



SPI - UVM Verification Project:

SPI Slave Wrapper with Single Port RAM

Presented For: Digital Verification Using SV & UVM Diploma - V11

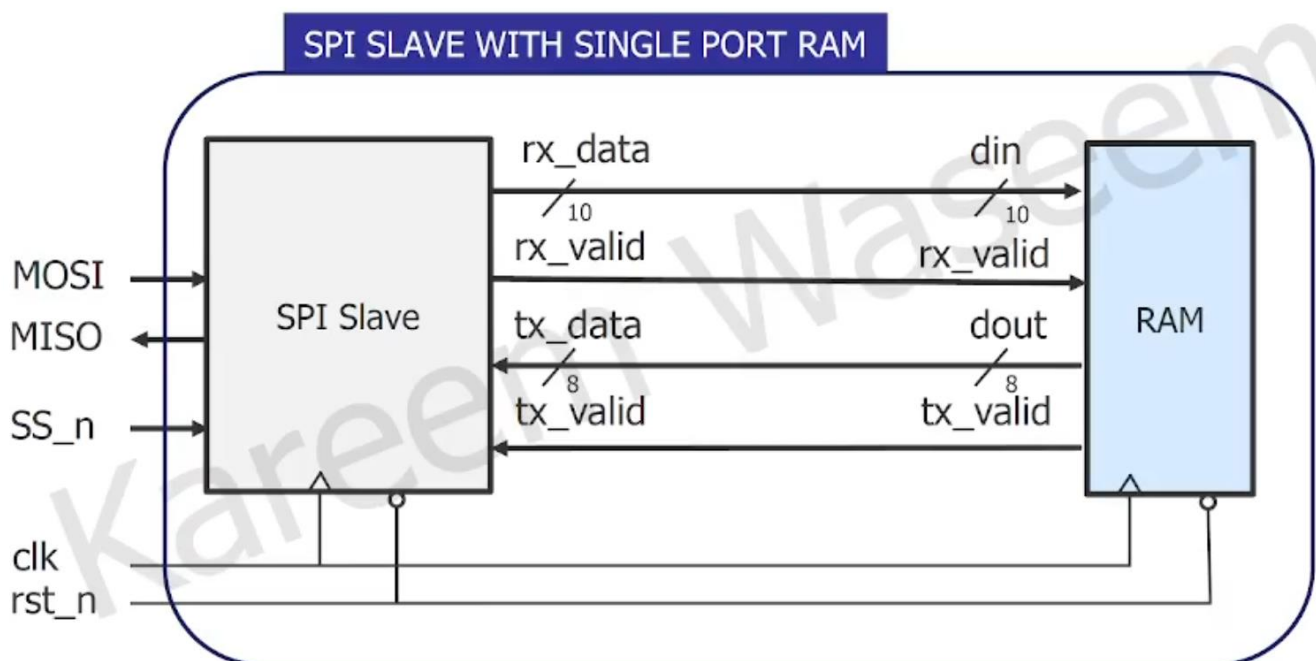
Under Supervision of: Eng. Kareem Waseem ❤️

❤️ Digital Verification Engineers 😎😂

Magdy Ahmed Abbas Abdelhamid

Ahmed Adel Younis Sayed

Kareem Ayman Mohamed



SPI SLAVE WITH SINGLE PORT RAM**Full RAM Environment****SPI RAM RTL WITHOUT BUGS**

```
// SPI RAM RTL Code After Bug Fixing
module project_ram(din, rx_valid, dout, tx_valid, clk, rst_n);
    parameter MEM_DEPTH = 256;
    parameter ADDR_SIZE = 8;
    input rx_valid, clk, rst_n;
    input [9:0] din;
    output reg tx_valid;
    output reg [7:0] dout;
    reg [ADDR_SIZE - 1:0] addr_rd, addr_wr;
    reg [7:0] mem [MEM_DEPTH - 1:0];
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            // make integer i = 0 to be int i = 0 inside the for loop to be in its scope and
            // Not to take in code coverage report as it can't make 100 % its 32 bits Toggle
            for (int i = 0 ; i < MEM_DEPTH ; i = i + 1) begin
                mem [i] <= 0;
            end
            dout <= 0;    // "Bug Fix" Zero For all the Following outside For Loop
            tx_valid <= 0;
            addr_rd <= 0;    // "Bug Fix" make addr_rd <= 0 When rst_n
            addr_wr <= 0;    // "Bug Fix" make addr_wr <= 0 When rst_n
        end
        else if (rx_valid) begin
            if (din[9:8] == 2'b00) begin
                addr_wr <= din[7:0];
                tx_valid <= 0;
            end
            else if (din[9:8] == 2'b01) begin
                mem [addr_wr] <= din[7:0];
                tx_valid <= 0;
            end
        end
    end
end
```

```

else if (din[9:8] == 2'b10) begin
    addr_rd <= din[7:0];
    tx_valid <= 0;
end
else if (din[9:8] == 2'b11) begin
    dout <= mem[addr_rd];
    tx_valid <= 1;
end
else begin
    dout <= 0;
    tx_valid <= 0;
end
end
end
endmodule

