LAB-10 Report

Jashwanth Chandhra Adama

UIN: 133008552

Testcases :

Short_packet_random_test:

```
class short_packet_random_test extends base_test;
	`uvm_component_utils(short_packet_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_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 short packet random
test",UVM_NONE)
	endtask : run_phas
endclass : short_packet_random_test
//////////////////////////////// VIRTUAL SEQUENCE ////////////////////////////
class short_packet_random_vsequence extends htax_base_vseq;
  `uvm_object_utils(short_packet_random_vsequence
  htax_packet_c packet0, packet1, packet2, packet3;
  rand int length;
  function new (string name = "short_packet_random_vsequence");
    super.new(name);
    packet0 = new();
    packet1 = new();
    packet2 = new();

    packet3 = new();


  endfunction : new

  task body();

                        // Exectuing 10 TXNs on ports {0,1,2,3} randomly

    repeat(100) begin

             fork

    //  `uvm_do_on(req, p_sequencer.htax_seqr[port])

                                     //USE `uvm_do_on_with to add constraints on req

              `uvm_do_on_with(packet0, p_sequencer.htax_seqr[0], {packet0.length inside
{[3:10]} ; }) ;

              `uvm_do_on_with(packet1, p_sequencer.htax_seqr[1], {packet1.length inside
{[3:10]} ; }) ;

              `uvm_do_on_with(packet2, p_sequencer.htax_seqr[2], {packet2.length inside
{[3:10]} ; }) ;

              `uvm_do_on_with(packet3, p_sequencer.htax_seqr[3], {packet3.length inside
{[3:10]} ; }) ;

             join

    end

  endtask : body

endclass : short_packet_random_vsequence
```

Medium_packet_random_test:

```
class medium_packet_random_test extends base_test;

             `uvm_component_utils(medium_packet_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", medium_packet_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 medium packet random
test",UVM_NONE)

          endtask : run_phase

endclass : medium_packet_random_test

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

class medium_packet_random_vsequence extends htax_base_vseq;

  `uvm_object_utils(medium_packet_random_vsequence

  htax_packet_c req[4];

  rand int length;

  function new (string name = "medium_packet_random_vsequence");

    super.new(name);

    req[0] = new();

    req[1] = new();

    req[2] = new();

    req[3] = new()

  endfunction : ne

  task body();

                    // Exectuing 10 TXNs on ports {0,1,2,3} randomly

    repeat(10) begin

    //  `uvm_do_on(req, p_sequencer.htax_seqr[port])

                              //USE `uvm_do_on_with to add constraints on req

           `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0], {req[0].length inside {[10:40]} ;
req[0].delay > 5; })

           `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1], {req[1].length inside {[10:40]} ;
req[1].delay > 5; })
```

```
            `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2], {req[2].length inside {[10:40]} ;
req[2].delay > 5; })
            `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3], {req[3].length inside {[10:40]} ;
req[3].delay > 5; })
    end
  endtask : body
endclass : medium_packet_random_vsequence
```

Long_packet_random_test:

```
class long_packet_random_test extends base_test;
            `uvm_component_utils(long_packet_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_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 long packet random test",UVM_NONE)
            endtask : run_phas
endclass : long_packet_random_test
//////////////////////////// VIRTUAL SEQUENCE //////////////////////////
class long_packet_random_vsequence extends htax_base_vseq;
  `uvm_object_utils(long_packet_random_vsequence
            htax_packet_c packet0, packet1, packet2, packet3;
  rand int length
  function new (string name = "long_packet_random_vsequence");
    super.new(name);
    packet0 = new();
    packet1 = new();
    packet2 = new();
    packet2 = new();
  endfunction : new
  task body();
                    // Exectuing 10 TXNs on ports {0,1,2,3} randomly
    repeat(100) begin
                    fork
    //  `uvm_do_on(req, p_sequencer.htax_seqr[port])
                              //USE `uvm_do_on_with to add constraints on req
            `uvm_do_on_with(packet0, p_sequencer.htax_seqr[0], {packet0.length inside
{[41:63]} ; }) ;
            `uvm_do_on_with(packet1, p_sequencer.htax_seqr[1], {packet1.length inside
{[41:63]}; });
            `uvm_do_on_with(packet2, p_sequencer.htax_seqr[2], {packet2.length inside
{[41:63]}; });
            `uvm_do_on_with(packet3, p_sequencer.htax_seqr[3], {packet3.length inside
{[41:63]}; });
                    join
            end
  endtask : body
endclass : long_packet_random_vsequence
```

