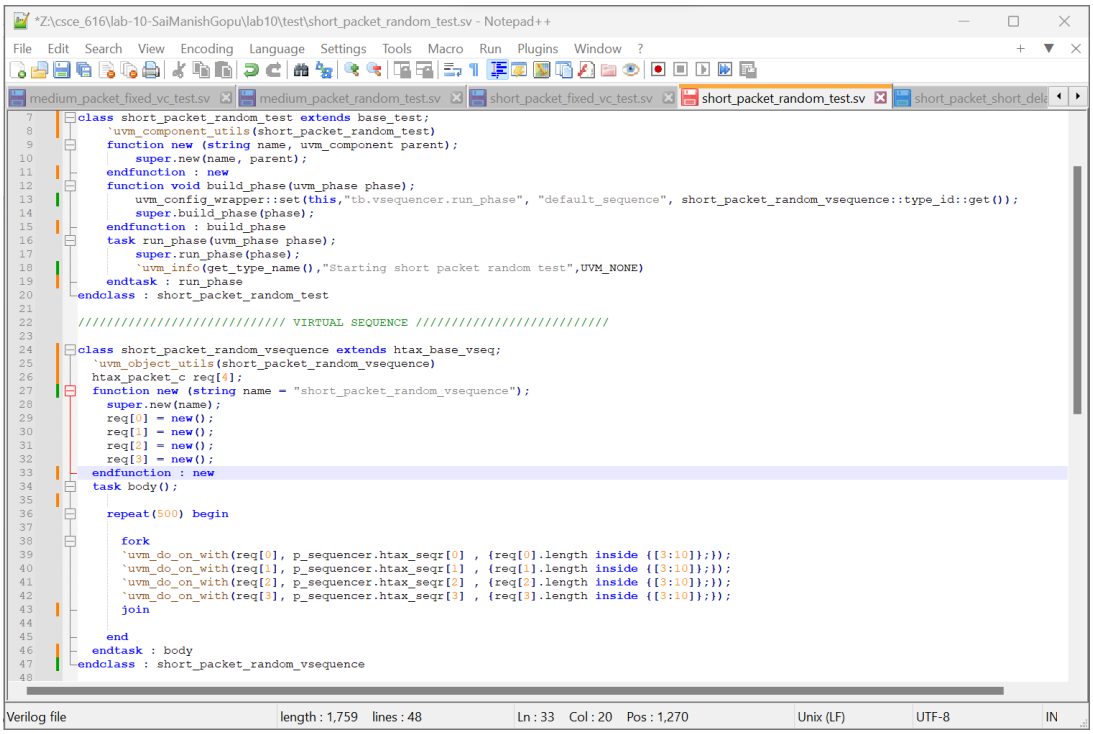
CSCE 616 Lab 10

HTAX Regressions

Name: Sai Manish Gopu UIN: 334003269

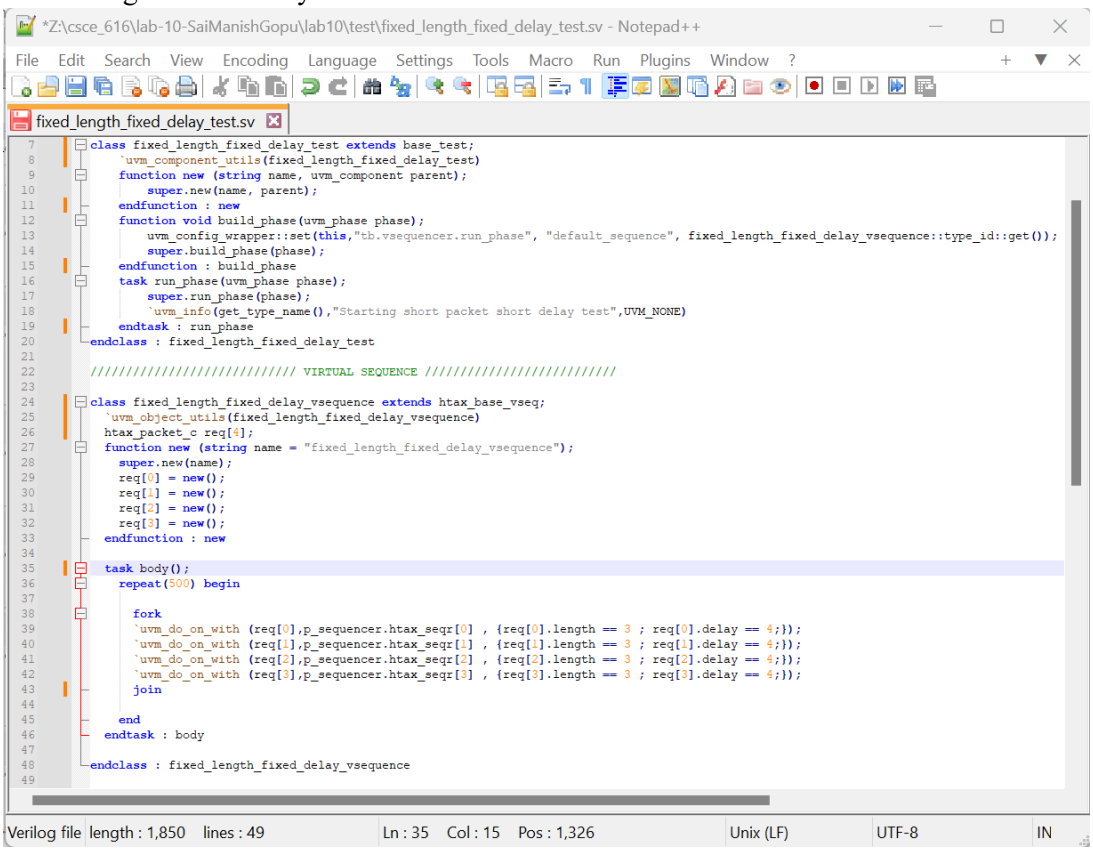
TEST CASES

Short Packet Random Test



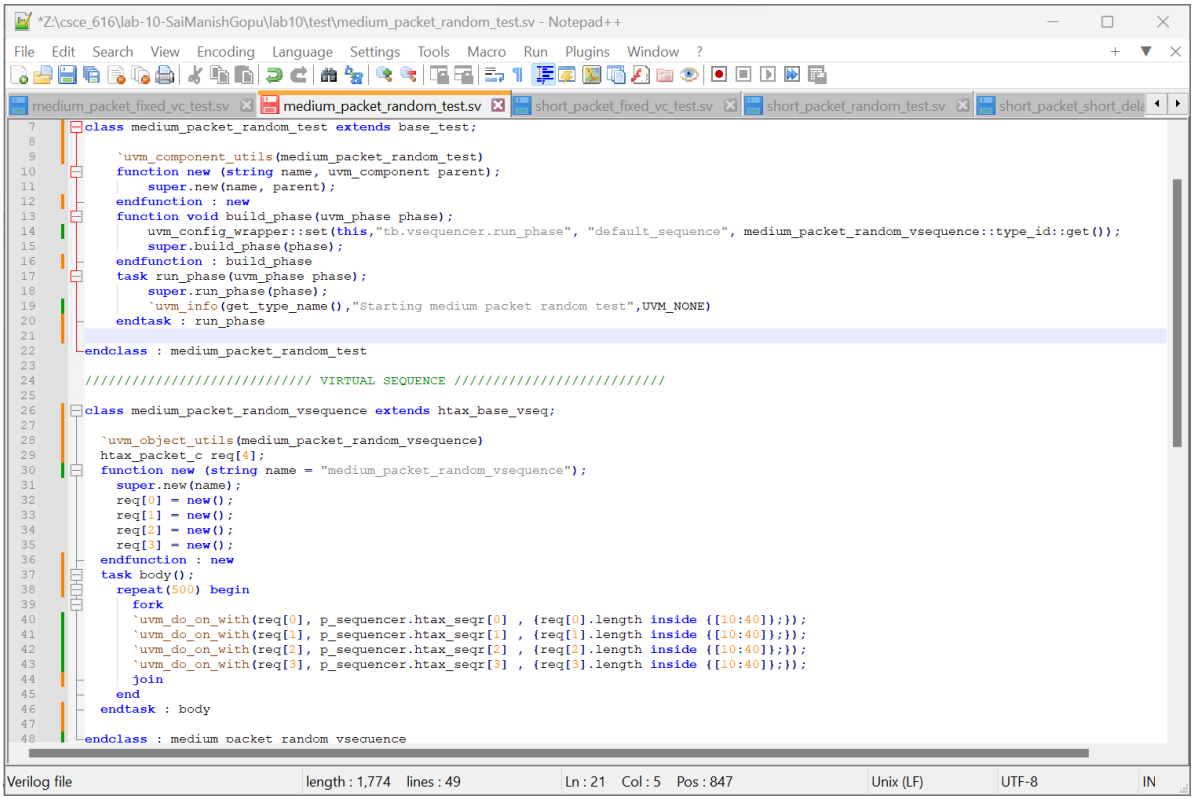
```
class short_packet_random_test extends base_test;
    `uvm_component_utils(short_packet_random_test)
    function new (string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "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_phase
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 req[4];
  function new (string name = "short_packet_random_vsequence");
    super.new(name);
    req[0] = new();
    req[1] = new();
    req[2] = new();
    req[3] = new();
  endfunction : new
  task body();

    repeat(500) begin

      fork
      `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0] , {req[0].length inside {[3:10]};});
      `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1] , {req[1].length inside {[3:10]};});
      `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2] , {req[2].length inside {[3:10]};});
      `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3] , {req[3].length inside {[3:10]};});
      join

    end
  endtask : body
endclass : short_packet_random_vsequence
```