```

SPI RAM TOP

```

import ram_test_pkg::*;
import uvm_pkg::*;
import ram_config_pkg::*;
`include "uvm_macros.svh"
module top();
    bit clk;
    initial begin
        clk = 0;
        forever #1 clk = !clk;
    end

    ram_if ramif(clk);
    project_ram dut (ramif.din, ramif.rx_valid, ramif.dout, ramif.tx_valid, ramif.clk, ramif.rst_n);
    bind project_ram SPI_RAM_SVA SVA_inst(ramif.din, ramif.rx_valid, ramif.tx_valid, ramif.clk);
    initial begin
        uvm_config_db#(virtual ram_if)::set(null, "uvm_test_top", "ram_IF", ramif);
        run_test("ram_test");
    end
end
endmodule

```

TEST

```

package ram_test_pkg;
import uvm_pkg::*;
import sequence_pkg::*;
import ram_config_pkg::*;
`include "uvm_macros.svh"
import ram_env_pkg::*;
class ram_test extends uvm_test;
  `uvm_component_utils(ram_test)
  ram_env env;
  ram_config ram_cfg;
  virtual ram_if ramif;
  ram_read_and_write_sequence read_and_write_seq;
  ram_reset_sequence reset_seq;
  ram_read_sequence read_seq;
  ram_write_sequence write_seq;
  function new(string name, uvm_component parent = null);
    super.new(name, parent);
  endfunction
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    env = ram_env::type_id::create("env", this);
    ram_cfg = ram_config::type_id::create("alu_cfg", this);
    read_and_write_seq
= ram_read_and_write_sequence::type_id::create("read_and_write_seq", this);
    read_seq = ram_read_sequence::type_id::create("read_seq", this);
    write_seq = ram_write_sequence::type_id::create("write_seq", this);
    reset_seq = ram_reset_sequence::type_id::create("reset_seq", this);
    if(!uvm_config_db #(virtual ram_if)::get(this, "", "ram_IF", ram_cfg.ramif))
      `uvm_fatal("build_phase", "test
        – unable to get the virtual interface of alu from uvm_config_db");
    uvm_config_db #(ram_config)::set(this, " * ", "CFG", ram_cfg);
  endfunction
  task run_phase(uvm_phase phase);
    super.run_phase(phase);
    phase.raise_objection(this);
    `uvm_info("run_phase", "reset_asserted", UVM_LOW);
    reset_seq.start(env.agt.sqr);
    `uvm_info("run_phase", "reset_deasserted", UVM_LOW);
    `uvm_info("run_phase", "stimulus_generation_started(write_only)", UVM_LOW);
    write_seq.start(env.agt.sqr);

```

```

`uvm_info("run_phase", "stimulus generation finished(write_only) ", UVM_LOW);
`uvm_info("run_phase", "stimulus_generation_started(read_only)", UVM_LOW);
read_seq.start(env.agt.sqr);
`uvm_info("run_phase", "stimulus generation finished(read_only) ", UVM_LOW);
`uvm_info("run_phase", "stimulus_generation_started(read & write)", UVM_LOW);
read_and_write_seq.start(env.agt.sqr);
`uvm_info("run_phase", "stimulus generation finished(read & write ) ", UVM_LOW);
phase.drop_objection(this);
endtask
endclass
endpackage

