_counter == 6'd29) ? 1'b1:1'b0;
579 wire count_2 = (out_fifo_2_counter == 6'd29) ? 1'b1:1'b0;
580
581     always@(posedge clk)
582     begin
583         if(!reset)
584             begin
585                 soft_reset_0<=0;
586                 out_fifo_0_counter<=0;
587             end
588         else if(!vld_out_0)
589             begin
590                 soft_reset_0<=0;
591                 out_fifo_0_counter<=0;
592             end
593         else if (read_enb_0)
594             begin
595                 soft_reset_0<=0;
596                 out_fifo_0_counter <=0;
597             end
598         else if(count_0 )

```

```

599         begin
600             out_fifo_0_counter <=0;
601             soft_reset_0<=1;
602         end
603     else
604         begin
605             out_fifo_0_counter <= out_fifo_0_counter+1;
606             soft_reset_0<=0;
607         end
608     end
609
610     always@(posedge clk)
611 begin
612     if(!reset)
613         begin
614             soft_reset_1<=0;
615             out_fifo_1_counter<=0;
616         end
617     else if(!vld_out_1)
618         begin
619             soft_reset_1<=0;
620             out_fifo_1_counter<=0;
621         end
622     else if (read_enb_1)
623         begin
624             soft_reset_1 <=0;
625             out_fifo_1_counter <=0;
626         end
627     else if(count_1)
628         begin
629             out_fifo_1_counter <=0;
630             soft_reset_1<=1;
631         end
632     else
633         begin
634             out_fifo_1_counter <= out_fifo_1_counter+1;
635             soft_reset_1<=0;
636         end
637     end
638
639     always@(posedge clk)
640 begin
641     if(!reset)
642         begin
643             soft_reset_2<=0;
644             out_fifo_2_counter<=0;
645         end
646     else if(!vld_out_2)
647         begin
648             soft_reset_2<=0;
649             out_fifo_2_counter<=0;
650         end
651     else if (read_enb_2)
652         begin
653             soft_reset_2 <=0;
654             out_fifo_2_counter <=0;
655         end
656     else if(count_2)
657         begin
658             out_fifo_2_counter <=0;
659             soft_reset_2<=1;
660         end
661     else
662         begin
663             out_fifo_2_counter <= out_fifo_2_counter+1;
664             soft_reset_2<=0;
665         end
666     end
667 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
701 fsm_router_controller(clk,reset,pkt_valid,busy,parity_done,data_in,soft_reset_0,soft_rese
702 t_1,soft_reset_2,fifo_full,low_pkt_valid,fifo_empty_0,fifo_empty_1,fifo_empty_2,detect_ad
703 d,ld_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state);
704 */
705
706 fsm_router_controller FSM(clk,reset,pkt_valid,busy,parity_done,data_in[1:0],soft_reset_0,
707 soft_reset_1,soft_reset_2,fifo_full,low_pkt_valid,empty_0,empty_1,empty_2,detect_add,
708 ld_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state);
709
710 /*
711 module
712 synchronizer_router(detect_add,data_in,write_enb_reg,clk,reset,vld_out_0,vld_out_1,vld_ou
713 t_2,read_enb_0,read_enb_1,read_enb_2,write_enb,fifo_full,empty_0,empty_1,empty_2,soft_res
714 et_0,soft_reset_1,soft_reset_2,full_0,full_1,full_2);
715 */
716
717 synchronizer_router SYNCHRONIZER(
718     detect_add,data_in[1:0],write_enb_reg,clk,reset,valid_out_0,valid_out_1,valid_out_2,
719     read_enb_0,read_enb_1,read_enb_2,write_enb,fifo_full,empty_0,empty_1,empty_2,
720     soft_reset_0,soft_reset_1,soft_reset_2,full_0,full_1,full_2);
721
722 /*
723 module
724 register_router(clk,reset,pkt_valid,data_in,fifo_full,rst_int_reg,detect_add,ld_state,laf
725 _state,full_state,lfd_state,parity_done,low_pkt_valid,error,dout);
726 */
727
728 register_router REGISTER(clk,reset,pkt_valid,data_in,fifo_full,rst_int_reg,detect_add,
729 ld_state,laf_state,full_state,lfd_state,parity_done,low_pkt_valid,error,data_out);
730
731 /*
732 module
733 fifo_16x9_router(clk,reset,data_in,write_enb,read_enb,soft_reset,lfd_state,full,empty,dat

```

```

722     a_out);
723     */
724     fifo_16x9_router FIFO1(clk,reset,data_out,write_enb[0],read_enb_0,soft_reset_0,lfd_state,
725     full_0,empty_0,data_out_0);
726     fifo_16x9_router FIFO2(clk,reset,data_out,write_enb[1],read_enb_1,soft_reset_1,lfd_state,
727     full_1,empty_1,data_out_1);
728     fifo_16x9_router FIFO3(clk,reset,data_out,write_enb[2],read_enb_2,soft_reset_2,lfd_state,
729     full_2,empty_2,data_out_2);
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     logic [7:0]data_in;
747     bit reset;
748     bit pkt_valid;
749     bit error;
750     bit busy;
751
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

```

```

856         bit clk = 1'b0;
857         always #10 clk = ~clk;
858
859     // instantiating interace
860
861     router_source_if VIF(clk);
862     router_destin_if VIF0(clk);
863     router_destin_if VIF1(clk);
864     router_destin_if VIF2(clk);
865
866
867
868     // instantiating DUT
869     // top module
870     //module
871     router_top(clk,reset,read_enb_0,read_enb_1,read_enb_2,data_in,pkt_valid,data_out_0,data_o
872     ut_1,data_out_2,valid_out_0,valid_out_1,valid_out_2,error,busy);
873
874     router_top DUT(
875         .clk(clk),
876         .reset(VIF.reset),
877         .read_enb_0(VIF0.read_enb),
878         .read_enb_1(VIF1.read_enb),
879         .read_enb_2(VIF2.read_enb),
880         .data_in(VIF.data_in),
881         .pkt_valid(VIF.pkt_valid),
882         .data_out_0(VIF0.data_out),
883         .data_out_1(VIF1.data_out),
884         .data_out_2(VIF2.data_out),
885         .valid_out_0(VIF0.valid_out),
886         .valid_out_1(VIF1.valid_out),
887         .valid_out_2(VIF2.valid_out),
888         .error(VIF.error),
889         .busy(VIF.busy)
890     );
891
892
893     initial begin
894
895         `ifdef VCS
896             $fsdbDumpvars(0,top);
897         `endif
898
899     // setting the virtual interface for source agent and 3 destination agents
900     uvm_config_db #(virtual router_source_if) :: set(null,"*", "vif",VIF);
901     uvm_config_db #(virtual router_destin_if) :: set(null,"*", "vif0",VIF0);
902     uvm_config_db #(virtual router_destin_if) :: set(null,"*", "vif1",VIF1);
903     uvm_config_db #(virtual router_destin_if) :: set(null,"*", "vif2",VIF2);
904     // calling the runtest()
905
906     run_test();
907
908     end
909
910 endmodule
911
912
913
914
915
916
917 // class for destinationj agetn configratorion data base object
918
919
920 class destin_agent_config extends uvm_object;
921
922

```

```

923 // factory registration
924
925     `uvm_object_utils(destin_agent_config)
926
927 // declare the variable required
928
929     uvm_active_passive_enum is_active = UVM_ACTIVE;
930 // declare the virtual interface handle
931
932     virtual router_destin_if vif;
933
934
935 // function new construction
936
937     function new(string name = "destin_agent_config");
938
939         super.new(name);
940
941     endfunction
942
943
944
945 endclass
946
947
948
949
950
951
952
953 // source agent config
954
955 class source_agent_config extends uvm_object;
956
957
958 // factory registration
959
960     `uvm_object_utils(source_agent_config)
961
962
963     uvm_active_passive_enum is_active = UVM_ACTIVE;
964
965
966 // declare virtual interface handle
967
968     virtual router_source_if vif;
969
970 // function new constructor
971
972
973     function new(string name = "source_agent_config" );
974
975         super.new(name);
976
977     endfunction
978
979
980
981
982
983 endclass
984
985
986
987
988
989 //destination driver class
990
991 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 // function 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 agentn config
            handle driver m_cfg from desti agentn 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 send 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

```



```

1058         @(vif.destin_drv);
1059     end
1060
1061     repeat(req.delay)
1062     begin
1063         @(vif.destin_drv);
1064     end
1065     vif.destin_drv.read_enb <= 1'b1;
1066
1067
1068     while(vif.destin_drv.valid_out !== 1'b0)
1069     begin
1070         @(vif.destin_drv);
1071     end
1072
1073     vif.destin_drv.read_enb <= 1'b0;
1074     @(vif.destin_drv);
1075
1076
1077     endtask
1078
1079
1080
1081
1082 endclass
1083
1084
1085
1086
1087
1088
1089
1090
1091 // destination monitor class
1092
1093 class destin_monitor extends uvm_monitor;
1094
1095 // factory registration
1096
1097     `uvm_component_utils(destin_monitor)
1098
1099 // declare virtual interface handle
1100
1101     destin_agent_config m_cfg;
1102     virtual router_destin_if.D_MON vif;
1103 // declaring the destin transaction
1104
1105     destin_xtn dmon;
1106
1107 // tlm port for write to score board
1108
1109     uvm_analysis_port #(destin_xtn) dmon_port;
1110
1111 // function new constructor
1112
1113
1114     function new(string name = "destin_monitor" , uvm_component parent = null );
1115
1116         super.new(name,parent);
1117         dmon_port = new ("dmon_port",this);
1118
1119     endfunction
1120
1121 // build phase
1122     function void build_phase(uvm_phase phase);
1123
1124         super.build_phase(phase);
1125         // `uvm_info(get_full_name(),"this is monitor destin",UVM_NONE)
1126         if(!uvm_config_db #(destin_agent_config) :: get(this,"",

```

