LPFFIR Easier UVM

Author: Vladimir Armstrong
vladimirarmstrong@opencores.org

Rev. 1.0
April 27, 2019

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
1.0	04/27/2019	Vladimir Armstrong	First Draft

Contents

1 Introduction

This document describes the verification of LPFFIR RTL module [1] by using the Easier UVM Code Generator [2]. The verification flow has 4 basic steps and is shown in Figure 1; starting with UVM architecture specifications Figure 2 from which templet files [3] are created and are used as input to Perl script that generates System Verilog UVM testbench Figure 3.

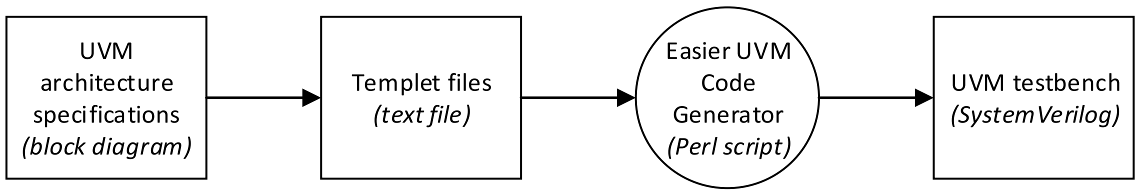
UVM architecture specifications (block diagram)
Templet files (text file)
Easier UVM Code Generator (Perl script)
UVM testbench (SystemVerilog)

Figure 1 Easier UVM verification flow.

2

UVM Architecture Specifications

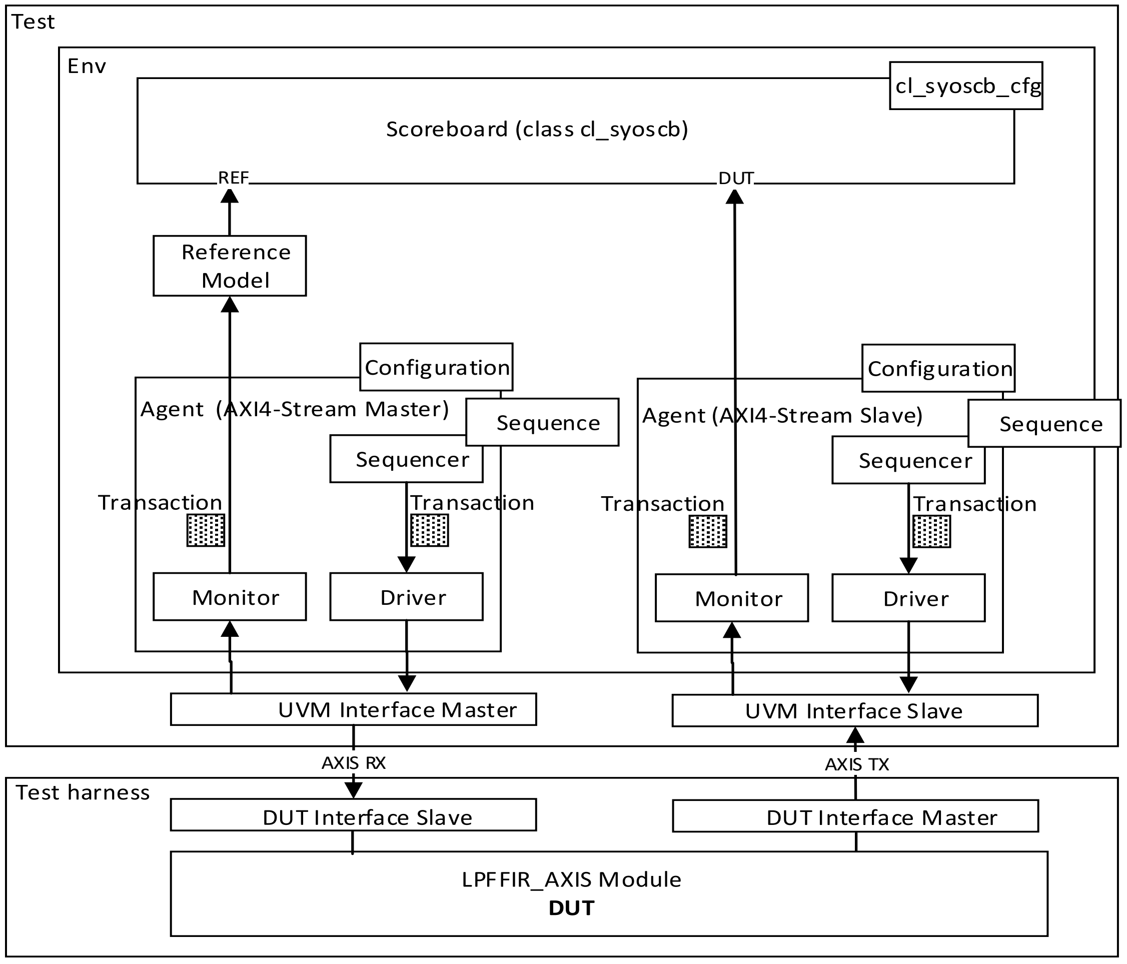

Figure 2 UVM architecture specifications.

3

Templet Files

common.tpl

```
dut_top                 = lpffir_axis
nested_config_objects = yes

tb_prepend_to_initial = vcd_dump.sv inline

#Path ignored on EDA Playground
#syosil_scoreboard_src_path = ../../syosil/src

ref_model_input  = reference m_data_input_agent

ref_model_output = reference m_data_output_agent

ref_model_compare_method    = reference iop

ref_model_inc_inside_class  = reference reference_inc_inside_class.sv
inline
ref_model_inc_after_class   = reference reference_inc_after_class.sv
inline

top_default_seq_count = 10

uvm_cmdline = +UVM_VERBOSITY=UVM_HIGH
```

data_input.tpl

```
agent_name = data_input

number_of_instances = 1

trans_item = input_tx
trans_var  = rand logic [15:0] data;

trans_var  = constraint c_data { 0 <= data; data < 128; }

driver_inc_inside_class = data_input_driver_inc_inside_class.sv  inline
driver_inc_after_class  = data_input_driver_inc_after_class.sv   inline
monitor_inc             = data_input_do_mon.sv                   inline
agent_cover_inc         = data_input_cover_inc.sv                inline

if_port  = logic last;
if_port  = logic valid;
if_port  = logic ready;
if_port  = logic [15:0] data;
if_port  = logic clk;
if_port  = logic reset;
if_clock = clk
if_reset = reset
```

data_output.tpl

```
agent_name = data_output

number_of_instances = 1

trans_item = output_tx
trans_var  = rand logic [15:0] data;

agent_coverage_enable = no

driver_inc_inside_class = data_output_driver_inc_inside_class.sv  inline
driver_inc_after_class  = data_output_driver_inc_after_class.sv   inline
monitor_inc = data_output_do_mon.sv inline

if_port  = logic last;
if_port  = logic valid;
if_port  = logic ready;
if_port  = logic [15:0] data;
if_port  = logic clk;
if_port  = logic reset;
if_clock = clk
if_reset = reset
```

pinlist

```
!data_input_if_0
rx_tlast_i last
rx_tvalid_i valid
rx_tready_o ready
rx_tdata_i data

!data_output_if_0
tx_tlast_o last
tx_tvalid_o valid
tx_tready_i ready
tx_tdata_o data

!
aclk_i  clock
aresetn_i reset
```

data_input_cover_inc.sv

```
covergroup m_cov;
  option.per_instance = 1;

  cp_data: coverpoint m_item.data {
    bins data_values[] = {[0:127]};
  }
endgroup
```

data_input_do_mon.sv

```
task data_input_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    wait (vif.reset == 1);
    if (vif.valid && vif.ready)
    begin
      m_trans.data = vif.data;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(), $sformatf("Input data = %0d",
m_trans.data), UVM_HIGH)
    end
  end
endtask
```

data_input_driver_inc_after_class.sv

```
task data_input_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever @(posedge vif.clk)
  begin
    seq_item_port.get_next_item(req);
    phase.raise_objection(this);
    wait (vif.reset == 1);
    vif.data <= req.data;
    vif.valid  <= 1;
    vif.last  <= 0;
    wait (vif.ready == 1);

    fork
      begin
        repeat (10) @(posedge vif.clk);
        phase.drop_objection(this);
      end
    join_none
    seq_item_port.item_done();
  end
endtask : run_phase
```

data_input_driver_inc_inside_class.sv

```
extern task run_phase(uvm_phase phase);
```

data_output_do_mon.sv

```
task data_output_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    wait (vif.reset == 1);
    if (vif.valid && vif.ready)
    begin
      m_trans.data = vif.data;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(), $sformatf("Output data =
%0d",m_trans.data), UVM_HIGH)
    end
  end
endtask
```