```

SVA

```

module SPI_RAM_SVA(din , rx_valid , tx_valid , clk);
    input [9: 0] din;
    input rx_valid, tx_valid, clk;
sequence A;
    ((din[9] == 1'b1) && (din[8] == 1'b1));
endsequence
property tx_valid_chk_11;
    @(posedge clk) A |-> ##1 $rose(tx_valid) |-> ##1 !tx_valid;
endproperty
tx_chk_11: cover property(tx_valid_chk_11);
tx_asser_11: assert property(tx_valid_chk_11);m
sequence B;
    ((din[9] == 1'b0) && (din[8] == 1'b0)) || ((din[9] == 1'b0) && (din[8] == 1'b1));
endsequence
property tx_valid_chk_00_01;
    @(posedge clk) B | => !(tx_valid);
endproperty
tx_chk_00_01: cover property(tx_valid_chk_00_01);
tx_asser_00_01: assert property(tx_valid_chk_00_01);

sequence C;
    ((din[9] == 1'b1) && (din[8] == 1'b0));
endsequence
property tx_valid_chk_10;
    @(posedge clk) C |-> ##1 (tx_valid == 0);
endproperty
tx_chk_10: cover property(tx_valid_chk_10);
tx_asser_10: assert property(tx_valid_chk_10);
endmodule

```

SEQUENCER

```
package sequencer_pkg;
import uvm_pkg::*;
import sequence_item_pkg::*;
`include "uvm_macros.svh"
class ram_sequencer extends uvm_sequencer #( ram_seq_item) ;
`uvm_component_utils(ram_sequencer)
function new(string name = "ram_sequencer", uvm_component parent = null);
super.new(name, parent);
endfunction
endclass
endpackage
```

SEQUENCE

```
package sequence_pkg;
import uvm_pkg::*;
import sequence_item_pkg::*;
`include "uvm_macros.svh"
class ram_reset_sequence extends uvm_sequence;
`uvm_object_utils(ram_reset_sequence)
function new(string name = "ram_reset_sequence");
super.new(name);
endfunction //new()
virtual task body();
ram_seq_item item;
item = ram_seq_item::type_id::create("item");
start_item(item);
item.din = 0;
item.rst_n = 0;
item.rx_valid = 0;
finish_item(item);
endtask
endclass
class ram_write_sequence extends uvm_sequence;
`uvm_object_utils(ram_write_sequence)
function new(string name = "ram_read_write_sequence");
super.new(name);
endfunction //new()
```

```

virtual task body();
repeat(10000) begin
    ram_seq_item item;
    item = ram_seq_item::type_id::create("item");
    start_item(item);
    item.constraint_mode(0);
    item.write_only.constraint_mode(1);
    assert(item.randomize());
    finish_item(item);
end
endtask
endclass

class ram_read_sequence extends uvm_sequence;
    `uvm_object_utils(ram_read_sequence)
    function new(string name = "ram_read_write_sequence");
        super.new(name);
    endfunction //new()
virtual task body();
repeat(10000) begin
    ram_seq_item item;
    item = ram_seq_item::type_id::create("item");
    start_item(item);
    item.constraint_mode(0);
    item.read_only.constraint_mode(1);
    assert(item.randomize());
    finish_item(item);
end
endtask
endclass

class ram_read_and_write_sequence extends uvm_sequence;
    `uvm_object_utils(ram_read_and_write_sequence)
    function new(string name = "ram_read_write_sequence");
        super.new(name);
    endfunction //new()
virtual task body();
repeat(10000) begin
    ram_seq_item item;
    item = ram_seq_item::type_id::create("item");
    start_item(item);
    item.constraint_mode(0);
    item.read_and_write.constraint_mode(1);
    assert(item.randomize());
    finish_item(item);
end
endtask
endclass
endpackage