```

1127         "destin_agent_config",m_cfg))
1128         `uvm_fatal(get_full_name(),"cannot get the destin agetrn config
1129         handle monitor m_cfg from desti agetrn top")
1130
1131     endfunction
1132
1133     // connect_phase connecting the virtual interfcece to local interface handle
1134
1135     function void connect_phase(uvm_phase phase);
1136
1137         vif = m_cfg.vif;
1138
1139     endfunction
1140
1141     // run phase
1142
1143     task run_phase(uvm_phase phase);
1144
1145         forever
1146             begin
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 endclass
1185
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 registration
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
1225                         destinaion sequence check it")
1226                     end
1227             finish_item(req);
1228
1229     endtask
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 source",UVM_NONE)
1299         if(!uvm_config_db #(source_agent_config) :: get(this,"",
1300             "source_agent_config",m_cfg))
1301             `uvm_fatal(get_full_name(),"cannot get the source agent config
1302             handle driver m_cfg from source agent top")
1303
1304     endfunction
1305
1306 // connect phase
1307
1308     function void connect_phase(uvm_phase phase);
1309
1310         vif = m_cfg.vif;
1311
1312     endfunction
1313
1314 // task run phase and send to dut method calling
1315
1316     task run_phase(uvm_phase phase);
1317         super.run_phase(phase);
1318         @(vif.source_drv);
1319         vif.source_drv.reset <= 1'b0;
1320         @(vif.source_drv);
1321         vif.source_drv.reset <= 1'b1;
1322
1323
1324     forever begin
1325
1326         seq_item_port.get_next_item(req);
1327
1328         send_to_dut(req);
1329         seq_item_port.item_done();
1330         req.print();

```

```

1329         end
1330     endtask
1331
1332     // send to dut
1333
1334     task send_to_dut(source_xtn req);
1335
1336         while(vif.source_drv.busy != 1'b0)
1337         begin
1338             @(vif.source_drv);
1339         end
1340         vif.source_drv.pkt_valid <= 1'b1;
1341         vif.source_drv.data_in  <= req.header_byte;
1342         @(vif.source_drv);
1343         foreach(req.payload[i])
1344             begin
1345                 while(vif.source_drv.busy != 1'b0)
1346                 begin
1347                     @(vif.source_drv);
1348                 end
1349                 vif.source_drv.data_in <= req.payload[i];
1350                 @(vif.source_drv);
1351             end
1352
1353         vif.source_drv.pkt_valid <= 1'b0;
1354         vif.source_drv.data_in <= req.parity_byte;
1355         repeat(2)
1356             @(vif.source_drv);
1357     endtask
1358
1359
1360
1361 endclass
1362
1363
1364
1365
1366
1367
1368
1369 // class source monitor
1370
1371 class source_monitor extends uvm_monitor;
1372
1373 // factory registration
1374
1375     `uvm_component_utils(source_monitor)
1376
1377 // dfeclare virtual interface
1378
1379     virtual router_source_if.S_MON vif;
1380     source_agent_config m_cfg;
1381     source_xtn smon;
1382
1383 // declaring the anlysis port
1384
1385     uvm_analysis_port #(source_xtn) smon_port;
1386
1387
1388 // function new constructor
1389
1390     function new (string name = "source_monitor",uvm_component parent = null );
1391
1392         super.new(name,parent);
1393         smon_port = new("smon_port",this);
1394
1395     endfunction
1396
1397 // 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,"",
1404             "source_agent_config",m_cfg))
1405             `uvm_fatal(get_full_name(),"cannot get the source agent config
1406             handle monitor m_cfg from source agent top")
1407
1408     endfunction
1409
1410 // connect phase
1411
1412     function void connect_phase(uvm_phase phase);
1413
1414         vif = m_cfg.vif;
1415
1416     endfunction
1417
1418 // run phase
1419
1420     task run_phase(uvm_phase phase);
1421
1422         forever
1423             begin
1424
1425                 super.run_phase(phase);
1426                 collect_data();
1427                 smon.print();
1428                 smon_port.write(smon);
1429
1430             end
1431         endtask
1432
1433 // collect task for from interace to monnitor
1434
1435     task collect_data();
1436         smon = source_xtn :: type_id :: create("smon");
1437         while(vif.source_mon.busy != 1'b0)
1438             @(vif.source_mon);
1439         while(vif.source_mon.pkt_valid != 1'b1)
1440             @(vif.source_mon);
1441         smon.header_byte = vif.source_mon.data_in;
1442         @(vif.source_mon);
1443         smon.payload = new[smon.header_byte[7:2]];
1444         foreach(smon.payload[i])
1445             begin
1446                 while(vif.source_mon.busy != 1'b0)
1447                     begin
1448                         @(vif.source_mon);
1449                         end
1450                         smon.payload[i] = vif.source_mon.data_in;
1451                         @(vif.source_mon);
1452                     end
1453             end
1454         smon.parity_byte = vif.source_mon.data_in;
1455
1456         @(vif.source_mon);
1457         @(vif.source_mon);
1458         smon.error = vif.source_mon.error;
1459         $display("Signal Error From Source Monitor = %0b",smon.error);
1460
1461     endtask
1462
1463 endclass
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 // address 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 source",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
1513                 set it check")
1514
1515         req = source_xtn :: type_id :: create("req");
1516         start_item(req);
1517         assert(req.randomize() with {req.header_byte[1:0] == addr;req.header_byte[7:2]
1518             inside{[1:21]};} )
1519         else
1520             begin
1521                 `uvm_info(get_full_name(),"!!!!!!!!!!!!randomization is failed in
1522                     source seqs at small seq!!!!!!!!!!!!",UVM_LOW)
1523             end
1524
1525         finish_item(req);
1526
1527     endtask
1528
1529 endclass
1530
1531 // ----- 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
1551                 set it check")
1552
1553         req = source_xtn :: type_id :: create("req");
1554         start_item(req);
1555         assert(req.randomize() with {req.header_byte[7:2] inside {[22:41]}};
1556             req.header_byte[1:0] == addr; )
1557
1558         else
1559             begin
1560                 `uvm_info(get_full_name(),"!!!!!!!randomization is failed in
1561                     source seqs at medium seq!!!!!!!",UVM_LOW)
1562
1563             end
1564         finish_item(req);
1565
1566     endtask
1567
1568 endclass
1569
1570 // large sequence for generating the from 42 to 63
1571
1572 class large_seq extends source_seq;
1573
1574 // factory registration
1575     `uvm_object_utils(large_seq)
1576
1577 // function new constructor
1578
1579     function new(string name = "large_seq" );
1580
1581         super.new(name);
1582
1583     endfunction
1584
1585 // task body for 42 to 63
1586
1587     task body();
1588         if(!uvm_config_db #(bit[2])::get(null,get_full_name(),"bit",addr))
1589             `uvm_fatal(get_full_name(),"cannot get the address from test class are
1590                 set it check")
1591
1592         req = source_xtn :: type_id :: create("req");
1593         start_item(req);
1594         assert(req.randomize() with {req.header_byte[7:2] inside {[42:63]}};
1595             req.header_byte[1:0] == addr; )
1596
1597         else
1598             begin
1599                 `uvm_info(get_full_name(),"!!!!!!!randomization is failed in
1600                     source seqs at medium seq!!!!!!!",UVM_LOW)

```



```

1596             end
1597         finish_item(req);
1598     endtask
1599
1600 endclass
1601
1602
1603
1604
1605
1606
1607 // source sequence class
1608
1609 class source_seqr extends uvm_sequencer #(source_xtn);
1610
1611
1612 // factory registration
1613
1614     `uvm_component_utils(source_seqr)
1615
1616
1617 // function new constructor
1618
1619
1620     function new(string name ="source_seqr", uvm_component parent = null);
1621
1622         super.new(name,parent);
1623
1624
1625     endfunction:new
1626
1627
1628
1629
1630
1631 endclass
1632
1633
1634
1635
1636
1637
1638 // virtual sequence class
1639
1640 class virtual_seq extends uvm_sequence #(uvm_sequence_item);
1641
1642 // factory registration
1643
1644     `uvm_object_utils(virtual_seq)
1645
1646 // declaring the handle for sequences
1647
1648         small_seq    s_seq;
1649         medium_seq    m_seq;
1650         large_seq     l_seq;
1651
1652 // declaring the handle for virtual sequencer
1653
1654         virtual_seqr v_seqr;
1655
1656 // declare dynamic handle for physical seqr
1657
1658         source_seqr ps_seqr[];
1659
1660 // declaring the handle for env_config to getting
1661
1662         router_env_config m_cfg;
1663
1664 // function new constructor

```