reference_inc_after_class.sv

```
function void reference::write_reference_0(input_tx t);
  send(t);
endfunction

function void reference::send(input_tx t);
  output_tx tx;
  tx = output_tx::type_id::create("tx");
  if (init_flag == 1)
    begin
      init_flag = 0;
      foreach(tx_save[j])
        tx_save[j] = 0;
    end
  if (save_pnt == 5)
    save_pnt = 0;
  else
  save_pnt++;
  tx_save[save_pnt] = t.data;
  tx.data = tx_save[0] + tx_save[1] + tx_save[2] + tx_save[3] +
tx_save[4] + tx_save[5];
  analysis_port_0.write(tx);
  `uvm_info(get_type_name(), $sformatf("Reference Model save_pnt = %0d,
data = %0d",save_pnt, tx.data), UVM_HIGH)
endfunction
```

reference_inc_inside_class.sv

```
extern function void send(input_tx t);

int save_pnt = 5;
logic [15:0] tx_save [0:5];
int init_flag = 1;
```

data_output_driver_inc_inside_class

```
extern task run_phase(uvm_phase phase);
```

data_output_driver_inc_after_class

```
task data_output_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever @(posedge vif.clk)
  begin
    seq_item_port.get_next_item(req);
    phase.raise_objection(this);

    vif.ready  <= 1;
    wait (vif.reset == 1);

    fork
      begin
        repeat (10) @(posedge vif.clk);
        phase.drop_objection(this);
      end
    join_none
    seq_item_port.item_done();
  end
endtask : run_phase
```

design.sv

```
module lpffir_axis (
                    input                aclk_i,
                    input                aresetn_i,
                    // AXI-Stream RX interface
                    input                rx_tlast_i,
                    input                rx_tvalid_i,
                    output logic         rx_tready_o,
                    input [15:0]         rx_tdata_i,
                    // AXI-Stream TX interface
                    output logic         tx_tlast_o,
                    output reg           tx_tvalid_o,
                    input                tx_tready_i,
                    output logic [15:0]  tx_tdata_o
                    );

   wire lpffir_en;
   assign                       lpffir_en = rx_tvalid_i && tx_tready_i;

   // AXI-Stream interface
   assign rx_tready_o = lpffir_en;
   assign tx_tvalid_o = lpffir_en;
   assign tx_tlast_o  = rx_tlast_i;

  // DEBUG
  always @(posedge aclk_i or negedge aresetn_i)
    if (aresetn_i)
      $display("DUT: rx_tdata_i %0d, tx_tdata_o %0d", rx_tdata_i,
tx_tdata_o);

   // LPFFIR
   lpffir_core lpffir_core(
                           .clk_i(aclk_i),
                           .rstn_i(aresetn_i),
                           .en_i(lpffir_en),
                           .x_i(rx_tdata_i),
                           .y_o(tx_tdata_o)
                           );

endmodule
```