Fixed Length Fixed Delay Test

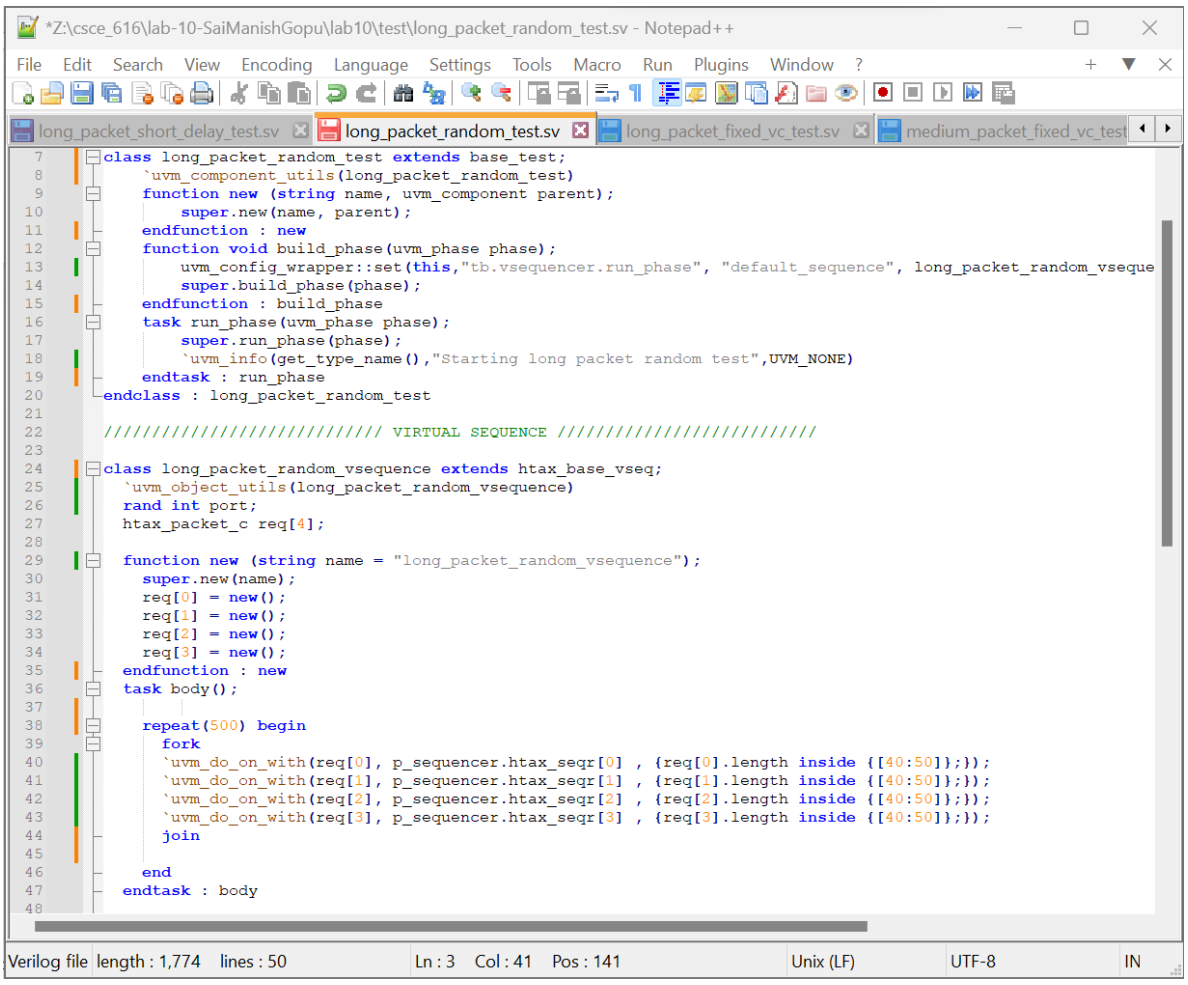


```
class fixed_length_fixed_delay_test extends base_test;
    `uvm_component_utils(fixed_length_fixed_delay_test)
    function new (string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence", fixed_length_fixed_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 short packet short delay test",UVM_NONE)
    endtask : run_phase
endclass : fixed_length_fixed_delay_test

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

class fixed_length_fixed_delay_vsequence extends htax_base_vseq;
  `uvm_object_utils(fixed_length_fixed_delay_vsequence)
  htax_packet_c req[4];
  function new (string name = "fixed_length_fixed_delay_vsequence");
    super.new(name);
    req[0] = new();
    req[1] = new();
    req[2] = new();
    req[3] = new();
  endfunction : new

  task body();
    repeat(500) begin

      fork
      `uvm_do_on_with (req[0],p_sequencer.htax_seqr[0] , {req[0].length == 3 ; req[0].delay == 4;});
      `uvm_do_on_with (req[1],p_sequencer.htax_seqr[1] , {req[1].length == 3 ; req[1].delay == 4;});
      `uvm_do_on_with (req[2],p_sequencer.htax_seqr[2] , {req[2].length == 3 ; req[2].delay == 4;});
      `uvm_do_on_with (req[3],p_sequencer.htax_seqr[3] , {req[3].length == 3 ; req[3].delay == 4;});
      join

    end
  endtask : body

endclass : fixed_length_fixed_delay_vsequence
```