Fixed_delay_test:

```
class fixed_length_test extends base_test;
            `uvm_component_utils(fixed_length_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", fixed_length_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 fixed length random test",UVM_NONE)

		endtask : run_phase

endclass : fixed_length_test


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



class fixed_length_random_vsequence extends htax_base_vseq;

  `uvm_object_utils(fixed_length_random_vsequence)

  htax_packet_c packet0, packet1, packet2, packet3;

  rand int length;

  function new (string name = "fixed_length_random_vsequence");

    super.new(name);

    packet0 = new();

    packet1 = new();

    packet2 = new();

    packet3 = new();

  endfunction : new


  task body();

			// Exectuing 10 TXNs on ports {0,1,2,3} randomly

    repeat(100) begin

		fork

    //  `uvm_do_on(req, p_sequencer.htax_seqr[port])
```

//USE `uvm_do_on_with to add constraints on req

`uvm_do_on_with(packet0, p_sequencer.htax_seqr[0], {packet0.length == 5; packet0.delay < 6; })

`uvm_do_on_with(packet1, p_sequencer.htax_seqr[1], {packet1.length == 5; packet1.delay < 6; })

`uvm_do_on_with(packet2, p_sequencer.htax_seqr[2], {packet2.length == 5; packet2.delay < 6; })

`uvm_do_on_with(packet3, p_sequencer.htax_seqr[3], {packet3.length == 5; packet3.delay < 6; })

