

# Lab10-Final Report

## Tests Added:

```
class long_pkt_test extends base_test;
`uvm_component_utils(long_pkt_test)

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

function void build_phase(uvm_phase phase);
    uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence",
long_pkt_sequence::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 long packet test",UVM_NONE)
endtask : run_phase
endclass : long_pkt_test

class short_pkt_test extends base_test;
`uvm_component_utils(short_pkt_test)

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

function void build_phase(uvm_phase phase);
    uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence",
short_pkt_sequence::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 short packet test",UVM_NONE)
endtask : run_phase
endclass : short_pkt_test

class random_pkt_len_test extends base_test;
`uvm_component_utils(random_pkt_len_test)

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

function void build_phase(uvm_phase phase);
    uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence",
random_pkt_len_sequence::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 short packet test",UVM_NONE)
endtask : run_phase
endclass : random_pkt_len_test
```

```

class mid_pkt_test extends base_test;
`uvm_component_utils(mid_pkt_test)

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

function void build_phase(uvm_phase phase);
    uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
mid_pkt_sequence::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 mid packet test", UVM_NONE)
endtask : run_phase
endclass : mid_pkt_test

class serial_multiport_test extends base_test;
`uvm_component_utils(serial_multiport_test)

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

function void build_phase(uvm_phase phase);
    uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
serial_multiport_sequence::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 simultaneous packet test", UVM_NONE)
endtask : run_phase
endclass : serial_multiport_test

class parallel_multiport_test extends base_test;
`uvm_component_utils(parallel_multiport_test)

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

function void build_phase(uvm_phase phase);
    uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
parallel_multiport_sequence::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 simultaneous packet test", UVM_NONE)
endtask : run_phase
endclass : parallel_multiport_test

class parallel_complete_test extends base_test;
`uvm_component_utils(parallel_complete_test)

```

```

function new (string name, uvm_component parent);
    super.new(name, parent);
endfunction
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
parallel_complete_sequence::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 simultaneous packet test", UVM_NONE)
    endtask : run_phase
endclass : parallel_complete_test

```

```

class dest_port_cross_length_test extends base_test;
`uvm_component_utils(dest_port_cross_length_test)

```

```

function new (string name, uvm_component parent);
    super.new(name, parent);
endfunction
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this, "tb.vsequencer.run_phase", "default_sequence",
dest_port_cross_length_sequence::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 simultaneous packet test", UVM_NONE)
    endtask : run_phase
endclass : dest_port_cross_length_test

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Virtual Sequences////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

class dest_port_cross_length_sequence extends htax_base_vseq;

    `uvm_object_utils(dest_port_cross_length_sequence)

    rand int port;
    randc int len0;
    randc int len1;
    randc int len2;
    randc int len3;
    function new (string name = "dest_port_cross_length_sequence");
        super.new(name);
    endfunction : new
    htax_packet_c req1; // Declare the sequence item

```

```

htax_packet_c req2; // Declare the sequence item
htax_packet_c req3; // Declare the sequence item
htax_packet_c req4; // Declare the sequence item

```

```

task body();
req1 = htax_packet_c::type_id::create("req1"); // Create the sequence item
req2 = htax_packet_c::type_id::create("req2"); // Create the sequence item
req3 = htax_packet_c::type_id::create("req3"); // Create the sequence item
req4 = htax_packet_c::type_id::create("req4"); // Create the sequence item
//Run 10 random
fork
begin
    for (int i=1; i<=3;i++)begin
        for (int j=3; j<=63; j++) begin
            for(int k=0; k<=3; k++) begin
                `uvm_do_on_with(req1, p_sequencer.htax_seqr[0], {length == j; vc == i; dest_port == k;})
            end
        end
    end
    `uvm_do_on_with(req1, p_sequencer.htax_seqr[0], {length == 12; vc == 2; dest_port == 3;})
    `uvm_do_on_with(req1, p_sequencer.htax_seqr[0], {length == 16; vc == 3; dest_port == 3;})
end
begin
    for (int i=1; i<=3;i++)begin
        for (int j=3; j<=63; j++) begin
            for(int k=0; k<=3; k++) begin
                `uvm_do_on_with(req3, p_sequencer.htax_seqr[1], {length == j; vc == i; dest_port == k;})
            end
        end
    end
end
begin
    for (int i=1; i<=3;i++)begin
        for (int j=3; j<=63; j++) begin
            for(int k=0; k<=3; k++) begin
                `uvm_do_on_with(req4, p_sequencer.htax_seqr[2], {length == j; vc == i; dest_port == k;})
            end
        end
    end
    `uvm_do_on_with(req4, p_sequencer.htax_seqr[2], {length == 12; vc == 1; dest_port == 3;})
    `uvm_do_on_with(req4, p_sequencer.htax_seqr[2], {length == 16; vc == 2; dest_port == 3;})
end
begin
    for (int i=1; i<=3;i++)begin
        for (int j=3; j<=63; j++) begin
            for(int k=0; k<=3; k++) begin
                `uvm_do_on_with(req2, p_sequencer.htax_seqr[3], {length == j; vc == i; dest_port == k;})
            end
        end
    end
    `uvm_do_on_with(req2, p_sequencer.htax_seqr[3], {length == 12; vc == 2; dest_port == 3;})
    `uvm_do_on_with(req2, p_sequencer.htax_seqr[3], {length == 16; vc == 3; dest_port == 3;})
end

```

```

end

join
$display("All ports completed");
endtask : body

endclass : dest_port_cross_length_sequence


class random_pkt_len_sequence extends htax_base_vseq;

`uvm_object_utils(random_pkt_len_sequence)

rand int port;
randc int len0;
randc int len1;
randc int len2;
randc int len3;

function new (string name = "random_pkt_len_sequence");
    super.new(name);
endfunction : new
htax_packet_c req1; // Declare the sequence item
htax_packet_c req2; // Declare the sequence item
htax_packet_c req3; // Declare the sequence item
htax_packet_c req0; // Declare the sequence item

task body();
req1 = htax_packet_c::type_id::create("req1"); // Create the sequence item
req2 = htax_packet_c::type_id::create("req2"); // Create the sequence item
req3 = htax_packet_c::type_id::create("req3"); // Create the sequence item
req0 = htax_packet_c::type_id::create("req0"); // Create the sequence item

    //Run 10 random
repeat(1000) begin
    port = $urandom_range(0,3);
    len0 = $urandom_range(3,10);
    len1 = $urandom_range(20,40);
    len2 = $urandom_range(50,63);
    len3 = $urandom_range(25,45);
    fork
        `uvm_do_on_with(req0, p_sequencer.htax_seqr[0], {length == len0; delay <10;})
        `uvm_do_on_with(req1, p_sequencer.htax_seqr[1], {length == len1; delay <10;})
        `uvm_do_on_with(req2, p_sequencer.htax_seqr[2], {length == len2; delay <10;})
        `uvm_do_on_with(req3, p_sequencer.htax_seqr[3], {length == len3; delay <10;})
    join
end
endtask : body

endclass : random_pkt_len_sequence
class parallel_complete_sequence extends htax_base_vseq;