```

1665         function new(string name = "virtual_seq");
1666
1667             super.new(name);
1668
1669         endfunction:new
1670
1671 // task body for assging the v sequencer to m_sequencer and assingn the these handle to
1672 env_config class
1673
1674     task body();
1675
1676         if(! uvm_config_db #(router_env_config) :: get(null,get_full_name
1677             (), "router_env_config", m_cfg) )
1678             `uvm_fatal(get_full_name(), "cannot get the
1679                 router_env_config handle m_cdfg from test in
1680                 virtual sequence")
1681
1682         ps_seqr = new[m_cfg.no_of_source];
1683
1684 // assiging the m_sequencer to physical virtual sequencer through $cast
1685
1686         assert($cast(v_seqr, m_sequencer))
1687             else
1688                 begin
1689                     `uvm_error(get_full_name(), "error becuase of
1690                         cast not working in the virtual sequence check
1691                         that")
1692                 end
1693             foreach(ps_seqr[i])
1694                 begin
1695                     ps_seqr[i] = v_seqr.s_seqr[i];
1696                 end
1697
1698     endtask:body
1699
1700 endclass
1701
1702 // Extended class from virtual_seq to small seq sequence class
1703
1704 class virtual_small_seq extends virtual_seq;
1705
1706 // factory registration
1707
1708     `uvm_object_utils(virtual_small_seq)
1709
1710 // function new constructor
1711
1712     function new( string name = "virtual_small_seq");
1713
1714         super.new(name);
1715
1716     endfunction
1717
1718 // task body
1719
1720     task body();
1721
1722         s_seq = small_seq :: type_id :: create("s_seq");
1723         super.body();
1724         for(int i=0; i < m_cfg.no_of_source ; i++)
1725             begin
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     task body();
1787         l_seq = large_seq :: type_id :: create("l_seq");
1788         super.body();
1789         for(int i=0; i< m_cfg.no_of_source ; i++)
1790             begin
1791
1792                 l_seq.start(ps_seqr[i]);
1793
1794             end
1795     endtask
1796

```

```

1797
1798 endclass
1799
1800
1801 // virtual sequencer 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,"",
1829             "router_env_config",m_cfg))
1830             `uvm_fatal(get_full_name()," cannot get the router_env_config
1831             handle mcfg from tesdt class in virtual sequncerr")
1832
1833         s_seqr = new[m_cfg.no_of_source];
1834     endfunction
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
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
267
```

```

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,"",
1944             "router_env_config",m_cfg))
1945             begin
1946                 `uvm_fatal(get_full_name(),"cannot get he router_env_config
1947                 handle m_cg in scoreboard ")
1948             end
1949
1950         source_fifo = new [m_cfg.no_of_source];
1951
1952         foreach(source_fifo[i])
1953             begin
1954                 source_fifo[i] = new ("source_fifo",this);
1955             end
1956
1957         destin_fifo = new[m_cfg.no_of_destin];
1958
1959         foreach(destin_fifo[i])
1960             begin
1961                 destin_fifo[i] = new($sformatf("destin_fifo[%0d]",i),this);
1962             end
1963     endfunction
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"
2113             ,m_cfg))
2114             `uvm_fatal(get_full_name(),"cannot get the router_env_config handle
2115             m_cfg from test in env_tb ")
2116         source_agth = source_agent_top :: type_id :: create("source_agth",this);
2117         destin_agth = destin_agent_top :: type_id :: create("destin_agth",this);
2118         sb           = tb_scoreboard :: type_id :: create("sb",this);
2119         v_seqr       = virtual_seqr :: type_id :: create("v_seqr",this);
2120
2121     endfunction
2122
2123 // connect phase
2124
2125     function void connect_phase(uvm_phase phase);
2126
2127         foreach(v_seqr.s_seqr[i])
2128             begin
2129                 v_seqr.s_seqr[i] = source_agth.agth[i].seqr;
2130             end
2131
2132         for(int j = 0; j < m_cfg.no_of_source;j++)
2133             begin
2134                 source_agth.agth[j].monh.smon_port.connect(sb.source_fifo[j].
2135                     analysis_export);
2136             end
2137
2138         for(int i = 0; i < m_cfg.no_of_destin ; i++)
2139             begin
2140                 destin_agth.agth[i].monh.dmon_port.connect(sb.destin_fifo[i].
2141                     analysis_export);
2142             end
2143
2144     endfunction:connect_phase
2145
2146 endclass:tb_env
2147
2148 // test class
2149
2150     class base_test extends uvm_test;
2151
2152         // factory registration
2153
2154         `uvm_component_utils(base_test)
2155
2156         // declaring the test bench env class handle and env_config handles
2157
2158         tb_env envh;
2159         router_env_config m_cfg;
2160
2161         source_agent_config m_source_agth[];
2162         destin_agent_config m_destin_agth[];
2163
2164         int no_of_source = 1;
2165         int no_of_destin = 3;
2166
2167         // header byte destination address setting
2168
2169         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
2191                     :: create($sformatf("m_source_agth[%0d]",i));
2192                 m_source_agth[i].is_active = UVM_ACTIVE;
2193                 if(!uvm_config_db #(virtual router_source_if) ::
2194                     get(this,"","vif",m_source_agth[i].vif))
2195                     `uvm_fatal(get_full_name(),"cannot get the
2196                         interface handle from source are set in top"
2197                     )
2198             end
2199         foreach(m_destin_agth[i])
2200             begin
2201                 m_destin_agth[i] = destin_agent_config :: type_id
2202                     :: create($sformatf("m_destin_agth[%0d]",i));
2203                 m_destin_agth[i].is_active = UVM_ACTIVE;
2204                 if(!uvm_config_db #(virtual router_destin_if) ::
2205                     get(this,"",$sformatf("vif%0d",i),m_destin_agth[i]
2206                     ].vif))
2207                     `uvm_fatal(get_full_name(),"cannot get the
2208                         interface handle from destination are set
2209                         in top")
2210             end
2211
2212         m_cfg.no_of_source = no_of_source;
2213         m_cfg.no_of_destin = no_of_destin;
2214
2215     endfunction
2216
2217 // fucniton build_phase
2218
2219     function void build_phase(uvm_phase phase);
2220
2221         super.build_phase(phase);
2222         m_cfg = router_env_config :: type_id :: create("m_cfg");
2223
2224         m_cfg.m_source_agth = new[no_of_source];
2225         m_cfg.m_destin_agth = new[no_of_destin];
2226
2227         config_data();
2228
2229         foreach(m_source_agth[i])
2230             m_cfg.m_source_agth[i] = m_source_agth[i];
2231
2232         foreach(m_destin_agth[i])
2233             m_cfg.m_destin_agth[i] = m_destin_agth[i];
2234
2235         uvm_config_db #(router_env_config) :: set(this,"*", "router_env_config",
2236             m_cfg);
2237         envh = tb_env :: type_id :: create("envh",this);
2238

```

```

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 sdeg 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

```

```

2370 // declare the handle of large seq
2371
2372     //large_seq seqh;
2373
2374     virtual_large_seq l_seqh;
2375     destin_seq dseq;
2376
2377
2378 // build phase
2379
2380     function void build_phase(uvm_phase phase);
2381
2382         super.build_phase(phase);
2383
2384     endfunction
2385
2386
2387 // function new constructor
2388
2389     function new(string name = "large_seq_test" , uvm_component parent);
2390
2391         super.new(name,parent);
2392         addr = 2'd2;
2393
2394         uvm_config_db #(bit[2])::set(this,"*", "bit",addr);
2395
2396
2397     endfunction
2398
2399 // run_phase of large test seq
2400
2401
2402     task run_phase(uvm_phase phase);
2403         phase.raise_objection(this);
2404         super.run_phase(phase);
2405         l_seqh = virtual_large_seq :: type_id :: create("l_seqh");
2406         dseq = destin_seq :: type_id :: create("dseq");
2407         repeat(5)
2408             begin
2409                 fork
2410                     l_seqh.start(envh.v_seqr);
2411                     dseq.start(envh.destin_agth.agth[addr].seqr);
2412                 join
2413             end
2414
2415         phase.drop_objection(this);
2416     endtask
2417
2418
2419
2420 endclass
2421
2422
2423 // packaGES files
2424
2425
2426 package router_pkg;
2427
2428 // including the all ruoter file
2429
2430 import uvm_pkg ::*;
2431
2432 `include "uvm_macros.svh"
2433
2434 `include "destin_agent_config.sv"
2435 `include "source_agent_config.sv"
2436 `include "router_env_config.sv"
2437
2438

```