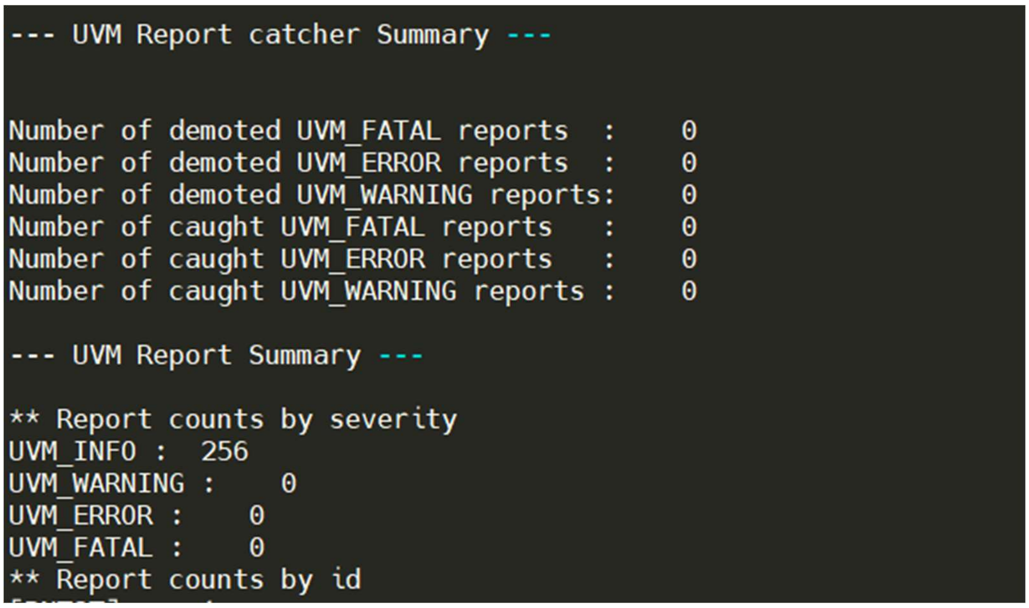
join

end

endtask : body

endclass : fixed_length_random_vsequence

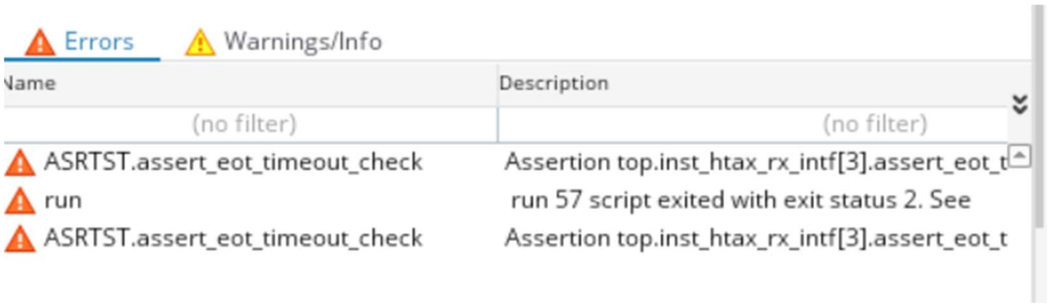
UVM No failure :



```
--- 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 :  256
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
```

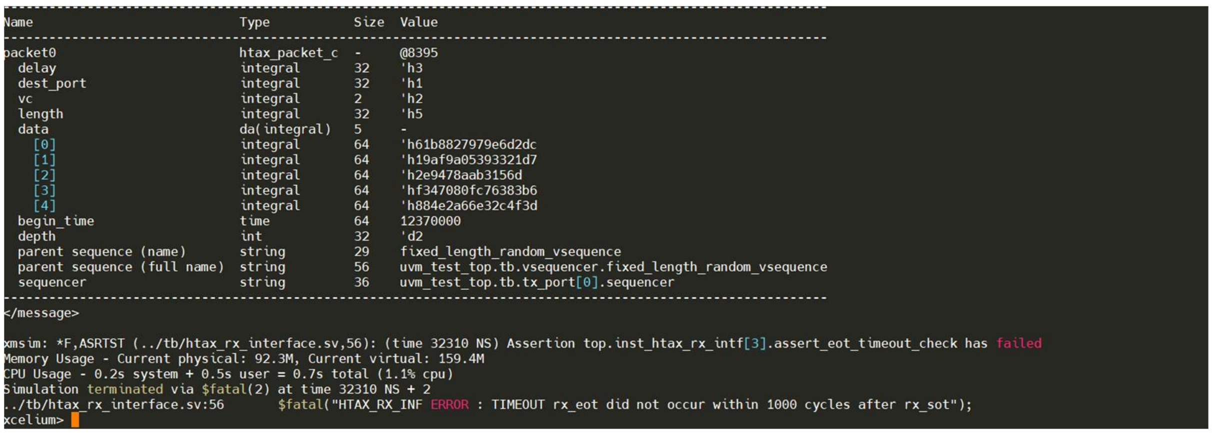
BUG Report:

I have written testcases namely short_packet, medium packet, long packet, fixed length testcases . I ran those testcases seed multiple times. While running the regression run, the below bug has been found.



Name	Description
(no filter)	(no filter)
ASRTST.assert_eot_timeout_check	Assertion top.inst_htax_rx_intf[3].assert_eot_t
run	run 57 script exited with exit status 2. See
ASRTST.assert_eot_timeout_check	Assertion top.inst_htax_rx_intf[3].assert_eot_t