```

```

`uvm_object_utils(parallel_complete_sequence)

rand int port;
randc int len0;
randc int len1;
randc int len2;
randc int len3;
function new (string name = "parallel_complete_sequence");
    super.new(name);
endfunction : new
htax_packet_c req1; // Declare the sequence item
htax_packet_c req2; // Declare the sequence item
htax_packet_c req3; // Declare the sequence item
htax_packet_c req4; // Declare the sequence item

task body();
req1 = htax_packet_c::type_id::create("req1"); // Create the sequence item
req2 = htax_packet_c::type_id::create("req2"); // Create the sequence item
req3 = htax_packet_c::type_id::create("req3"); // Create the sequence item
req4 = htax_packet_c::type_id::create("req4"); // Create the sequence item
    //Run 10 random
fork
begin
    repeat(50) begin
        for (int i = 1; i < 3; i++) begin
            for (int j = 3; j < 63; j++) begin
                for (int k = 1; k < 20; k++) begin
                    `uvm_do_on_with(req1, p_sequencer.htax_seqr[0], {length == j; vc == i; dest_port == 1;
delay == k;})
                end
            end
        end
    end
end
begin
    repeat(50) begin
        for (int i = 1; i < 3; i++) begin
            for (int j = 3; j < 63; j++) begin
                for (int k = 1; k < 20; k++) begin
                    `uvm_do_on_with(req3, p_sequencer.htax_seqr[1], {length == j; vc == i; dest_port == 2;
delay == k;})
                end
            end
        end
    end
end
begin
    repeat(50) begin
        for (int i = 1; i < 3; i++) begin
            for (int j = 3; j < 63; j++) begin
                for (int k = 1; k < 2; k++) begin

```

```

        `uvm_do_on_with(req4, p_sequencer.htax_seqr[2], {length == j; vc == i; dest_port == k;})
        end
    end
end

end
begin
repeat(50) begin
    for (int i = 1; i < 3; i++) begin
        for (int j = 3; j < 63; j++) begin
            for (int k = 1; k < 2; k++) begin
                `uvm_do_on_with(req2, p_sequencer.htax_seqr[3], {length == j; vc == i; dest_port == k;})
                end
            end
        end
    end
end

end
join
$display("All ports completed");
endtask : body

endclass : parallel_complete_sequence

```

```

class serial_multiport_sequence extends htax_base_vseq;

```

```

    `uvm_object_utils(serial_multiport_sequence)

```

```

    rand int port;
    randc int len0;
    randc int len1;
    randc int len2;
    randc int len3;
    semaphore sem1 = new(1);
    function new (string name = "serial_multiport_sequence");
        super.new(name);
    endfunction : new

```

```

    task body();
        //Run 10 random

```

```

    fork
    begin
        repeat(50) begin
            len0 = $urandom_range(20,50);
            sem1.get(1);
            `uvm_do_on_with(req, p_sequencer.htax_seqr[0], {length == len0;})
            sem1.put(1);
        end
    end
    begin
        repeat(50) begin
            len1 = $urandom_range(20,50);

```

```

    sem1.get(1);
    `uvm_do_on_with(req, p_sequencer.htax_seqr[1], {length == len1;})
    sem1.put(1);
end
end
begin
    repeat(50) begin
        len2 = $urandom_range(20,50);
        sem1.get(1);
        `uvm_do_on_with(req, p_sequencer.htax_seqr[2], {length == len2;})
        sem1.put(1);
    end
end
begin
    repeat(50) begin
        len3 = $urandom_range(20,50);
        sem1.get(1);
        `uvm_do_on_with(req, p_sequencer.htax_seqr[3], {length == len3;})
        sem1.put(1);
    end
end
join
    $display("All ports completed");
endtask : body

endclass : serial_multiport_sequence

```

```

class parallel_multiport_sequence extends htax_base_vseq;

```

```

    `uvm_object_utils(parallel_multiport_sequence)

    rand int port;
    randc int len0;
    randc int len1;
    randc int len2;
    randc int len3;
    function new (string name = "parallel_multiport_sequence");
        super.new(name);
    endfunction : new
    htax_packet_c req1; // Declare the sequence item
    htax_packet_c req2; // Declare the sequence item
    htax_packet_c req3; // Declare the sequence item
    htax_packet_c req4; // Declare the sequence item

    task body();
    req1 = htax_packet_c::type_id::create("req1"); // Create the sequence item
    req2 = htax_packet_c::type_id::create("req2"); // Create the sequence item
    req3 = htax_packet_c::type_id::create("req3"); // Create the sequence item
    req4 = htax_packet_c::type_id::create("req4"); // Create the sequence item
        //Run 10 random
    fork

```



```

begin
  repeat(50) begin
    len0 = $urandom_range(20,50);
    `uvm_do_on_with(req1, p_sequencer.htax_seqr[0], {length == len0;})
  end
end
begin
  repeat(50) begin
    len1 = $urandom_range(20,50);
    `uvm_do_on_with(req2, p_sequencer.htax_seqr[1], {length == len1;})
  end
end
begin
  repeat(50) begin
    len2 = $urandom_range(20,50);
    `uvm_do_on_with(req3, p_sequencer.htax_seqr[2], {length == len2;})
  end
end
begin
  repeat(50) begin
    len3 = $urandom_range(20,50);
    `uvm_do_on_with(req4, p_sequencer.htax_seqr[3], {length == len3;})
  end
end
join
$display("All ports completed");
endtask : body

endclass : parallel_multiport_sequence

```

```

class mid_pkt_sequence extends htax_base_vseq;

  `uvm_object_utils(mid_pkt_sequence)

  rand int port;
  randc int len;

  function new (string name = "mid_pkt_sequence");
    super.new(name);
  endfunction : new

  task body();
    //Run 10 random
    repeat(50) begin
      port = $urandom_range(0,3);
      len = $urandom_range(20,50);
      `uvm_do_on_with(req, p_sequencer.htax_seqr[1], {length == len;})
    end
  endtask : body

endclass : mid_pkt_sequence