```

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
2488 +incdir+../destin_agent_top
2489 SVTB2 = ../test/router_pkg.sv
2490 VSIMOPT= -vopt -voptargs=+acc
2491 VSIMCOV= -coverage -sva
2492 VSIMBATCH1= -c -do " log -r /* ;coverage save -onexit mem_cov1;run -all; exit"
2493 VSIMBATCH2= -c -do " log -r /* ;coverage save -onexit mem_cov2;run -all; exit"
2494 VSIMBATCH3= -c -do " log -r /* ;coverage save -onexit mem_cov3;run -all; exit"
2495 VSIMBATCH4= -c -do " log -r /* ;coverage save -onexit mem_cov4;run -all; exit"
2496
2497 help:
2498     @echo
2499     =====
2500     @echo "! USAGE          --  make target
2501     !"
2502     @echo "! clean          =>  clean the earlier log and intermediate
2503     files.                  !"
2504     @echo "! sv_cmp         =>  Create library and compile the
2505     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 -cvlg -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 -cvlg -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.fsd
2611
2612 report_VCS:
2613     urg -dir mem_cov1.vdb mem_cov2.vdb mem_cov3.vdb mem_cov4.vdb -dbname
2614     merged_dir/merged_test -format both -report urgReport
2615
2616 regress_VCS: clean_VCS sv_cmp_VCS run_test_VCS run_test1_VCS run_test2_VCS run_test3_VCS
2617 report_VCS
2618
2619 cov_VCS:
2620     verdi -cov -covdir merged_dir.vdb
2621
2622 clean_VCS:
2623     rm -rf simv* csrc* *.tmp *.vpd *.vdb *.key *.log *hdrs.h urgReport* *.fsdb novas*
2624     verdi*
2625     clear
2626
2627 # ----- END of Definitions for Synopsys's VCS Specific Targets
2628 -----#
2629
2630 */
2631
2632 // OUTPUT ::: FOR 3 TEST CASES Small,Medium,Large
2633
2634 // Small PACKETS
2635
2636 UVM_INFO @ 0: reporter [RNTST] Running test small_seq_test...
2637 UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
2638 -----
2639
2640 Name                                     Type                                     Size  Value
2641 -----
2642 uvm_test_top                           small_seq_test                           -    @474
2643     envh                               tb_env                                   -    @499
2644     destin_agth                         destin_agent_top                         -    @516
2645         agth[0]                         destin_agent                             -    @660
2646         drvh                             destin_driver                             -    @708
2647             rsp_port                     uvm_analysis_port                       -    @725
2648             seq_item_port                 uvm_seq_item_pull_port                   -    @716
2649     monh                               destin_monitor                           -    @691
2650         dmon_port                       uvm_analysis_port                       -    @699
2651     seqr                               destin_seqr                             -    @734
2652         rsp_export                       uvm_analysis_export                     -    @742
2653         seq_item_export                   uvm_seq_item_pull_imp                    -    @848
2654         arbitration_queue                 array                                    0    -
2655         lock_queue                       array                                    0    -
2656         num_last_reqs                     integral                                32    'd1
2657         num_last_rsps                     integral                                32    'd1
2658     agth[1]                             destin_agent                             -    @669
2659         drvh                             destin_driver                             -    @886
2660             rsp_port                     uvm_analysis_port                       -    @903
2661             seq_item_port                 uvm_seq_item_pull_port                   -    @894
2662     monh                               destin_monitor                           -    @869
2663         dmon_port                       uvm_analysis_port                       -    @877
2664     seqr                               destin_seqr                             -    @912
2665         rsp_export                       uvm_analysis_export                     -    @920
2666         seq_item_export                   uvm_seq_item_pull_imp                    -    @1026
2667         arbitration_queue                 array                                    0    -
2668         lock_queue                       array                                    0    -
2669         num_last_reqs                     integral                                32    'd1
2670         num_last_rsps                     integral                                32    'd1
2671     agth[2]                             destin_agent                             -    @678
2672         drvh                             destin_driver                             -    @1064
2673             rsp_port                     uvm_analysis_port                       -    @1081
2674             seq_item_port                 uvm_seq_item_pull_port                   -    @1072
2675     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

-----

Signal Error From Source Monitor = 1

Name	Type	Size	Value
source_mon	source_xtn	-	@1648
destin_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

```
2742     payload[5]      integral      8      207
2743     payload[6]      integral      8      173
2744     payload[7]      integral      8      158
2745     payload[8]      integral      8      'd91
2746     payload[9]      integral      8      'd0
2747     payload[10]     integral      8      220
2748     payload[11]     integral      8      230
2749     payload[12]     integral      8      'd113
2750     payload[13]     integral      8      'd40
2751     payload[14]     integral      8      130
2752     payload[15]     integral      8      'd93
2753     payload[16]     integral      8      'd55
2754     payload[17]     integral      8      'd18
2755     payload[18]     integral      8      'd78
2756     payload[19]     integral      8      188
2757     parity_byte     integral      8      'd102
```

```
2758 -----
2759 -----
2760 Name                Type                Size  Value
2761 -----
2762 destin_mon          destin_xtn    -    @1630
2763   des_address       integral      2    'd0
2764   pay_lenth         integral      6    'd20
2765   header_byte       integral      8    'd80
2766   payload[0]        integral      8    157
2767   payload[1]        integral      8    'd15
2768   payload[2]        integral      8    'd66
2769   payload[3]        integral      8    'd39
2770   payload[4]        integral      8    'd13
2771   payload[5]        integral      8    207
2772   payload[6]        integral      8    173
2773   payload[7]        integral      8    158
2774   payload[8]        integral      8    'd91
2775   payload[9]        integral      8    'd0
2776   payload[10]       integral      8    220
2777   payload[11]       integral      8    230
2778   payload[12]       integral      8    'd113
2779   payload[13]       integral      8    'd40
2780   payload[14]       integral      8    130
2781   payload[15]       integral      8    'd93
2782   payload[16]       integral      8    'd55
2783   payload[17]       integral      8    'd18
2784   payload[18]       integral      8    'd78
2785   payload[19]       integral      8    188
2786   parity_byte       integral      8    'd102
2787   delay             integral      6    'd0
2788 -----
```

```
2789
2790 ===== Header_Byte Matched SuccessFull =====
2791
2792
2793 ===== Payload Matched SuccessFull =====
2794
2795
2796 ===== Parity_Byte Matched SuccessFull =====
2797
2798
2799 =====
2800
2801
2802     SUCCESSFULLY MATCHED
2803
2804
2805 =====
2806
2807
2808 =====
2809
2810
```

2811 source side functional coverage = 38.889  
2812  
2813  
2814 destin side functional coverage = 41.667  
2815

2817  
2818  
2819 Signal Error From Source Monitor = 1  
2820

2821 Name Type Size Value  
2822  
2823 source\_mon source\_xtn - @1803  
2824 destin\_address integral 2 'd0  
2825 pay\_lenth integral 6 'd8  
2826 header\_byte integral 8 'd32  
2827 payload[0] integral 8 205  
2828 payload[1] integral 8 'd57  
2829 payload[2] integral 8 251  
2830 payload[3] integral 8 251  
2831 payload[4] integral 8 232  
2832 payload[5] integral 8 147  
2833 payload[6] integral 8 'd78  
2834 payload[7] integral 8 188  
2835 parity\_byte integral 8 'd93

2837  
2838 Name Type Size Value  
2839  
2840 destin\_mon destin\_xtn - @1807  
2841 des\_address integral 2 'd0  
2842 pay\_lenth integral 6 'd8  
2843 header\_byte integral 8 'd32  
2844 payload[0] integral 8 205  
2845 payload[1] integral 8 'd57  
2846 payload[2] integral 8 251  
2847 payload[3] integral 8 251  
2848 payload[4] integral 8 232  
2849 payload[5] integral 8 147  
2850 payload[6] integral 8 'd78  
2851 payload[7] integral 8 188  
2852 parity\_byte integral 8 'd93  
2853 delay integral 6 'd0

2855  
2856 ===== Header\_Byte Matched SuccessFull =====  
2857

2858  
2859 ===== Payload Matched SuccessFull =====  
2860

2861  
2862 ===== Parity\_Byte Matched SuccessFull =====  
2863

2864  
2865  
2866  
2867  
2868 SUCCESSFULLY MATCHED  
2869

2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877 source side functional coverage = 38.889  
2878  
2879

```

2880     destin    side functional coverage = 41.667
2881
2882
2883 =====
2884
2885 Signal Error From Source Monitor = 1
2886 -----
2887 Name                Type                Size  Value
2888 -----
2889 source_mon          source_xtn      -      @1826
2890   destin_address    integral        2      'd0
2891   pay_lenth         integral        6      'd2
2892   header_byte       integral        8      'd8
2893   payload[0]        integral        8      129
2894   payload[1]        integral        8      186
2895   parity_byte       integral        8      'd51
2896 -----
2897 -----
2898 Name                Type                Size  Value
2899 -----
2900 destin_mon          destin_xtn      -      @1830
2901   des_address       integral        2      'd0
2902   pay_lenth         integral        6      'd2
2903   header_byte       integral        8      'd8
2904   payload[0]        integral        8      129
2905   payload[1]        integral        8      186
2906   parity_byte       integral        8      'd51
2907   delay             integral        6      'd0
2908 -----
2909
2910 ===== Header_Byte Matched SuccessFull =====
2911
2912
2913 ===== Payload Matched SuccessFull =====
2914
2915
2916 ===== Parity_Byte Matched SuccessFull =====
2917
2918
2919 =====
2920
2921
2922     SUCCESSFULLY MATCHED
2923
2924
2925 =====
2926
2927
2928 =====
2929
2930
2931     source    side functional coverage = 38.889
2932
2933
2934     destin    side functional coverage = 41.667
2935
2936
2937 =====
2938
2939 Signal Error From Source Monitor = 1
2940 -----
2941 Name                Type                Size  Value
2942 -----
2943 source_mon          source_xtn      -      @1848
2944   destin_address    integral        2      'd0
2945   pay_lenth         integral        6      'd18
2946   header_byte       integral        8      'd72
2947   payload[0]        integral        8      203
2948   payload[1]        integral        8      191