4 Generated UVM Testbench

```
--------------------------------------------------------------------------------
Name                                  Type                            Size  Value
--------------------------------------------------------------------------------
uvm_test_top                          top_test                        -     @347
  m_env                               top_env                         -     @360
    m_converter_m_data_output_agent   uvm_component                   -     @417
      a_port                          uvm_analysis_port               -     @436
      analysis_imp                    uvm_analysis_imp                -     @426
    m_data_input_agent                data_input_agent                -     @455
      analysis_port                   uvm_analysis_port               -     @464
      m_driver                        data_input_driver               -     @552
        rsp_port                      uvm_analysis_port               -     @571
        seq_item_port                 uvm_seq_item_pull_port          -     @561
      m_monitor                       data_input_monitor              -     @532
        analysis_port                 uvm_analysis_port               -     @541
      m_sequencer                     uvm_sequencer                   -     @581
        rsp_export                    uvm_analysis_export             -     @590
        seq_item_export               uvm_seq_item_pull_imp           -     @708
        arbitration_queue             array                           0     -
        lock_queue                    array                           0     -
        num_last_reqs                 integral                        32    'd1
        num_last_rsps                 integral                        32    'd1
    m_data_input_coverage             data_input_coverage             -     @474
      analysis_imp                    uvm_analysis_imp                -     @483
    m_data_output_agent               data_output_agent               -     @493
      analysis_port                   uvm_analysis_port               -     @502
      m_driver                        data_output_driver              -     @745
        rsp_port                      uvm_analysis_port               -     @764
        seq_item_port                 uvm_seq_item_pull_port          -     @754
      m_monitor                       data_output_monitor             -     @725
        analysis_port                 uvm_analysis_port               -     @734
      m_sequencer                     uvm_sequencer                   -     @774
        rsp_export                    uvm_analysis_export             -     @783
        seq_item_export               uvm_seq_item_pull_imp           -     @901
        arbitration_queue             array                           0     -
        lock_queue                    array                           0     -
        num_last_reqs                 integral                        32    'd1
        num_last_rsps                 integral                        32    'd1
    m_data_output_coverage            data_output_coverage            -     @512
      analysis_imp                    uvm_analysis_imp                -     @521
    m_reference                       reference                       -     @388
      analysis_export_0               uvm_analysis_imp_reference_0    -     @397
      analysis_port_0                 uvm_analysis_port              -     @407
    m_reference_scoreboard            cl_syoscb                      -     @446
      DUT                             cl_syoscb_queue_std            -     @918
        cfg                           cl_syoscb_cfg                  -     @382
          queues                      aa(object,string)              2     -
            [DUT]                     cl_syoscb_queue_std            -     @918
            [REF]                     cl_syoscb_queue_std            -     @929
              cfg                     cl_syoscb_cfg                  -     @382
              iter_idx                integral                       32    'h0
              cnt_add_item            integral                       32    'h0
              items                   da(object)                     0     -
          producers                   aa(object,string)              1     -
            [m_data_output_agent]     cl_syoscb_cfg_pl               -     @383
              list                    da(string)                     2     -
                [0]                   string                         3     DUT
                [1]                   string                         3     REF
          primary_queue               string                         3     DUT
          disable_clone               integral                       1     'h0
          max_queue_size              aa(int,string)                 2     -
            [DUT]                     integral                       32    'h0
            [REF]                     integral                       32    'h0
          scb_name                    string                         22
m_reference_scoreboard
        iter_idx                      integral                       32    'h0
        cnt_add_item                  integral                       32    'h0
        items                         da(object)                     0     -
      REF                             cl_syoscb_queue_std            -     @929
        cfg                           cl_syoscb_cfg                  -     @382
          queues                      aa(object,string)              2     -
            [DUT]                     cl_syoscb_queue_std            -     @918
              cfg                     cl_syoscb_cfg                  -     @382
              iter_idx                integral                       32    'h0
              cnt_add_item            integral                       32    'h0
              items                   da(object)                     0     -
            [REF]                     cl_syoscb_queue_std            -     @929
          producers                   aa(object,string)              1     -
            [m_data_output_agent]     cl_syoscb_cfg_pl               -     @383
              list                    da(string)                     2     -
                [0]                   string                         3     DUT
                [1]                   string                         3     REF
          primary_queue               string                         3     DUT
          disable_clone               integral                       1     'h0
          max_queue_size              aa(int,string)                 2     -
            [DUT]                     integral                       32    'h0
            [REF]                     integral                       32    'h0
          scb_name                    string                         22
m_reference_scoreboard
        iter_idx                      integral                       32    'h0
        cnt_add_item                  integral                       32    'h0
        items                         da(object)                     0     -
      compare_strategy                cl_syoscb_compare              -     @940
        cfg                           cl_syoscb_cfg                  -     @382
          queues                      aa(object,string)              2     -
            [DUT]                     cl_syoscb_queue_std            -     @918
              cfg                     cl_syoscb_cfg                  -     @382
              iter_idx                integral                       32    'h0
              cnt_add_item            integral                       32    'h0
              items                   da(object)                     0     -
            [REF]                     cl_syoscb_queue_std            -     @929
              cfg                     cl_syoscb_cfg                  -     @382
              iter_idx                integral                       32    'h0
              cnt_add_item            integral                       32    'h0
              items                   da(object)                     0     -
          producers                 aa(object,string)          1      -
            [m_data_output_agent]   cl_syoscb_cfg_pl           -      @383
              list                  da(string)                 2      -
                [0]                 string                     3      DUT
                [1]                 string                     3      REF
          primary_queue             string                     3      DUT
          disable_clone             integral                   1      'h0
          max_queue_size            aa(int,string)             2      -
            [DUT]                   integral                   32     'h0
            [REF]                   integral                   32     'h0
          scb_name                  string                     22
m_reference_scoreboard
        compare_algo                cl_syoscb_compare_iop      -      @991
          cfg                       cl_syoscb_cfg              -      @382
            queues                  aa(object,string)          2      -
              [DUT]                 cl_syoscb_queue_std        -      @918
                cfg                 cl_syoscb_cfg              -      @382
                iter_idx            integral                   32     'h0
                cnt_add_item        integral                   32     'h0
                items               da(object)                 0      -
              [REF]                 cl_syoscb_queue_std        -      @929
                cfg                 cl_syoscb_cfg              -      @382
                iter_idx            integral                   32     'h0
                cnt_add_item        integral                   32     'h0
                items               da(object)                 0      -
            producers               aa(object,string)          1      -
              [m_data_output_agent] cl_syoscb_cfg_pl           -      @383
                list                da(string)                 2      -
                  [0]               string                     3      DUT
                  [1]               string                     3      REF
            primary_queue           string                     3      DUT
            disable_clone           integral                   1      'h0
            max_queue_size          aa(int,string)             2      -
              [DUT]                 integral                   32     'h0
              [REF]                 integral                   32     'h0
            scb_name                string                     22
m_reference_scoreboard
      m_data_output_agent_DUT_subscr cl_syoscb_subscriber      -      @950
        analysis_imp                uvm_analysis_imp           -      @959
        queue_name                  string                     3      DUT
        producer                    string                     19
m_data_output_agent
      m_data_output_agent_REF_subscr cl_syoscb_subscriber      -      @969
        analysis_imp                uvm_analysis_imp           -      @978
        queue_name                  string                     3      REF
        producer                    string                     19
m_data_output_agent
      cfg                           cl_syoscb_cfg              -      @382
        queues                      aa(object,string)          2      -
          [DUT]                     cl_syoscb_queue_std        -      @918
            cfg                     cl_syoscb_cfg              -      @382
            iter_idx                integral                   32     'h0
            cnt_add_item            integral                   32     'h0
            items                   da(object)                 0      -
          [REF]                     cl_syoscb_queue_std        -      @929
            cfg                     cl_syoscb_cfg              -      @382
            iter_idx                integral                   32     'h0
            cnt_add_item            integral                   32     'h0
            items                   da(object)                 0      -
        producers                   aa(object,string)          1      -
          [m_data_output_agent]     cl_syoscb_cfg_pl           -      @383
            list                    da(string)                 2      -
              [0]                   string                     3      DUT
              [1]                 string                         3     REF
        primary_queue             string                         3     DUT
        disable_clone             integral                       1     'h0
        max_queue_size            aa(int,string)                 2     -
          [DUT]                   integral                       32    'h0
          [REF]                   integral                       32    'h0
        scb_name                  string                         22
m_reference_scoreboard
      queues                      da(object)                     2     -
        [0]                       cl_syoscb_queue_std            -     @918
          cfg                     cl_syoscb_cfg                  -     @382
            queues                aa(object,string)              2     -
              [DUT]               cl_syoscb_queue_std            -     @918
              [REF]               cl_syoscb_queue_std            -     @929
                cfg               cl_syoscb_cfg                  -     @382
                iter_idx          integral                       32    'h0
                cnt_add_item      integral                       32    'h0
                items             da(object)                     0     -
            producers             aa(object,string)              1     -
              [m_data_output_agent]  cl_syoscb_cfg_pl            -     @383
                list              da(string)                     2     -
                  [0]             string                         3     DUT
                  [1]             string                         3     REF
            primary_queue         string                         3     DUT
            disable_clone         integral                       1     'h0
            max_queue_size        aa(int,string)                 2     -
              [DUT]               integral                       32    'h0
              [REF]               integral                       32    'h0
            scb_name              string                         22
m_reference_scoreboard
          iter_idx                integral                       32    'h0
          cnt_add_item            integral                       32    'h0
          items                   da(object)                     0     -
        [1]                       cl_syoscb_queue_std            -     @929
          cfg                     cl_syoscb_cfg                  -     @382
            queues                aa(object,string)              2     -
              [DUT]               cl_syoscb_queue_std            -     @918
                cfg               cl_syoscb_cfg                  -     @382
                iter_idx          integral                       32    'h0
                cnt_add_item      integral                       32    'h0
                items             da(object)                     0     -
              [REF]               cl_syoscb_queue_std            -     @929
            producers             aa(object,string)              1     -
              [m_data_output_agent]  cl_syoscb_cfg_pl            -     @383
                list              da(string)                     2     -
                  [0]             string                         3     DUT
                  [1]             string                         3     REF
            primary_queue         string                         3     DUT
            disable_clone         integral                       1     'h0
            max_queue_size        aa(int,string)                 2     -
              [DUT]               integral                       32    'h0
              [REF]               integral                       32    'h0
            scb_name              string                         22
m_reference_scoreboard
          iter_idx                integral                       32    'h0
          cnt_add_item            integral                       32    'h0
          items                   da(object)                     0     -
      compare_strategy            cl_syoscb_compare              -     @940
        cfg                       cl_syoscb_cfg                  -     @382
          queues                  aa(object,string)              2     -
            [DUT]                 cl_syoscb_queue_std            -     @918
              cfg                 cl_syoscb_cfg                  -     @382
              iter_idx            integral                       32    'h0
              cnt_add_item          integral                  32    'h0
              items                 da(object)                0     -
            [REF]                   cl_syoscb_queue_std       -     @929
              cfg                   cl_syoscb_cfg             -     @382
              iter_idx              integral                  32    'h0
              cnt_add_item          integral                  32    'h0
              items                 da(object)                0     -
          producers                 aa(object,string)         1     -
            [m_data_output_agent]   cl_syoscb_cfg_pl          -     @383
              list                  da(string)                2     -
                [0]                 string                    3     DUT
                [1]                 string                    3     REF
          primary_queue             string                    3     DUT
          disable_clone             integral                  1     'h0
          max_queue_size            aa(int,string)            2     -
            [DUT]                   integral                  32    'h0
            [REF]                   integral                  32    'h0
          scb_name                  string                    22
m_reference_scoreboard
        compare_algo                cl_syoscb_compare_iop     -     @991
          cfg                       cl_syoscb_cfg             -     @382
            queues                  aa(object,string)         2     -
              [DUT]                 cl_syoscb_queue_std       -     @918
                cfg                 cl_syoscb_cfg             -     @382
                iter_idx            integral                  32    'h0
                cnt_add_item        integral                  32    'h0
                items               da(object)                0     -
              [REF]                 cl_syoscb_queue_std       -     @929
                cfg                 cl_syoscb_cfg             -     @382
                iter_idx            integral                  32    'h0
                cnt_add_item        integral                  32    'h0
                items               da(object)                0     -
            producers               aa(object,string)         1     -
              [m_data_output_agent] cl_syoscb_cfg_pl          -     @383
                list                da(string)                2     -
                  [0]               string                    3     DUT
                  [1]               string                    3     REF
            primary_queue           string                    3     DUT
            disable_clone           integral                  1     'h0
            max_queue_size          aa(int,string)            2     -
              [DUT]                 integral                  32    'h0
              [REF]                 integral                  32    'h0
            scb_name                string                    22
m_reference_scoreboard
      subscribers                   aa(object,string)         2     -
        [DUTm_data_output_agent]    cl_syoscb_subscriber      -     @950
          analysis_imp              uvm_analysis_imp          -     @959
          queue_name                string                    3     DUT
          producer                  string                    19
m_data_output_agent
        [REFm_data_output_agent]    cl_syoscb_subscriber      -     @969
          analysis_imp              uvm_analysis_imp          -     @978
          queue_name                string                    3     REF
          producer                  string                    19
m_data_output_agent
-------------------------------------------------------------------------------
```

Figure 3 UVM testbench topology

```
tb
├── data_input
│   └── sv
│       ├── data_input_agent.sv
│       ├── data_input_config.sv
│       ├── data_input_coverage.sv
│       ├── data_input_driver.sv
│       ├── data_input_if.sv
│       ├── data_input_input_tx.sv
│       ├── data_input_monitor.sv
│       ├── data_input_pkg.sv
│       ├── data_input_seq_lib.sv
│       └── data_input_sequencer.sv
├── data_output
│   └── sv
│       ├── data_output_agent.sv
│       ├── data_output_config.sv
│       ├── data_output_coverage.sv
│       ├── data_output_driver.sv
│       ├── data_output_if.sv
│       ├── data_output_monitor.sv
│       ├── data_output_output_tx.sv
│       ├── data_output_pkg.sv
│       ├── data_output_seq_lib.sv
│       └── data_output_sequencer.sv
├── include
│   ├── data_input_cover_inc.sv
│   ├── data_input_do_mon.sv
│   ├── data_input_driver_inc_after_class.sv
│   ├── data_input_driver_inc_inside_class.sv
│   ├── data_output_do_mon.sv
│   ├── data_output_driver_inc_after_class.sv
│   ├── data_output_driver_inc_inside_class.sv
│   ├── reference_inc_after_class.sv
│   ├── reference_inc_inside_class.sv
│   └── vcd_dump.sv
├── top
│   └── sv
│       ├── port_converter.sv
│       ├── reference.sv
│       ├── top_config.sv
│       ├── top_env.sv
│       ├── top_pkg.sv
│       └── top_seq_lib.sv
├── top_tb
│   └── sv
│       ├── top_tb.sv
│       └── top_th.sv
└── top_test
    └── sv
        ├── top_test_pkg.sv
        └── top_test.sv
```