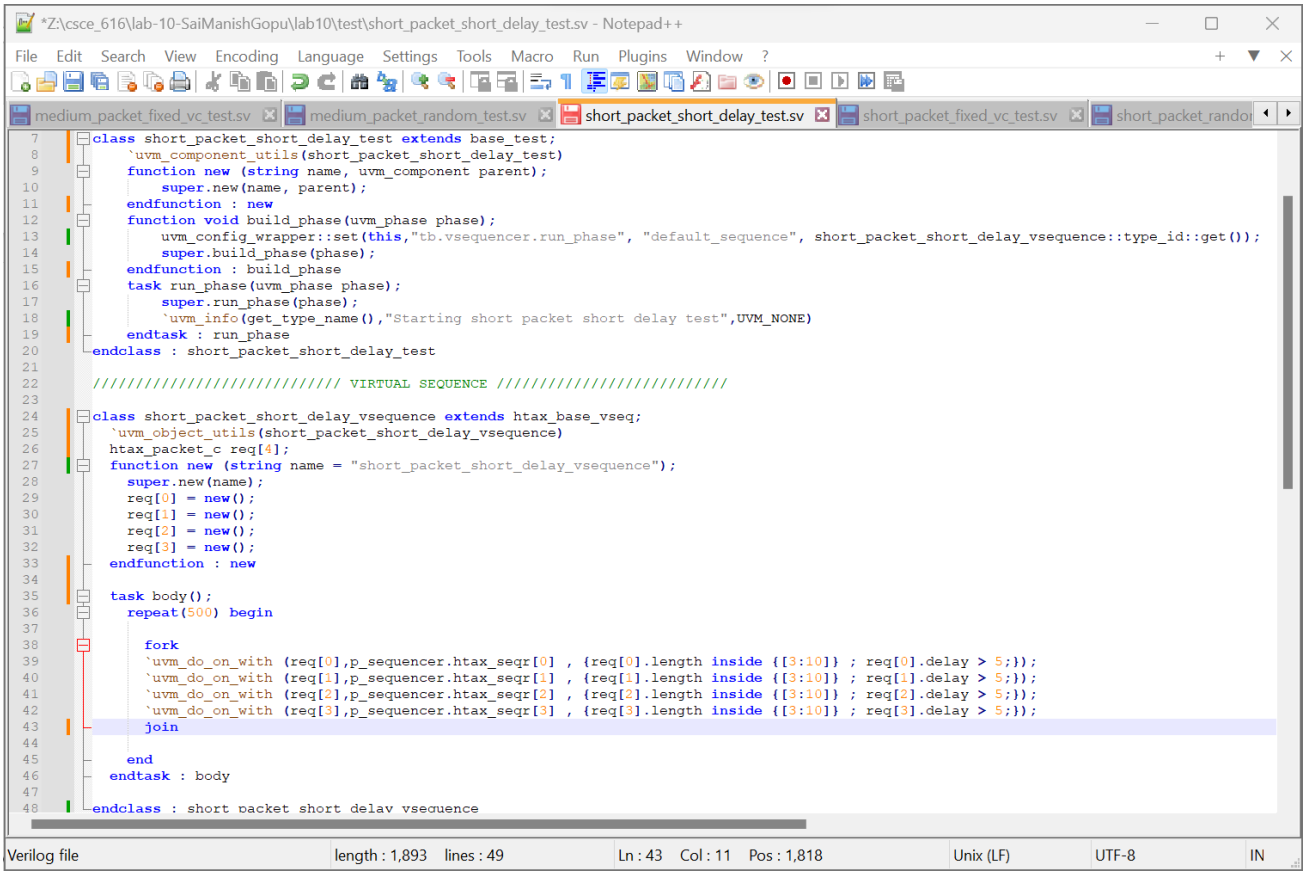
Medium Packet Random Test



Long Packet Random Test

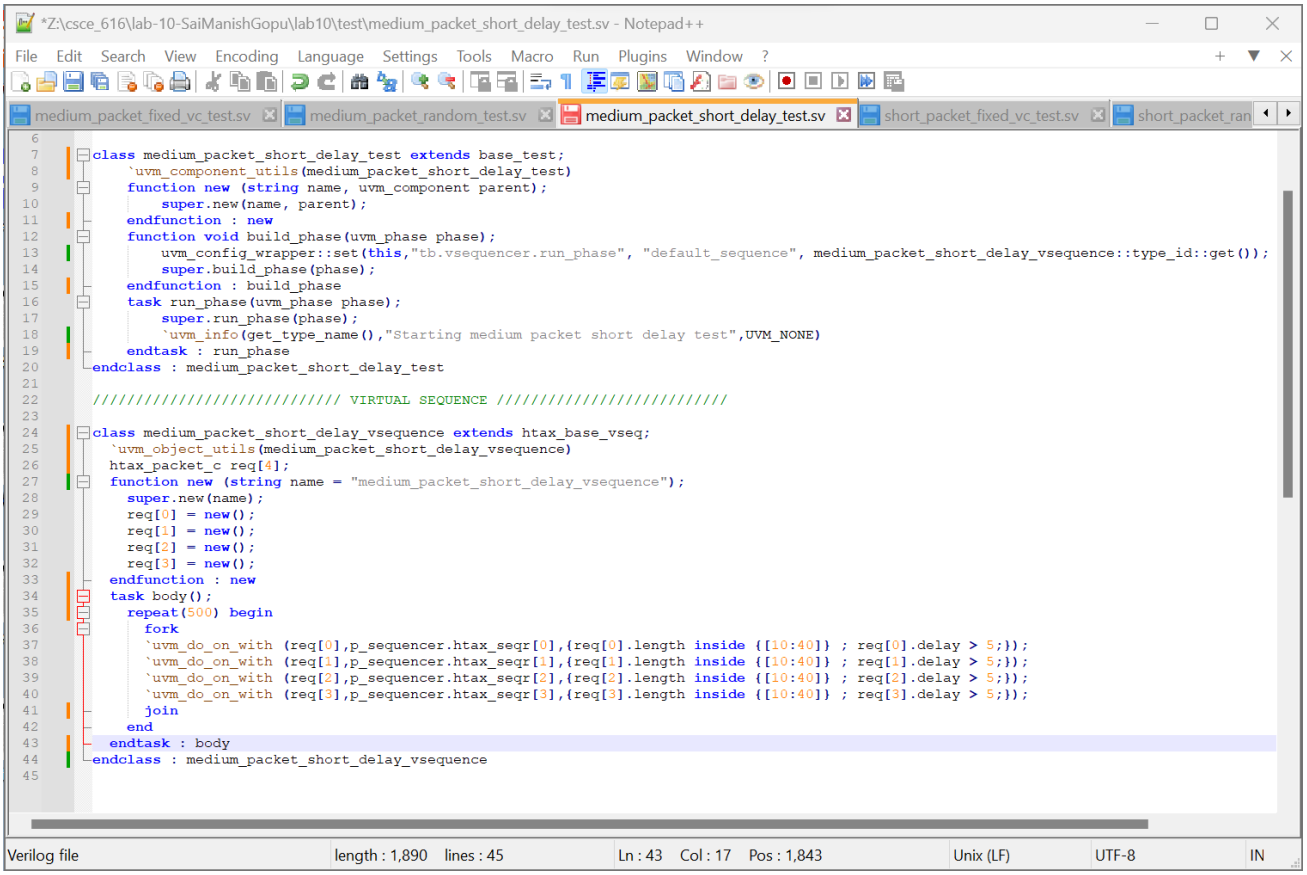


Short Packet Short Delay Test



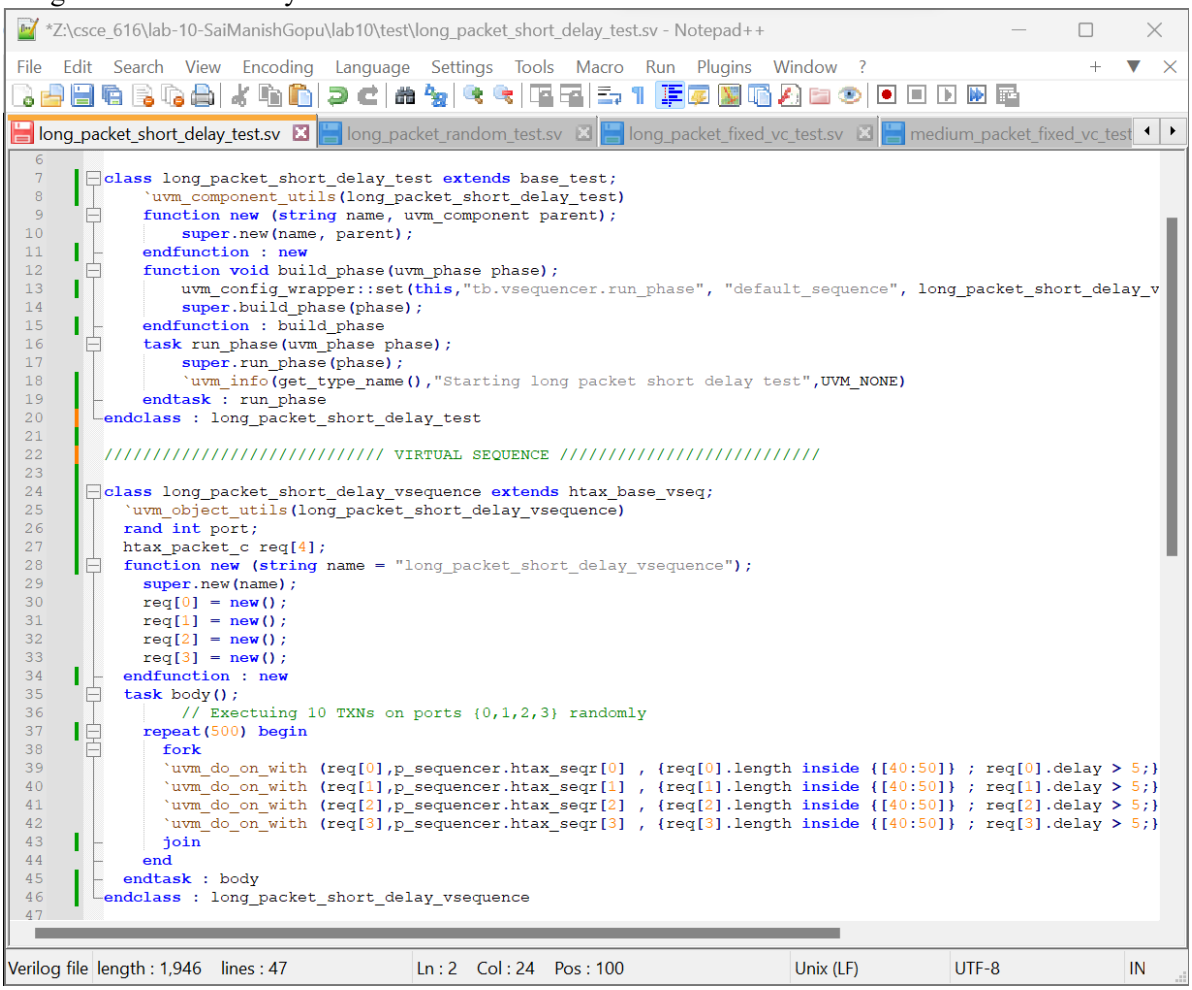
```
class short_packet_short_delay_test extends base_test;
    `uvm_component_utils(short_packet_short_delay_test)
    function new (string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence", short_packet_short_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 short packet short delay test",UVM_NONE)
    endtask : run_phase
endclass : short_packet_short_delay_test

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

class short_packet_short_delay_vsequence extends htax_base_vseq;
  `uvm_object_utils(short_packet_short_delay_vsequence)
  htax_packet_c req[4];
  function new (string name = "short_packet_short_delay_vsequence");
    super.new(name);
    req[0] = new();
    req[1] = new();
    req[2] = new();
    req[3] = new();
  endfunction : new

  task body();
    repeat(500) begin

      fork
      `uvm_do_on_with (req[0],p_sequencer.htax_seqr[0] , {req[0].length inside {[3:10]} ; req[0].delay > 5;});
      `uvm_do_on_with (req[1],p_sequencer.htax_seqr[1] , {req[1].length inside {[3:10]} ; req[1].delay > 5;});
      `uvm_do_on_with (req[2],p_sequencer.htax_seqr[2] , {req[2].length inside {[3:10]} ; req[2].delay > 5;});
      `uvm_do_on_with (req[3],p_sequencer.htax_seqr[3] , {req[3].length inside {[3:10]} ; req[3].delay > 5;});
      join

    end
  endtask : body

endclass : short_packet_short_delay_vsequence
```