```

```
2949     payload[2]      integral      8      'd123
2950     payload[3]      integral      8      182
2951     payload[4]      integral      8      174
2952     payload[5]      integral      8      158
2953     payload[6]      integral      8      'd92
2954     payload[7]      integral      8      138
2955     payload[8]      integral      8      'd116
2956     payload[9]      integral      8      'd90
2957     payload[10]     integral      8      'd95
2958     payload[11]     integral      8      159
2959     payload[12]     integral      8      190
2960     payload[13]     integral      8      189
2961     payload[14]     integral      8      'd66
2962     payload[15]     integral      8      'd101
2963     payload[16]     integral      8      'd6
2964     payload[17]     integral      8      136
2965     parity_byte     integral      8      'd83
```

```
2966     -----
2967     -----
2968     Name              Type              Size  Value
2969     -----
2970     destin_mon        destin_xtn  -    @1852
2971     des_address       integral     2     'd0
2972     pay_lenth         integral     6     'd18
2973     header_byte       integral     8     'd72
2974     payload[0]        integral     8     203
2975     payload[1]        integral     8     191
2976     payload[2]        integral     8     'd123
2977     payload[3]        integral     8     182
2978     payload[4]        integral     8     174
2979     payload[5]        integral     8     158
2980     payload[6]        integral     8     'd92
2981     payload[7]        integral     8     138
2982     payload[8]        integral     8     'd116
2983     payload[9]        integral     8     'd90
2984     payload[10]       integral     8     'd95
2985     payload[11]       integral     8     159
2986     payload[12]       integral     8     190
2987     payload[13]       integral     8     189
2988     payload[14]       integral     8     'd66
2989     payload[15]       integral     8     'd101
2990     payload[16]       integral     8     'd6
2991     payload[17]       integral     8     136
2992     parity_byte       integral     8     'd83
2993     delay             integral     6     'd0
2994     -----
```

```
2995
2996     ===== Header_Byte Matched SuccessFull =====
2997
```

```
2998
2999     ===== Payload Matched SuccessFull =====
3000
```

```
3001
3002     ===== Parity_Byte Matched SuccessFull =====
3003
```

```
3004
3005     =====
3006
```

```
3007
3008     SUCCESSFULLY MATCHED
3009
```

```
3010
3011     =====
3012
```

```
3013
3014     =====
3015
```

```
3016
3017     source side functional coverage = 38.889
```

```
3018
3019
3020     destin   side functional coverage = 41.667
3021
3022
3023 =====
3024
3025 Signal Error From Source Monitor = 1
3026 -----
3027 Name                Type                Size  Value
3028 -----
3029 source_mon          source_xtn      -    @1870
3030     destin_address  integral        2    'd0
3031     pay_lenth       integral        6    'd11
3032     header_byte     integral        8    'd44
3033     payload[0]      integral        8    128
3034     payload[1]      integral        8    'd61
3035     payload[2]      integral        8    'd12
3036     payload[3]      integral        8    'd90
3037     payload[4]      integral        8    193
3038     payload[5]      integral        8    'd56
3039     payload[6]      integral        8    131
3040     payload[7]      integral        8    255
3041     payload[8]      integral        8    229
3042     payload[9]      integral        8    139
3043     payload[10]     integral        8    'd14
3044     parity_byte     integral        8    'd34
3045 -----
3046 -----
3047 Name                Type                Size  Value
3048 -----
3049 destin_mon          destin_xtn      -    @1874
3050     des_address     integral        2    'd0
3051     pay_lenth       integral        6    'd11
3052     header_byte     integral        8    'd44
3053     payload[0]      integral        8    128
3054     payload[1]      integral        8    'd61
3055     payload[2]      integral        8    'd12
3056     payload[3]      integral        8    'd90
3057     payload[4]      integral        8    193
3058     payload[5]      integral        8    'd56
3059     payload[6]      integral        8    131
3060     payload[7]      integral        8    255
3061     payload[8]      integral        8    229
3062     payload[9]      integral        8    139
3063     payload[10]     integral        8    'd14
3064     parity_byte     integral        8    'd34
3065     delay           integral        6    'd0
3066 -----
3067
3068 ===== Header_Byte Matched SuccessFull =====
3069
3070
3071 ===== Payload Matched SuccessFull =====
3072
3073
3074 ===== Parity_Byte Matched SuccessFull =====
3075
3076
3077 =====
3078
3079     SUCCESSFULLY MATCHED
3080
3081
3082
3083 =====
3084
3085
3086 =====
```

```

3087
3088
3089     source    side functional coverage = 38.889
3090
3091
3092     destin    side functional coverage = 41.667
3093
3094
3095 =====
3096
3097 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
3098
3099 --- UVM Report Summary ---
3100
3101 ** Report counts by severity
3102 UVM_INFO :      3
3103 UVM_WARNING :    0
3104 UVM_ERROR :    0
3105 UVM_FATAL :    0
3106 ** Report counts by id
3107 [RNTST]      1
3108 [TEST_DONE]  1
3109 [UVMTOP]     1
3110 $finish called from file
"/home/cad/eda/SYNOPSYS/VCS/vcs/T-2022.06-SP1/etc/uvm/base/uvm_root.svh", line 437.
3111 $finish at simulation time 3650000
3112         V C S   S i m u l a t i o n   R e p o r t
3113 Time: 3650000 ps
3114
3115
3116 // Medium PACKETS
3117
3118 UVM_INFO @ 0: reporter [RNTST] Running test medium_seq_test...
3119 UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
3120 -----
3121 Name                                     Type                                     Size  Value
3122 -----
3123 uvm_test_top                           medium_seq_test                           -    @474
3124     envh                               tb_env                                   -    @499
3125     destin_agth                         destin_agent_top                         -    @516
3126         agth[0]                         destin_agent                             -    @660
3127             drvh                         destin_driver                             -    @708
3128                 rsp_port                 uvm_analysis_port                       -    @725
3129                     seq_item_port         uvm_seq_item_pull_port                   -    @716
3130     monh                               destin_monitor                           -    @691
3131         dmon_port                       uvm_analysis_port                       -    @699
3132             seqr                         destin_seqr                             -    @734
3133                 rsp_export               uvm_analysis_export                     -    @742
3134                     seq_item_export       uvm_seq_item_pull_imp                   -    @848
3135             arbitration_queue            array                                   0    -
3136                 lock_queue               array                                   0    -
3137                     num_last_reqs         integral                               32    'd1
3138                         num_last_rsps     integral                               32    'd1
3139         agth[1]                         destin_agent                             -    @669
3140             drvh                         destin_driver                             -    @886
3141                 rsp_port                 uvm_analysis_port                       -    @903
3142                     seq_item_port         uvm_seq_item_pull_port                   -    @894
3143     monh                               destin_monitor                           -    @869
3144         dmon_port                       uvm_analysis_port                       -    @877
3145             seqr                         destin_seqr                             -    @912
3146                 rsp_export               uvm_analysis_export                     -    @920
3147                     seq_item_export       uvm_seq_item_pull_imp                   -    @1026
3148             arbitration_queue            array                                   0    -
3149                 lock_queue               array                                   0    -
3150                     num_last_reqs         integral                               32    'd1
3151                         num_last_rsps     integral                               32    'd1
3152         agth[2]                         destin_agent                             -    @678
3153             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

-----

Signal Error From Source Monitor = 1

Name	Type	Size	Value
source_mon	source_xtn	-	@1648
destin_address	integral	2	'd1
pay_lenth	integral	6	'd27
header_byte	integral	8	'd109
payload[0]	integral	8	195
payload[1]	integral	8	218

3223	payload[2]	integral	8	227
3224	payload[3]	integral	8	'd122
3225	payload[4]	integral	8	'd64
3226	payload[5]	integral	8	191
3227	payload[6]	integral	8	'd24
3228	payload[7]	integral	8	138
3229	payload[8]	integral	8	'd22
3230	payload[9]	integral	8	235
3231	payload[10]	integral	8	139
3232	payload[11]	integral	8	'd42
3233	payload[12]	integral	8	197
3234	payload[13]	integral	8	'd44
3235	payload[14]	integral	8	249
3236	payload[15]	integral	8	141
3237	payload[16]	integral	8	198
3238	payload[17]	integral	8	220
3239	payload[18]	integral	8	'd93
3240	payload[19]	integral	8	224
3241	payload[20]	integral	8	235
3242	payload[21]	integral	8	'd45
3243	payload[22]	integral	8	'd42
3244	payload[23]	integral	8	214
3245	payload[24]	integral	8	149
3246	payload[25]	integral	8	196
3247	payload[26]	integral	8	'd31
3248	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