Figure 4 UVM testbench directory structure

5 top_tb

top_tb.sv

```
module top_tb;

  timeunit      1ns;
  timeprecision 1ps;

  `include "uvm_macros.svh"

  import uvm_pkg::*;

  import top_test_pkg::*;
  import top_pkg::top_config;

  // Configuration object for top-level environment
  top_config top_env_config;

  // Test harness
  top_th th();

  // You can insert code here by setting tb_inc_inside_module in file
common.tpl

  // You can remove the initial block below by setting
tb_generate_run_test = no in file common.tpl

  initial
  begin
    // Start of inlined include file generated_tb/tb/include/vcd_dump.sv
    $dumpfile("dump.vcd");
    $dumpvars;
    // End of inlined include file

    // Create and populate top-level configuration object
    top_env_config = new("top_env_config");
    if ( !top_env_config.randomize() )
      `uvm_error("top_tb", "Failed to randomize top-level configuration
object" )
```

```
    top_env_config.m_data_input_config.vif  = th.data_input_if_0;
    top_env_config.m_data_output_config.vif = th.data_output_if_0;

    uvm_config_db #(top_config)::set(null, "uvm_test_top", "config",
top_env_config);
    uvm_config_db #(top_config)::set(null, "uvm_test_top.m_env",
"config", top_env_config);

    // You can insert code here by setting tb_inc_before_run_test in
file common.tpl

    run_test();
  end

endmodule
```

top_th.sv

```
module top_th;

  timeunit      1ns;
  timeprecision 1ps;


  // You can remove clock and reset below by setting
th_generate_clock_and_reset = no in file common.tpl

  // Example clock and reset declarations
  logic clock = 0;
  logic reset;

  // Example clock generator process
  always #10 clock = ~clock;

  // Example reset generator process
  initial
  begin
    reset = 0;         // Active low reset in this example
    #75 reset = 1;
  end

  assign data_input_if_0.reset  = reset;
  assign data_output_if_0.reset = reset;

  assign data_input_if_0.clk    = clock;
  assign data_output_if_0.clk   = clock;

  // You can insert code here by setting th_inc_inside_module in file
common.tpl

  // Pin-level interfaces connected to DUT
  // You can remove interface instances by setting
generate_interface_instance = no in the interface template file

  data_input_if   data_input_if_0 ();
  data_output_if  data_output_if_0 ();

  lpffir_axis uut (
    .rx_tlast_i  (data_input_if_0.last),
    .rx_tvalid_i(data_input_if_0.valid),
    .rx_tready_o(data_input_if_0.ready),
    .rx_tdata_i  (data_input_if_0.data),
    .tx_tlast_o  (data_output_if_0.last),
    .tx_tvalid_o(data_output_if_0.valid),
    .tx_tready_i(data_output_if_0.ready),
    .tx_tdata_o  (data_output_if_0.data),
    .aclk_i      (clock),
    .aresetn_i   (reset)
  );

endmodule
```

6 top_test

top_test_pkg.sv

```
package top_test_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;

  import data_input_pkg::*;
  import data_output_pkg::*;
  import top_pkg::*;

  `include "top_test.sv"

endpackage : top_test_pkg
```

top_test.sv

```
class top_test extends uvm_test;

  `uvm_component_utils(top_test)

  top_env m_env;

  extern function new(string name, uvm_component parent);

  // You can remove build_phase method by setting
test_generate_methods_inside_class = no in file common.tpl

  extern function void build_phase(uvm_phase phase);

  // You can insert code here by setting test_inc_inside_class in file
common.tpl

endclass : top_test


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


// You can remove build_phase method by setting
test_generate_methods_after_class = no in file common.tpl

function void top_test::build_phase(uvm_phase phase);

  // You can insert code here by setting test_prepend_to_build_phase in
file common.tpl

  // You could modify any test-specific configuration object variables
here



  m_env = top_env::type_id::create("m_env", this);

  // You can insert code here by setting test_append_to_build_phase in
file common.tpl

endfunction : build_phase
```

7

top

port_converter.sv

```
class port_converter #(type T = uvm_sequence_item) extends
uvm_subscriber #(T);
  `uvm_component_param_utils(port_converter#(T))

  // For connecting analysis port of monitor to analysis export of
Syosil scoreboard

  uvm_analysis_port #(uvm_sequence_item) analysis_port;

  function new(string name, uvm_component parent);
    super.new(name, parent);
    analysis_port = new("a_port", this);
  endfunction

  function void write(T t);
    analysis_port.write(t);
  endfunction

endclass
```

reference.sv

```
`uvm_analysis_imp_decl(_reference_0)

class reference extends uvm_component;
  `uvm_component_utils(reference)

  uvm_analysis_imp_reference_0 #(input_tx, reference) analysis_export_0;
// m_data_input_agent

  uvm_analysis_port #(uvm_sequence_item) analysis_port_0; //
m_data_output_agent

  extern function new(string name, uvm_component parent);
  extern function void write_reference_0(input input_tx t);

  // Start of inlined include file
generated_tb/tb/include/reference_inc_inside_class.sv
  extern function void send(input_tx t);

  int save_pnt = 5;
  logic [15:0] tx_save [0:5];
  int init_flag = 1;  // End of inlined include file

endclass


function reference::new(string name, uvm_component parent);
  super.new(name, parent);
  analysis_export_0 = new("analysis_export_0", this);
  analysis_port_0   = new("analysis_port_0",   this);
endfunction : new


// Start of inlined include file
generated_tb/tb/include/reference_inc_after_class.sv
function void reference::write_reference_0(input_tx t);
  send(t);
endfunction

function void reference::send(input_tx t);
  output_tx tx;
  tx = output_tx::type_id::create("tx");
  if (init_flag == 1)
    begin
      init_flag = 0;
      foreach(tx_save[j])
        tx_save[j] = 0;
    end
  if (save_pnt == 5)
    save_pnt = 0;
  else
  save_pnt++;
  tx_save[save_pnt] = t.data;
  tx.data = tx_save[0] + tx_save[1] + tx_save[2] + tx_save[3] +
tx_save[4] + tx_save[5];
  analysis_port_0.write(tx);
  `uvm_info(get_type_name(), $sformatf("Reference Model save_pnt = %0d,
data = %0d",save_pnt, tx.data), UVM_HIGH)
endfunction
```

top_config.sv

```
class top_config extends uvm_object;

  // Do not register config class with the factory

  rand data_input_config   m_data_input_config;
  rand data_output_config  m_data_output_config;

  // You can insert variables here by setting config_var in file
common.tpl

  // You can remove new by setting
top_env_config_generate_methods_inside_class = no in file common.tpl

  extern function new(string name = "");

  // You can insert code here by setting top_env_config_inc_inside_class
in file common.tpl

endclass : top_config


// You can remove new by setting
top_env_config_generate_methods_after_class = no in file common.tpl

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

  m_data_input_config                  = new("m_data_input_config");
  m_data_input_config.is_active        = UVM_ACTIVE;
  m_data_input_config.checks_enable    = 1;
  m_data_input_config.coverage_enable  = 1;

  m_data_output_config                 = new("m_data_output_config");
  m_data_output_config.is_active       = UVM_ACTIVE;
  m_data_output_config.checks_enable   = 1;
  m_data_output_config.coverage_enable = 0;

  // You can insert code here by setting top_env_config_append_to_new in
file common.tpl

endfunction : new
```

