TEST CODE

```
////////////////////////////////////////////////////////////////////////
// Avinash Singh
// CSCE 616 Hardware Design Verification
// UIN : 435009714
////////////////////////////////////////////////////////////////////////

class multiport_parallel_random_test extends base_test;

   `uvm_component_utils(multiport_parallel_random_test)

   function new(string name, uvm_component parent);
      super.new(name, parent);
   endfunction : new

   function void build_phase(uvm_phase phase);
      uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
parallel_random_vsequence::type_id::get());
      super.build_phase(phase);
   endfunction : build_phase

   task run_phase(uvm_phase phase);
      super.run_phase(phase);
      `uvm_info(get_type_name(), "Starting multiport parallel random test", UVM_NONE)
   endtask : run_phase

endclass : multiport_parallel_random_test


/// short packet test
class short_packet_parallel_random_test extends base_test;

   `uvm_component_utils(short_packet_parallel_random_test)

   function new(string name, uvm_component parent);
      super.new(name, parent);
   endfunction : new

   function void build_phase(uvm_phase phase);
```

```
        uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
short_packet_vsequence::type_id::get());
        super.build_phase(phase);
    endfunction : build_phase

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        `uvm_info(get_type_name(), "Starting multiport parallel random test", UVM_NONE)
    endtask : run_phase

endclass : short_packet_parallel_random_test


//// long packet test
class long_packet_parallel_random_test extends base_test;

    `uvm_component_utils(long_packet_parallel_random_test)

    function new(string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new

    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
long_packet_vsequence::type_id::get());
        super.build_phase(phase);
    endfunction : build_phase

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        `uvm_info(get_type_name(), "Starting multiport parallel random test", UVM_NONE)
    endtask : run_phase

endclass : long_packet_parallel_random_test


/// same delay test
class same_delay_parallel_random_test extends base_test;

    `uvm_component_utils(same_delay_parallel_random_test)
    function new(string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new

    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
same_delay_vsequence::type_id::get());
        super.build_phase(phase);
    endfunction : build_phase

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        `uvm_info(get_type_name(), "Starting multiport parallel random test", UVM_NONE)
    endtask : run_phase

endclass : same_delay_parallel_random_test

/// same delay, same length, same port test
class same_delay_same_length_same_port_parallel_random_test extends base_test;

    `uvm_component_utils(same_delay_same_length_same_port_parallel_random_test)

    function new(string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new

    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
same_delay_same_length_same_port_vsequence::type_id::get());
        super.build_phase(phase);
    endfunction : build_phase

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        `uvm_info(get_type_name(), "Starting multiport parallel random test", UVM_NONE)
    endtask : run_phase

endclass : same_delay_same_length_same_port_parallel_random_test
```

```
/// same delay, same length, all port test
class same_delay_same_length_all_port_parallel_random_test extends base_test;

   `uvm_component_utils(same_delay_same_length_all_port_parallel_random_test)

   function new(string name, uvm_component parent);
      super.new(name, parent);
   endfunction : new

   function void build_phase(uvm_phase phase);
      uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
same_delay_same_length_all_port_vsequence::type_id::get());
      super.build_phase(phase);
   endfunction : build_phase

   task run_phase(uvm_phase phase);
      super.run_phase(phase);
      `uvm_info(get_type_name(), "Starting multiport parallel random test", UVM_NONE)
   endtask : run_phase

endclass : same_delay_same_length_all_port_parallel_random_test

/////////////////////////// VIRTUAL SEQUENCE /////////////////////////

class parallel_random_vsequence extends htax_base_vseq;

   `uvm_object_utils(parallel_random_vsequence)

   function new(string name = "parallel_random_vsequence");
      super.new(name);
   endfunction : new
   htax_packet_c req[0:3];
   task body();
      semaphore sem;
      sem = new(1);
    //repeat (10) begin
      fork
         begin
            //sem.get();
            for (int j=0;j<4;j++) begin
```

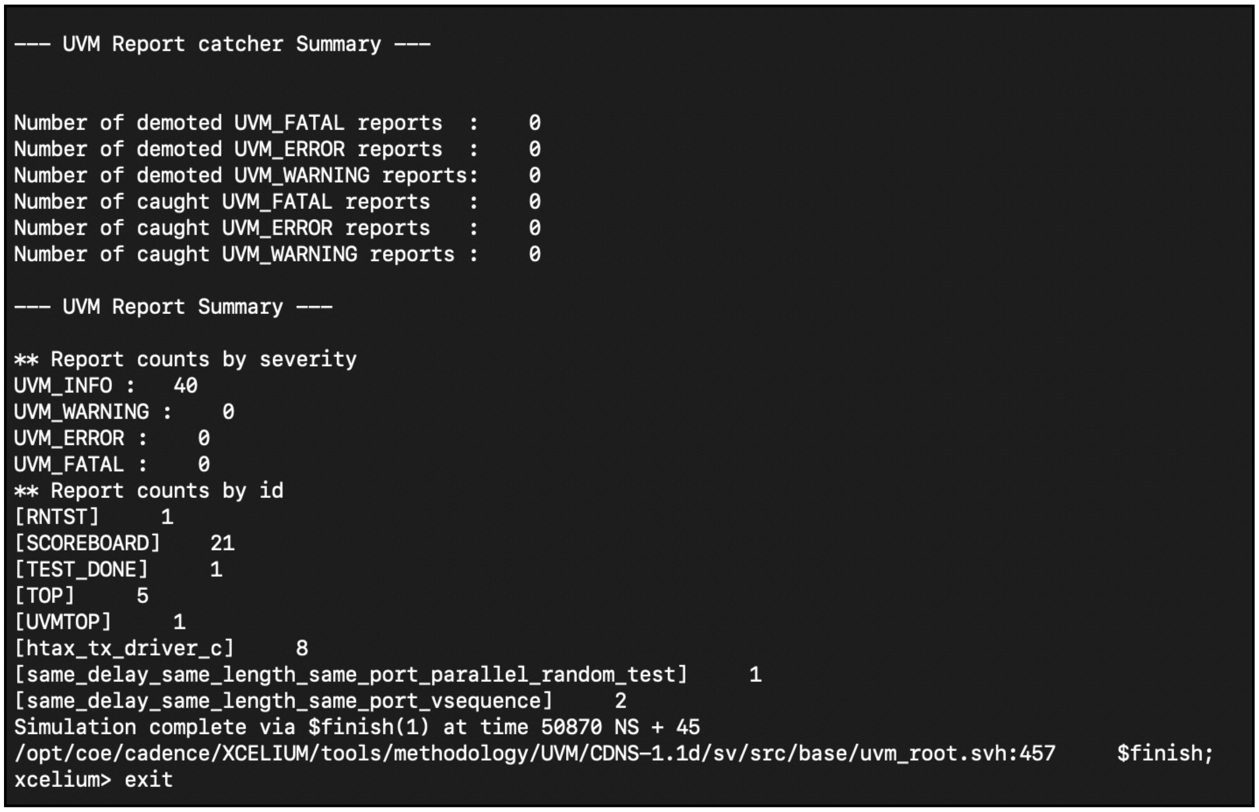
```
                              for(int i=3;i< 64 ;i++) begin
                        `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0],{length == i; dest_port ==
j; })
                  end
               end
               //sem.put();
            end
            begin
               //sem.get();
               for (int j=0;j<4;j++) begin
                              for(int i=3;i< 64 ;i++) begin
                        `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1],{length == i; dest_port ==
j;})
                  end
               end
               //sem.put();
            end
            begin
               //sem.get();
               for (int j=0;j<4;j++) begin
                              for(int i=3;i< 64 ;i++) begin
                        `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2],{length == i; dest_port ==
j;})
                  end
               end
               //sem.put();
            end
            begin
               //sem.get();
               for (int j=0;j<4;j++) begin
                              for(int i=3;i< 64 ;i++) begin
                        `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3],{length == i; dest_port ==
j;})
                  end
               end
               //sem.put();
            end
         join
      //end
endtask : body
```

```
endclass : parallel_random_vsequence

// short packet sequence

class short_packet_vsequence extends htax_base_vseq;

   `uvm_object_utils(short_packet_vsequence)

   function new(string name = "short_packet_vsequence");
      super.new(name);
   endfunction : new
   htax_packet_c req[0:3];
   task body();

      fork
         begin