```

```
3361 pay_lenth    integral    6      'd29
3362 header_byte  integral    8      'd117
3363 payload[0]    integral    8      'd107
3364 payload[1]    integral    8      'd22
3365 payload[2]    integral    8      250
3366 payload[3]    integral    8      133
3367 payload[4]    integral    8      'd106
3368 payload[5]    integral    8      'd88
3369 payload[6]    integral    8      'd61
3370 payload[7]    integral    8      'd16
3371 payload[8]    integral    8      194
3372 payload[9]    integral    8      'd56
3373 payload[10]   integral    8      237
3374 payload[11]   integral    8      'd37
3375 payload[12]   integral    8      156
3376 payload[13]   integral    8      'd34
3377 payload[14]   integral    8      194
3378 payload[15]   integral    8      210
3379 payload[16]   integral    8      'd115
3380 payload[17]   integral    8      152
3381 payload[18]   integral    8      242
3382 payload[19]   integral    8      'd105
3383 payload[20]   integral    8      237
3384 payload[21]   integral    8      246
3385 payload[22]   integral    8      175
3386 payload[23]   integral    8      'd14
3387 payload[24]   integral    8      249
3388 payload[25]   integral    8      'd44
3389 payload[26]   integral    8      'd104
3390 payload[27]   integral    8      186
3391 payload[28]   integral    8      162
3392 parity_byte   integral    8      155
3393 delay         integral    6      'd0
3394 -----
3395
3396 ===== Header_Byte Matched SuccessFull =====
3397
3398
3399 ===== Payload Matched SuccessFull =====
3400
3401
3402 ===== Parity_Byte Matched SuccessFull =====
3403
3404
3405 =====
3406
3407
3408             SUCCESSFULLY MATCHED
3409
3410
3411 =====
3412
3413
3414 =====
3415
3416
3417     source   side functional coverage = 38.889
3418
3419
3420     destin   side functional coverage = 41.667
3421
3422
3423 =====
3424
3425 Signal Error From Source Monitor = 1
3426 -----
3427 Name                Type                Size  Value
3428 -----
3429 source_mon          source_xtn    -      @1826
```

```

3430     destin_address    integral    2      'd1
3431     pay_lenth         integral    6      'd25
3432     header_byte      integral    8      'd101
3433     payload[0]        integral    8      'd76
3434     payload[1]        integral    8      238
3435     payload[2]        integral    8      216
3436     payload[3]        integral    8      'd123
3437     payload[4]        integral    8      'd86
3438     payload[5]        integral    8      'd7
3439     payload[6]        integral    8      243
3440     payload[7]        integral    8      220
3441     payload[8]        integral    8      130
3442     payload[9]        integral    8      156
3443     payload[10]       integral    8      142
3444     payload[11]       integral    8      132
3445     payload[12]       integral    8      150
3446     payload[13]       integral    8      'd13
3447     payload[14]       integral    8      142
3448     payload[15]       integral    8      195
3449     payload[16]       integral    8      'd5
3450     payload[17]       integral    8      239
3451     payload[18]       integral    8      'd119
3452     payload[19]       integral    8      159
3453     payload[20]       integral    8      'd69
3454     payload[21]       integral    8      162
3455     payload[22]       integral    8      235
3456     payload[23]       integral    8      'd74
3457     payload[24]       integral    8      202
3458     parity_byte      integral    8      'd86

```

```

3459     -----
3460     -----
3461     Name                Type                Size  Value
3462     -----
3463     destin_mon          destin_xtn    -    @1830
3464     des_address         integral    2      'd1
3465     pay_lenth           integral    6      'd25
3466     header_byte        integral    8      'd101
3467     payload[0]          integral    8      'd76
3468     payload[1]          integral    8      238
3469     payload[2]          integral    8      216
3470     payload[3]          integral    8      'd123
3471     payload[4]          integral    8      'd86
3472     payload[5]          integral    8      'd7
3473     payload[6]          integral    8      243
3474     payload[7]          integral    8      220
3475     payload[8]          integral    8      130
3476     payload[9]          integral    8      156
3477     payload[10]         integral    8      142
3478     payload[11]         integral    8      132
3479     payload[12]         integral    8      150
3480     payload[13]         integral    8      'd13
3481     payload[14]         integral    8      142
3482     payload[15]         integral    8      195
3483     payload[16]         integral    8      'd5
3484     payload[17]         integral    8      239
3485     payload[18]         integral    8      'd119
3486     payload[19]         integral    8      159
3487     payload[20]         integral    8      'd69
3488     payload[21]         integral    8      162
3489     payload[22]         integral    8      235
3490     payload[23]         integral    8      'd74
3491     payload[24]         integral    8      202
3492     parity_byte         integral    8      'd86
3493     delay               integral    6      'd0
3494     -----

```

```

3495
3496     ===== Header_Byte Matched SuccessFull =====
3497
3498

```

```

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

```

```
3568     payload[1]    integral    8      215
3569     payload[2]    integral    8      185
3570     payload[3]    integral    8      'd37
3571     payload[4]    integral    8      229
3572     payload[5]    integral    8      'd89
3573     payload[6]    integral    8      148
3574     payload[7]    integral    8      'd118
3575     payload[8]    integral    8      151
3576     payload[9]    integral    8      182
3577     payload[10]   integral    8      'd76
3578     payload[11]   integral    8      'd54
3579     payload[12]   integral    8      'd48
3580     payload[13]   integral    8      177
3581     payload[14]   integral    8      'd26
3582     payload[15]   integral    8      211
3583     payload[16]   integral    8      'd39
3584     payload[17]   integral    8      'd43
3585     payload[18]   integral    8      'd26
3586     payload[19]   integral    8      'd5
3587     payload[20]   integral    8      'd17
3588     payload[21]   integral    8      181
3589     payload[22]   integral    8      'd99
3590     payload[23]   integral    8      196
3591     payload[24]   integral    8      244
3592     parity_byte   integral    8      'd74
3593     delay          integral    6      'd0
3594     -----
3595
3596     ===== Header_Byte Matched SuccessFull =====
3597
3598
3599     ===== Payload Matched SuccessFull =====
3600
3601
3602     ===== Parity_Byte Matched SuccessFull =====
3603
3604
3605     =====
3606
3607
3608     SUCCESSFULLY MATCHED
3609
3610
3611     =====
3612
3613
3614     =====
3615
3616
3617     source   side functional coverage = 38.889
3618
3619
3620     destin   side functional coverage = 41.667
3621
3622
3623     =====
3624
3625     Signal Error From Source Monitor = 1
3626     -----
3627     Name              Type              Size  Value
3628     -----
3629     source_mon        source_xtn    -      @1870
3630     destin_address    integral      2      'd1
3631     pay_lenth         integral      6      'd22
3632     header_byte       integral      8      'd89
3633     payload[0]        integral      8      254
3634     payload[1]        integral      8      184
3635     payload[2]        integral      8      226
3636     payload[3]        integral      8      'd56
```

```
3637     payload[4]      integral      8      254
3638     payload[5]      integral      8      'd69
3639     payload[6]      integral      8      'd109
3640     payload[7]      integral      8      232
3641     payload[8]      integral      8      232
3642     payload[9]      integral      8      'd57
3643     payload[10]     integral      8      139
3644     payload[11]     integral      8      185
3645     payload[12]     integral      8      230
3646     payload[13]     integral      8      175
3647     payload[14]     integral      8      'd85
3648     payload[15]     integral      8      'd123
3649     payload[16]     integral      8      240
3650     payload[17]     integral      8      166
3651     payload[18]     integral      8      'd104
3652     payload[19]     integral      8      167
3653     payload[20]     integral      8      240
3654     payload[21]     integral      8      'd27
3655     parity_byte     integral      8      'd13
```

```
3656 -----
3657 -----
3658 Name                Type                Size  Value
3659 -----
3660 destin_mon          destin_xtn    -    @1874
3661   des_address       integral      2    'd1
3662   pay_lenth         integral      6    'd22
3663   header_byte       integral      8    'd89
3664   payload[0]        integral      8    254
3665   payload[1]        integral      8    184
3666   payload[2]        integral      8    226
3667   payload[3]        integral      8    'd56
3668   payload[4]        integral      8    254
3669   payload[5]        integral      8    'd69
3670   payload[6]        integral      8    'd109
3671   payload[7]        integral      8    232
3672   payload[8]        integral      8    232
3673   payload[9]        integral      8    'd57
3674   payload[10]       integral      8    139
3675   payload[11]       integral      8    185
3676   payload[12]       integral      8    230
3677   payload[13]       integral      8    175
3678   payload[14]       integral      8    'd85
3679   payload[15]       integral      8    'd123
3680   payload[16]       integral      8    240
3681   payload[17]       integral      8    166
3682   payload[18]       integral      8    'd104
3683   payload[19]       integral      8    167
3684   payload[20]       integral      8    240
3685   payload[21]       integral      8    'd27
3686   parity_byte       integral      8    'd13
3687   delay             integral      6    'd0
```

```
3688 -----
3689
3690 ===== Header_Byte Matched SuccessFull =====
3691
3692
3693 ===== Payload Matched SuccessFull =====
3694
3695
3696 ===== Parity_Byte Matched SuccessFull =====
3697
3698
3699 =====
3700
3701
3702     SUCCESSFULLY MATCHED
3703
3704
3705 =====
```



```

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 -