Medium Packet Short Delay Test



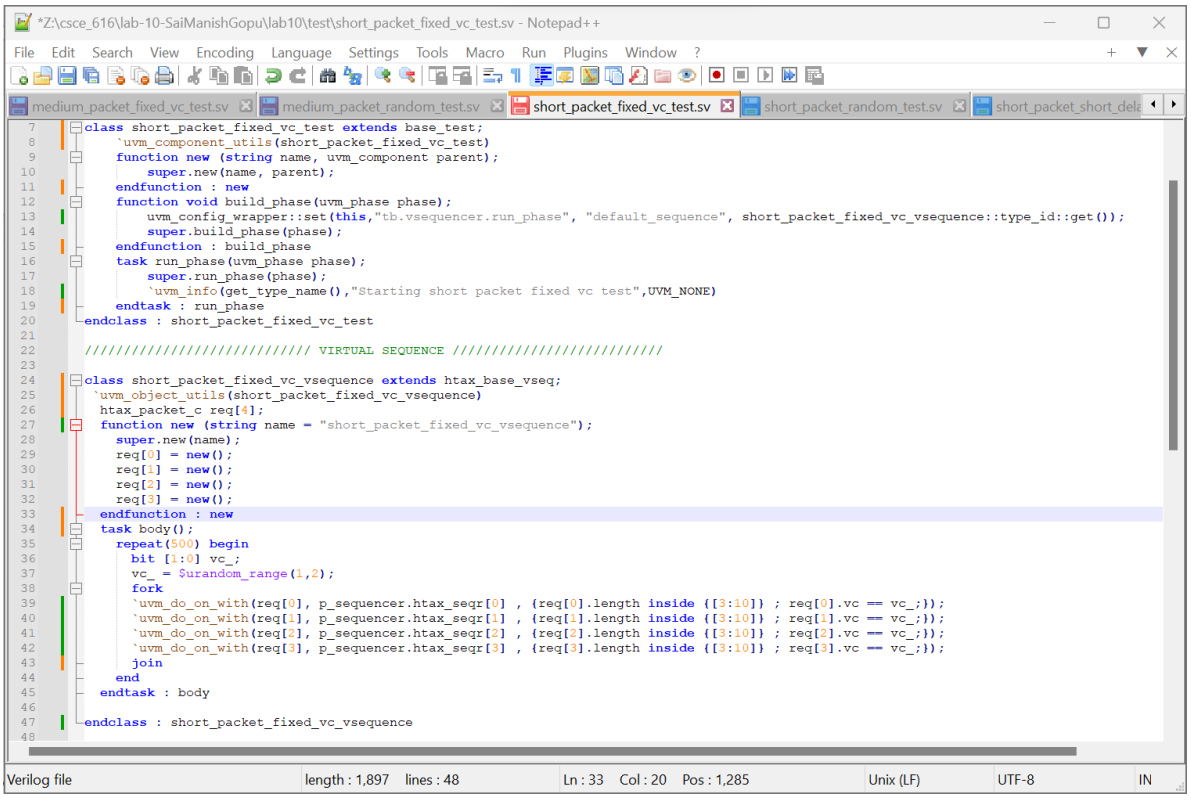
```
class medium_packet_short_delay_test extends base_test;
    `uvm_component_utils(medium_packet_short_delay_test)
    function new (string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence", medium_packet_short_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 medium packet short delay test",UVM_NONE)
    endtask : run_phase
endclass : medium_packet_short_delay_test

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

class medium_packet_short_delay_vsequence extends htax_base_vseq;
  `uvm_object_utils(medium_packet_short_delay_vsequence)
  htax_packet_c req[4];
  function new (string name = "medium_packet_short_delay_vsequence");
    super.new(name);
    req[0] = new();
    req[1] = new();
    req[2] = new();
    req[3] = new();
  endfunction : new
  task body();
    repeat(500) begin
      fork
      `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;});
      join
    end
  endtask : body
endclass : medium_packet_short_delay_vsequence
```

Long Packet Short Delay Test



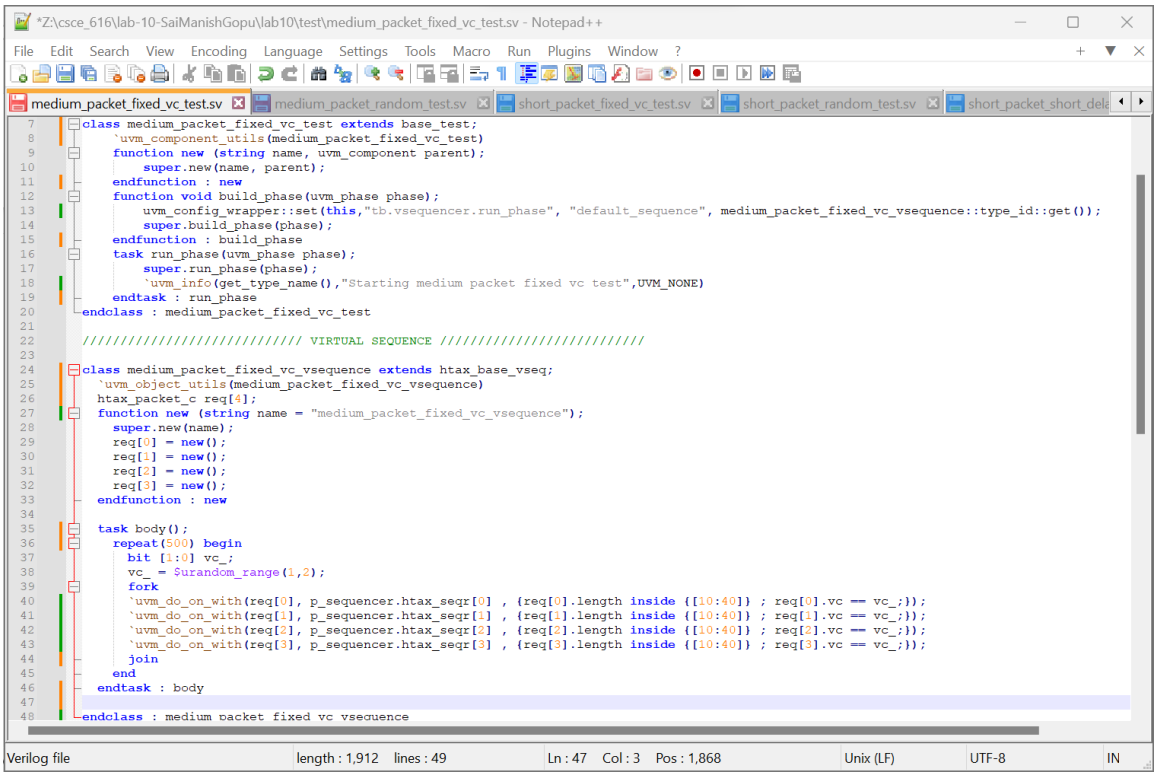
```
class long_packet_short_delay_test extends base_test;
    `uvm_component_utils(long_packet_short_delay_test)
    function new (string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence", long_packet_short_delay_v
        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 short delay test",UVM_NONE)
    endtask : run_phase
endclass : long_packet_short_delay_test

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

class long_packet_short_delay_vsequence extends htax_base_vseq;
  `uvm_object_utils(long_packet_short_delay_vsequence)
  rand int port;
  htax_packet_c req[4];
  function new (string name = "long_packet_short_delay_vsequence");
    super.new(name);
    req[0] = new();
    req[1] = new();
    req[2] = new();
    req[3] = new();
  endfunction : new
  task body();
      // Exectuing 10 TXNs on ports {0,1,2,3} randomly
    repeat(500) begin
      fork
      `uvm_do_on_with (req[0],p_sequencer.htax_seqr[0] , {req[0].length inside {[40:50]} ; req[0].delay > 5;}
      `uvm_do_on_with (req[1],p_sequencer.htax_seqr[1] , {req[1].length inside {[40:50]} ; req[1].delay > 5;}
      `uvm_do_on_with (req[2],p_sequencer.htax_seqr[2] , {req[2].length inside {[40:50]} ; req[2].delay > 5;}
      `uvm_do_on_with (req[3],p_sequencer.htax_seqr[3] , {req[3].length inside {[40:50]} ; req[3].delay > 5;}
      join
    end
  endtask : body
endclass : long_packet_short_delay_vsequence
```