            `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0],{length > 3; length < 20;})
         end
         begin
            `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1],{length > 3; length < 20;})
         end
         begin
            `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2],{length > 3; length < 20;})
         end
         begin
            `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3],{length > 3; length < 20;})
         end
      join
     //end
endtask : body


endclass : short_packet_vsequence


// long packet sequence
class long_packet_vsequence extends htax_base_vseq;
```

```
    `uvm_object_utils(long_packet_vsequence)

    function new(string name = "long_packet_vsequence");
       super.new(name);
    endfunction : new
    htax_packet_c req[0:3];
    task body();

       fork
          begin
             `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0],{length > 25; length < 60;})
          end
          begin
             `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1],{length > 25; length < 60;})
          end
          begin
             `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2],{length > 25; length < 60;})
          end
          begin
             `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3],{length > 25; length < 60;})
          end
       join
      //end
endtask : body


endclass : long_packet_vsequence


// short delay sequence

class same_delay_vsequence extends htax_base_vseq;
          `uvm_object_utils(same_delay_vsequence)
    function new(string name = "same_delay_vsequence");
       super.new(name);
    endfunction : new
    htax_packet_c req[0:3];
    task body();
      //repeat(10) begin
       fork
```

```
          begin
             `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0],{delay == 5;})
          end
          begin
             `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1],{delay == 5;})
          end
          begin
             `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2],{delay == 5;})
          end
          begin
             `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3],{delay == 5;})
          end
        join
      //end
    endtask : body
endclass : same_delay_vsequence

//same delay same length same port
class same_delay_same_length_same_port_vsequence extends htax_base_vseq;

    `uvm_object_utils(same_delay_same_length_same_port_vsequence)
   function new(string name = "same_delay_same_length_same_port_vsequence");
        super.new(name);
    endfunction : new
    htax_packet_c req[0:3];
    task body();
      //repeat(10) begin
        fork
          begin
             `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0],{length == 8;dest_port==0;delay
> 3; delay < 5;})
          end
          begin
             `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1],{length == 8;dest_port==0;delay
> 3; delay < 5;})
          end
          begin
             `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2],{length == 8;dest_port==0;delay
> 3; delay < 5;})
          end
          begin
             `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3],{length == 8;dest_port==0;delay
> 3; delay < 5;})
          end
       join
     //end
endtask : body
endclass : same_delay_same_length_same_port_vsequence

//same delay same length all port
class same_delay_same_length_all_port_vsequence extends htax_base_vseq;

   `uvm_object_utils(same_delay_same_length_all_port_vsequence)

   function new(string name = "same_delay_same_length_all_port_vsequence");
      super.new(name);
   endfunction : new
   htax_packet_c req[0:3];
   task body();
     //repeat(10) begin
       fork
          begin
             `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0],{length == 8;dest_port==0;delay
> 3; delay < 5;})
          end
          begin
             `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1],{length == 8;dest_port==1;delay
> 3; delay < 5;})
          end
          begin
             `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2],{length == 8;dest_port==2;delay
> 3; delay < 5;})
          end
          begin
             `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3],{length == 8;dest_port==3;delay
> 3; delay < 5;})
          end
       join
     //end
endtask : body
```

endclass: same_delay_same_length_all_port_vsequence



```
--- UVM Report catcher Summary ---


Number of demoted UVM_FATAL reports  :    0
Number of demoted UVM_ERROR reports  :    0
Number of demoted UVM_WARNING reports:    0
Number of caught UVM_FATAL reports   :    0
Number of caught UVM_ERROR reports   :    0
Number of caught UVM_WARNING reports :    0

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   40
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[SCOREBOARD]    21
[TEST_DONE]     1
[TOP]     5
[UVMTOP]     1
[htax_tx_driver_c]     8
[same_delay_same_length_same_port_parallel_random_test]     1
[same_delay_same_length_same_port_vsequence]     2
Simulation complete via $finish(1) at time 50870 NS + 45
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

All Testcase Summary

I performed a series of tests to identify and isolate the bug in the DUT (Design Under Test). Starting:-

Test7 (multiport_parallel_random_test), we verified the DUT's ability to handle parallel communication across multiple ports with random data. This test passed, indicating the DUT could manage multiple ports in parallel.

Next, **Test8** (short_packet_parallel_random_test) tested the DUT's performance with short data packets, which also passed, confirming efficient handling of small packets across multiple ports.

In **Test9** (long_packet_parallel_random_test), we assessed the DUT's capability to manage long packets in parallel across multiple ports. The DUT passed this test as well, demonstrating its ability to handle larger data sizes.

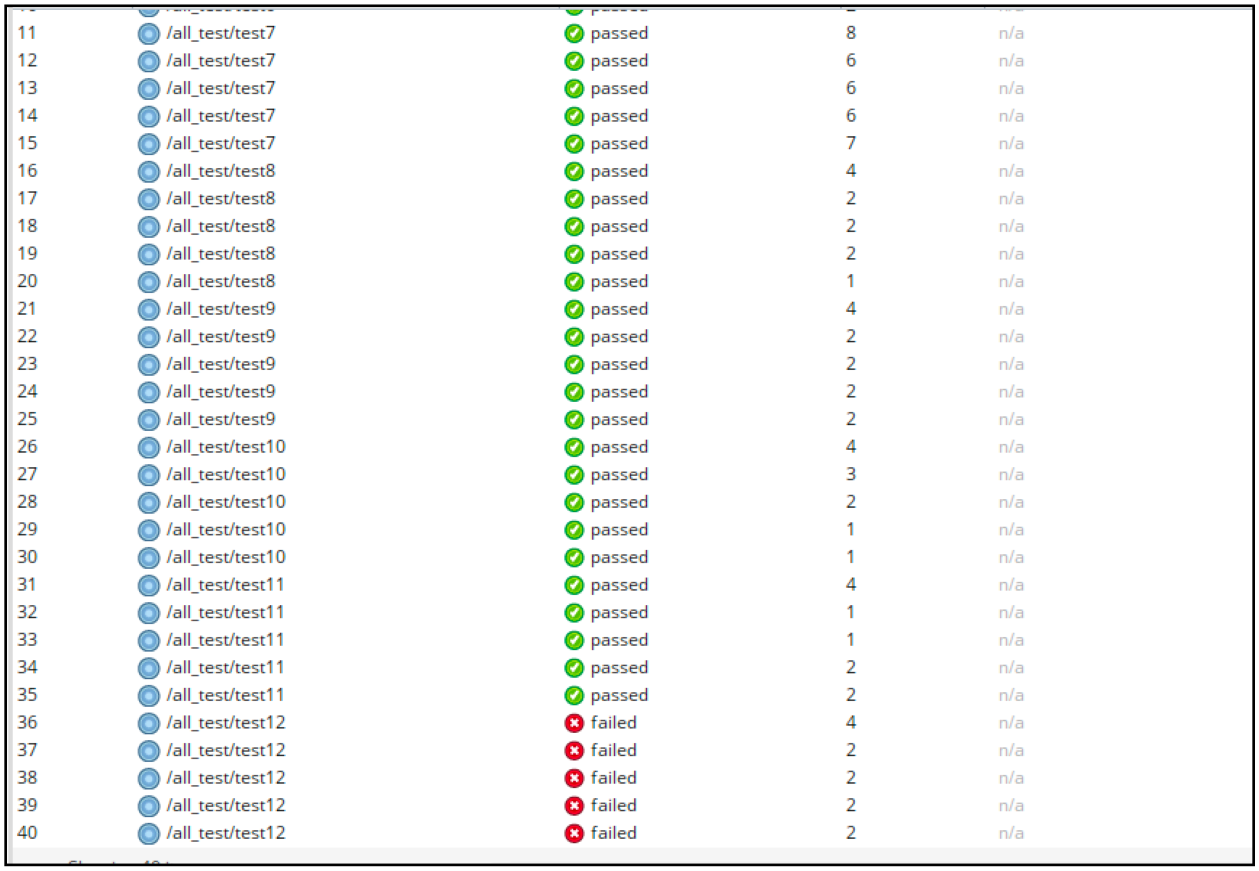
Then, **Test10** (same_delay_parallel_random_test) checked if the DUT could synchronize multiple ports with identical delays. The DUT successfully passed this test, ensuring proper timing synchronization.

Next, **Test11** (same_delay_same_length_same_port_parallel_random_test) evaluated the DUT's ability to handle multiple ports with the same delay, packet length, and configuration. The test passed, confirming the DUT's consistency across uniform conditions.