```

3773	lock_queue	array	0	-
3774	num_last_reqs	integral	32	'd1
3775	num_last_rsps	integral	32	'd1
3776	agth[2]	destin_agent	-	@678
3777	drvh	destin_driver	-	@1064
3778	rsp_port	uvm_analysis_port	-	@1081
3779	seq_item_port	uvm_seq_item_pull_port	-	@1072
3780	monh	destin_monitor	-	@1047
3781	dmon_port	uvm_analysis_port	-	@1055
3782	seqr	destin_seqr	-	@1090
3783	rsp_export	uvm_analysis_export	-	@1098
3784	seq_item_export	uvm_seq_item_pull_imp	-	@1204
3785	arbitration_queue	array	0	-
3786	lock_queue	array	0	-
3787	num_last_reqs	integral	32	'd1
3788	num_last_rsps	integral	32	'd1
3789	sb	tb_scoreboard	-	@524
3790	destin_fifo[0]	uvm_tlm_analysis_fifo #(T)	-	@1274
3791	analysis_export	uvm_analysis_imp	-	@1318
3792	get_ap	uvm_analysis_port	-	@1309
3793	get_peek_export	uvm_get_peek_imp	-	@1291
3794	put_ap	uvm_analysis_port	-	@1300
3795	put_export	uvm_put_imp	-	@1282
3796	destin_fifo[1]	uvm_tlm_analysis_fifo #(T)	-	@1327
3797	analysis_export	uvm_analysis_imp	-	@1371
3798	get_ap	uvm_analysis_port	-	@1362
3799	get_peek_export	uvm_get_peek_imp	-	@1344
3800	put_ap	uvm_analysis_port	-	@1353
3801	put_export	uvm_put_imp	-	@1335
3802	destin_fifo[2]	uvm_tlm_analysis_fifo #(T)	-	@1380
3803	analysis_export	uvm_analysis_imp	-	@1424
3804	get_ap	uvm_analysis_port	-	@1415
3805	get_peek_export	uvm_get_peek_imp	-	@1397
3806	put_ap	uvm_analysis_port	-	@1406
3807	put_export	uvm_put_imp	-	@1388
3808	source_fifo	uvm_tlm_analysis_fifo #(T)	-	@1221
3809	analysis_export	uvm_analysis_imp	-	@1265
3810	get_ap	uvm_analysis_port	-	@1256
3811	get_peek_export	uvm_get_peek_imp	-	@1238
3812	put_ap	uvm_analysis_port	-	@1247
3813	put_export	uvm_put_imp	-	@1229
3814	source_agth	source_agent_top	-	@508
3815	agth[0]	source_agent	-	@1438
3816	drvh	source_driver	-	@1468
3817	rsp_port	uvm_analysis_port	-	@1485
3818	seq_item_port	uvm_seq_item_pull_port	-	@1476
3819	monh	source_monitor	-	@1451
3820	smon_port	uvm_analysis_port	-	@1459
3821	seqr	source_seqr	-	@1494
3822	rsp_export	uvm_analysis_export	-	@1502
3823	seq_item_export	uvm_seq_item_pull_imp	-	@1608
3824	arbitration_queue	array	0	-
3825	lock_queue	array	0	-
3826	num_last_reqs	integral	32	'd1
3827	num_last_rsps	integral	32	'd1
3828	v_seqr	virtual_seqr	-	@532
3829	rsp_export	uvm_analysis_export	-	@540
3830	seq_item_export	uvm_seq_item_pull_imp	-	@646
3831	arbitration_queue	array	0	-
3832	lock_queue	array	0	-
3833	num_last_reqs	integral	32	'd1
3834	num_last_rsps	integral	32	'd1

---

```

3835
3836
3837 Signal Error From Source Monitor = 1
3838
3839 Name                Type                Size  Value
3840 -----
3841 source_mon           source_xtn          -      @1648

```

3842	destin_address	integral	2	2
3843	pay_lenth	integral	6	57
3844	header_byte	integral	8	230
3845	payload[0]	integral	8	'd16
3846	payload[1]	integral	8	161
3847	payload[2]	integral	8	'd50
3848	payload[3]	integral	8	245
3849	payload[4]	integral	8	'd74
3850	payload[5]	integral	8	'd71
3851	payload[6]	integral	8	202
3852	payload[7]	integral	8	'd23
3853	payload[8]	integral	8	'd73
3854	payload[9]	integral	8	241
3855	payload[10]	integral	8	'd80
3856	payload[11]	integral	8	'd8
3857	payload[12]	integral	8	'd79
3858	payload[13]	integral	8	199
3859	payload[14]	integral	8	'd54
3860	payload[15]	integral	8	'd103
3861	payload[16]	integral	8	192
3862	payload[17]	integral	8	138
3863	payload[18]	integral	8	'd125
3864	payload[19]	integral	8	'd104
3865	payload[20]	integral	8	157
3866	payload[21]	integral	8	'd86
3867	payload[22]	integral	8	'd112
3868	payload[23]	integral	8	'd11
3869	payload[24]	integral	8	'd118
3870	payload[25]	integral	8	133
3871	payload[26]	integral	8	'd24
3872	payload[27]	integral	8	171
3873	payload[28]	integral	8	133
3874	payload[29]	integral	8	'd120
3875	payload[30]	integral	8	'd9
3876	payload[31]	integral	8	'd0
3877	payload[32]	integral	8	'd83
3878	payload[33]	integral	8	210
3879	payload[34]	integral	8	224
3880	payload[35]	integral	8	204
3881	payload[36]	integral	8	'd14
3882	payload[37]	integral	8	'd106
3883	payload[38]	integral	8	'd122
3884	payload[39]	integral	8	'd5
3885	payload[40]	integral	8	'd119
3886	payload[41]	integral	8	'd116
3887	payload[42]	integral	8	'd23
3888	payload[43]	integral	8	'd43
3889	payload[44]	integral	8	'd91
3890	payload[45]	integral	8	'd60
3891	payload[46]	integral	8	'd91
3892	payload[47]	integral	8	252
3893	payload[48]	integral	8	'd42
3894	payload[49]	integral	8	'd94
3895	payload[50]	integral	8	154
3896	payload[51]	integral	8	'd71
3897	payload[52]	integral	8	'd67
3898	payload[53]	integral	8	251
3899	payload[54]	integral	8	'd75
3900	payload[55]	integral	8	214
3901	payload[56]	integral	8	'd117
3902	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
```

```
3980
3981 =====
3982
3983
3984     SUCCESSFULLY MATCHED
3985
3986
3987 =====
3988
3989
3990 =====
3991
3992
3993     source   side functional coverage = 38.889
3994
3995
3996     destin   side functional coverage = 41.667
3997
3998
3999 =====
4000
4001     Signal Error From Source Monitor = 1
4002     -----
4003     Name                Type                Size  Value
4004     -----
4005     source_mon          source_xtn    -      @1803
4006     dest_in_address     integral      2        2
4007     pay_lenth           integral      6        56
4008     header_byte         integral      8       226
4009     payload[0]          integral      8       167
4010     payload[1]          integral      8       212
4011     payload[2]          integral      8       211
4012     payload[3]          integral      8      'd71
4013     payload[4]          integral      8       227
4014     payload[5]          integral      8      'd91
4015     payload[6]          integral      8       188
4016     payload[7]          integral      8       207
4017     payload[8]          integral      8      'd47
4018     payload[9]          integral      8      'd0
4019     payload[10]         integral      8      'd117
4020     payload[11]         integral      8       148
4021     payload[12]         integral      8       206
4022     payload[13]         integral      8       190
4023     payload[14]         integral      8      'd39
4024     payload[15]         integral      8       173
4025     payload[16]         integral      8       206
4026     payload[17]         integral      8       205
4027     payload[18]         integral      8      'd76
4028     payload[19]         integral      8      'd58
4029     payload[20]         integral      8       143
4030     payload[21]         integral      8      'd20
4031     payload[22]         integral      8       128
4032     payload[23]         integral      8       195
4033     payload[24]         integral      8      'd50
4034     payload[25]         integral      8      'd79
4035     payload[26]         integral      8      'd29
4036     payload[27]         integral      8      'd31
4037     payload[28]         integral      8       236
4038     payload[29]         integral      8       137
4039     payload[30]         integral      8       245
4040     payload[31]         integral      8       185
4041     payload[32]         integral      8       152
4042     payload[33]         integral      8      'd110
4043     payload[34]         integral      8       239
4044     payload[35]         integral      8       205
4045     payload[36]         integral      8       234
4046     payload[37]         integral      8      'd69
4047     payload[38]         integral      8       171
4048     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

```

4118     payload[44]    integral      8      231
4119     payload[45]    integral      8      231
4120     payload[46]    integral      8      'd21
4121     payload[47]    integral      8      'd50
4122     payload[48]    integral      8      'd41
4123     payload[49]    integral      8      'd89
4124     payload[50]    integral      8      'd24
4125     payload[51]    integral      8      'd6
4126     payload[52]    integral      8      'd103
4127     payload[53]    integral      8      143
4128     payload[54]    integral      8      208
4129     payload[55]    integral      8      'd91
4130     parity_byte    integral      8      190
4131     delay           integral      6      'd0
4132     -----
4133
4134     ===== Header_Byte Matched SuccessFull =====
4135
4136
4137     ===== Payload Matched SuccessFull =====
4138
4139
4140     ===== Parity_Byte Matched SuccessFull =====
4141
4142
4143     =====
4144
4145
4146     SUCCESSFULLY MATCHED
4147
4148
4149     =====
4150
4151
4152     =====
4153
4154
4155     source   side functional coverage = 38.889
4156
4157
4158     destin   side functional coverage = 41.667
4159
4160
4161     =====
4162
4163     Signal Error From Source Monitor = 1
4164     -----
4165     Name                Type                Size  Value
4166     -----
4167     source_mon           source_xtn      -    @1826
4168     destin_address       integral        2      2
4169     pay_lenth            integral        6      44
4170     header_byte          integral        8      178
4171     payload[0]           integral        8      'd39
4172     payload[1]           integral        8      'd125
4173     payload[2]           integral        8      216
4174     payload[3]           integral        8      215
4175     payload[4]           integral        8      'd0
4176     payload[5]           integral        8      'd125
4177     payload[6]           integral        8      'd42
4178     payload[7]           integral        8      173
4179     payload[8]           integral        8      'd0
4180     payload[9]           integral        8      'd1
4181     payload[10]          integral        8      'd63
4182     payload[11]          integral        8      'd89
4183     payload[12]          integral        8      149
4184     payload[13]          integral        8      200
4185     payload[14]          integral        8      203
4186     payload[15]          integral        8      'd43