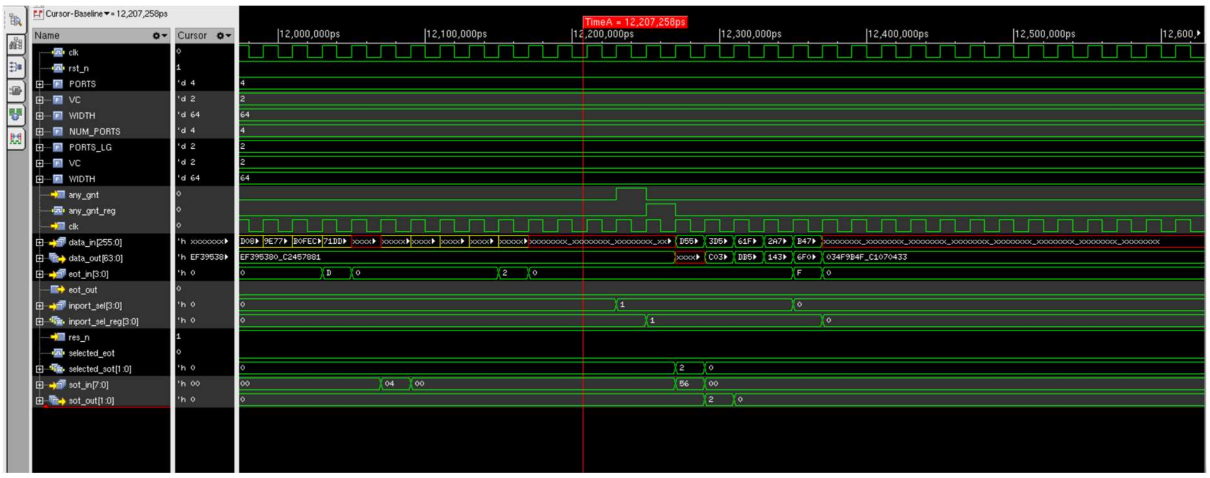
Then I run the failure seed with irun as mentioned below:



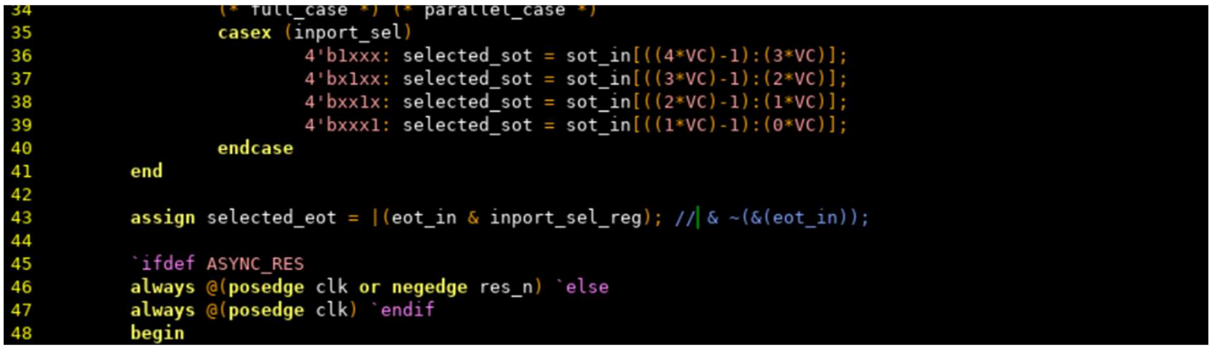
```
-------------------------------------------------------------------------------------------------------
Name                         Type           Size  Value
-------------------------------------------------------------------------------------------------------
packet0                      htax_packet_c  -     @8395
  delay                      integral       32    'h3
  dest_port                  integral       32    'h1
  vc                         integral       2     'h2
  length                     integral       32    'h5
  data                       da(integral)   5     -
    [0]                      integral       64    'h61b8827979e6d2dc
    [1]                      integral       64    'h19af9a05393321d7
    [2]                      integral       64    'h2e9478aab3156d
    [3]                      integral       64    'hf347080fc76383b6
    [4]                      integral       64    'h884e2a66e32c4f3d
  begin_time                 time           64    12370000
  depth                      int            32    'd2
  parent sequence (name)     string         29    fixed_length_random_vsequence
  parent sequence (full name) string        56    uvm_test_top.tb.vsequencer.fixed_length_random_vsequence
  sequencer                  string         36    uvm_test_top.tb.tx_port[0].sequencer
-------------------------------------------------------------------------------------------------------
</message>

xmsim: *F,ASRTST (../tb/htax_rx_interface.sv,56): (time 32310 NS) Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed
Memory Usage - Current physical: 92.3M, Current virtual: 159.4M
CPU Usage - 0.2s system + 0.5s user = 0.7s total (1.1% cpu)
Simulation terminated via $fatal(2) at time 32310 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>
```