```

SEQUENCE ITEM

```

package sequence_item_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ram_seq_item extends uvm_sequence_item;
`uvm_object_utils(ram_seq_item)
rand bit [9:0] din;
rand bit rx_valid, rst_n;
bit [7:0] dout;
bit tx_valid;
function new(string name = "ram_seq_item");
super.new(name);
endfunction
function string convert2string();
    return $sformatf("%s rst_n = 0b%0b din = 0b%0b rx_valid = 0b%0b dout = 0b%0b
                    tx_valid = 0b%0b ", super.convert2string(), rst_n, din, rx_valid, dout, tx_valid);
endfunction
function string convert2string_stimulus();
    return $sformatf("rst_n = 0b%0b din = 0b%0b rx_valid = 0b%0b ", rst_n, din, rx_valid);
endfunction
    constraint write_only{
rx_valid dist {1: = 90,0: = 10};
rst_n dist {1: = 98,0: = 2};
din[9:8] inside {2'b00,2'b01};    }
    constraint read_only{
rx_valid dist {1: = 90,0: = 10};
rst_n dist {1: = 98,0: = 2};
din[9:8] inside {2'b10,2'b11};    }
    constraint read_and_write{
rx_valid dist {1: = 90,0: = 10};
rst_n dist {1: = 98,0: = 2};    }
endclass
endpackage

```

INTERFACE

```

interface ram_if (clk);
input clk;
logic rst_n;
logic rx_valid;
logic tx_valid;
logic [9:0] din;
logic [7:0] dout;
endinterface

```


RAM CONFIGURATION

```
package ram_config_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ram_config extends uvm_object;
`uvm_object_utils(ram_config);
virtual ram_if ramif;
    function new(string name = "ram_config");
        super.new(name);
    endfunction
endclass
endpackage
```

RAM SCOREBOARD

```
package score_board_pkg;
import uvm_pkg::*;
import sequence_item_pkg::*;
`include "uvm_macros.svh"
class ram_scoreboard extends uvm_scoreboard;
`uvm_component_utils(ram_scoreboard)
uvm_analysis_export #(ram_seq_item) sb_export;
uvm_tlm_analysis_fifo #(ram_seq_item) sb_fifo;
ram_seq_item seq_item_sb;
logic [7:0] dout_ref;
logic tx_valid_ref;
logic [7:0] rd_add_ref;
logic [7:0] wr_add_ref;
logic [7:0] mem_ref [255:0];
int error_count = 0;
int correct_count = 0;
    function new(string name = "ram_scoreboard", uvm_component parent = null);
        super.new(name, parent);
    endfunction //new()
function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    seq_item_sb = new("seq_item_sb");
    sb_export = new("sb_export", this);
    sb_fifo = new("sb_fifo", this);
endfunction
```

```

function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    sb_export.connect(sb_fifo.analysis_export);
endfunction
task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
        sb_fifo.get(seq_item_sb);
        ref_model(seq_item_sb);

        if((seq_item_sb.dout != dout_ref)|| (seq_item_sb.tx_valid != tx_valid_ref))
            begin
                error_count = error_count + 1;
                `uvm_error("run_phase", $sformatf("comparasion_failed, transaction recieved
by dut : %s while the refrence dout: 0b%0b and tx_valid_ref
: 0b%0b ", seq_item_sb.convert2string(), dout_ref, tx_valid_ref))
            end
        else begin
            correct_count = correct_count + 1 ;
            `uvm_info("run_phase", $sformatf("correct data out
: %s", seq_item_sb.convert2string()), UVM_HIGH)
        end
    end
endtask
task ref_model(ram_seq_item seq_item_chk);
    if(! seq_item_chk.rst_n)
        begin
            dout_ref = 0;
            rd_add_ref = 0;
            wr_add_ref = 0;
            tx_valid_ref = 0;
            foreach (mem_ref[i]) begin
                mem_ref[i] = 0;
            end
        end else
        begin
            if(seq_item_chk.rx_valid)
                begin
                    case (seq_item_chk.din[9: 8])
                        2'b00: begin
                            wr_add_ref = seq_item_chk.din[7: 0];
                            tx_valid_ref = 0;
                        end
                    end
                end
        end
    end
end

```

```

2'b01: begin
    mem_ref[wr_add_ref] = seq_item_chk.din[7:0];
    tx_valid_ref = 0;
end
2'b10: begin
    rd_add_ref = seq_item_chk.din[7:0];
    tx_valid_ref = 0;
end
2'b11: begin
    dout_ref = mem_ref[rd_add_ref];
    tx_valid_ref = 1;
end          endcase
    end
    end
endtask
function void report_phase(uvm_phase phase) ;
    super.report_phase(phase);
    `uvm_info("report phase", $sformatf("total successful transactions is %0d",
                                         correct_count), UVM_MEDIUM)
    `uvm_info("report phase", $sformatf("total wrong transactions is %0d",
                                         error_count), UVM_MEDIUM)
endfunction
endclass //className extends superClass
endpackage