Short Packet Fixed VC Test

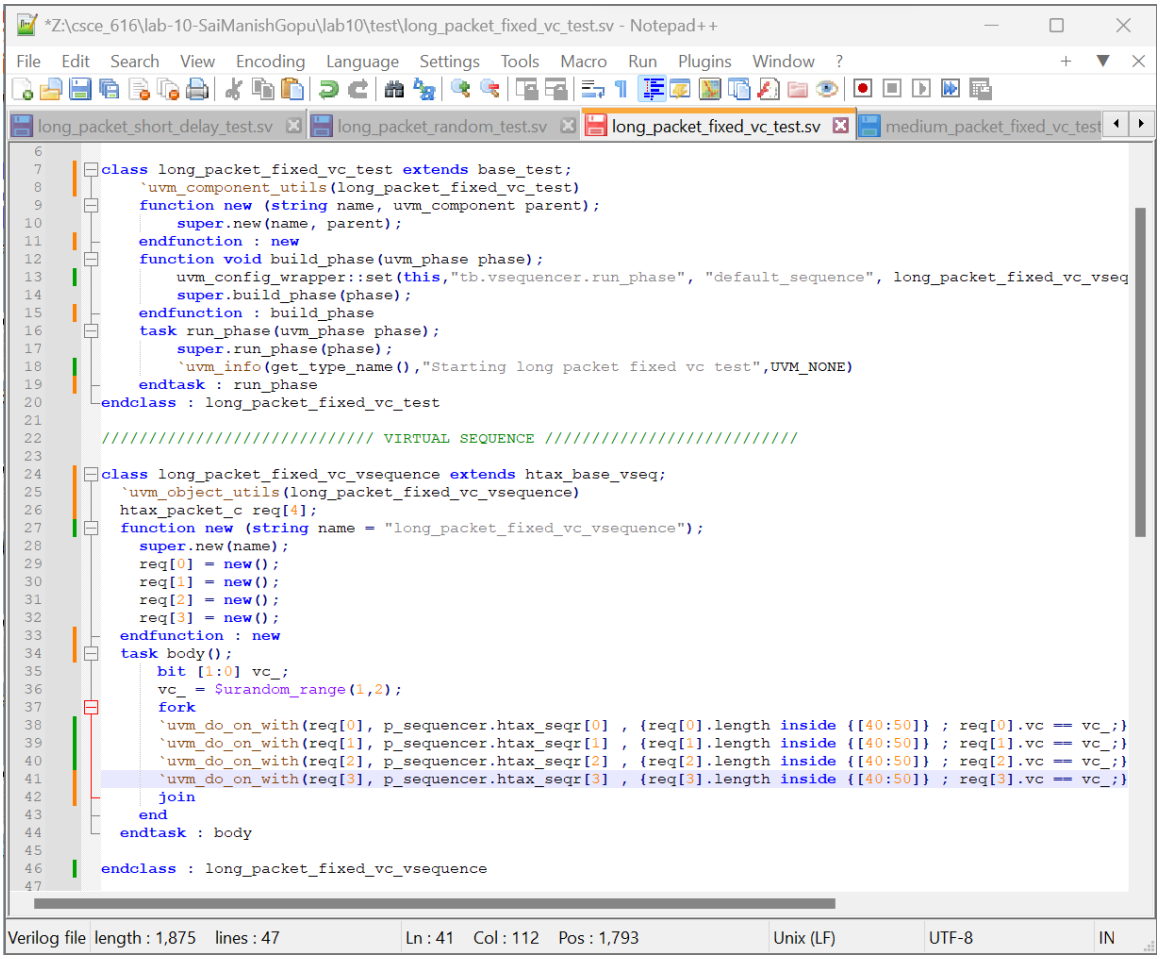


```
class short_packet_fixed_vc_test extends base_test;
    `uvm_component_utils(short_packet_fixed_vc_test)
    function new (string name, uvm_component parent);
        super.new(name, parent);
    endfunction : new
    function void build_phase(uvm_phase phase);
        uvm_config_wrapper::set(this,"tb.vsequencer.run_phase", "default_sequence", short_packet_fixed_vc_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 fixed vc test",UVM_NONE)
    endtask : run_phase
endclass : short_packet_fixed_vc_test

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

class short_packet_fixed_vc_vsequence extends htax_base_vseq;
  `uvm_object_utils(short_packet_fixed_vc_vsequence)
  htax_packet_c req[4];
  function new (string name = "short_packet_fixed_vc_vsequence");
    super.new(name);
    req[0] = new();
    req[1] = new();
    req[2] = new();
    req[3] = new();
  endfunction : new
  task body();
    repeat(500) begin
      bit [1:0] vc_;
      vc_ = $urandom_range(1,2);
      fork
      `uvm_do_on_with(req[0], p_sequencer.htax_seqr[0] , {req[0].length inside {[3:10]} ; req[0].vc == vc_;});
      `uvm_do_on_with(req[1], p_sequencer.htax_seqr[1] , {req[1].length inside {[3:10]} ; req[1].vc == vc_;});
      `uvm_do_on_with(req[2], p_sequencer.htax_seqr[2] , {req[2].length inside {[3:10]} ; req[2].vc == vc_;});
      `uvm_do_on_with(req[3], p_sequencer.htax_seqr[3] , {req[3].length inside {[3:10]} ; req[3].vc == vc_;});
      join
    end
  endtask : body

endclass : short_packet_fixed_vc_vsequence
```