Currently, to debug, we'll see the waveforms where it appears that the end-of-transmission (EOT) signal isn't activated for the RX interface on port 3. From the code ,

The selected_eot signal is sent to `eot_out` to mark the end of a transaction. `eot_in` combines EOT signals from all units; when all are on, it's `4’b1111`. Initially flawed logic caused `selected_eot` to be low when all `eot_in` were on. To fix this, `selected_eot` is now set using `selected_eot = |(eot_in & inport_sel_reg)`.

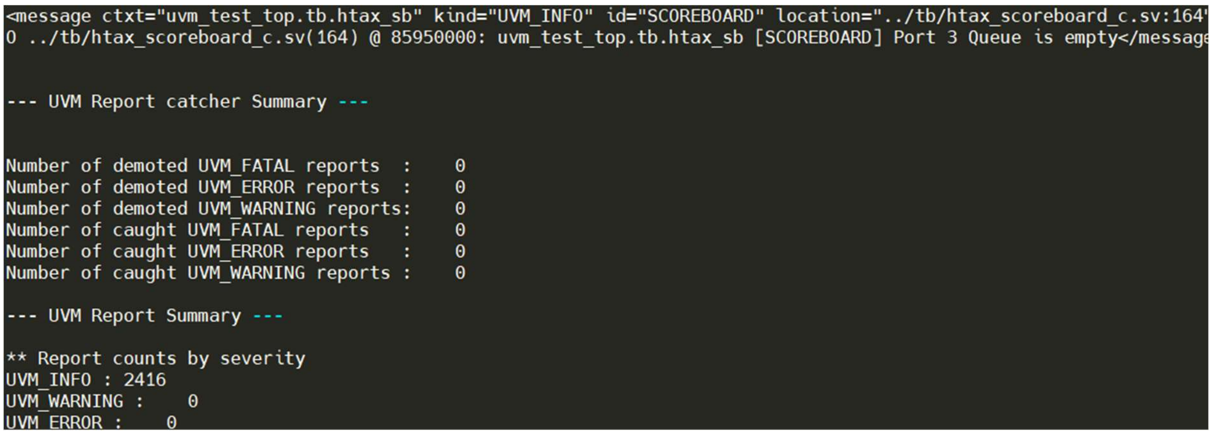


To resolve this issue, make the changes the below changes in the code.



```
                (* full_case *) (* parallel_case *)
                casex (inport_sel)
                        4'b1xxx: selected_sot = sot_in[((4*VC)-1):(3*VC)];
                        4'bx1xx: selected_sot = sot_in[((3*VC)-1):(2*VC)];
                        4'bxx1x: selected_sot = sot_in[((2*VC)-1):(1*VC)];
                        4'bxxx1: selected_sot = sot_in[((1*VC)-1):(0*VC)];
                endcase
        end

        assign selected_eot = |(eot_in & inport_sel_reg); //| & ~(&(eot_in));

        `ifdef ASYNC_RES
        always @(posedge clk or negedge res_n) `else
        always @(posedge clk) `endif
        begin
```

Result: ran same failing seed again, now the testcase has passed.



```
<message ctxt="uvm_test_top.tb.htax_sb" kind="UVM_INFO" id="SCOREBOARD" location="../tb/htax_scoreboard_c.sv:164'
0 ../tb/htax_scoreboard_c.sv(164) @ 85950000: uvm_test_top.tb.htax_sb [SCOREBOARD] Port 3 Queue is empty</message


--- 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 : 2416
UVM_WARNING :    0
UVM_ERROR :    0
```