```

```

class short_pkt_sequence extends htax_base_vseq;

  `uvm_object_utils(short_pkt_sequence)

  rand int port;
  randc int len;

  function new (string name = "short_pkt_sequence");
    super.new(name);
  endfunction : new
  htax_packet_c req1; // Declare the sequence item
  htax_packet_c req2; // Declare the sequence item
  htax_packet_c req3; // Declare the sequence item
  htax_packet_c req0; // Declare the sequence item

  task body();
    req1 = htax_packet_c::type_id::create("req1"); // Create the sequence item
    req2 = htax_packet_c::type_id::create("req2"); // Create the sequence item
    req3 = htax_packet_c::type_id::create("req3"); // Create the sequence item
    req0 = htax_packet_c::type_id::create("req0"); // Create the sequence item

    //Run 10 random
    repeat(1000) begin
      port = $urandom_range(0,3);
      len = $urandom_range(3,10);
      fork
        `uvm_do_on_with(req0, p_sequencer.htax_seqr[0], {length == len; delay <10;})
        `uvm_do_on_with(req1, p_sequencer.htax_seqr[1], {length == len; delay <10;})
        `uvm_do_on_with(req2, p_sequencer.htax_seqr[2], {length == len; delay <10;})
        `uvm_do_on_with(req3, p_sequencer.htax_seqr[3], {length == len; delay <10;})
      join
    end
  endtask : body

endclass : short_pkt_sequence

class long_pkt_sequence extends htax_base_vseq;

  `uvm_object_utils(long_pkt_sequence)

  rand int port;
  randc int len;

  function new (string name = "long_pkt_sequence");
    super.new(name);
  endfunction : new

  task body();
    //Run 10 random
    repeat(10) begin
      port = $urandom_range(0,3);
      len = $urandom_range(54,63);
      `uvm_do_on_with(req, p_sequencer.htax_seqr[port], {length == len;})
    end
  endtask : body

```

```

//USE `uvm_do_on_with` to add constraints on req

end
endtask : body

endclass : long_pkt_sequence

```

## Regression Script Code:

```

session htax_regress
{
    top_dir : $ENV(PWD)/regression/ ;
    pre_session_script : <text> echo "pre_session_script" </text>;
};

group all_test {

    run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=simple_random_test" ;
    scan_script: "vm_scan.pl ius.flt shell.flt" ;
    count : 1;
    pre_commands: "";
    timeout: 13000000;
    sv_seed: random;

    -----
    -- Simulation runs in the test container -----
    -----

    test test1 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=simple_random_test
-define TEST1" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 1;
    };

    test test2 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=simple_random_test -define
TEST2" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 1;
    };

    test test3 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=simple_random_test -define
TEST3" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
    };
}

```

```

        count : 1;
    };

    test test4 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=simple_random_test
-define TEST4" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 1;
    };

    test test5 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=simple_random_test
-define TEST5" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 1;
    };

    test test6 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f
+UVM_TESTNAME=multiport_sequential_random_test -define TEST6" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 5;
    };

-----
-- Add your tests here -----
-----

    test test7 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=long_pkt_test -define
TEST7" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 10;
    };
    test test8 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=short_pkt_test -define
TEST8" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 10;
    };
    test test9 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=mid_pkt_test -define
TEST9" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 10;
    };
    test test10 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=serial_multiport_test
-define TEST10" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;

```

```

        count : 100;
    };
    test test11 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=parallel_multiport_test
-define TEST11" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 10;
    };
    test test12 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=parallel_complete_test
-define TEST12" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 1;
    };
    test test13 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f +UVM_TESTNAME=random_pkt_len_test
-define TEST13" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 10;
    };
    test test14 {
        run_script: "cd $ENV(PWD) ; xrun -f run_vm.f
+UVM_TESTNAME=dest_port_cross_length_test -define TEST14" ;
        scan_script: "vm_scan.pl ius.flt shell.flt" ;
        count : 10;
    };
};

```

## Coverage report

### Failing Regression:

The screenshot displays the 'Tests Hierarchy' window with the 'Test-Case Model' tree. The 'all\_test' folder is expanded, showing a list of test cases from test1 to test14. Test8 is highlighted, showing an overall average grade of 90% and 9/10 coverage. A tooltip for test8 shows an overall average grade of 100%. Below the hierarchy, the 'Runs' section shows a list of test runs for test8. Run 29 is marked as failed, while all other runs are passed. The 'Errors' section at the bottom shows the failure details for run 29, indicating a timeout in the assertion 'assert\_eot\_timeout\_check'.

Name	Overall Average Grade	Overall Covered	Test Status
(no filter)	(no filter)	(no filter)	(no filter)
Test-Case Model	99.29%	170 / 171 (99.42%)	99.42%
default	99.29%	170 / 171 (99.42%)	99.42%
all_test	99.29%	170 / 171 (99.42%)	99.42%
test1	100%	1 / 1 (100%)	100%
test2	100%	1 / 1 (100%)	100%
test3	100%	1 / 1 (100%)	100%
test4	100%	1 / 1 (100%)	100%
test5	100%	1 / 1 (100%)	100%
test6	100%	5 / 5 (100%)	100%
test7	100%	10 / 10 (100%)	100%
test8	90%	9 / 10 (90%)	90%
test9	100%	10 / 10 (100%)	100%
test10	100%	100 / 100 (100%)	100%
test11	100%	10 / 10 (100%)	100%
test12	100%	1 / 1 (100%)	100%
test13	100%	10 / 10 (100%)	100%
test14	100%	10 / 10 (100%)	100%

Index	Name	Status	Duration (sec.)	Top Files	Start Time
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
30	/all_test/test8	passed	16	n/a	12/1/24 3:31 PM
29	/all_test/test8	failed	10	n/a	12/1/24 3:30 PM
28	/all_test/test8	passed	15	n/a	12/1/24 3:30 PM
27	/all_test/test8	passed	15	n/a	12/1/24 3:30 PM
26	/all_test/test8	passed	14	n/a	12/1/24 3:29 PM
25	/all_test/test8	passed	16	n/a	12/1/24 3:29 PM

Name	Description
(no filter)	(no filter)
VMRFEEXEC	'cd /home/grads/s/shubkumr7/UVM_LABs/lab-10-ShubhamKumartamu/work/sim ; xrun -f run_vm.f +UVM_TESTNAME=short_pkt_test -define TEST8' failed to execute!
ASRTST.assert_eot_timeout_check	Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed
RUN	run 29 script 'cd /home/grads/s/shubkumr7/UVM_LABs/lab-10-ShubhamKumartamu/work/sim ; xrun -f run_vm.f +UVM_TESTNAME=short_pkt_test -define TEST8' exited with exit code 2
ASRTST.assert_eot_timeout_check	Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed

As per Regression Run, short\_packet\_test is failing in Assertion assert\_eot\_timeout\_check as shown in this snapshot.