top_env.sv

```
import pk_syoscb::*;

class top_env extends uvm_env;

  `uvm_component_utils(top_env)

  extern function new(string name, uvm_component parent);

  // Reference model and Syosil scoreboard
  typedef port_converter #(output_tx) converter_m_data_output_agent_t;

  converter_m_data_output_agent_t m_converter_m_data_output_agent;

  reference                         m_reference;
  cl_syoscb                         m_reference_scoreboard;
  cl_syoscb_cfg                     m_reference_config;

  // Child agents
  data_input_config     m_data_input_config;
  data_input_agent      m_data_input_agent;
  data_input_coverage   m_data_input_coverage;

  data_output_config    m_data_output_config;
  data_output_agent     m_data_output_agent;
  data_output_coverage  m_data_output_coverage;

  top_config            m_config;

  // You can remove build/connect/run_phase by setting
top_env_generate_methods_inside_class = no in file common.tpl

  extern function void build_phase(uvm_phase phase);
  extern function void connect_phase(uvm_phase phase);
  extern function void end_of_elaboration_phase(uvm_phase phase);
  extern task          run_phase(uvm_phase phase);

  // You can insert code here by setting top_env_inc_inside_class in
file common.tpl

endclass : top_env


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


// You can remove build/connect/run_phase by setting
top_env_generate_methods_after_class = no in file common.tpl

function void top_env::build_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "In build_phase", UVM_HIGH)
  // You can insert code here by setting top_env_prepend_to_build_phase
in file common.tpl

  if (!uvm_config_db #(top_config)::get(this, "", "config", m_config))
    `uvm_error(get_type_name(), "Unable to get top_config")

  m_data_input_config = m_config.m_data_input_config;

  // You can insert code here by setting agent_copy_config_vars in file
data_input.tpl

  uvm_config_db #(data_input_config)::set(this, "m_data_input_agent",
"config", m_data_input_config);
  if (m_data_input_config.is_active == UVM_ACTIVE )
    uvm_config_db #(data_input_config)::set(this,
"m_data_input_agent.m_sequencer", "config", m_data_input_config);
  uvm_config_db #(data_input_config)::set(this, "m_data_input_coverage",
"config", m_data_input_config);

  m_data_output_config = m_config.m_data_output_config;

  // You can insert code here by setting agent_copy_config_vars in file
data_output.tpl

  uvm_config_db #(data_output_config)::set(this, "m_data_output_agent",
"config", m_data_output_config);
  if (m_data_output_config.is_active == UVM_ACTIVE )
    uvm_config_db #(data_output_config)::set(this,
"m_data_output_agent.m_sequencer", "config", m_data_output_config);
  uvm_config_db #(data_output_config)::set(this,
"m_data_output_coverage", "config", m_data_output_config);

  // Default factory overrides for Syosil scoreboard

cl_syoscb_queue::type_id::set_type_override(cl_syoscb_queue_std::type_id
::get());

  begin
    bit ok;
    uvm_factory factory = uvm_factory::get();

    if
(factory.find_override_by_type(cl_syoscb_compare_base::type_id::get(),
"*") == cl_syoscb_compare_base::type_id::get())

cl_syoscb_compare_base::type_id::set_inst_override(cl_syoscb_compare_iop
::type_id::get(), "m_reference_scoreboard.*", this);

    // Configuration object for Syosil scoreboard
    m_reference_config =
cl_syoscb_cfg::type_id::create("m_reference_config");
    m_reference_config.set_queues( {"DUT", "REF"} );
    ok = m_reference_config.set_primary_queue("DUT");
    assert(ok);
    ok = m_reference_config.set_producer("m_data_output_agent", {"DUT",
"REF"} );
    assert(ok);
    uvm_config_db#(cl_syoscb_cfg)::set(this, "m_reference_scoreboard",
"cfg", m_reference_config);

    // Instantiate reference model and Syosil scoreboard
    m_reference                     = reference
::type_id::create("m_reference", this);
    m_converter_m_data_output_agent =
converter_m_data_output_agent_t::type_id::create("m_converter_m_data_out
put_agent", this);
    m_reference_scoreboard          = cl_syoscb
::type_id::create("m_reference_scoreboard", this);
  end


  m_data_input_agent     = data_input_agent
::type_id::create("m_data_input_agent", this);
  m_data_input_coverage  = data_input_coverage
::type_id::create("m_data_input_coverage", this);

  m_data_output_agent    = data_output_agent
::type_id::create("m_data_output_agent", this);
  m_data_output_coverage =
data_output_coverage::type_id::create("m_data_output_coverage", this);

  // You can insert code here by setting top_env_append_to_build_phase
in file common.tpl

endfunction : build_phase


function void top_env::connect_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "In connect_phase", UVM_HIGH)


m_data_input_agent.analysis_port.connect(m_data_input_coverage.analysis_
export);


m_data_output_agent.analysis_port.connect(m_data_output_coverage.analysi
s_export);

  begin
    // Connect reference model and Syosil scoreboard
    cl_syoscb_subscriber subscriber;


m_data_input_agent.analysis_port.connect(m_reference.analysis_export_0);

    subscriber = m_reference_scoreboard.get_subscriber("REF",
"m_data_output_agent");
    m_reference.analysis_port_0.connect(subscriber.analysis_export);

    subscriber = m_reference_scoreboard.get_subscriber("DUT",
"m_data_output_agent");
```

```
m_data_output_agent.analysis_port.connect(m_converter_m_data_output_agen
t.analysis_export);

m_converter_m_data_output_agent.analysis_port.connect(subscriber.analysi
s_export);
  end

  // You can insert code here by setting top_env_append_to_connect_phase
in file common.tpl

endfunction : connect_phase


// You can remove end_of_elaboration_phase by setting
top_env_generate_end_of_elaboration = no in file common.tpl

function void top_env::end_of_elaboration_phase(uvm_phase phase);
  uvm_factory factory = uvm_factory::get();
  `uvm_info(get_type_name(), "Information printed from
top_env::end_of_elaboration_phase method", UVM_MEDIUM)
  `uvm_info(get_type_name(), $sformatf("Verbosity threshold is %d",
get_report_verbosity_level()), UVM_MEDIUM)
  uvm_top.print_topology();
  factory.print();
endfunction : end_of_elaboration_phase


// You can remove run_phase by setting top_env_generate_run_phase = no
in file common.tpl

task top_env::run_phase(uvm_phase phase);
  top_default_seq vseq;
  vseq = top_default_seq::type_id::create("vseq");
  vseq.set_item_context(null, null);
  if ( !vseq.randomize() )
    `uvm_fatal(get_type_name(), "Failed to randomize virtual sequence")
  vseq.m_data_input_agent  = m_data_input_agent;
  vseq.m_data_output_agent = m_data_output_agent;
  vseq.set_starting_phase(phase);
  vseq.start(null);

  // You can insert code here by setting top_env_append_to_run_phase in
file common.tpl

endtask : run_phase
```

top_pkg.sv

```
package top_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;

  import data_input_pkg::*;
  import data_output_pkg::*;

  `include "top_config.sv"
  `include "top_seq_lib.sv"
  `include "port_converter.sv"
  `include "reference.sv"
  `include "top_env.sv"

endpackage : top_pkg
```

top_seq_lib.sv

```
class top_default_seq extends uvm_sequence #(uvm_sequence_item);

  `uvm_object_utils(top_default_seq)

  data_input_agent   m_data_input_agent;
  data_output_agent  m_data_output_agent;

  // Number of times to repeat child sequences
  int m_seq_count = 10;

  extern function new(string name = "");
  extern task body();
  extern task pre_start();
  extern task post_start();

`ifndef UVM_POST_VERSION_1_1
  // Functions to support UVM 1.2 objection API in UVM 1.1
  extern function uvm_phase get_starting_phase();
  extern function void set_starting_phase(uvm_phase phase);
`endif

endclass : top_default_seq


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


task top_default_seq::body();
  `uvm_info(get_type_name(), "Default sequence starting", UVM_HIGH)


  repeat (m_seq_count)
  begin
    fork
      if (m_data_input_agent.m_config.is_active == UVM_ACTIVE)
      begin
        data_input_default_seq seq;
        seq = data_input_default_seq::type_id::create("seq");
        seq.set_item_context(this, m_data_input_agent.m_sequencer);
        if ( !seq.randomize() )
          `uvm_error(get_type_name(), "Failed to randomize sequence")
        seq.set_starting_phase( get_starting_phase() );
        seq.start(m_data_input_agent.m_sequencer, this);
      end
      if (m_data_output_agent.m_config.is_active == UVM_ACTIVE)
      begin
        data_output_default_seq seq;
        seq = data_output_default_seq::type_id::create("seq");
        seq.set_item_context(this, m_data_output_agent.m_sequencer);
        if ( !seq.randomize() )
          `uvm_error(get_type_name(), "Failed to randomize sequence")
        seq.set_starting_phase( get_starting_phase() );
        seq.start(m_data_output_agent.m_sequencer, this);
      end
    join
  end

  `uvm_info(get_type_name(), "Default sequence completed", UVM_HIGH)
endtask : body


task top_default_seq::pre_start();
  uvm_phase phase = get_starting_phase();
  if (phase != null)
    phase.raise_objection(this);
endtask: pre_start


task top_default_seq::post_start();
  uvm_phase phase = get_starting_phase();
  if (phase != null)
    phase.drop_objection(this);
endtask: post_start


`ifndef UVM_POST_VERSION_1_1
function uvm_phase top_default_seq::get_starting_phase();
  return starting_phase;
endfunction: get_starting_phase


function void top_default_seq::set_starting_phase(uvm_phase phase);
  starting_phase = phase;
endfunction: set_starting_phase
```

8 data_input

data_input_agent.sv

```
class data_input_agent extends uvm_agent;

  `uvm_component_utils(data_input_agent)

  uvm_analysis_port #(input_tx) analysis_port;

  data_input_config       m_config;
  data_input_sequencer_t  m_sequencer;
  data_input_driver       m_driver;
  data_input_monitor      m_monitor;

  local int m_is_active = -1;

  extern function new(string name, uvm_component parent);

  // You can remove build/connect_phase and get_is_active by setting
agent_generate_methods_inside_class = no in file data_input.tpl

  extern function void build_phase(uvm_phase phase);
  extern function void connect_phase(uvm_phase phase);
  extern function uvm_active_passive_enum get_is_active();

  // You can insert code here by setting agent_inc_inside_class in file
data_input.tpl

endclass : data_input_agent


function  data_input_agent::new(string name, uvm_component parent);
  super.new(name, parent);
  analysis_port = new("analysis_port", this);
endfunction : new


// You can remove build/connect_phase and get_is_active by setting
agent_generate_methods_after_class = no in file data_input.tpl
```

```
function void data_input_agent::build_phase(uvm_phase phase);

  // You can insert code here by setting agent_prepend_to_build_phase in
file data_input.tpl

  if (!uvm_config_db #(data_input_config)::get(this, "", "config",
m_config))
    `uvm_error(get_type_name(), "data_input config not found")

  m_monitor     = data_input_monitor    ::type_id::create("m_monitor",
this);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver    = data_input_driver     ::type_id::create("m_driver",
this);
    m_sequencer = data_input_sequencer_t::type_id::create("m_sequencer",
this);
  end

  // You can insert code here by setting agent_append_to_build_phase in
file data_input.tpl

endfunction : build_phase


function void data_input_agent::connect_phase(uvm_phase phase);
  if (m_config.vif == null)
    `uvm_warning(get_type_name(), "data_input virtual interface is not
set!")

  m_monitor.vif = m_config.vif;
  m_monitor.analysis_port.connect(analysis_port);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver.seq_item_port.connect(m_sequencer.seq_item_export);
    m_driver.vif = m_config.vif;
  end

  // You can insert code here by setting agent_append_to_connect_phase
in file data_input.tpl

endfunction : connect_phase


function uvm_active_passive_enum data_input_agent::get_is_active();
  if (m_is_active == -1)
  begin
    if (uvm_config_db#(uvm_bitstream_t)::get(this, "", "is_active",
m_is_active))
    begin
      if (m_is_active != m_config.is_active)
        `uvm_warning(get_type_name(), "is_active field in config_db
conflicts with config object")
    end
    else
```

```
      m_is_active = m_config.is_active;
  end
  return uvm_active_passive_enum'(m_is_active);
endfunction : get_is_active
```