```

RAM MONITOR

```

package monitor_pkg;
import sequence_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ram_monitor extends uvm_monitor;
`uvm_component_utils(ram_monitor)
virtual ram_if ramif;
ram_seq_item rsp_seq_item;
uvm_analysis_port #(ram_seq_item) mon_ap;
    function new(string name = "ram_monitor", uvm_component parent = null);
        super.new(name, parent);
        mon_ap = new("mon_ap", this);
    endfunction //new()
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
    endtask
endclass

```

```

    forever begin
        rsp_seq_item = ram_seq_item::type_id::create("rsp_seq_item");
        @(negedge ramif.clk);
        rsp_seq_item.din = ramif.din;
        rsp_seq_item.rx_valid = ramif.rx_valid;
        rsp_seq_item.rst_n = ramif.rst_n;
        rsp_seq_item.dout = ramif.dout;
        rsp_seq_item.tx_valid = ramif.tx_valid;
        mon_ap.write(rsp_seq_item);
        `uvm_info("run_phase",rsp_seq_item.convert2string(),UVM_HIGH)
    end
endtask
endclass //className extends superClass
endpackage

```

RAM ENVIRONMENT

```

package ram_env_pkg;
import ram_driver_pkg::*;
import sequencer_pkg::*;
import uvm_pkg::*;
import agent_pkg::*;
import sequence_item_pkg::*;
import score_board_pkg::*;
import coverage_collector::*;
`include "uvm_macros.svh"
class ram_env extends uvm_env;
    `uvm_component_utils(ram_env)
    ram_coverage cov;
    ram_scoreboard sb;
    ram_agent agt;
    function new(string name = "ram_env",uvm_component parent = null);
        super.new(name,parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        agt = ram_agent::type_id::create("agt",this);
        sb = ram_scoreboard::type_id::create("sb",this);
        cov = ram_coverage::type_id::create("cov",this);
    endfunction

```

```

function void connect_phase(uvm_phase phase);
  super.connect_phase(phase);
  agt.agt_ap.connect(sb.sb_export);
  agt.agt_ap.connect(cov.cov_export);
endfunction
endclass
endpackage

```

RAM DRIVER

```

package ram_driver_pkg;
import uvm_pkg::*;
import ram_config_pkg::*;
import sequence_item_pkg::*;
`include "uvm_macros.svh"
class ram_driver extends uvm_driver #(ram_seq_item);
  `uvm_component_utils(ram_driver)
  virtual ram_if ramif;
  ram_seq_item stim_seq_item;
  function new( string name = "ram_driver", uvm_component parent = null);
    super.new(name, parent);
  endfunction
  task run_phase(uvm_phase phase);
  super.run_phase(phase);
    forever begin
      stim_seq_item = ram_seq_item::type_id::create("stim_seq_item");
      seq_item_port.get_next_item(stim_seq_item);
      ramif.din = stim_seq_item.din;
      ramif.rx_valid = stim_seq_item.rx_valid;
      ramif.rst_n = stim_seq_item.rst_n;
      @(negedge ramif.clk);
      seq_item_port.item_done();
      `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH)
    end
  endtask
endclass
endpackage