### Passing Regression after the failing regression:

After fixing the bug and rerunning the regression, it passed, same is verified by re running the failing test case with seed value from the failing case in regression.

Session Status	Name	Total Runs	#Passed	#Failed	#Running	#Waiting	#Other
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
completed	htax_regress.shubkumr7.24_12_03_02_26_55...	171	171	0	0	0	0
completed	htax_regress.shubkumr7.24_12_01_15_25_21...	171	170	1	0	0	0
completed	htax_regress.shubkumr7.24_12_01_15_11_15...	10	10	0	0	0	0
completed	htax_regress.shubkumr7.24_12_01_15_07_22...	10	10	0	0	0	0
completed	htax_regress.shubkumr7.24_12_01_14_56_41...	10	10	0	0	0	0
completed	htax_regress.shubkumr7.24_11_30_23_44_22...	171	171	0	0	0	0
completed	htax_regress.shubkumr7.24_11_30_22_45_19...	162	162	0	0	0	0
stopped	htax_regress.shubkumr7.24_11_30_17_36_14...	162	161	0	0	0	1
completed	htax_regress.shubkumr7.24_11_30_00_55_32...	161	161	0	0	0	0
pre_session_script_done	htax_regress.shubkumr7.24_11_29_15_38_58...	440	249	0	1	190	0
completed	htax_regress.shubkumr7.24_11_28_21_10_56...	240	240	0	0	0	0
completed	htax_regress.shubkumr7.24_11_27_10_58_49...	14	14	0	0	0	0
completed	htax_regress.shubkumr7.24_11_26_22_00_04...	14	13	1	0	0	0

































## TestCases Mapped:

The screenshot displays the 'vPlan Hierarchy' and 'Metrics Mapping' interface. On the left, the 'vPlan Hierarchy' pane shows a tree structure under 'CSCE 616 Project Fall2024'. The '1.2 TX Interface' is selected, and its sub-items are listed: '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 Parallel Multiport Test', '1.2.1.5 test14', '1.2.1.6 Complete Parallel Test', '1.2.1.7 Mid Pkt Test', '1.2.1.8 Long Pkt Test', '1.2.1.9 Sequential multiport test', and '1.2.2 Assertions\_slash\_Checkers for TX interf'. A tooltip 'Section: TX Interface' is visible over the mapping area. The right pane, 'Metrics Mapping', shows a table with 'Name' and 'Valid Metrics'. The 'Tests' section is expanded, showing a list of tests: 'all\_test', 'test1', 'test2', 'test3', 'test4', 'test5', 'test6', 'test7', 'test8', 'test9', 'test10', 'test11', 'test12', 'test13', and 'test14'. The 'Valid Metrics' column for all tests is 'n/a'.













All test cases are mapped as shown in the snapshot for TX Interface as well as RX Interface

The screenshot displays the 'vPlan Hierarchy' pane with the '1.3 RX Interface' section expanded. The '1.3.1 Testcases to verify RX interface' is also expanded, showing the following sub-items: '1.3.1.1 Simple Port-Port Test', '1.3.1.2 Random Test', '1.3.1.3 Short Packet Test', '1.3.1.4 Parallel Multiport Test', '1.3.1.5 Complete Paralle Test', '1.3.1.6 Mid Pkt Test', '1.3.1.7 Long Pkt Test', and '1.3.1.8 Sequential multiport test'.

























## Assertions/Checkers for Tx interface: All TX assertions PASS after Bug Fix

CSCE 616 Project Fall2024		48.38%	14318 / 14455 (99.05%)		96.65%
1 HTAX_v014		48.38%	14318 / 14455 (99.05%)		96.65%
1.1 System Interface		0%	0 / 2 (0%)		0%
1.2 TX Interface		100%	211 / 211 (100%)		100%
1.2.1 Testcases to verify TX interface		100%	171 / 171 (100%)		n/a
1.2.2 Assertions_slash_Checkers for TX inte		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_gnt_0		100%	4 / 4 (100%)		100%
1.2.2.9 tx_vc_req_outport_req		100%	4 / 4 (100%)		100%
1.2.2.10 tx_outport_req_vc_req		100%	4 / 4 (100%)		100%













## Assertions/Checkers for Rx interface: All RX assertions PASS after Bug Fix

1.3 RX Interface		100%	27 / 27 (100%)		100%
1.3.1 Testcases to verify RX interface		100%	15 / 15 (100%)		n/a
1.3.2 Assertions_slash_Checkers for RX inte		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 interface:























































































1.6 Functional Coverage		40%	636 / 639 (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 Grant		100%	12 / 12 (100%)		n/a
1.6.2.5 Packet VC		100%	12 / 12 (100%)		n/a
1.6.2.6 Packet Dest Port		100%	16 / 16 (100%)		n/a
1.6.2.7 Cross Dest Port and VC		100%	48 / 48 (100%)		n/a
1.6.2.8 Cross Dest Port and Length		100%	256 / 256 (100%)		n/a
1.6.2.9 Cross VC and Length		100%	192 / 192 (100%)		n/a

## Functional Coverage for Rx interface

1.6 Functional Coverage		40%	636 / 639 (99.53%)		0%
1.6.1 System Interface		0%	0 / 1 (0%)		0%
1.6.2 TX Interface		100%	628 / 628 (100%)		n/a
1.6.3 RX Interface		100%	8 / 8 (100%)		n/a
1.6.3.1 RX_DATA		100%	4 / 4 (100%)		n/a
1.6.3.2 RX_EOT		100%	4 / 4 (100%)		n/a