```

4187	payload[16]	integral	8	212
4188	payload[17]	integral	8	'd113
4189	payload[18]	integral	8	215
4190	payload[19]	integral	8	134
4191	payload[20]	integral	8	'd106
4192	payload[21]	integral	8	215
4193	payload[22]	integral	8	241
4194	payload[23]	integral	8	'd2
4195	payload[24]	integral	8	'd113
4196	payload[25]	integral	8	135
4197	payload[26]	integral	8	'd115
4198	payload[27]	integral	8	225
4199	payload[28]	integral	8	'd27
4200	payload[29]	integral	8	190
4201	payload[30]	integral	8	255
4202	payload[31]	integral	8	171
4203	payload[32]	integral	8	181
4204	payload[33]	integral	8	247
4205	payload[34]	integral	8	208
4206	payload[35]	integral	8	'd120
4207	payload[36]	integral	8	'd56
4208	payload[37]	integral	8	147
4209	payload[38]	integral	8	191
4210	payload[39]	integral	8	225
4211	payload[40]	integral	8	'd14
4212	payload[41]	integral	8	'd43
4213	payload[42]	integral	8	158
4214	payload[43]	integral	8	221
4215	parity_byte	integral	8	145

4216	-----			
4217	-----			
4218	Name	Type	Size	Value
4219	-----			
4220	destin_mon	destin_xtn	-	@1830
4221	des_address	integral	2	2
4222	pay_lenth	integral	6	44
4223	header_byte	integral	8	178
4224	payload[0]	integral	8	'd39
4225	payload[1]	integral	8	'd125
4226	payload[2]	integral	8	216
4227	payload[3]	integral	8	215
4228	payload[4]	integral	8	'd0
4229	payload[5]	integral	8	'd125
4230	payload[6]	integral	8	'd42
4231	payload[7]	integral	8	173
4232	payload[8]	integral	8	'd0
4233	payload[9]	integral	8	'd1
4234	payload[10]	integral	8	'd63
4235	payload[11]	integral	8	'd89
4236	payload[12]	integral	8	149
4237	payload[13]	integral	8	200
4238	payload[14]	integral	8	203
4239	payload[15]	integral	8	'd43
4240	payload[16]	integral	8	212
4241	payload[17]	integral	8	'd113
4242	payload[18]	integral	8	215
4243	payload[19]	integral	8	134
4244	payload[20]	integral	8	'd106
4245	payload[21]	integral	8	215
4246	payload[22]	integral	8	241
4247	payload[23]	integral	8	'd2
4248	payload[24]	integral	8	'd113
4249	payload[25]	integral	8	135
4250	payload[26]	integral	8	'd115
4251	payload[27]	integral	8	225
4252	payload[28]	integral	8	'd27
4253	payload[29]	integral	8	190
4254	payload[30]	integral	8	255
4255	payload[31]	integral	8	171



```

4256     payload[32]    integral      8      181
4257     payload[33]    integral      8      247
4258     payload[34]    integral      8      208
4259     payload[35]    integral      8      'd120
4260     payload[36]    integral      8      'd56
4261     payload[37]    integral      8      147
4262     payload[38]    integral      8      191
4263     payload[39]    integral      8      225
4264     payload[40]    integral      8      'd14
4265     payload[41]    integral      8      'd43
4266     payload[42]    integral      8      158
4267     payload[43]    integral      8      221
4268     parity_byte    integral      8      145
4269     delay           integral      6      'd0
4270     -----
4271
4272     ===== Header_Byte Matched SuccessFull =====
4273
4274
4275     ===== Payload Matched SuccessFull =====
4276
4277
4278     ===== Parity_Byte Matched SuccessFull =====
4279
4280
4281     =====
4282
4283
4284     SUCCESSFULLY MATCHED
4285
4286
4287     =====
4288
4289
4290     =====
4291
4292
4293     source   side functional coverage = 38.889
4294
4295
4296     destin   side functional coverage = 41.667
4297
4298
4299     =====
4300
4301     Signal Error From Source Monitor = 1
4302     -----
4303     Name                Type                Size  Value
4304     -----
4305     source_mon           source_xtn      -    @1848
4306     destin_address       integral        2      2
4307     pay_lenth            integral        6      61
4308     header_byte          integral        8      246
4309     payload[0]           integral        8      'd81
4310     payload[1]           integral        8      'd100
4311     payload[2]           integral        8      'd82
4312     payload[3]           integral        8      169
4313     payload[4]           integral        8      203
4314     payload[5]           integral        8      'd49
4315     payload[6]           integral        8      201
4316     payload[7]           integral        8      244
4317     payload[8]           integral        8      'd13
4318     payload[9]           integral        8      'd5
4319     payload[10]          integral        8      'd112
4320     payload[11]          integral        8      'd105
4321     payload[12]          integral        8      'd32
4322     payload[13]          integral        8      216
4323     payload[14]          integral        8      158
4324     payload[15]          integral        8      'd68

```

4325	payload[16]	integral	8	156
4326	payload[17]	integral	8	179
4327	payload[18]	integral	8	'd69
4328	payload[19]	integral	8	'd64
4329	payload[20]	integral	8	'd109
4330	payload[21]	integral	8	130
4331	payload[22]	integral	8	'd13
4332	payload[23]	integral	8	'd14
4333	payload[24]	integral	8	172
4334	payload[25]	integral	8	153
4335	payload[26]	integral	8	'd65
4336	payload[27]	integral	8	207
4337	payload[28]	integral	8	'd57
4338	payload[29]	integral	8	'd71
4339	payload[30]	integral	8	251
4340	payload[31]	integral	8	'd59
4341	payload[32]	integral	8	'd21
4342	payload[33]	integral	8	'd88
4343	payload[34]	integral	8	255
4344	payload[35]	integral	8	'd55
4345	payload[36]	integral	8	'd53
4346	payload[37]	integral	8	135
4347	payload[38]	integral	8	203
4348	payload[39]	integral	8	189
4349	payload[40]	integral	8	170
4350	payload[41]	integral	8	226
4351	payload[42]	integral	8	243
4352	payload[43]	integral	8	'd111
4353	payload[44]	integral	8	180
4354	payload[45]	integral	8	168
4355	payload[46]	integral	8	'd2
4356	payload[47]	integral	8	141
4357	payload[48]	integral	8	'd92
4358	payload[49]	integral	8	'd90
4359	payload[50]	integral	8	'd57
4360	payload[51]	integral	8	'd4
4361	payload[52]	integral	8	'd18
4362	payload[53]	integral	8	'd76
4363	payload[54]	integral	8	176
4364	payload[55]	integral	8	'd108
4365	payload[56]	integral	8	215
4366	payload[57]	integral	8	'd86
4367	payload[58]	integral	8	235
4368	payload[59]	integral	8	'd98
4369	payload[60]	integral	8	175
4370	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     =====
```

4463  
4464  
4465 source side functional coverage = 38.889  
4466  
4467  
4468 destin side functional coverage = 41.667  
4469  
4470

4471  
4472  
4473 Signal Error From Source Monitor = 1  
4474

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
```

```
-----
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
```