```

RAM COVERAGE COLLECTOR

```

package coverage_collector;
import uvm_pkg:.*;
import sequence_item_pkg:.*;
`include "uvm_macros.svh"
class ram_coverage extends uvm_component;
    `uvm_component_utils(ram_coverage)
    uvm_analysis_export #(ram_seq_item) cov_export;
    uvm_tlm_analysis_fifo #(ram_seq_item) cov_fifo;
    ram_seq_item seq_item_cov;
    covergroup g;
        din_cp: coverpoint seq_item_cov.din ;
        rx_valid_cp: coverpoint seq_item_cov.rx_valid;
        the_order_cp: coverpoint seq_item_cov.din[9:8]{
            bins read_add = {2'b10};
            bins read_data = {2'b11};
            bins write_add = {2'b00};
            bins write_data = {2'b01};        }
        rst_n_cp: coverpoint seq_item_cov.rst_n;
        the_order_with_rx_valid_cp: cross the_order_cp, rx_valid_cp;
        rx_valid_with_rst_n_cp: cross rx_valid_cp, the_order_cp;
        the_order_with_rst_n_cp: cross the_order_cp, rst_n_cp;
    endgroup
    function new(string name = "ram_coverage", uvm_component parent = null);
        super.new(name, parent);
        g = new;
    endfunction
    function void build_phase( uvm_phase phase);
        super.build_phase(phase);
        cov_export = new("cov_export", this);
        cov_fifo = new("cov_fifo", this);
    endfunction
    function void connect_phase(uvm_phase phase);
        super.connect_phase(phase);
        cov_export.connect(cov_fifo.analysis_export);
    endfunction
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            cov_fifo.get(seq_item_cov);
            g.sample();
        end
    endtask
endclass
endpackage

```

RAM AGENT

```

package agent_pkg;
`include "uvm_macros.svh"
import uvm_pkg::*;
import squencer_pkg::*;
import ram_config_pkg::*;
import monitor_pkg::*;
import sequence_item_pkg::*;
import ram_driver_pkg::*;
class ram_agent extends uvm_agent;
`uvm_component_utils(ram_agent)
    ram_driver driver;
    ram_monitor mon;
    ram_squencer sqr;
    ram_config ram_cfg;
    uvm_analysis_port #(ram_seq_item) agt_ap;
    function new(string name = "ram_agent", uvm_component parent = null);
        super.new(name, parent);
        agt_ap = new("agt_ap", this);
    endfunction
    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        if(! uvm_config_db #(ram_config)::get(this, "", "CFG", ram_cfg))
            `uvm_fatal("build_phase", "unable to get configuration object")
        driver = ram_driver::type_id::create("driver", this);
        sqr = ram_squencer::type_id::create("sqr", this);
        mon = ram_monitor::type_id::create("mon", this);
    endfunction

    function void connect_phase(uvm_phase phase);
        driver.ramif = ram_cfg.ramif;
        mon.ramif = ram_cfg.ramif;
        driver.seq_item_port.connect(sqr.seq_item_export);
        mon.mon_ap.connect(agt_ap);
    endfunction
endclass
endpackage