## Code Coverage:

1.7 Code Coverage	 98.69%	13444 / 13572 (99.06%)	 100%
1.7.1 Block	 98.69%	3361 / 3393 (99.06%)	 100%
top	 98.69%	3361 / 3393 (99.06%)	 100%
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%)	 100%
inst_htax_rx_intf[2]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[1]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[0]	 100%	85 / 85 (100%)	 100%
inst_htax_top	 96.86%	2651 / 2679 (98.95%)	n/a
1.7.2 Expression	 98.69%	3361 / 3393 (99.06%)	 100%
top	 98.69%	3361 / 3393 (99.06%)	 100%
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%)	 100%
inst_htax_rx_intf[2]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[1]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[0]	 100%	85 / 85 (100%)	 100%
inst_htax_top	 96.86%	2651 / 2679 (98.95%)	n/a
1.7.3 Toggle	 98.69%	3361 / 3393 (99.06%)	 100%
top	 98.69%	3361 / 3393 (99.06%)	 100%
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%)	 100%
inst_htax_rx_intf[2]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[1]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[0]	 100%	85 / 85 (100%)	 100%
inst_htax_top	 96.86%	2651 / 2679 (98.95%)	n/a
1.7.4 FSM	 98.69%	3361 / 3393 (99.06%)	 100%
top	 98.69%	3361 / 3393 (99.06%)	 100%
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%)	 100%
inst_htax_rx_intf[2]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[1]	 100%	85 / 85 (100%)	 100%
inst_htax_rx_intf[0]	 100%	85 / 85 (100%)	 100%
inst_htax_top	 96.86%	2651 / 2679 (98.95%)	n/a

## Code Coverage Holes:

Block	Index	Block Type	Source Line	Score	Enclosing Entity
...	...	true part of	288	0	top.inst_htax_top
...	...	true part of	280	0	top.inst_htax_top
...	...	true part of	272	0	top.inst_htax_top
...	...	true part of	264	0	top.inst_htax_top
...	...	code block	178	1	top.inst_htax_top
...	...	false part of	290	1	top.inst_htax_top
...	...	code block	286	1	top.inst_htax_top
...	...	false part of	282	1	top.inst_htax_top
...	...	code block	278	1	top.inst_htax_top
...	...	false part of	274	1	top.inst_htax_top
...	...	code block	270	1	top.inst_htax_top
...	...	false part of	266	1	top.inst_htax_top
...	...	code block	262	1	top.inst_htax_top
...	...	false part of	187	1	top.inst_htax_top
...	...	true part of	180	1	top.inst_htax_top

```
htax_top.v * htax_top.sv *
264 0% fu0_sot_and_release_seen <= 1'b1;
265 else
266 100% fu0_sot_and_release_seen <= 1'b0;
267 end
268
269 always @(posedge clk)
270 begin
271 if(fu1_tx_sot && fu1_tx_release_gnt)
272 0% fu1_sot_and_release_seen <= 1'b1;
273 else
274 100% fu1_sot_and_release_seen <= 1'b0;
275 end
276
277 always @(posedge clk)
278 begin
279 if(fu2_tx_sot && fu2_tx_release_gnt)
```

Block	Index	Block Type	Source Line	Score	Enclosing Entity
...	...	true part of	288	0	top.inst_htax_top
...	...	true part of	280	0	top.inst_htax_top
...	...	true part of	272	0	top.inst_htax_top
...	...	true part of	264	0	top.inst_htax_top
...	...	code block	178	1	top.inst_htax_top
...	...	false part of	290	1	top.inst_htax_top
...	...	code block	286	1	top.inst_htax_top
...	...	false part of	282	1	top.inst_htax_top
...	...	code block	278	1	top.inst_htax_top
...	...	false part of	274	1	top.inst_htax_top
...	...	code block	270	1	top.inst_htax_top
...	...	false part of	266	1	top.inst_htax_top
...	...	code block	262	1	top.inst_htax_top

```
htax_top.v * htax_top.sv *
280 0% fu2_sot_and_release_seen <= 1'b1;
281 else
282 100% fu2_sot_and_release_seen <= 1'b0;
283 end
284
285 always @(posedge clk)
286 begin
287 if(fu3_tx_sot && fu3_tx_release_gnt)
288 0% fu3_sot_and_release_seen <= 1'b1;
289 else
290 100% fu3_sot_and_release_seen <= 1'b0;
291 end
292
293 assign fu0_tx_release_gnt_delayed = fu0_sot_and_release_seen && !fu0_tx_rel
294 assign fu1_tx_release_gnt_delayed = fu1_sot_and_release_seen && !fu1_tx_rel
295 assign fu2_tx_release_gnt_delayed = fu2_sot_and_release_seen && !fu2_tx_rel
```

Block Coverage holes : Module: inst\_htax\_top

This can only be covered with length <3 which gives Assertion Failures

Instance (default): top inst\_htax\_top

Overall Covered Grade: 98.95% Code Covered Grade: 98.95% Block Covered Grade: 97.63% Statement Covered Grade: n/a Expression Covered Grade: n/a

Block Expression Toggle Statement

Top Level Expressions

Ex	UNR	Index	Overall Average Grade	Overall Covered	Source Line	Enclosing Entity
		(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
		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 (66.67%)	263	top.inst_htax_top
		6	66.67%	2 / 3 (66.67%)	271	top.inst_htax_top
		7	66.67%	2 / 3 (66.67%)	279	top.inst_htax_top
		8	66.67%	2 / 3 (66.67%)	287	top.inst_htax_top

Showing 8 items

Source

Overall Average Grade

```

259 assign fu3_tx_release_gnt_mask = (!fu3_tx_sot) && fu3_tx_release_gnt;
260
261 always @(posedge clk)
262 begin
263   if(fu0_tx_sot && fu0_tx_release_gnt)
264     fu0_sot_and_release_seen <= 1'b1;
265   else
266     fu0_sot_and_release_seen <= 1'b0;
267 end

```

Coverage Table

Ex	UNR	Index	T1	T2	Score
		(no filter)			(no filter)
		1	0	-	1
		2	-	0	1
		3	1	1	0

Showing 3 items

**Expression Coverage Holes: Module: inst\_htax\_top**  
 This can only be covered with length <3 which gives Assertion Failures

Instance (default): top inst\_htax\_top

Overall Covered Grade: 98.95% Code Covered Grade: 98.95% Block Covered Grade: 97.63% Statement Covered Grade: n/a Expression Covered Grade: n/a