data_input_config.sv

```
class data_input_config extends uvm_object;

  // Do not register config class with the factory

  virtual data_input_if    vif;

  uvm_active_passive_enum  is_active = UVM_ACTIVE;
  bit                      coverage_enable;
  bit                      checks_enable;

  // You can insert variables here by setting config_var in file 
data_input.tpl

  // You can remove new by setting 
agent_config_generate_methods_inside_class = no in file data_input.tpl

  extern function new(string name = "");

  // You can insert code here by setting agent_config_inc_inside_class 
in file data_input.tpl

endclass : data_input_config


// You can remove new by setting 
agent_config_generate_methods_after_class = no in file data_input.tpl

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

data_input_coverage.sv

```
class data_input_coverage extends uvm_subscriber #(input_tx);

  `uvm_component_utils(data_input_coverage)

  data_input_config m_config;
  bit               m_is_covered;
  input_tx          m_item;

  // Start of inlined include file
generated_tb/tb/include/data_input_cover_inc.sv
  covergroup m_cov;
    option.per_instance = 1;

    cp_data: coverpoint m_item.data {
      bins data_values[] = {[0:127]};
    }
  endgroup
  // End of inlined include file

  // You can remove new, write, and report_phase by setting
agent_cover_generate_methods_inside_class = no in file data_input.tpl

  extern function new(string name, uvm_component parent);
  extern function void write(input input_tx t);
  extern function void build_phase(uvm_phase phase);
  extern function void report_phase(uvm_phase phase);

  // You can insert code here by setting agent_cover_inc_inside_class in
file data_input.tpl

endclass : data_input_coverage


// You can remove new, write, and report_phase by setting
agent_cover_generate_methods_after_class = no in file data_input.tpl

function data_input_coverage::new(string name, uvm_component parent);
  super.new(name, parent);
  m_is_covered = 0;
  m_cov = new();
endfunction : new


function void data_input_coverage::write(input input_tx t);
  m_item = t;
  if (m_config.coverage_enable)
  begin
    m_cov.sample();
    // Check coverage - could use m_cov.option.goal instead of 100 if
your simulator supports it
    if (m_cov.get_inst_coverage() >= 100) m_is_covered = 1;
  end
endfunction : write
```

```
function void data_input_coverage::build_phase(uvm_phase phase);
  if (!uvm_config_db #(data_input_config)::get(this, "", "config",
m_config))
    `uvm_error(get_type_name(), "data_input config not found")
endfunction : build_phase


function void data_input_coverage::report_phase(uvm_phase phase);
  if (m_config.coverage_enable)
    `uvm_info(get_type_name(), $sformatf("Coverage score = %3.1f%%",
m_cov.get_inst_coverage()), UVM_MEDIUM)
  else
    `uvm_info(get_type_name(), "Coverage disabled for this agent",
UVM_MEDIUM)
endfunction : report_phase
```

data_input_driver.sv

```
class data_input_driver extends uvm_driver #(input_tx);

  `uvm_component_utils(data_input_driver)

  virtual data_input_if vif;

  extern function new(string name, uvm_component parent);

  // Start of inlined include file
generated_tb/tb/include/data_input_driver_inc_inside_class.sv
  extern task run_phase(uvm_phase phase);
  // End of inlined include file

endclass : data_input_driver


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


// Start of inlined include file
generated_tb/tb/include/data_input_driver_inc_after_class.sv
task data_input_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever @(posedge vif.clk)
  begin
    seq_item_port.get_next_item(req);
    phase.raise_objection(this);
    wait (vif.reset == 1);
    vif.data <= req.data;
    vif.valid  <= 1;
    vif.last  <= 0;
    wait (vif.ready == 1);

    fork
      begin
        repeat (10) @(posedge vif.clk);
        phase.drop_objection(this);
      end
    join_none
    seq_item_port.item_done();
  end
endtask : run_phase
```

data_input_if.sv

```
interface data_input_if();

  timeunit      1ns;
  timeprecision 1ps;

  import data_input_pkg::*;

  logic last;
  logic valid;
  logic ready;
  logic [15:0] data;
  logic clk;
  logic reset;

  // You can insert properties and assertions here

  // You can insert code here by setting if_inc_inside_interface in file
data_input.tpl

endinterface : data_input_if
```

data_input_input_tx.sv

```
class input_tx extends uvm_sequence_item;

  `uvm_object_utils(input_tx)

  // To include variables in copy, compare, print, record, pack, unpack,
and compare2string, define them using trans_var in file data_input.tpl
  // To exclude variables from compare, pack, and unpack methods, define
them using trans_meta in file data_input.tpl

  // Transaction variables
  rand logic [15:0] data;
  constraint c_data { 0 <= data; data < 128; }


  extern function new(string name = "");

  // You can remove do_copy/compare/print/record and convert2string
method by setting trans_generate_methods_inside_class = no in file
data_input.tpl
  extern function void do_copy(uvm_object rhs);
  extern function bit  do_compare(uvm_object rhs, uvm_comparer
comparer);
  extern function void do_print(uvm_printer printer);
  extern function void do_record(uvm_recorder recorder);
  extern function void do_pack(uvm_packer packer);
  extern function void do_unpack(uvm_packer packer);
  extern function string convert2string();

  // You can insert code here by setting trans_inc_inside_class in file
data_input.tpl

endclass : input_tx


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


// You can remove do_copy/compare/print/record and convert2string method
by setting trans_generate_methods_after_class = no in file
data_input.tpl

function void input_tx::do_copy(uvm_object rhs);
  input_tx rhs_;
  if (!$cast(rhs_, rhs))
    `uvm_fatal(get_type_name(), "Cast of rhs object failed")
  super.do_copy(rhs);
  data = rhs_.data;
endfunction : do_copy
```

```
function bit input_tx::do_compare(uvm_object rhs, uvm_comparer
comparer);
  bit result;
  input_tx rhs_;
  if (!$cast(rhs_, rhs))
    `uvm_fatal(get_type_name(), "Cast of rhs object failed")
  result = super.do_compare(rhs, comparer);
  result &= comparer.compare_field("data", data, rhs_.data,
$bits(data));
  return result;
endfunction : do_compare


function void input_tx::do_print(uvm_printer printer);
  if (printer.knobs.sprint == 0)
    `uvm_info(get_type_name(), convert2string(), UVM_MEDIUM)
  else
    printer.m_string = convert2string();
endfunction : do_print


function void input_tx::do_record(uvm_recorder recorder);
  super.do_record(recorder);
  // Use the record macros to record the item fields:
  `uvm_record_field("data", data)
endfunction : do_record


function void input_tx::do_pack(uvm_packer packer);
  super.do_pack(packer);
  `uvm_pack_int(data)
endfunction : do_pack


function void input_tx::do_unpack(uvm_packer packer);
  super.do_unpack(packer);
  `uvm_unpack_int(data)
endfunction : do_unpack


function string input_tx::convert2string();
  string s;
  $sformat(s, "%s\n", super.convert2string());
  $sformat(s, {"%s\n",
    "data = 'h%0h  'd%0d\n"},
    get_full_name(), data, data);
  return s;
endfunction : convert2string
```

data_input_monitor.sv

```
class data_input_monitor extends uvm_monitor;

  `uvm_component_utils(data_input_monitor)

  virtual data_input_if vif;

  uvm_analysis_port #(input_tx) analysis_port;

  input_tx m_trans;

  extern function new(string name, uvm_component parent);

  // Methods build_phase, run_phase, and do_mon generated by setting
monitor_inc in file data_input.tpl
  extern function void build_phase(uvm_phase phase);
  extern task run_phase(uvm_phase phase);
  extern task do_mon();

  // You can insert code here by setting monitor_inc_inside_class in
file data_input.tpl

endclass : data_input_monitor


function data_input_monitor::new(string name, uvm_component parent);
  super.new(name, parent);
  analysis_port = new("analysis_port", this);
endfunction : new


function void data_input_monitor::build_phase(uvm_phase phase);
endfunction : build_phase


task data_input_monitor::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  m_trans = input_tx::type_id::create("m_trans");
  do_mon();
endtask : run_phase