However,when we performed **Test12** (same_delay_same_length_all_port_parallel_random_test), which required the DUT to manage all ports simultaneously with identical delay and packet length conditions, the **DUT failed.**
Failure in Test12: This test revealed a bug in the DUT related to its inability to handle all ports in parallel under identical conditions. Debugging showed that the DUT struggled to maintain proper synchronization across all ports when subjected to the same delay and packet length conditions.

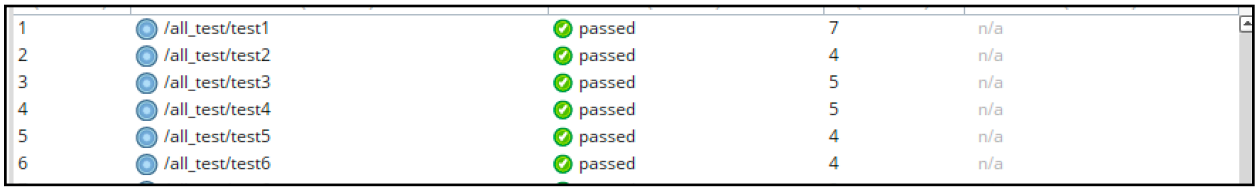
COVERAGE REPORT:

Failing Regression Test 12

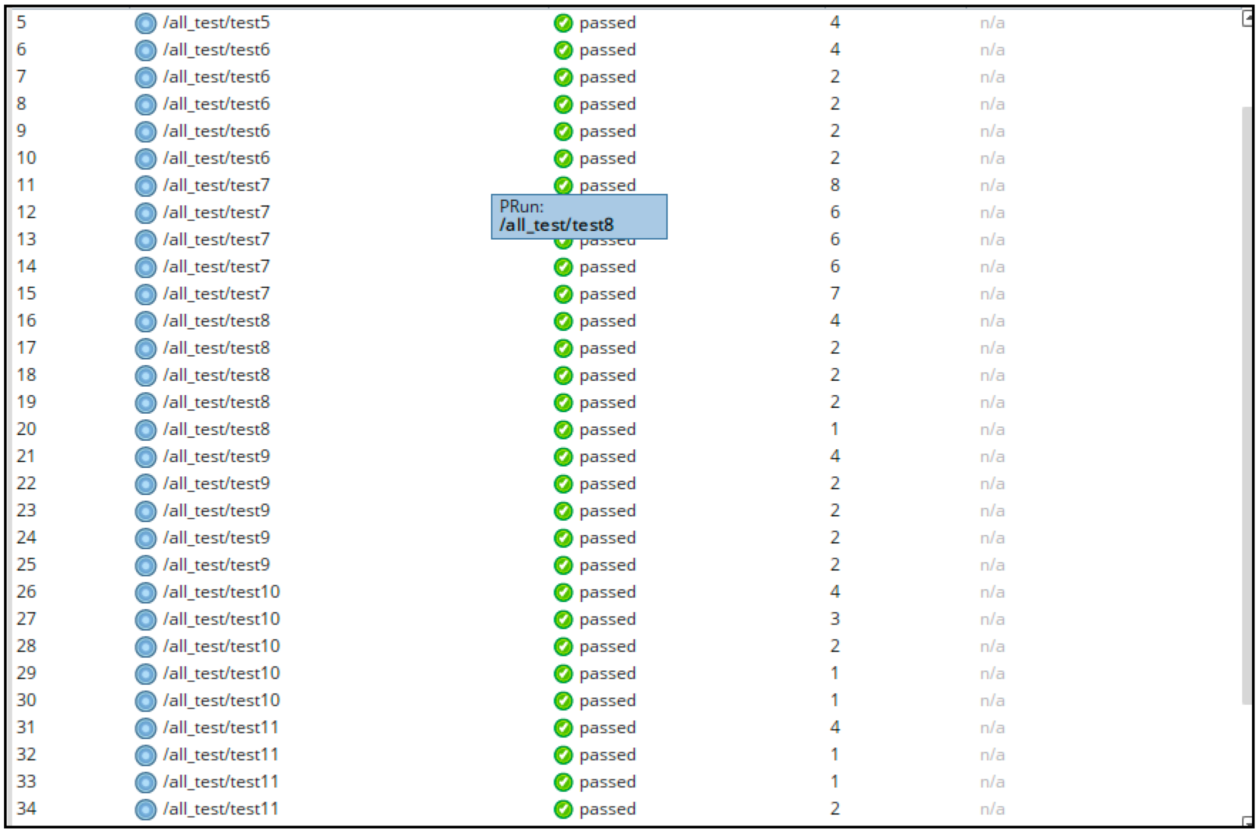


11	/all_test/test7	✓ passed	8	n/a
12	/all_test/test7	✓ passed	6	n/a
13	/all_test/test7	✓ passed	6	n/a
14	/all_test/test7	✓ passed	6	n/a
15	/all_test/test7	✓ passed	7	n/a
16	/all_test/test8	✓ passed	4	n/a
17	/all_test/test8	✓ passed	2	n/a
18	/all_test/test8	✓ passed	2	n/a
19	/all_test/test8	✓ passed	2	n/a
20	/all_test/test8	✓ passed	1	n/a
21	/all_test/test9	✓ passed	4	n/a
22	/all_test/test9	✓ passed	2	n/a
23	/all_test/test9	✓ passed	2	n/a
24	/all_test/test9	✓ passed	2	n/a
25	/all_test/test9	✓ passed	2	n/a
26	/all_test/test10	✓ passed	4	n/a
27	/all_test/test10	✓ passed	3	n/a
28	/all_test/test10	✓ passed	2	n/a
29	/all_test/test10	✓ passed	1	n/a
30	/all_test/test10	✓ passed	1	n/a
31	/all_test/test11	✓ passed	4	n/a
32	/all_test/test11	✓ passed	1	n/a
33	/all_test/test11	✓ passed	1	n/a
34	/all_test/test11	✓ passed	2	n/a
35	/all_test/test11	✓ passed	2	n/a
36	/all_test/test12	✗ failed	4	n/a
37	/all_test/test12	✗ failed	2	n/a
38	/all_test/test12	✗ failed	2	n/a
39	/all_test/test12	✗ failed	2	n/a
40	/all_test/test12	✗ failed	2	n/a

Passing Regression

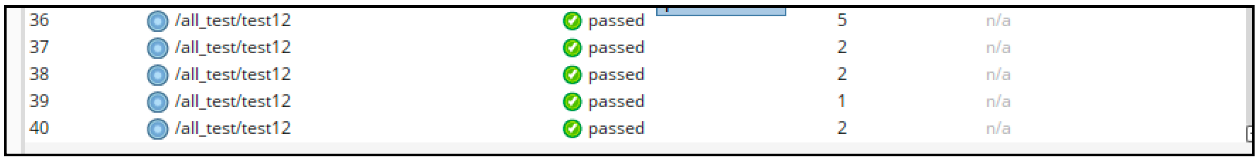


1	/all_test/test1	passed	7	n/a
2	/all_test/test2	passed	4	n/a
3	/all_test/test3	passed	5	n/a
4	/all_test/test4	passed	5	n/a
5	/all_test/test5	passed	4	n/a
6	/all_test/test6	passed	4	n/a



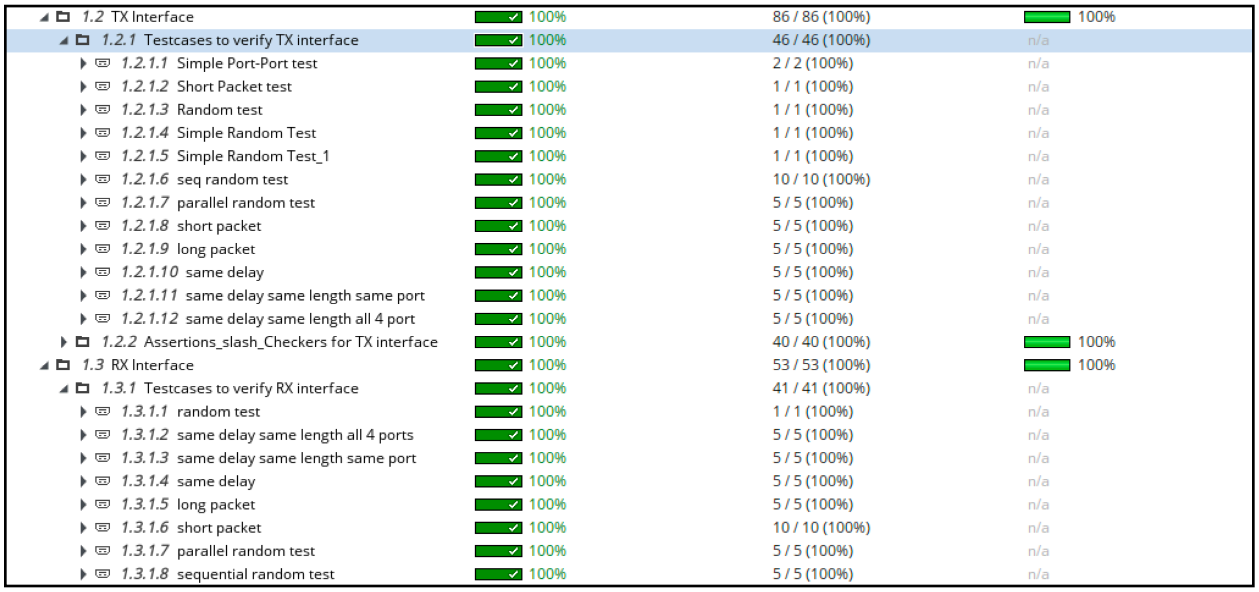
5	/all_test/test5	passed	4	n/a
6	/all_test/test6	passed	4	n/a
7	/all_test/test6	passed	2	n/a
8	/all_test/test6	passed	2	n/a
9	/all_test/test6	passed	2	n/a
10	/all_test/test6	passed	2	n/a
11	/all_test/test7	passed	8	n/a
12	/all_test/test7		6	n/a
13	/all_test/test7	passed	6	n/a
14	/all_test/test7	passed	6	n/a
15	/all_test/test7	passed	7	n/a
16	/all_test/test8	passed	4	n/a
17	/all_test/test8	passed	2	n/a
18	/all_test/test8	passed	2	n/a
19	/all_test/test8	passed	2	n/a
20	/all_test/test8	passed	1	n/a
21	/all_test/test9	passed	4	n/a
22	/all_test/test9	passed	2	n/a
23	/all_test/test9	passed	2	n/a
24	/all_test/test9	passed	2	n/a
25	/all_test/test9	passed	2	n/a
26	/all_test/test10	passed	4	n/a
27	/all_test/test10	passed	3	n/a
28	/all_test/test10	passed	2	n/a