Medium Packet Fixed VC Test

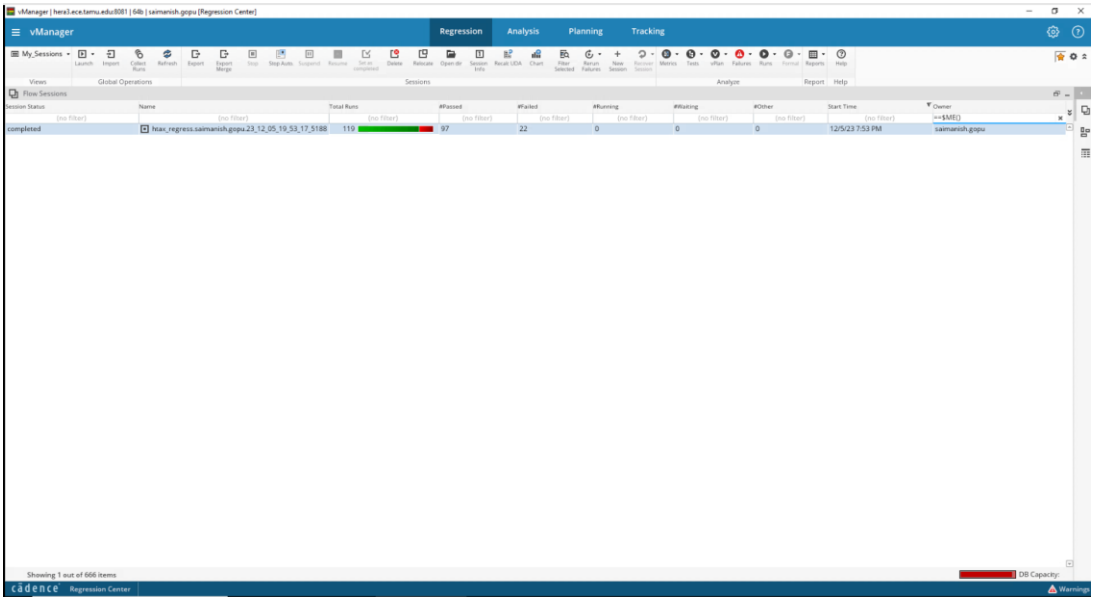


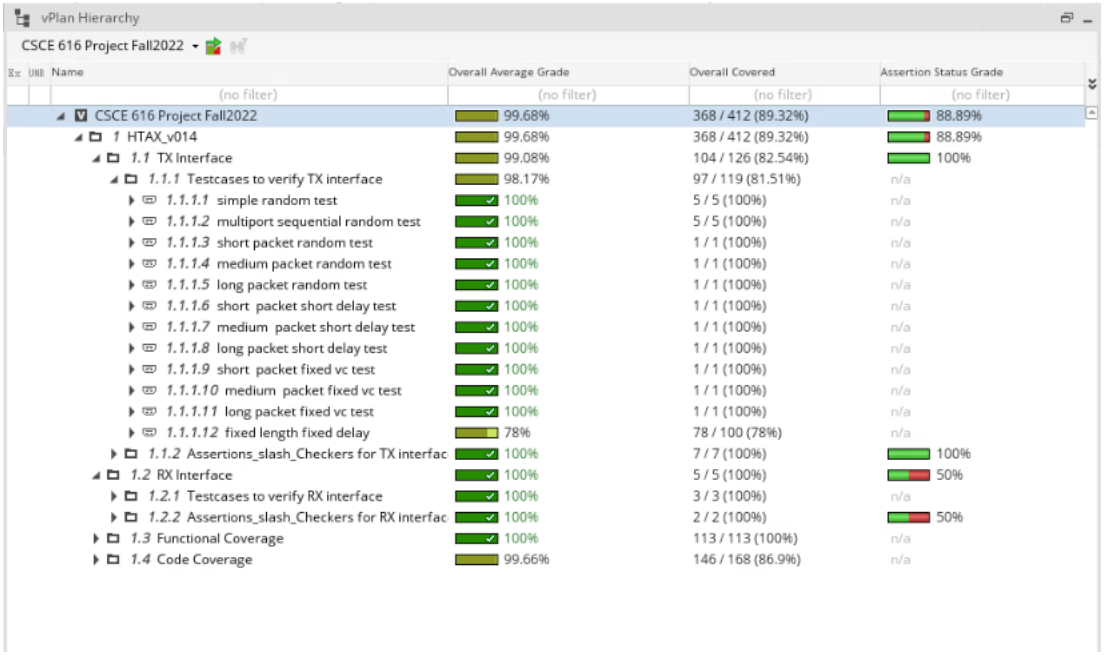
Long Packet Fixed VC Test



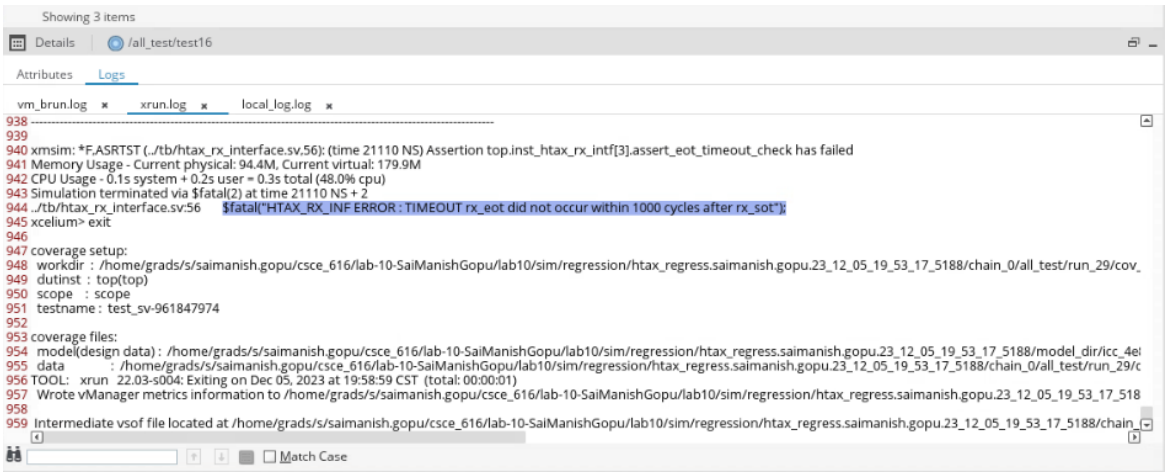
BUG REPORT

Regression Summary

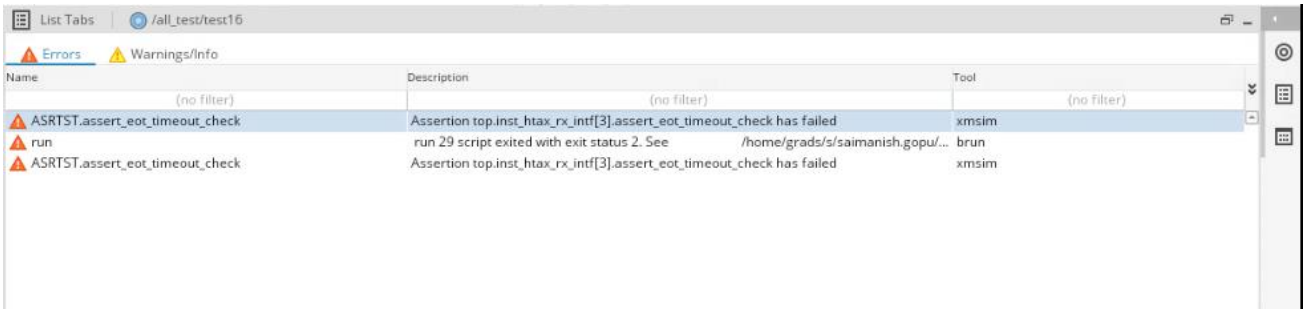




Xrun.log



Error Message



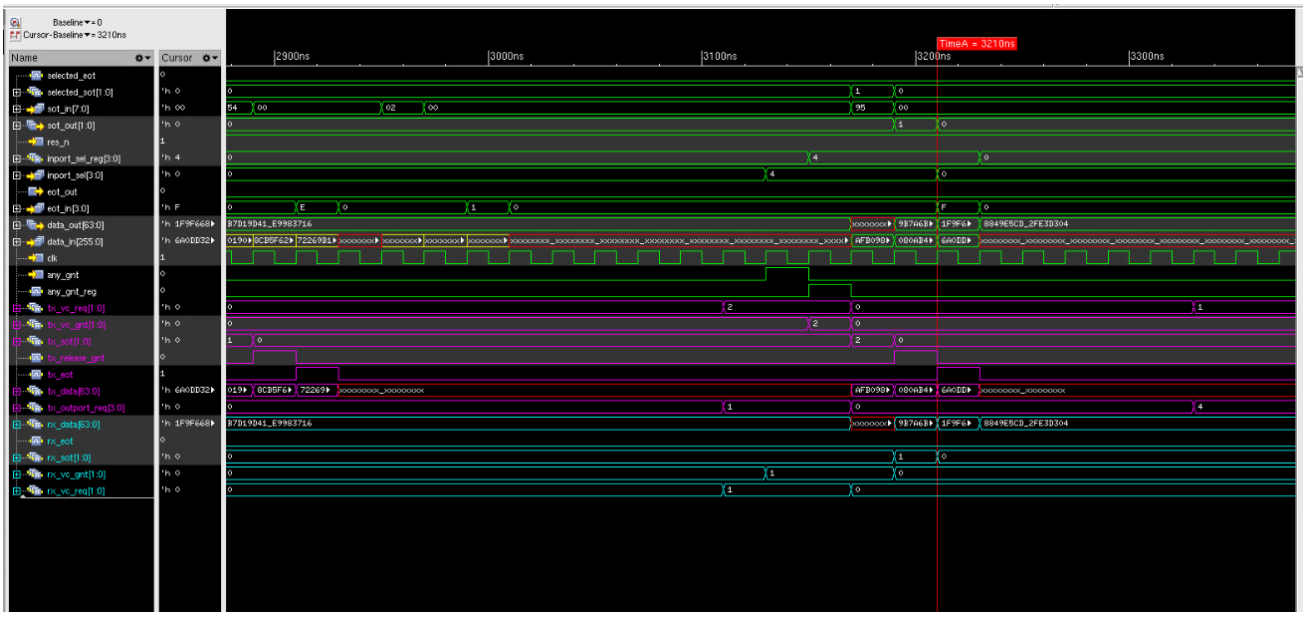
Debug Process

I ran multiple regressions with random seed number. Finally for the seed number 961847974 the test case fixed_length_fixed_delay_test failed and all other tests passed.

When the test case fixed_length_fixed_delay_test is ran isolated with the same seed number. It gave below error message. It says that the assertion assert_eot_timeout_check has failed.

```
xmsim: *F,ASRTST (../tb/htax_rx_interface.sv,56): (time 23210 NS) Assertion top.inst_htax_rx_intf[3].assert_eot_timeout_check has failed
Memory Usage - Current physical: 92.6M, Current virtual: 175.9M
CPU Usage - 0.1s system + 0.2s user = 0.3s total (67.0% cpu)
Simulation terminated via $fatal(2) at time 23210 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_sv961847974
```

To analyse this, I opened the simvision waveform which came out as below,



In the above waveform,

The signal rx_eot has not rised eventhough the transaction is tx_eot is raised which indicates the end of transaction. This could the reason for the assertion failure. When traced, it is found that the tx_eot signal value comes from signal eot_out from the htax_outport_mux_data from design folder. When the

htax_outport_mux_data is analysed , I came to know that the eot_out is coming from another signal selected_eot and selected_eot signal is assigned the value as selected_eot = |(eot_in & inport_sel_reg) & ~(&(eot_in)); if the if condition is satisfied i.e., true. But ideally this should not be the case.

It appears that the RX side receives the rx_sot signal, and data becomes visible, but the rx_eot signal fails to appear. The absence of rx_eot is attributed to the added term &~(&(eot_in)) in the assignment of selected_eot. This issue arises when all RX ports simultaneously request a transaction, and the packet length and delay are identical. In this scenario, all bits in eot_in[3:0] are expected to be high (indicated as 'F' in the waveform). However, as eot_in[3:0] goes high, ~(&(eot_in)) goes low. Consequently, when this low value is combined with other terms through bitwise AND, the result is zero.

The signals eot_out and selected_eot from the htax_outport_mux_data design file is as below

```
    // Muxes for data, eot, sot
    `ifdef ASYNC_RES
    always @(posedge clk or negedge res_n) `else
    always @(posedge clk) `endif
    begin
        if (!res_n)
        begin
            data_out <= {WIDTH {1'b0}};
            sot_out <= {VC {1'b0}};
            eot_out <= 1'b0;
        end
        else
            begin
            (* full_case *) (* parallel_case *)
            casex (inport_sel_reg)
                4'b1xxx: data_out <= data_in[((4*WIDTH)-1):(3*WIDTH)];
                4'bx1xx: data_out <= data_in[((3*WIDTH)-1):(2*WIDTH)];
                4'bxx1x: data_out <= data_in[((2*WIDTH)-1):(1*WIDTH)];
                4'bxxx1: data_out <= data_in[((1*WIDTH)-1):(0*WIDTH)];
            endcase
            if ((|inport_sel && !any_gnt) )
                sot_out <= selected_sot;
            else
                sot_out <=  {VC{1'b0}};
            if (|inport_sel_reg && (!eot_out || (|selected_sot && (|inport_sel && !any_gnt) )))
                eot_out <= selected_eot;
            else
                eot_out <=  1'b0;
        end
    end

endmodule
```

```
    reg                 any_gnt_reg;
    reg [NUM_PORTS-1:0]   inport_sel_reg;
    reg [VC-1:0]selected_sot;
    wire        selected_eot;

    always @( * )
    begin
        (* 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
    if (!res_n) begin
      inport_sel_reg <= {NUM_PORTS{1'b0}};
      any_gnt_reg    <= 0;
     end else begin
      any_gnt_reg <= any_gnt;

      //if(any_gnt_reg)
        inport_sel_reg <= inport_sel;
     end
    end
```

Recommended Fix for the Bug:

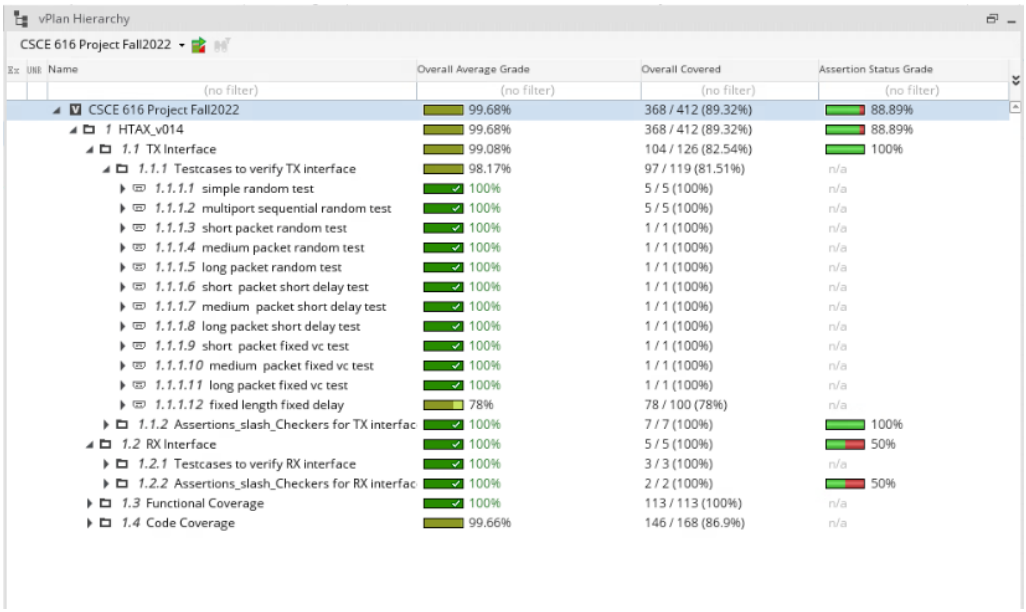
The bug can fixed by making modification to the selected_eot assign statement by removing ~(&(eot_in)).