// Start of inlined include file
generated_tb/tb/include/data_input_do_mon.sv
task data_input_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    wait (vif.reset == 1);
    if (vif.valid && vif.ready)
    begin
      m_trans.data = vif.data;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(), $sformatf("Input data = %0d",
m_trans.data), UVM_HIGH)
    end
  end
endtask
```

data_input_pkg.sv

```
package data_input_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;


  `include "data_input_input_tx.sv"
  `include "data_input_config.sv"
  `include "data_input_driver.sv"
  `include "data_input_monitor.sv"
  `include "data_input_sequencer.sv"
  `include "data_input_coverage.sv"
  `include "data_input_agent.sv"
  `include "data_input_seq_lib.sv"

endpackage : data_input_pkg
```

data_input_seq_lib.sv

```
class data_input_default_seq extends uvm_sequence #(input_tx);

  `uvm_object_utils(data_input_default_seq)

  extern function new(string name = "");
  extern task body();

`ifndef UVM_POST_VERSION_1_1
  // Functions to support UVM 1.2 objection API in UVM 1.1
  extern function uvm_phase get_starting_phase();
  extern function void set_starting_phase(uvm_phase phase);
`endif

endclass : data_input_default_seq


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


task data_input_default_seq::body();
  `uvm_info(get_type_name(), "Default sequence starting", UVM_HIGH)

  req = input_tx::type_id::create("req");
  start_item(req);
  if ( !req.randomize() )
    `uvm_error(get_type_name(), "Failed to randomize transaction")
  finish_item(req);

  `uvm_info(get_type_name(), "Default sequence completed", UVM_HIGH)
endtask : body


`ifndef UVM_POST_VERSION_1_1
function uvm_phase data_input_default_seq::get_starting_phase();
  return starting_phase;
endfunction: get_starting_phase


function void data_input_default_seq::set_starting_phase(uvm_phase
phase);
  starting_phase = phase;
endfunction: set_starting_phase
```

data_input_sequencer.sv

```
// Sequencer class is specialization of uvm_sequencer
typedef uvm_sequencer #(input_tx) data_input_sequencer_t;
```

9

data_output

data_output_agent.sv

```
class data_output_agent extends uvm_agent;

  `uvm_component_utils(data_output_agent)

  uvm_analysis_port #(output_tx) analysis_port;

  data_output_config       m_config;
  data_output_sequencer_t  m_sequencer;
  data_output_driver       m_driver;
  data_output_monitor      m_monitor;

  local int m_is_active = -1;

  extern function new(string name, uvm_component parent);

  // You can remove build/connect_phase and get_is_active by setting
agent_generate_methods_inside_class = no in file data_output.tpl

  extern function void build_phase(uvm_phase phase);
  extern function void connect_phase(uvm_phase phase);
  extern function uvm_active_passive_enum get_is_active();

  // You can insert code here by setting agent_inc_inside_class in file
data_output.tpl

endclass : data_output_agent


function  data_output_agent::new(string name, uvm_component parent);
  super.new(name, parent);
  analysis_port = new("analysis_port", this);
endfunction : new


// You can remove build/connect_phase and get_is_active by setting
agent_generate_methods_after_class = no in file data_output.tpl
```

```
function void data_output_agent::build_phase(uvm_phase phase);

  // You can insert code here by setting agent_prepend_to_build_phase in
file data_output.tpl

  if (!uvm_config_db #(data_output_config)::get(this, "", "config",
m_config))
    `uvm_error(get_type_name(), "data_output config not found")

  m_monitor     = data_output_monitor    ::type_id::create("m_monitor",
this);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver     = data_output_driver     ::type_id::create("m_driver",
this);
    m_sequencer =
data_output_sequencer_t::type_id::create("m_sequencer", this);
  end

  // You can insert code here by setting agent_append_to_build_phase in
file data_output.tpl

endfunction : build_phase


function void data_output_agent::connect_phase(uvm_phase phase);
  if (m_config.vif == null)
    `uvm_warning(get_type_name(), "data_output virtual interface is not
set!")

  m_monitor.vif = m_config.vif;
  m_monitor.analysis_port.connect(analysis_port);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver.seq_item_port.connect(m_sequencer.seq_item_export);
    m_driver.vif = m_config.vif;
  end

  // You can insert code here by setting agent_append_to_connect_phase
in file data_output.tpl

endfunction : connect_phase


function uvm_active_passive_enum data_output_agent::get_is_active();
  if (m_is_active == -1)
  begin
    if (uvm_config_db#(uvm_bitstream_t)::get(this, "", "is_active",
m_is_active))
    begin
      if (m_is_active != m_config.is_active)
        `uvm_warning(get_type_name(), "is_active field in config_db
conflicts with config object")
    end
    else
```

```
      m_is_active = m_config.is_active;
  end
  return uvm_active_passive_enum'(m_is_active);
endfunction : get_is_active
```

data_output_config.sv

```
class data_output_config extends uvm_object;

  // Do not register config class with the factory

  virtual data_output_if   vif;

  uvm_active_passive_enum  is_active = UVM_ACTIVE;
  bit                      coverage_enable;
  bit                      checks_enable;

  // You can insert variables here by setting config_var in file
data_output.tpl

  // You can remove new by setting
agent_config_generate_methods_inside_class = no in file data_output.tpl

  extern function new(string name = "");

  // You can insert code here by setting agent_config_inc_inside_class
in file data_output.tpl

endclass : data_output_config


// You can remove new by setting
agent_config_generate_methods_after_class = no in file data_output.tpl

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

data_output_coverage.sv

```
class data_output_coverage extends uvm_subscriber #(output_tx);

  `uvm_component_utils(data_output_coverage)

  data_output_config m_config;
  bit                m_is_covered;
  output_tx          m_item;

  // You can replace covergroup m_cov by setting agent_cover_inc in file
data_output.tpl
  // or remove covergroup m_cov by setting
agent_cover_generate_methods_inside_class = no in file data_output.tpl

  covergroup m_cov;
    option.per_instance = 1;
    // You may insert additional coverpoints here ...

    cp_data: coverpoint m_item.data;
    //  Add bins here if required

  endgroup

  // You can remove new, write, and report_phase by setting
agent_cover_generate_methods_inside_class = no in file data_output.tpl

  extern function new(string name, uvm_component parent);
  extern function void write(input output_tx t);
  extern function void build_phase(uvm_phase phase);
  extern function void report_phase(uvm_phase phase);

  // You can insert code here by setting agent_cover_inc_inside_class in
file data_output.tpl

endclass : data_output_coverage


// You can remove new, write, and report_phase by setting
agent_cover_generate_methods_after_class = no in file data_output.tpl

function data_output_coverage::new(string name, uvm_component parent);
  super.new(name, parent);
  m_is_covered = 0;
  m_cov = new();
endfunction : new


function void data_output_coverage::write(input output_tx t);
  m_item = t;
  if (m_config.coverage_enable)
  begin
    m_cov.sample();
    // Check coverage - could use m_cov.option.goal instead of 100 if
your simulator supports it
```

```
    if (m_cov.get_inst_coverage() >= 100) m_is_covered = 1;
  end
endfunction : write


function void data_output_coverage::build_phase(uvm_phase phase);
  if (!uvm_config_db #(data_output_config)::get(this, "", "config",
m_config))
    `uvm_error(get_type_name(), "data_output config not found")
endfunction : build_phase


function void data_output_coverage::report_phase(uvm_phase phase);
  if (m_config.coverage_enable)
    `uvm_info(get_type_name(), $sformatf("Coverage score = %3.1f%%",
m_cov.get_inst_coverage()), UVM_MEDIUM)
  else
    `uvm_info(get_type_name(), "Coverage disabled for this agent",
UVM_MEDIUM)
endfunction : report_phase
```

data_output_driver.sv

```
class data_output_driver extends uvm_driver #(output_tx);

  `uvm_component_utils(data_output_driver)

  virtual data_output_if vif;

  extern function new(string name, uvm_component parent);

  // Start of inlined include file
generated_tb/tb/include/data_output_driver_inc_inside_class.sv
  extern task run_phase(uvm_phase phase);
      // End of inlined include file

endclass : data_output_driver


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


// Start of inlined include file
generated_tb/tb/include/data_output_driver_inc_after_class.sv
task data_output_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever @(posedge vif.clk)
  begin
    seq_item_port.get_next_item(req);
    phase.raise_objection(this);

    vif.ready  <= 1;
    wait (vif.reset == 1);

    fork
      begin
        repeat (10) @(posedge vif.clk);
        phase.drop_objection(this);
      end
    join_none
    seq_item_port.item_done();
  end
endtask : run_phase
```

data_output_if.sv

```
interface data_output_if();

  timeunit      1ns;
  timeprecision 1ps;

  import data_output_pkg::*;

  logic last;
  logic valid;
  logic ready;
  logic [15:0] data;
  logic clk;
  logic reset;

  // You can insert properties and assertions here

  // You can insert code here by setting if_inc_inside_interface in file
data_output.tpl

endinterface : data_output_if
```