29	/all_test/test10	passed	1	n/a
30	/all_test/test10	passed	1	n/a
31	/all_test/test11	passed	4	n/a
32	/all_test/test11	passed	1	n/a
33	/all_test/test11	passed	1	n/a
34	/all_test/test11	passed	2	n/a

Passing Test 12 After design fix

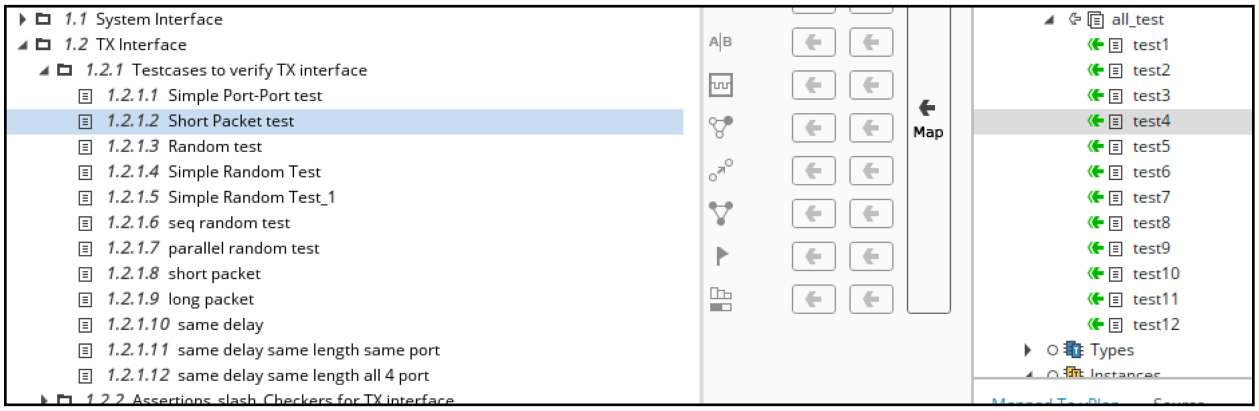


36	/all_test/test12	passed	5	n/a
37	/all_test/test12	passed	2	n/a
38	/all_test/test12	passed	2	n/a
39	/all_test/test12	passed	1	n/a
40	/all_test/test12	passed	2	n/a

Testcase Mapped

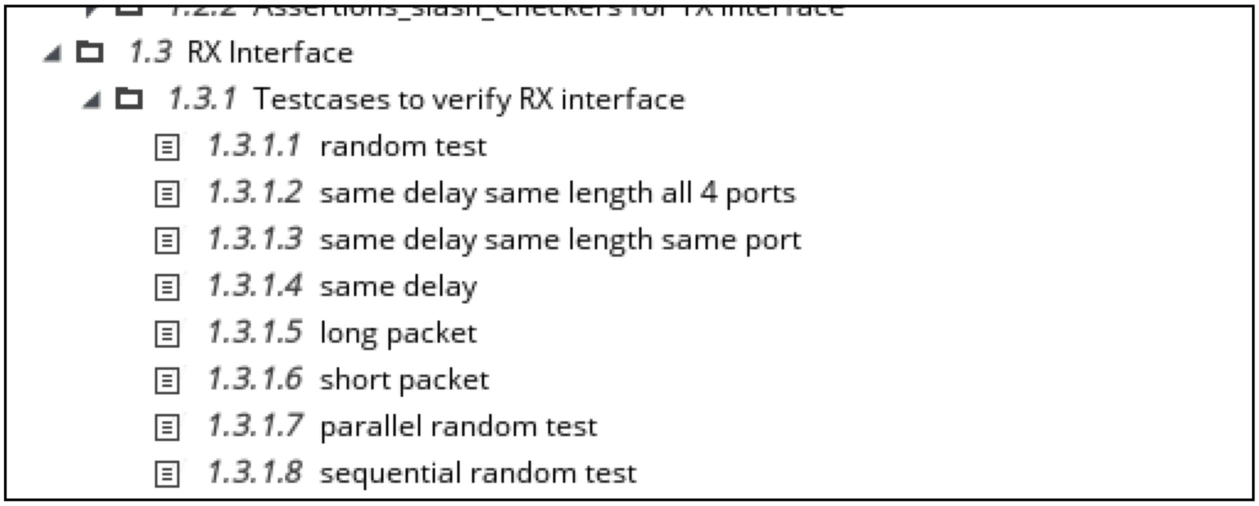


1.2 TX Interface	✓ 100%	86 / 86 (100%)	🟩 100%
1.2.1 Testcases to verify TX interface	✓ 100%	46 / 46 (100%)	n/a
1.2.1.1 Simple Port-Port test	✓ 100%	2 / 2 (100%)	n/a
1.2.1.2 Short Packet test	✓ 100%	1 / 1 (100%)	n/a
1.2.1.3 Random test	✓ 100%	1 / 1 (100%)	n/a
1.2.1.4 Simple Random Test	✓ 100%	1 / 1 (100%)	n/a
1.2.1.5 Simple Random Test_1	✓ 100%	1 / 1 (100%)	n/a
1.2.1.6 seq random test	✓ 100%	10 / 10 (100%)	n/a
1.2.1.7 parallel random test	✓ 100%	5 / 5 (100%)	n/a
1.2.1.8 short packet	✓ 100%	5 / 5 (100%)	n/a
1.2.1.9 long packet	✓ 100%	5 / 5 (100%)	n/a
1.2.1.10 same delay	✓ 100%	5 / 5 (100%)	n/a
1.2.1.11 same delay same length same port	✓ 100%	5 / 5 (100%)	n/a
1.2.1.12 same delay same length all 4 port	✓ 100%	5 / 5 (100%)	n/a
1.2.2 Assertions_slash_Checkers for TX interface	✓ 100%	40 / 40 (100%)	🟩 100%
1.3 RX Interface	✓ 100%	53 / 53 (100%)	🟩 100%
1.3.1 Testcases to verify RX interface	✓ 100%	41 / 41 (100%)	n/a
1.3.1.1 random test	✓ 100%	1 / 1 (100%)	n/a
1.3.1.2 same delay same length all 4 ports	✓ 100%	5 / 5 (100%)	n/a
1.3.1.3 same delay same length same port	✓ 100%	5 / 5 (100%)	n/a
1.3.1.4 same delay	✓ 100%	5 / 5 (100%)	n/a
1.3.1.5 long packet	✓ 100%	5 / 5 (100%)	n/a
1.3.1.6 short packet	✓ 100%	10 / 10 (100%)	n/a
1.3.1.7 parallel random test	✓ 100%	5 / 5 (100%)	n/a
1.3.1.8 sequential random test	✓ 100%	5 / 5 (100%)	n/a



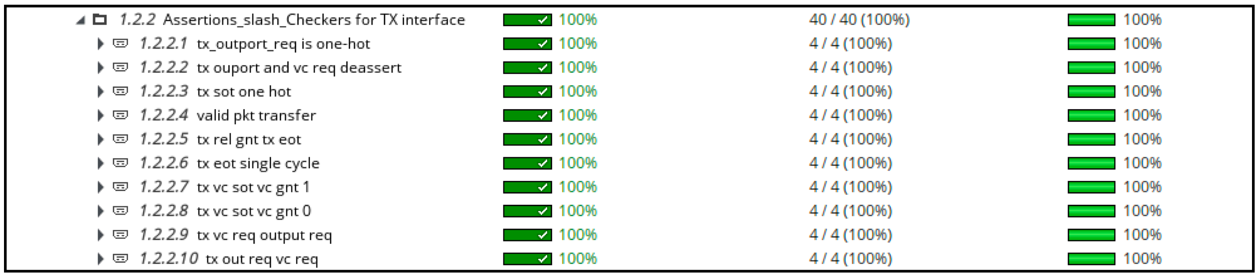
1.1 System Interface
1.2 TX Interface
1.2.1 Testcases to verify TX interface
1.2.1.1 Simple Port-Port test
1.2.1.2 Short Packet test
1.2.1.3 Random test
1.2.1.4 Simple Random Test
1.2.1.5 Simple Random Test_1
1.2.1.6 seq random test
1.2.1.7 parallel random test
1.2.1.8 short packet
1.2.1.9 long packet
1.2.1.10 same delay
1.2.1.11 same delay same length same port
1.2.1.12 same delay same length all 4 port
1.2.2 Assertions_slash_Checkers for TX interface

all_test
test1
test2
test3
test4
test5
test6
test7
test8
test9
test10
test11
test12
Types
Instances



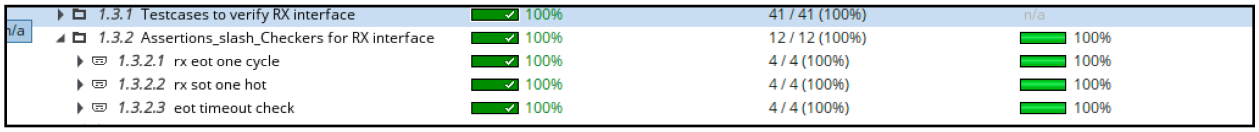
1.2.2 Assertions_slash_Checkers for TX interface
1.3 RX Interface
1.3.1 Testcases to verify RX interface
1.3.1.1 random test
1.3.1.2 same delay same length all 4 ports
1.3.1.3 same delay same length same port
1.3.1.4 same delay
1.3.1.5 long packet