Block Expression Toggle Statement

Variables

Ex	UNR	Name	Range	Overall Average Grade	Overall Covered	Enclos
		(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
		fu3_tx_release_gnt_delayed		0%	0 / 1 (0%)	top.ii
		fu2_tx_release_gnt_delayed		0%	0 / 1 (0%)	top.ii
		fu1_tx_release_gnt_delayed		0%	0 / 1 (0%)	top.ii
		fu0_tx_release_gnt_delayed		0%	0 / 1 (0%)	top.ii
		fu3_sot_and_release_seen		0%	0 / 1 (0%)	top.ii
		fu2_sot_and_release_seen		0%	0 / 1 (0%)	top.ii
		fu1_sot_and_release_seen		0%	0 / 1 (0%)	top.ii
		fu0_sot_and_release_seen		0%	0 / 1 (0%)	top.ii
		fu0_any_gnt		100%	1 / 1 (100%)	top.ii
		fu3_inport_ack	[3:0]	100%	4 / 4 (100%)	top.ii
		fu1_any_gnt		100%	1 / 1 (100%)	top.ii
		fu2_any_gnt		100%	1 / 1 (100%)	top.ii
		fu3_any_gnt		100%	1 / 1 (100%)	top.ii

Signal: fu1\_sot\_and\_release\_seen

Source

Overall Average Grade

```

246 ((fu1_tx_output_req[3], fu1_tx_output_req[3]) & fu1_tx_output_req[3]) & fu0_tx_output_req[3] & fu0_tx_output_req[3] & fu0_tx_output_req[3]
247
248 //Mask out release grant signal for small packets to avoid false arbitrations
249 //If sot is asserted in the same cycle the release_gnt signal is masked out.
250 //Instead the release_gnt signal is flopped in the release_gnt_delayed FF
251 //Result is that the release_gnt signal is delayed by one cycle for minimum size packets
252 wire fu0_tx_release_gnt_mask, fu1_tx_release_gnt_mask, fu2_tx_release_gnt_mask, fu3_tx_release_gnt_mask;
253 reg fu0_sot_and_release_seen, fu1_sot_and_release_seen, fu2_sot_and_release_seen, fu3_sot_and_release_seen;
254 wire fu0_tx_release_gnt_delayed, fu1_tx_release_gnt_delayed, fu2_tx_release_gnt_delayed, fu3_tx_release_gnt_delayed;
255 assign fu0_tx_release_gnt_mask = (!fu0_tx_sot) && fu0_tx_release_gnt;
256 assign fu1_tx_release_gnt_mask = (!fu1_tx_sot) && fu1_tx_release_gnt;
257 assign fu2_tx_release_gnt_mask = (!fu2_tx_sot) && fu2_tx_release_gnt;
258 assign fu3_tx_release_gnt_mask = (!fu3_tx_sot) && fu3_tx_release_gnt;
259
260 always @(posedge clk)
261 begin
262   fu0_sot_and_release_seen <= fu0_tx_release_gnt_mask & fu0_tx_release_gnt_delayed;
263   fu1_sot_and_release_seen <= fu1_tx_release_gnt_mask & fu1_tx_release_gnt_delayed;
264   fu2_sot_and_release_seen <= fu2_tx_release_gnt_mask & fu2_tx_release_gnt_delayed;
265   fu3_sot_and_release_seen <= fu3_tx_release_gnt_mask & fu3_tx_release_gnt_delayed;
266 end

```

**Toggle Coverage Holes: Module: inst\_htax\_top**  
 This can only be covered with length <3 which gives Assertion Failures

Instance (default): top > inst\_htax\_top > htax\_output\_arbiter\_FU0 > htax\_combinatoric\_arbiter\_I

Overall Covered Grade: 96.3% | Code Covered Grade: 96.3% | Block Covered Grade: 100% | Statement Covered Grade: n/a | Expression Covered Grad

Block Expression Toggle Statement

Variables

Ex	UNR	Name	Range	Overall Average Grade	Overall Covered	Enclos
		(no filter)	(no filter)	(no filter)	(no filter)	
		clk		100%	1 / 1 (100%)	top.i...
		res_n		100%	1 / 1 (100%)	top.i...
		reqs	[3:0]	100%	4 / 4 (100%)	top.i...
		stop		100%	1 / 1 (100%)	top.i...
		any_gnt		100%	1 / 1 (100%)	top.i...
		gnts	[3:0]	100%	4 / 4 (100%)	top.i...
		int_reqs	[3:0]	100%	4 / 4 (100%)	top.i...
		mask	[3:0]	75%	3 / 4 (75%)	top.i...
		gnts_w	[3:0]	100%	4 / 4 (100%)	top.i...
		any_gnt_w		100%	1 / 1 (100%)	top.i...
		g0	[3:0]	100%	4 / 4 (100%)	top.i...
		g1	[3:0]	100%	4 / 4 (100%)	top.i...
		p0	[3:0]	75%	3 / 4 (75%)	top.i...

Showing 19 items

Overall Average Grade

htax\_combinatoric\_arbiter.v x htax\_output\_arbiter.v x

```

57 MIX input wire clk, res_n,
58 100% input wire [3:0] reqs,
59 100% input wire stop,
60 100% output wire any_gnt,
61 100% output wire [3:0] gnts
62 ;
63
64 100% wire [3:0] int_reqs;
65 assign int_reqs = {reqs[0], reqs[1], reqs[2], reqs[3]};
66
67 75% reg [3:0] mask;
68
69 100% wire [3:0] gnts_w;
70 100% wire any_gnt_w;
71
72 100% wire [3:0] g0;
73 100% wire [3:0] p1;

```

Match Case

**Toggle Coverage Holes: Module: htax\_combinatoric\_arbiter\_I**  
**mask[3] is not getting toggled because mask is assigned by right shifting g1>>1; so 3rd bit will never toggle.**

Instance (default): top > inst\_htax\_top > htax\_output\_data\_mux\_FU0

Overall Covered Grade: 99.74% | Code Covered Grade: 99.74% | Block Covered Grade: 100% | Statement Covered Grade: n/a | Expression Covered Grad

Block Expression Toggle Statement

Top Level Expressions

Ex	UNR	Index	Overall Average Grade	Overall Covered	Source Line	Enclosing Entity
		(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
		1	100%	3 / 3 (100%)	80	top.inst_htax_top.htax_output_data_mux_FU0
		2	83.33%	5 / 6 (83.33%)	84	top.inst_htax_top.htax_output_data_mux_FU0