data_output_monitor.sv

```
class data_output_monitor extends uvm_monitor;

  `uvm_component_utils(data_output_monitor)

  virtual data_output_if vif;

  uvm_analysis_port #(output_tx) analysis_port;

  output_tx m_trans;

  extern function new(string name, uvm_component parent);

  // Methods build_phase, run_phase, and do_mon generated by setting
monitor_inc in file data_output.tpl
  extern function void build_phase(uvm_phase phase);
  extern task run_phase(uvm_phase phase);
  extern task do_mon();

  // You can insert code here by setting monitor_inc_inside_class in
file data_output.tpl

endclass : data_output_monitor


function data_output_monitor::new(string name, uvm_component parent);
  super.new(name, parent);
  analysis_port = new("analysis_port", this);
endfunction : new


function void data_output_monitor::build_phase(uvm_phase phase);
endfunction : build_phase


task data_output_monitor::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  m_trans = output_tx::type_id::create("m_trans");
  do_mon();
endtask : run_phase


// Start of inlined include file
generated_tb/tb/include/data_output_do_mon.sv
task data_output_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    wait (vif.reset == 1);
    if (vif.valid && vif.ready)
    begin
      m_trans.data = vif.data;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(), $sformatf("Output data =
%0d",m_trans.data), UVM_HIGH)
    end
  end
endtask
```

data_output_output_tx.sv

```
class output_tx extends uvm_sequence_item;

  `uvm_object_utils(output_tx)

  // To include variables in copy, compare, print, record, pack, unpack,
and compare2string, define them using trans_var in file data_output.tpl
  // To exclude variables from compare, pack, and unpack methods, define
them using trans_meta in file data_output.tpl

  // Transaction variables
  rand logic [15:0] data;


  extern function new(string name = "");

  // You can remove do_copy/compare/print/record and convert2string
method by setting trans_generate_methods_inside_class = no in file
data_output.tpl
  extern function void do_copy(uvm_object rhs);
  extern function bit  do_compare(uvm_object rhs, uvm_comparer
comparer);
  extern function void do_print(uvm_printer printer);
  extern function void do_record(uvm_recorder recorder);
  extern function void do_pack(uvm_packer packer);
  extern function void do_unpack(uvm_packer packer);
  extern function string convert2string();

  // You can insert code here by setting trans_inc_inside_class in file
data_output.tpl

endclass : output_tx


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


// You can remove do_copy/compare/print/record and convert2string method
by setting trans_generate_methods_after_class = no in file
data_output.tpl

function void output_tx::do_copy(uvm_object rhs);
  output_tx rhs_;
  if (!$cast(rhs_, rhs))
    `uvm_fatal(get_type_name(), "Cast of rhs object failed")
  super.do_copy(rhs);
  data = rhs_.data;
endfunction : do_copy


function bit output_tx::do_compare(uvm_object rhs, uvm_comparer
comparer);
```

```
  bit result;
  output_tx rhs_;
  if (!$cast(rhs_, rhs))
    `uvm_fatal(get_type_name(), "Cast of rhs object failed")
  result = super.do_compare(rhs, comparer);
  result &= comparer.compare_field("data", data, rhs_.data,
$bits(data));
  return result;
endfunction : do_compare


function void output_tx::do_print(uvm_printer printer);
  if (printer.knobs.sprint == 0)
    `uvm_info(get_type_name(), convert2string(), UVM_MEDIUM)
  else
    printer.m_string = convert2string();
endfunction : do_print


function void output_tx::do_record(uvm_recorder recorder);
  super.do_record(recorder);
  // Use the record macros to record the item fields:
  `uvm_record_field("data", data)
endfunction : do_record


function void output_tx::do_pack(uvm_packer packer);
  super.do_pack(packer);
  `uvm_pack_int(data)
endfunction : do_pack


function void output_tx::do_unpack(uvm_packer packer);
  super.do_unpack(packer);
  `uvm_unpack_int(data)
endfunction : do_unpack


function string output_tx::convert2string();
  string s;
  $sformat(s, "%s\n", super.convert2string());
  $sformat(s, {"%s\n",
    "data = 'h%0h  'd%0d\n"},
    get_full_name(), data, data);
  return s;
endfunction : convert2string
```

data_output_pkg.sv

```
package data_output_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;


  `include "data_output_output_tx.sv"
  `include "data_output_config.sv"
  `include "data_output_driver.sv"
  `include "data_output_monitor.sv"
  `include "data_output_sequencer.sv"
  `include "data_output_coverage.sv"
  `include "data_output_agent.sv"
  `include "data_output_seq_lib.sv"

endpackage : data_output_pkg
```

data_output_seq_lib.sv

```
class data_output_default_seq extends uvm_sequence #(output_tx);

  `uvm_object_utils(data_output_default_seq)

  extern function new(string name = "");
  extern task body();

`ifndef UVM_POST_VERSION_1_1
  // Functions to support UVM 1.2 objection API in UVM 1.1
  extern function uvm_phase get_starting_phase();
  extern function void set_starting_phase(uvm_phase phase);
`endif

endclass : data_output_default_seq


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


task data_output_default_seq::body();
  `uvm_info(get_type_name(), "Default sequence starting", UVM_HIGH)

  req = output_tx::type_id::create("req");
  start_item(req);
  if ( !req.randomize() )
    `uvm_error(get_type_name(), "Failed to randomize transaction")
  finish_item(req);

  `uvm_info(get_type_name(), "Default sequence completed", UVM_HIGH)
endtask : body


`ifndef UVM_POST_VERSION_1_1
function uvm_phase data_output_default_seq::get_starting_phase();
  return starting_phase;
endfunction: get_starting_phase


function void data_output_default_seq::set_starting_phase(uvm_phase
phase);
  starting_phase = phase;
endfunction: set_starting_phase
```

data_output_sequencer.sv

```
// Sequencer class is specialization of uvm_sequencer
typedef uvm_sequencer #(output_tx) data_output_sequencer_t;
```

10

include

data_input_cover_inc.sv

```
covergroup m_cov;
  option.per_instance = 1;

  cp_data: coverpoint m_item.data {
    bins data_values[] = {[0:127]};
  }
endgroup
```

data_input_do_mon.sv

```
task data_input_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    wait (vif.reset == 1);
    if (vif.valid && vif.ready)
    begin
      m_trans.data = vif.data;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(), $sformatf("Input data = %0d",
m_trans.data), UVM_HIGH)
    end
  end
endtask
```

data_input_driver_inc_after_class.sv

```
task data_input_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever @(posedge vif.clk)
  begin
    seq_item_port.get_next_item(req);
    phase.raise_objection(this);
    wait (vif.reset == 1);
    vif.data <= req.data;
    vif.valid  <= 1;
    vif.last  <= 0;
    wait (vif.ready == 1);

    fork
      begin
        repeat (10) @(posedge vif.clk);
        phase.drop_objection(this);
      end
    join_none
    seq_item_port.item_done();
  end
endtask : run_phase
```

data_input_driver_inc_inside_class.sv

```
extern task run_phase(uvm_phase phase);
```

data_output_do_mon.sv

```
task data_output_monitor::do_mon;
  forever @(posedge vif.clk)
  begin
    wait (vif.reset == 1);
    if (vif.valid && vif.ready)
    begin
      m_trans.data = vif.data;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(), $sformatf("Output data = 
%0d",m_trans.data), UVM_HIGH)
    end
  end
endtask
```

data_output_driver_inc_after_class.sv

```
task data_output_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever @(posedge vif.clk)
  begin
    seq_item_port.get_next_item(req);
    phase.raise_objection(this);

    vif.ready  <= 1;
    wait (vif.reset == 1);

    fork
      begin
        repeat (10) @(posedge vif.clk);
        phase.drop_objection(this);
      end
    join_none
    seq_item_port.item_done();
  end
endtask : run_phase
```

data_output_driver_inc_inside_class.sv

```
extern task run_phase(uvm_phase phase);
```

reference_inc_after_class.sv

```
function void reference::write_reference_0(input_tx t);
  send(t);
endfunction

function void reference::send(input_tx t);
  output_tx tx;
  tx = output_tx::type_id::create("tx");
  if (init_flag == 1)
    begin
      init_flag = 0;
      foreach(tx_save[j])
        tx_save[j] = 0;
    end
  if (save_pnt == 5)
    save_pnt = 0;
  else
  save_pnt++;
  tx_save[save_pnt] = t.data;
  tx.data = tx_save[0] + tx_save[1] + tx_save[2] + tx_save[3] +
tx_save[4] + tx_save[5];
  analysis_port_0.write(tx);
  `uvm_info(get_type_name(), $sformatf("Reference Model save_pnt = %0d,
data = %0d",save_pnt, tx.data), UVM_HIGH)
endfunction
```

reference_inc_inside_class.sv

```
extern function void send(input_tx t);

int save_pnt = 5;
logic [15:0] tx_save [0:5];
int init_flag = 1;
```

Index

1. OpenCores LPFFIR project SVN repository: https://opencores.org/projects/lpffir
2. Doulos. *Easier UVM*. Retrieved from https://www.doulos.com/knowhow/sysverilog/uvm/easier/
3. EDA Playground LPFFIR project UVM simulations https://www.edaplayground.com/x/4RFv
4. ARM publications (2010). AMBA 4 AXI4-Stream Protocol Specification Version 1.0 (ARM IHI 0051)