1.3.1.6 short packet
1.3.1.7 parallel random test
1.3.1.8 sequential random test

TX Interface Assertions



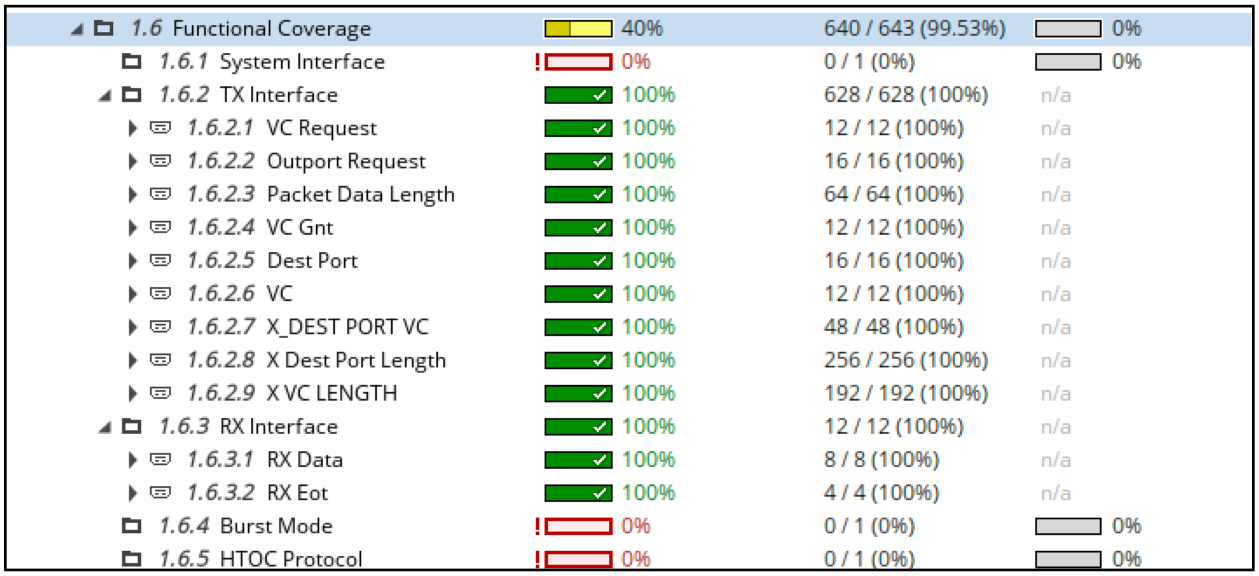
1.2.2 Assertions_slash_Checkers for TX interface	✓ 100%	40 / 40 (100%)	🟩 100%
1.2.2.1 tx_outport_req is one-hot	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.2 tx ouport and vc req deassert	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.3 tx sot one hot	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.4 valid pkt transfer	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.5 tx rel gnt tx eot	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.6 tx eot single cycle	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.7 tx vc sot vc gnt 1	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.8 tx vc sot vc gnt 0	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.9 tx vc req output req	✓ 100%	4 / 4 (100%)	🟩 100%
1.2.2.10 tx out req vc req	✓ 100%	4 / 4 (100%)	🟩 100%

Rx Interface Assertions



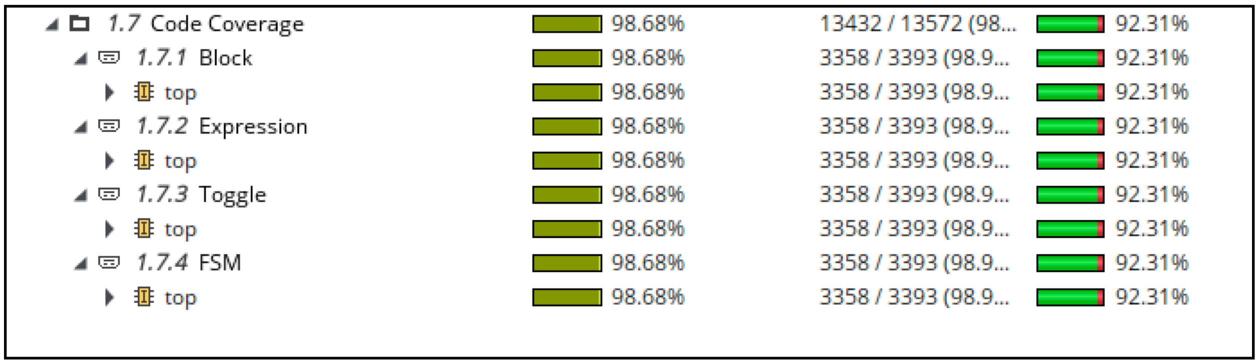
1.3.1 Testcases to verify RX interface	✓ 100%	41 / 41 (100%)	n/a
1.3.2 Assertions_slash_Checkers for RX interface	✓ 100%	12 / 12 (100%)	🟩 100%
1.3.2.1 rx eot one cycle	✓ 100%	4 / 4 (100%)	🟩 100%
1.3.2.2 rx sot one hot	✓ 100%	4 / 4 (100%)	🟩 100%
1.3.2.3 eot timeout check	✓ 100%	4 / 4 (100%)	🟩 100%

Functional Coverage for Tx and Rx Interface

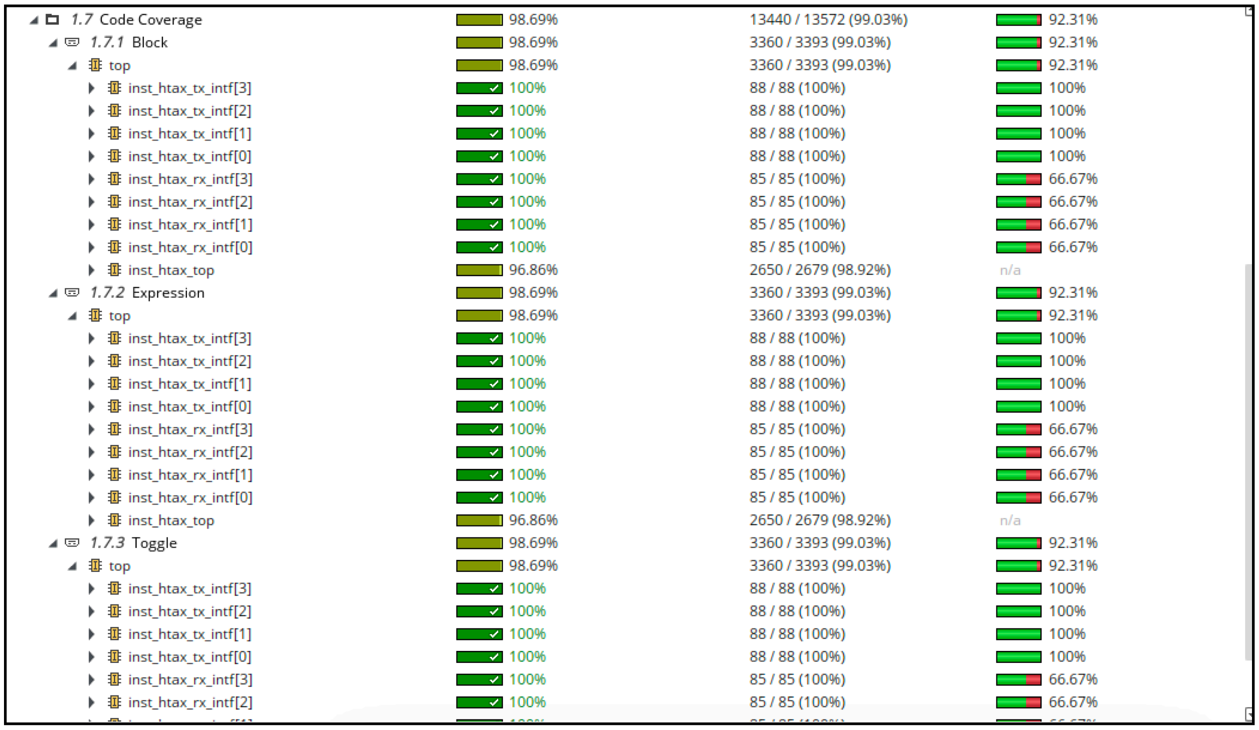


1.6 Functional Coverage	40%	640 / 643 (99.53%)	0%
1.6.1 System Interface	! 0%	0 / 1 (0%)	0%
1.6.2 TX Interface	✓ 100%	628 / 628 (100%)	n/a
1.6.2.1 VC Request	✓ 100%	12 / 12 (100%)	n/a
1.6.2.2 Outport Request	✓ 100%	16 / 16 (100%)	n/a
1.6.2.3 Packet Data Length	✓ 100%	64 / 64 (100%)	n/a
1.6.2.4 VC Gnt	✓ 100%	12 / 12 (100%)	n/a
1.6.2.5 Dest Port	✓ 100%	16 / 16 (100%)	n/a
1.6.2.6 VC	✓ 100%	12 / 12 (100%)	n/a
1.6.2.7 X_DEST PORT VC	✓ 100%	48 / 48 (100%)	n/a
1.6.2.8 X Dest Port Length	✓ 100%	256 / 256 (100%)	n/a
1.6.2.9 X VC LENGTH	✓ 100%	192 / 192 (100%)	n/a
1.6.3 RX Interface	✓ 100%	12 / 12 (100%)	n/a
1.6.3.1 RX Data	✓ 100%	8 / 8 (100%)	n/a
1.6.3.2 RX Eot	✓ 100%	4 / 4 (100%)	n/a
1.6.4 Burst Mode	! 0%	0 / 1 (0%)	0%
1.6.5 HTOC Protocol	! 0%	0 / 1 (0%)	0%

Code Coverage

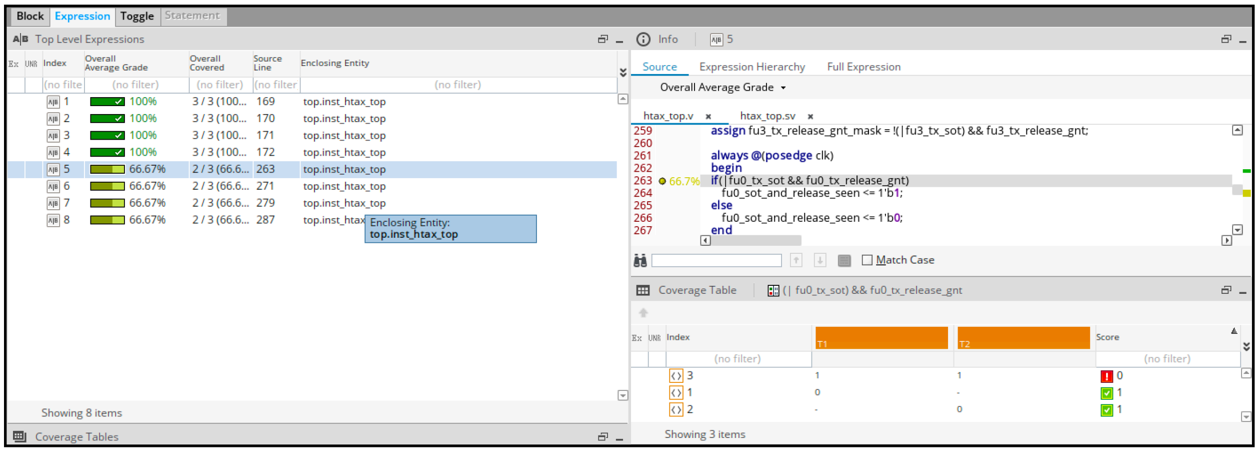


1.7 Code Coverage	98.68%	13432 / 13572 (98...	92.31%
1.7.1 Block	98.68%	3358 / 3393 (98.9...	92.31%
top	98.68%	3358 / 3393 (98.9...	92.31%
1.7.2 Expression	98.68%	3358 / 3393 (98.9...	92.31%
top	98.68%	3358 / 3393 (98.9...	92.31%
1.7.3 Toggle	98.68%	3358 / 3393 (98.9...	92.31%
top	98.68%	3358 / 3393 (98.9...	92.31%
1.7.4 FSM	98.68%	3358 / 3393 (98.9...	92.31%
top	98.68%	3358 / 3393 (98.9...	92.31%



1.7 Code Coverage	98.69%	13440 / 13572 (99.03%)	92.31%
1.7.1 Block	98.69%	3360 / 3393 (99.03%)	92.31%