Showing 2 items

Coverage Tables

Source

Expression Hierarchy

Full Expression

Overall Average Grade

htax\_output\_data\_mux.v (out of sync) x htax\_top.v x

```

80 if ((|inport_sel && !any_gnt))
81   sot_out <= selected_sot;
82 else
83   sot_out <= {VC{1'b0}};
84 if (|inport_sel_reg && (!eot_out || (!selected_sot && (!inport_sel && !any_gnt))))
85   eot_out <= selected_eot;
86 else
87   eot_out <= 1'b0;
88 end

```

Match Case

Coverage Table with Output

Ex	UNR	Index	T1	T2	T3	T4	T5	Score	Output
		(no filter)						(no filter)	
		1	1	-	1	1	0	1	1
		2	-	0	-	-	-	1	1
		3	-	1	-	-	1	0	0

Showing 6 items

**Expression Coverage Hole: Module: htax\_output\_data\_mux\_FU\***  
**Expression Missed: eot\_out =1 && any\_gnt =1; Based on Timing relationship between these two signals; these two can't be simultaneously HIGH.**

## Bugs report

### Bug 1

#### What is the bug?

If tx\_eot for all 4 instances come simultaneously, rx\_eot will not be generated and will lead to Assertion failure of EOT Timeout. This is due to bug in RTL which checks that rx\_eot can't be generated if all 4 TX\_EOTs are becoming high simultaneously.

#### Where is it?

Module name : **htax\_outport\_data\_mux**

Instance name : **htax\_outport\_data\_mux\_FU\*** in HTAX\_TOP

File: **htax\_outport\_data\_mux.v**

Line number(s): 43

#### How to reproduce:

We should send sequence items on all 4 ports parallel with same packet lengths and small delays i.e, less than 10 which is used. Also this bug is specifically coming for short packets test only. I have attached the sequence used for this case.

Sequence Used:

```
class short_pkt_sequence extends htax_base_vseq;
```

```
`uvm_object_utils(short_pkt_sequence)
```

```
rand int port;  
randc int len;
```

```
function new (string name = "short_pkt_sequence");  
    super.new(name);  
endfunction : new  
htax_packet_c req1; // Declare the sequence item  
htax_packet_c req2; // Declare the sequence item  
htax_packet_c req3; // Declare the sequence item  
htax_packet_c req0; // Declare the sequence item
```

```
task body();  
req1 = htax_packet_c::type_id::create("req1"); // Create the sequence item  
req2 = htax_packet_c::type_id::create("req2"); // Create the sequence item  
req3 = htax_packet_c::type_id::create("req3"); // Create the sequence item  
req0 = htax_packet_c::type_id::create("req0"); // Create the sequence item
```

```
    //Run 10 random  
    repeat(1000) begin  
        port = $urandom_range(0,3);  
        len = $urandom_range(3,10);
```

```

fork
`uvm_do_on_with(req0, p_sequencer.htax_seqr[0], {length == len; delay <10;})
`uvm_do_on_with(req1, p_sequencer.htax_seqr[1], {length == len; delay <10;})
`uvm_do_on_with(req2, p_sequencer.htax_seqr[2], {length == len; delay <10;})
`uvm_do_on_with(req3, p_sequencer.htax_seqr[3], {length == len; delay <10;})
join
end
endtask : body

endclass : short_pkt_sequence

```

For exactly recreating the bug we can run the following command with given seed value:  
**xrun -f run.f +UVM\_TESTNAME=short\_pkt\_test -seed -1313566714**

### Expected behavior:

It is expected that even if all 4 channels produce tx\_eot at the same time the DUT should be able to capture them simultaneously and generate RX\_EOT before the timeout. After this the next packets should be transferred correctly.

### How the Bug is Found:

After creating multiple test cases and running regression multiple times; I saw some Cross Coverage holes so I created directed test cases to cover it; when I created test case where I sent sequence items parallel to all the instances with same length and small delay values I was able to find the bug. You can see the number of times I updated and created new scenarios of test cases for regression.

Flow Sessions										
Session Status	Name	Total Runs	#Passed	#Failed	#Running	#Waiting	#Other	Start Time	Owner	
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	==\$ME()	
completed	htax_regress.shubkumr7.24_12_03_02_26_55...	171	171	0	0	0	0	12/3/24 2:26 AM	shubkumr7	
completed	htax_regress.shubkumr7.24_12_01_15_25_21...	171	170	1	0	0	0	12/1/24 3:25 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_12_01_15_11_15...	10	10	0	0	0	0	12/1/24 3:11 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_12_01_15_07_22...	10	10	0	0	0	0	12/1/24 3:07 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_12_01_14_56_41...	10	10	0	0	0	0	12/1/24 2:56 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_11_30_23_44_22...	171	171	0	0	0	0	11/30/24 11:44 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_11_30_22_45_19...	162	162	0	0	0	0	11/30/24 10:45 PM	shubkumr7	
stopped	htax_regress.shubkumr7.24_11_30_17_36_14...	162	161	0	0	0	1	11/30/24 5:36 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_11_30_00_55_32...	161	161	0	0	0	0	11/30/24 12:55 AM	shubkumr7	
pre_session_script_done	htax_regress.shubkumr7.24_11_29_15_38_58...	440	249	0	1	190	0	11/29/24 3:38 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_11_28_21_10_56...	240	240	0	0	0	0	11/28/24 9:10 PM	shubkumr7	
completed	htax_regress.shubkumr7.24_11_27_10_58_49...	14	14	0	0	0	0	11/27/24 10:58 AM	shubkumr7	
completed	htax_regress.shubkumr7.24_11_26_22_00_04...	14	13	1	0	0	0	11/26/24 10:00 PM	shubkumr7	

Once I found that one case is failing I got the seed value and name of test case so I ran the test separately and generated the waveforms and debugged the RTL to find the RTL bug.

### Actual behavior:

Our design doesn't generate RX\_EOT when TX\_EOT for all 4 instances come simultaneously. This leads to assertion failure as RX\_EOT is not generated till timeout. Due to this the next packets are also not captured as TX\_SOT itself is not getting generated for next packets.