Therefore, the correct logic for selected_eot is

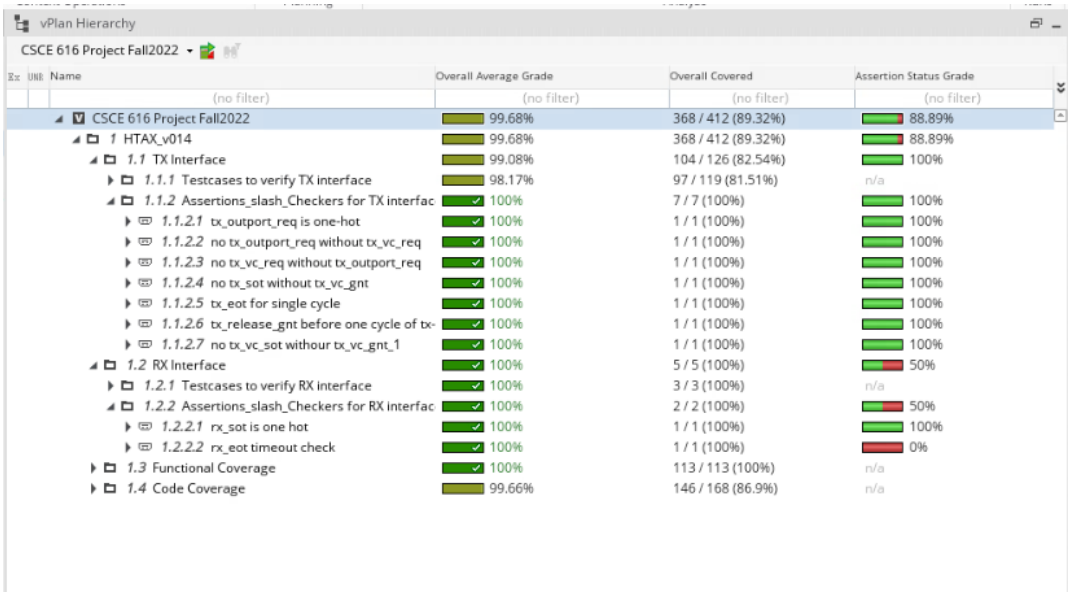
assign selected_eot = |(eot_in & inport_sel_reg);

COVERAGE REPORT

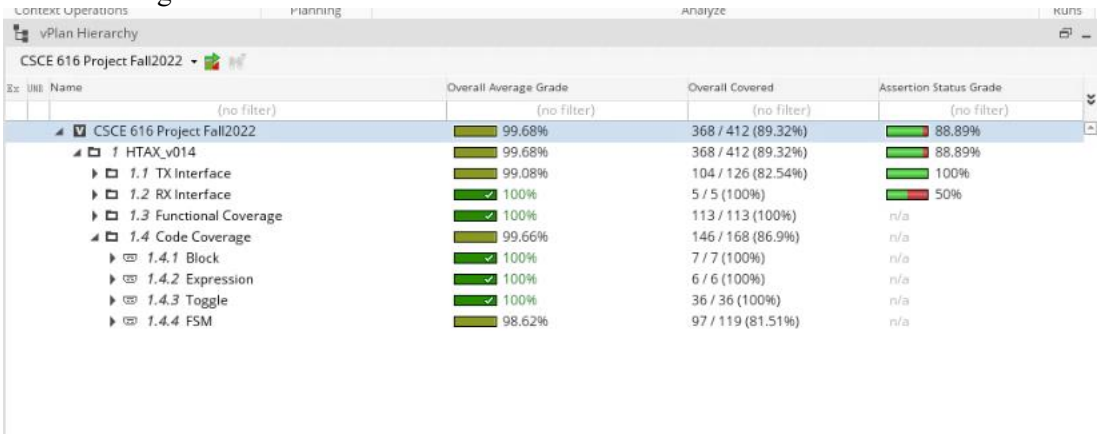
Test Coverage



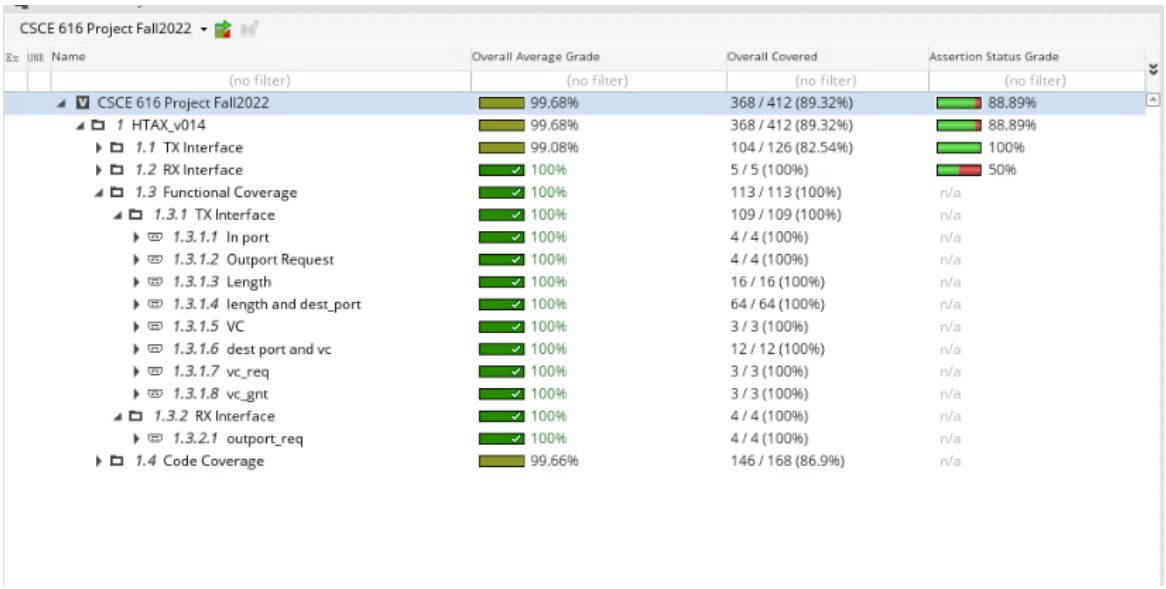
Assertion Coverage of TX and RX:



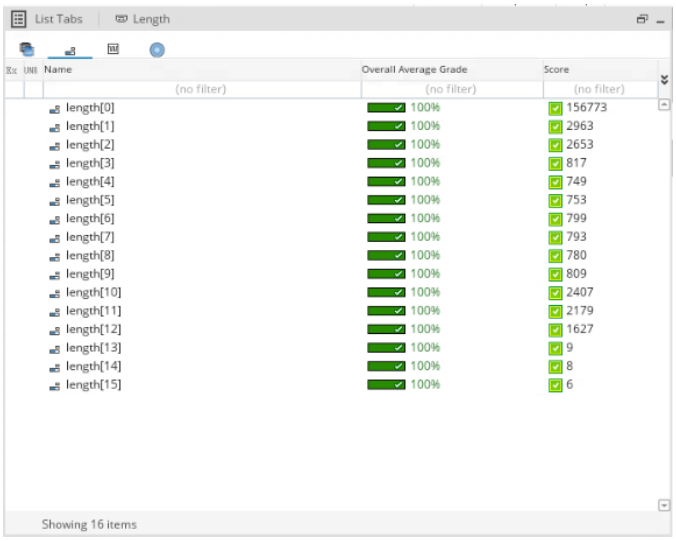
Code Coverage



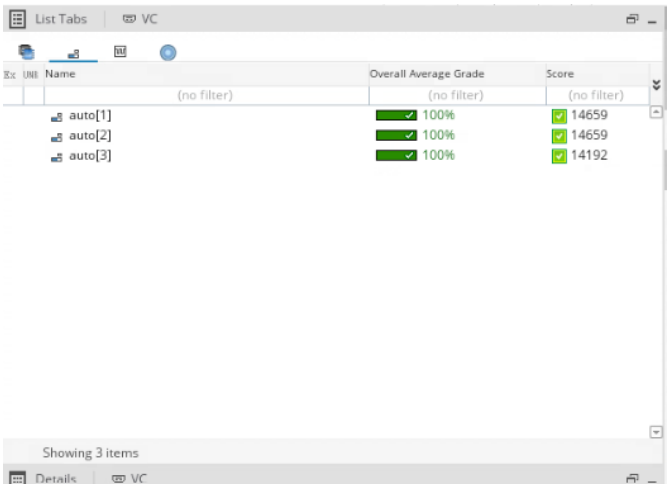
Functional Coverage



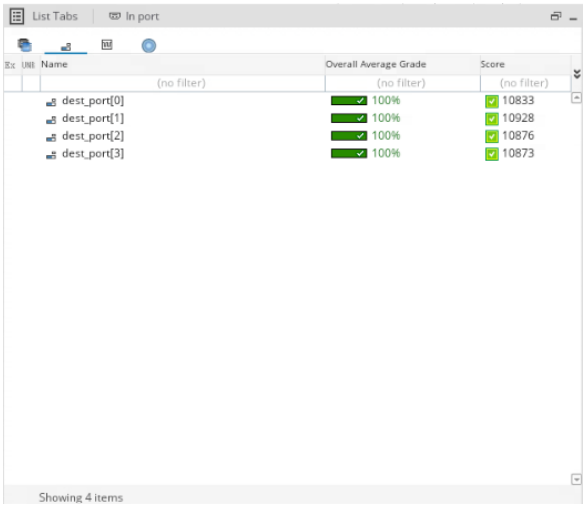
Length



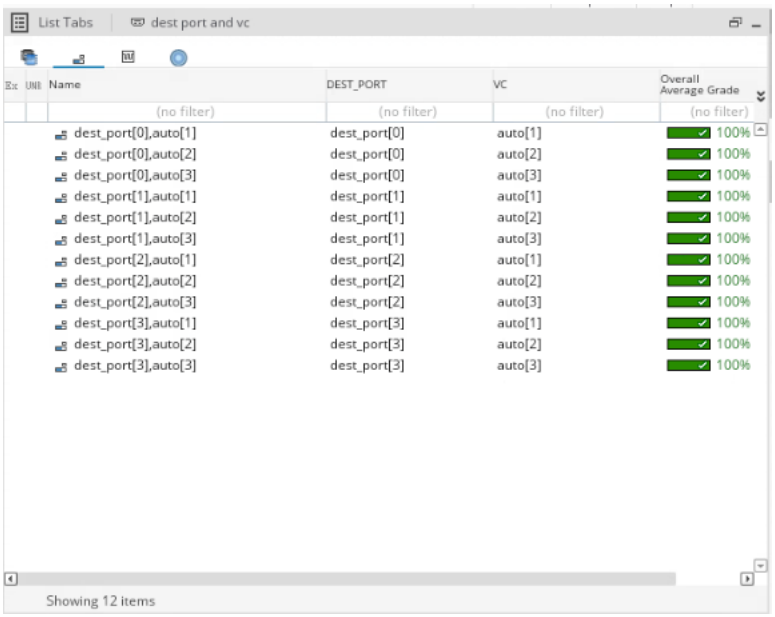
vc



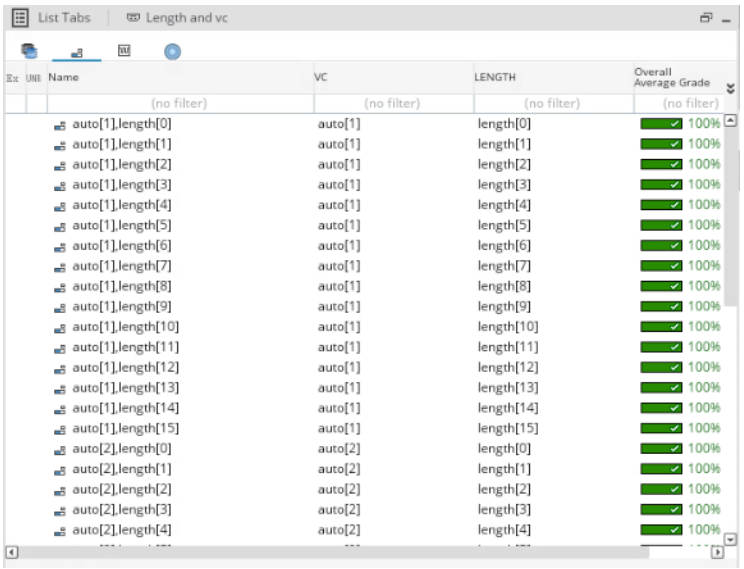
dest_port



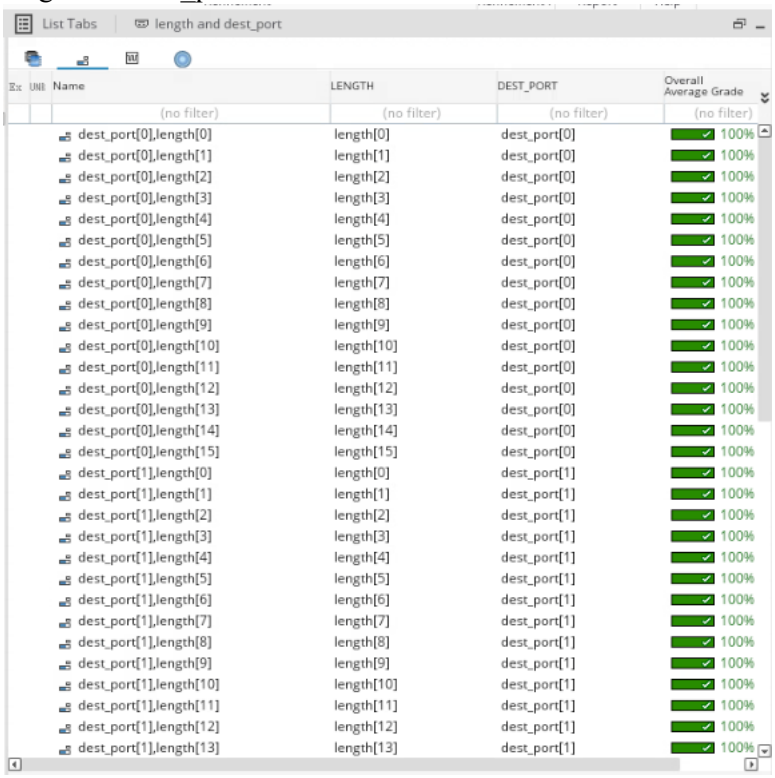
dest_port and vc



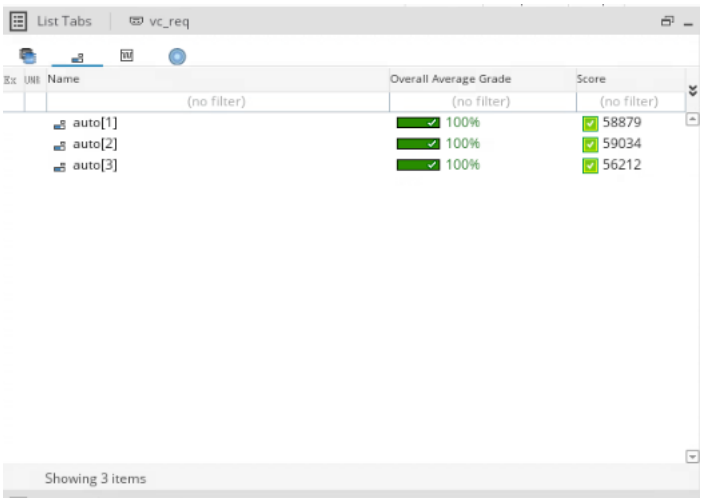
length and vc



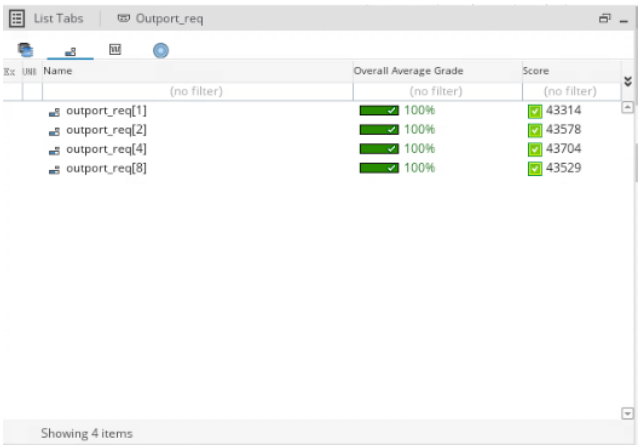
length and dest_port



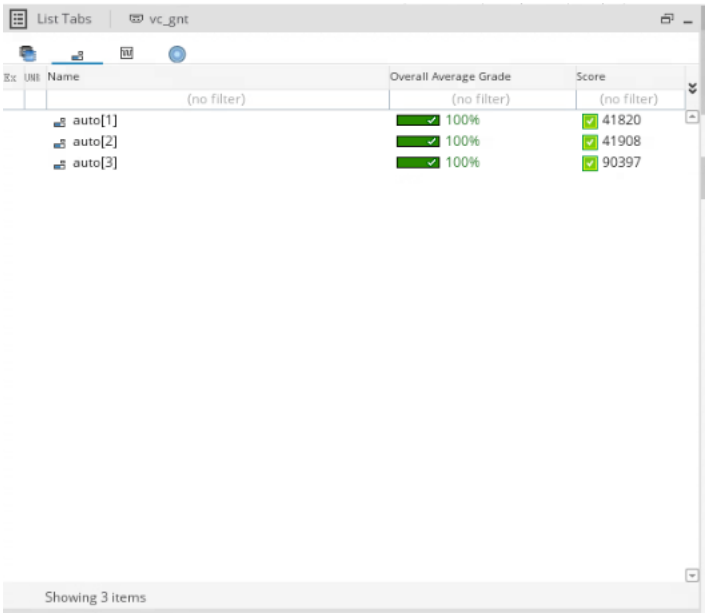
vc_req



outport_req



vc_gnt


List Tabs
vc_gnt
Name
Overall Average Grade
Score
(no filter)
(no filter)
(no filter)
auto[1]
100%
41820
auto[2]
100%
41908
auto[3]
100%
90397
Showing 3 items