top	98.69%	3360 / 3393 (99.03%)	92.31%
inst_htax_tx_intf[3]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[2]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[1]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[0]	✓ 100%	88 / 88 (100%)	100%
inst_htax_rx_intf[3]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[2]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[1]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[0]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_top	96.86%	2650 / 2679 (98.92%)	n/a
1.7.2 Expression	98.69%	3360 / 3393 (99.03%)	92.31%
top	98.69%	3360 / 3393 (99.03%)	92.31%
inst_htax_tx_intf[3]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[2]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[1]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[0]	✓ 100%	88 / 88 (100%)	100%
inst_htax_rx_intf[3]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[2]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[1]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[0]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_top	96.86%	2650 / 2679 (98.92%)	n/a
1.7.3 Toggle	98.69%	3360 / 3393 (99.03%)	92.31%
top	98.69%	3360 / 3393 (99.03%)	92.31%
inst_htax_tx_intf[3]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[2]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[1]	✓ 100%	88 / 88 (100%)	100%
inst_htax_tx_intf[0]	✓ 100%	88 / 88 (100%)	100%
inst_htax_rx_intf[3]	✓ 100%	85 / 85 (100%)	66.67%
inst_htax_rx_intf[2]	✓ 100%	85 / 85 (100%)	66.67%

Code Coverage Hole


Block | Expression | Toggle | Statement

Top Level Expressions

Ex	UNR	Index	Overall Average Grade	Overall Covered	Source Line	Enclosing Entity
		1	100%	3 / 3 (100...	169	top.inst_htax_top
		2	100%	3 / 3 (100...	170	top.inst_htax_top
		3	100%	3 / 3 (100...	171	top.inst_htax_top
		4	100%	3 / 3 (100...	172	top.inst_htax_top
		5	66.67%	2 / 3 (66.6...	263	top.inst_htax_top
		6	66.67%	2 / 3 (66.6...	271	top.inst_htax_top
		7	66.67%	2 / 3 (66.6...	279	top.inst_htax_top
		8	66.67%	2 / 3 (66.6...	287	top.inst_htax_top

Enclosing Entity: top.inst_htax_top

Showing 8 items

Coverage Tables

Info | 5

Source | Expression Hierarchy | Full Expression

Overall Average Grade

htax_top.v | htax_top.sv

```
assign fu3_tx_release_gnt_mask = !(|fu3_tx_sot) && fu3_tx_release_gnt;

always @(posedge clk)
begin
if(|fu0_tx_sot && fu0_tx_release_gnt)                      66.7%
  fu0_sot_and_release_seen <= 1'b1;
else
  fu0_sot_and_release_seen <= 1'b0;
end
```

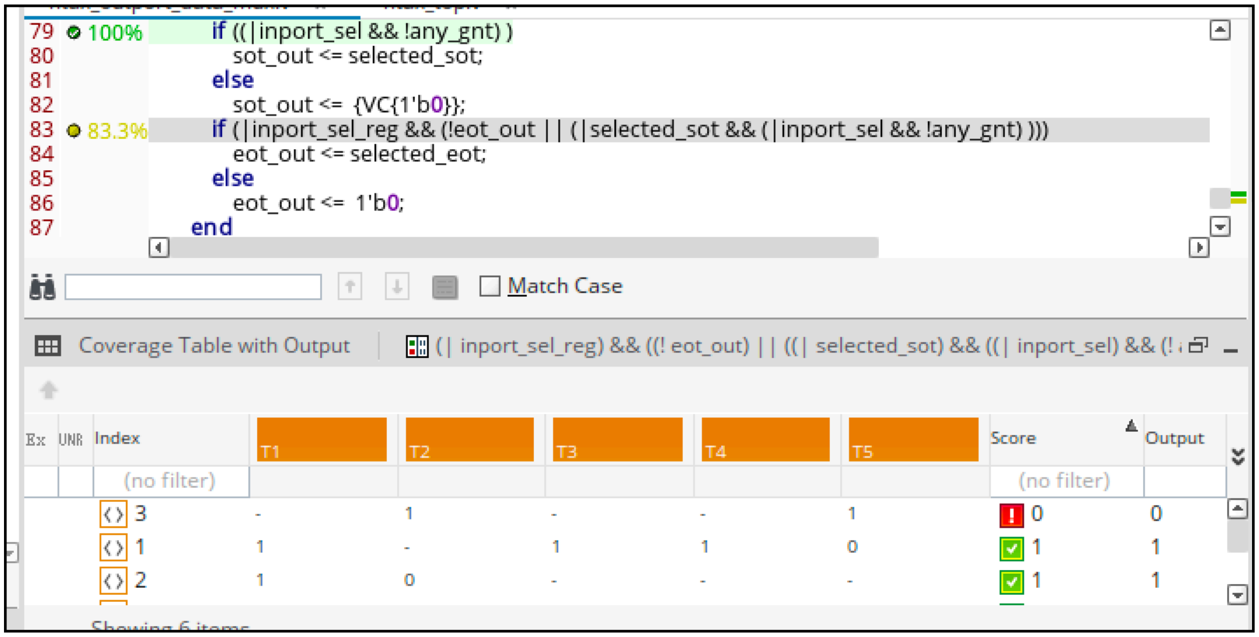
Match Case

Coverage Table | (| fu0_tx_sot) && fu0_tx_release_gnt

Ex	UNR	Index	T1	T2	Score
		3	1	1	0
		1	0	-	1
		2	-	0	1

Showing 3 items

If tx_release_gnt is asserted, it indicates that the current transaction is ending or should be released for arbitration to occur.