### Bug fix:

We should do the RTL fix and remove the dependency of RX\_EOT generation on the condition that TX\_EOT should not be simultaneously HIGH. **Line 43** is the code with Bug and **Line 44** is Bug fix.

```

30      wire          selected_eot;
31
32      always @( * )
33      begin
34          (* full_case *) (* parallel_case *)
35          casex (inport_sel)
36              4'b1xxx: selected_sot = sot_in[((4*VC)-1):(3*VC)];
37              4'bx1xx: selected_sot = sot_in[((3*VC)-1):(2*VC)];
38              4'bxx1x: selected_sot = sot_in[((2*VC)-1):(1*VC)];
39              4'bxxx1: selected_sot = sot_in[((1*VC)-1):(0*VC)];
40          endcase
41      end
42
43      //assign selected_eot = |(eot_in & inport_sel_reg) & ~(&(eot_in));
44      assign selected_eot = |(eot_in & inport_sel_reg);
45
46      `ifdef ASYNC_RES
47      always @(posedge clk or negedge res_n) `else
48      always @(posedge clk) `endif
49      begin
50          if (!res_n) begin
51              inport_sel_reg <= {NUM_PORTS{1'b0}};
52              any_gnt_reg    <= 0;
53          end else begin
54              any_gnt_reg <= any_gnt;
55
56              //if(any_gnt_reg)
57              inport_sel_reg <= inport_sel;
58          end
59      end
60      // Muxes for data, eot, sot

```

htax\_outport\_data\_mux.v [+]

## Failing Assertion:

We can see that **assert\_eot\_timeout\_check** has failed for seed value of **-1313566714** for **short\_pkt\_test**.

```

UVM_INFO ../tb/htax_tx_driver_c.sv(59) @ 296130000: uvm_test_top.tb.tx_port[0].tx_driver [htax_tx_driver_c] Input Data Packet to DUT :
-----
Name                                     Type      Size  Value
-----
req0                                     htax_packet_c  -    @8211
delay                                   integral     32    'h9
dest_port                               integral     32    'h1
vc                                       integral      2    'h1
length                                  integral     32    'h3
data                                     da(integral)  3    -
  [0]                                    integral     64    'h179e249cc9b27f5a
  [1]                                    integral     64    'h5747ca70b1bf6fd5
  [2]                                    integral     64    'h67e1f3bcb5d33ca2
begin_time                              time         64    296130000
depth                                    int           32    'd2
parent sequence (name)                  string        18    short_pkt_sequence
parent sequence (full name)              string        45    uvm_test_top.tb.vsequencer.short_pkt_sequence
sequencer                                string        36    uvm_test_top.tb.tx_port[0].sequencer
-----

xmsim: *F,ASRTST (../tb/htax_rx_interface.sv,56): (time 315970 NS) Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed
Memory Usage - Current physical: 115.6M, Current virtual: 162.8M
CPU Usage - 0.3s system + 2.7s user = 3.1s total (20.5% cpu)
Simulation terminated via $fatal(2) at time 315970 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

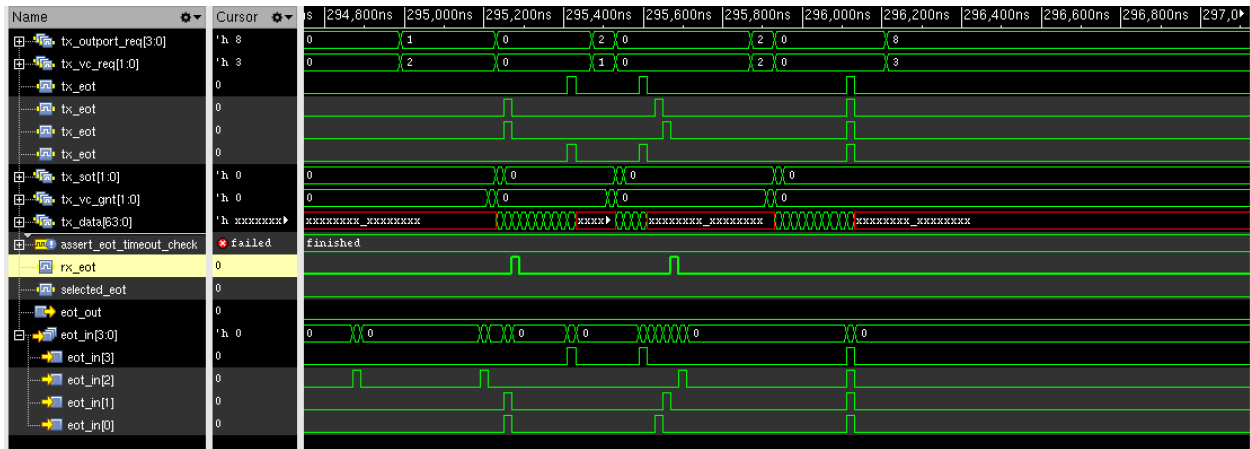
coverage setup:
workdir : ./cov_work
dutinst : top(top)
scope   : scope
testname : test_sv-1313566714

coverage files:
model(design data) : ./cov_work/scope/icc_4e8e3c4e_7997b529.ucm (reused)
data               : ./cov_work/scope/test_sv-1313566714/icc_4e8e3c4e_7997b529.ucd
TOOL: xrun 22.03-s012: Exiting on Dec 03, 2024 at 03:57:27 CST (total: 00:00:19)
[shubkumr7@n02-hera sim]$ xrun -f run.f +UVM_TESTNAME=short_pkt_test -seed -1313566714

```



## Failing Scenario Waveform:



As we can see that tx\_eot for all 4 instances are coming, then even after tx\_data has been sent completely RX\_EOT is not generated. For next packets even though we give tx\_outport\_req and tx\_vc\_req; tx\_sot is not coming for next packets.

## Failing Assertion Passing after the fix:

The test case short\_pkt\_test with the seed value of **-1313566714** which was failing earlier is now passing after the RTL fix.

```
--- 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 :24016
UVM_WARNING : 0
UVM_ERROR : 0
UVM_FATAL : 0
** Report counts by id
[RNTST] 1
[SCOREBOARD] 16005
[TEST_DONE] 1
[TOP] 5
[UVMTOP] 1
[hax_tx_driver_c] 8000
[short_pkt_sequence] 2
[short_pkt_test] 1
Simulation complete via $finish(1) at time 530350 NS + 45
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457 $finish;
xcelium> exit

coverage setup:
workdir : ./cov_work
dutinst : top(top)
scope : scope
testname : test_sv-1313566714

coverage files:
model(design data) : ./cov_work/scope/icc_4e8e3c4e_7997b529.ucm (reused)
data : ./cov_work/scope/test_sv-1313566714/icc_4e8e3c4e_7997b529.ucd
TOOL: xrun 22.03-s012: Exiting on Dec 03, 2024 at 04:34:12 CST (total: 00:00:26)
[shubkumr7@n02-hera sim]$ xrun -f run.f +UVM_TESTNAME=short_pkt_test -seed -1313566714
```