If tx_sot is asserted at the same time, it implies a new transaction is starting on the same virtual channel. This would mean two conflicting actions are happening: one signal is trying to start a new transaction, while the other is signaling the end or release of the current one. This scenario we can reproduce when length of packet is 2 and assertion will fail in this case


```
if ((|inport_sel && !any_gnt) )                              100%
    sot_out <= selected_sot;
else
    sot_out <= {VC{1'b0}};
if (|inport_sel_reg && (!eot_out | | (|selected_sot && (|inport_sel && !any_gnt) )))   83.3%
    eot_out <= selected_eot;
else
    eot_out <= 1'b0;
end
```

Match Case

Coverage Table with Output | (| inport_sel_reg) && ((! eot_out) | | ((| selected_sot) && ((| inport_sel) && (!

Ex	UNR	Index	T1	T2	T3	T4	T5	Score	Output
		3	-	1	-	-	1	0	0
		1	1	-	1	1	0	1	1
		2	1	0	-	-	-	1	1

Showing 6 items

.
As per timing relationship between these two signals "eot_out" and "any_gnt" can't be high together.

BUG REPORT

What is BUG ?

The DUT fails to maintain proper synchronization when the **EOT** signal becomes high at the same time across all ports when subjected to identical delay and packet length conditions, leading to failure in handling all ports in parallel. Bug is EOT timeout assertion is failing.

Where is it ?

- Module: htax_outport_data_mux
- File: htax_outport_data_mux.v
- Line number(s): 43

How to reproduce ?

I have given same packet length, same delay for all 4 ports in TEST 12. To reproduce this run command
“xrun -f run_vm.f
+UVM_TESTNAME=same_delay_same_length_all_port_parallel_random_test -seed
-1056087154

Expected Behavior

When the End-of-Transmission (EOT) signal becomes high simultaneously across all ports, In this case, the selected_eot signal should also go high, generating an EOT pulse and EOT timeout assertion should pass.

Actual Behavior

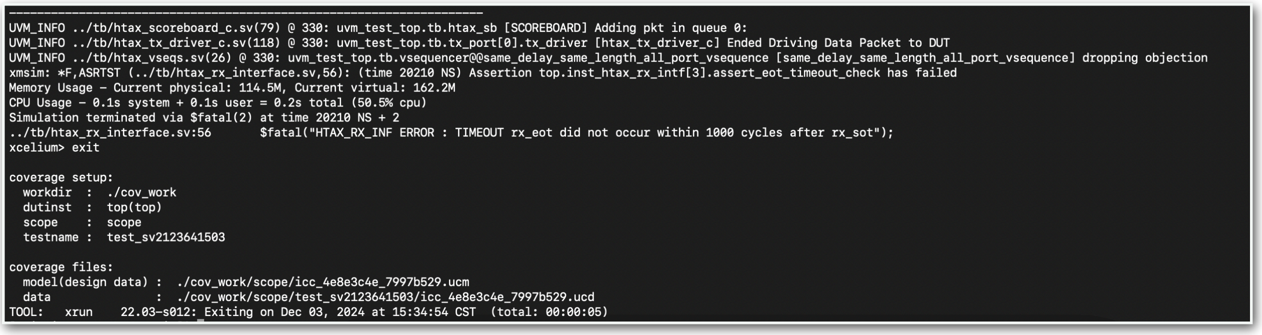
When the End-of-Transmission (EOT) signal becomes high simultaneously across all ports, the selected_eot signal goes low because of ~(&(eot_in)) generates a value of 0, causing the EOT timeout assertion to fail.

BUG FIX

To resolve this issue, **remove** the `~(&(eot_in))` component from the current logic:
assign selected_eot = |(eot_in & inport_sel_reg) & ~(&(eot_in));

By removing ~(&(eot_in)), the selected_eot signal will no longer be cleared when the EOT signal becomes high simultaneously on all ports. This change ensures that the selected_eot signal correctly asserts high when any port's EOT signal is active, even if all ports assert the EOT signal simultaneously. This adjustment should allow the EOT timeout assertion to pass.

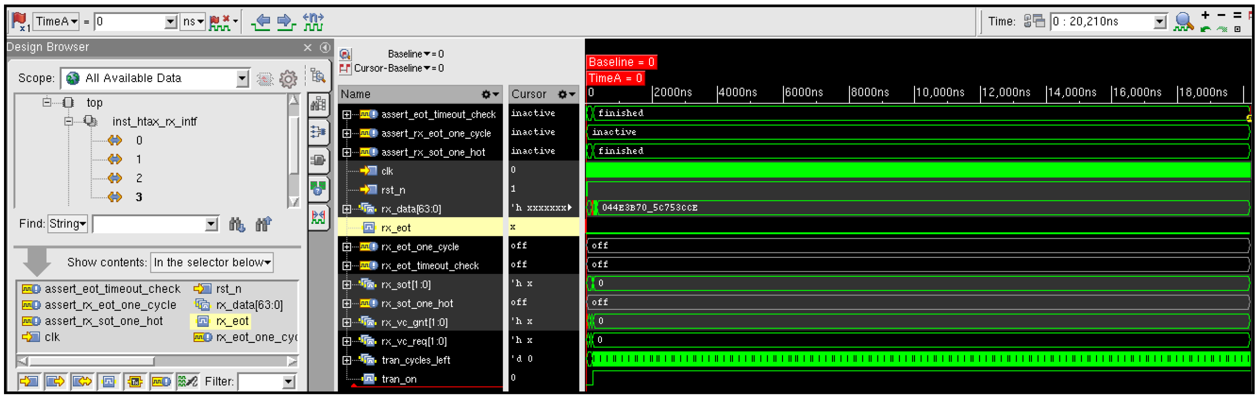
Failing Assertions



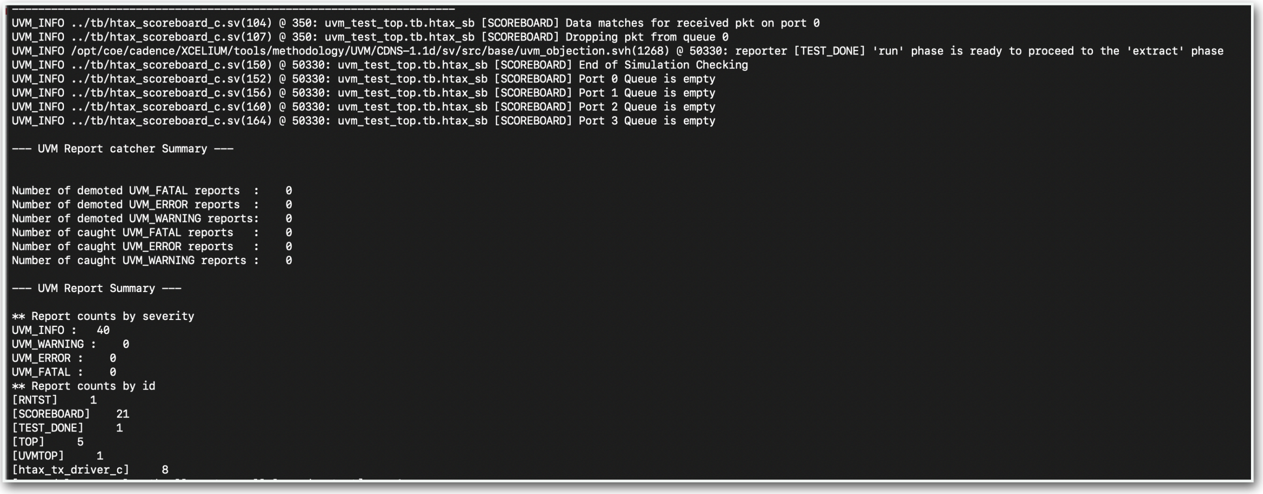
```
--------------------------------------------------------------------
UVM_INFO ../tb/htax_scoreboard_c.sv(79) @ 330: uvm_test_top.tb.htax_sb [SCOREBOARD] Adding pkt in queue 0:
UVM_INFO ../tb/htax_tx_driver_c.sv(118) @ 330: uvm_test_top.tb.tx_port[0].tx_driver [htax_tx_driver_c] Ended Driving Data Packet to DUT
UVM_INFO ../tb/htax_vseqs.sv(26) @ 330: uvm_test_top.tb.vsequencer@@same_delay_same_length_all_port_vsequence [same_delay_same_length_all_port_vsequence] dropping objection
xmsim: *F,ASRTST (../tb/htax_rx_interface.sv,56): (time 20210 NS) Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed
Memory Usage - Current physical: 114.5M, Current virtual: 162.2M
CPU Usage - 0.1s system + 0.1s user = 0.2s total (50.5% cpu)
Simulation terminated via $fatal(2) at time 20210 NS + 2
../tb/htax_rx_interface.sv:56       $fatal("HTAX_RX_INF ERROR : TIMEOUT rx_eot did not occur within 1000 cycles after rx_sot");
xcelium> exit

coverage setup:
  workdir  :  ./cov_work
  dutinst  :  top(top)
  scope    :  scope
  testname :  test_sv2123641503

coverage files:
  model(design data) :  ./cov_work/scope/icc_4e8e3c4e_7997b529.ucm
  data               :  ./cov_work/scope/test_sv2123641503/icc_4e8e3c4e_7997b529.ucd
TOOL:   xrun    22.03-s012: Exiting on Dec 03, 2024 at 15:34:54 CST  (total: 00:00:05)
```

Failure in Test12: This test revealed a bug in the DUT related to its inability to handle all ports in parallel under identical conditions.

Failing Scenario Waveform

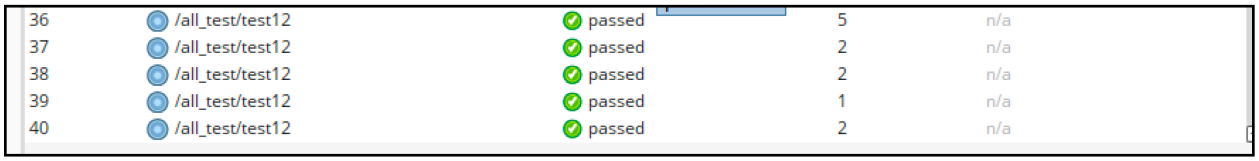


Failing Assertion passing after fix



```
--------------------------------------------------------------------
UVM_INFO ../tb/htax_scoreboard_c.sv(104) @ 350: uvm_test_top.tb.htax_sb [SCOREBOARD] Data matches for received pkt on port 0
UVM_INFO ../tb/htax_scoreboard_c.sv(107) @ 350: uvm_test_top.tb.htax_sb [SCOREBOARD] Dropping pkt from queue 0
UVM_INFO /opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_objection.svh(1268) @ 50330: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO ../tb/htax_scoreboard_c.sv(150) @ 50330: uvm_test_top.tb.htax_sb [SCOREBOARD] End of Simulation Checking
UVM_INFO ../tb/htax_scoreboard_c.sv(152) @ 50330: uvm_test_top.tb.htax_sb [SCOREBOARD] Port 0 Queue is empty
UVM_INFO ../tb/htax_scoreboard_c.sv(156) @ 50330: uvm_test_top.tb.htax_sb [SCOREBOARD] Port 1 Queue is empty
UVM_INFO ../tb/htax_scoreboard_c.sv(160) @ 50330: uvm_test_top.tb.htax_sb [SCOREBOARD] Port 2 Queue is empty
UVM_INFO ../tb/htax_scoreboard_c.sv(164) @ 50330: uvm_test_top.tb.htax_sb [SCOREBOARD] Port 3 Queue is empty

--- UVM Report catcher Summary ---


Number of demoted UVM_FATAL reports  :    0
Number of demoted UVM_ERROR reports  :    0
Number of demoted UVM_WARNING reports:    0
Number of caught UVM_FATAL reports   :    0
Number of caught UVM_ERROR reports   :    0
Number of caught UVM_WARNING reports :    0

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   40
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[SCOREBOARD]    21
[TEST_DONE]     1
[TOP]     5
[UVMTOP]     1
[htax_tx_driver_c]     8
```

Passing test 12 after fix



36	/all_test/test12	✓ passed	5	n/a
37	/all_test/test12	✓ passed	2	n/a
38	/all_test/test12	✓ passed	2	n/a
39	/all_test/test12	✓ passed	1	n/a
40	/all_test/test12	✓ passed	2	n/a