```

RAM DO FILE

```

vlib work

vlog *v +cover

vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover

add wave /top/ramif/*

coverage save top.ucdb -onexit -du work.project_ram

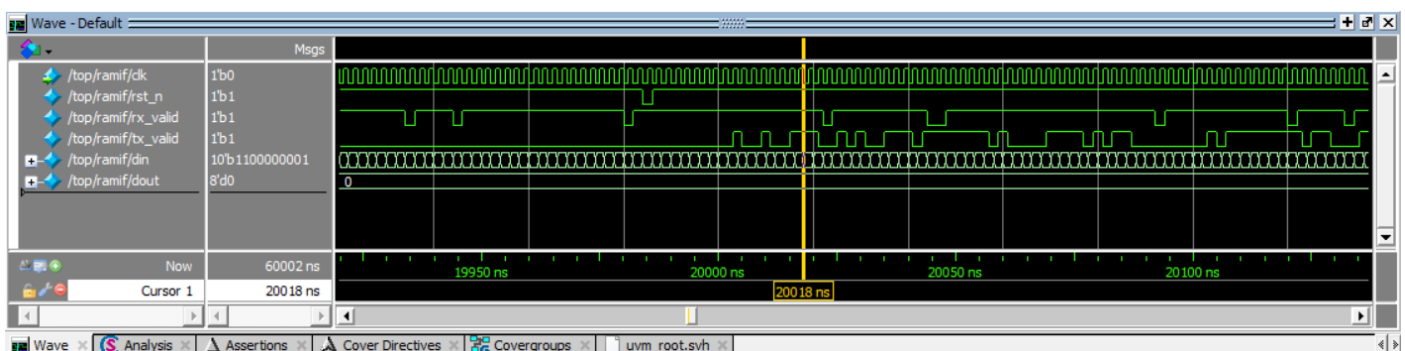
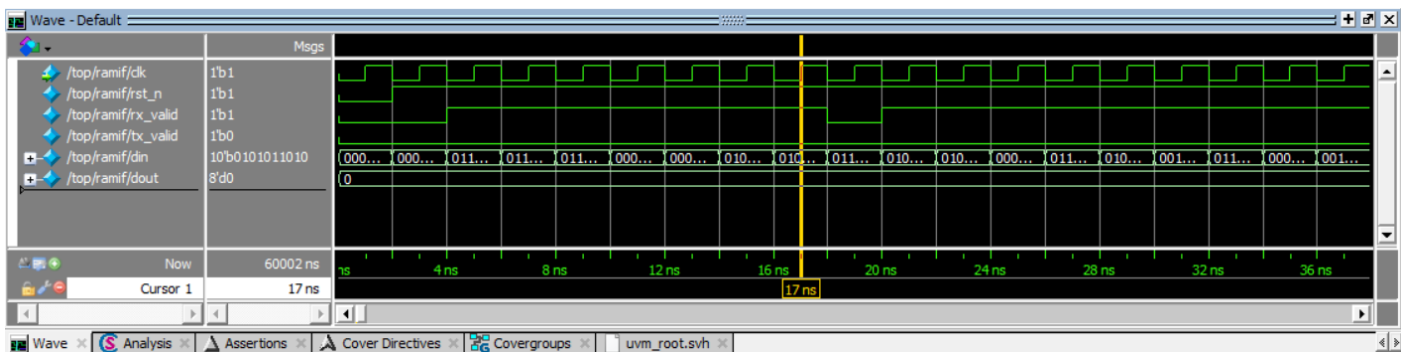
run -all

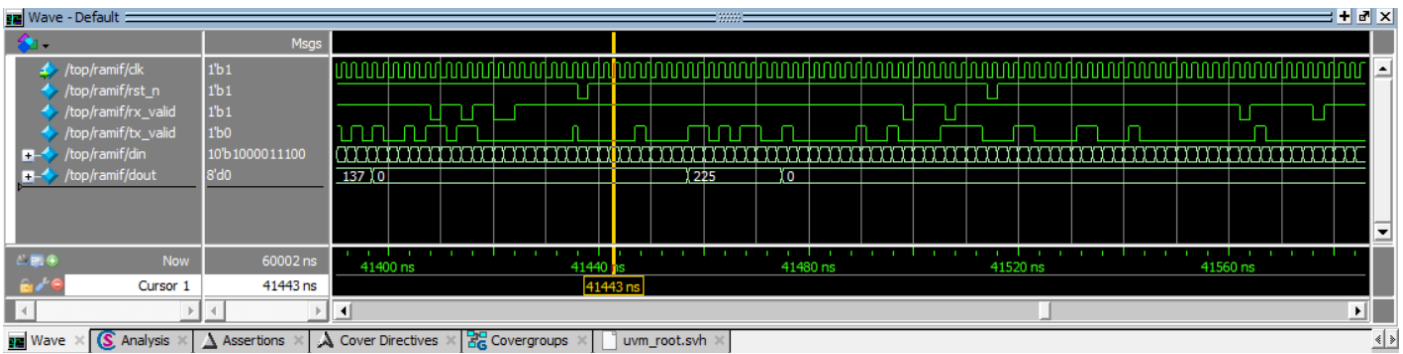
# quit -sim

# vcover report top.ucdb -details -annotate -all -output ram_coverage_rpt.txt

```

WAVEFORM





TRANSCRIPT

```
# UVM_INFO scoreboard.sv(85) @ 60002: uvm_test_top.env.sb [report phase] total successful transactions is 30001
# UVM_INFO scoreboard.sv(87) @ 60002: uvm_test_top.env.sb [report phase] total wrong transactions is 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 14
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# ** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [report phase] 2
# [run_phase] 8
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 60002 ns Iteration: 61 Instance: /top
```

Covergroups			
Name	Class Type	Coverage	Goal
+ /coverage_collector/ram_coverage		100.00%	

Cover Directives										
Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included
/top/dut/SVA_inst/...	SVA	✓	Off	2127	1	Unli...	1	100%		✓
/top/dut/SVA_inst/...	SVA	✓	Off	14927	1	Unli...	1	100%		✓
/top/dut/SVA_inst/...	SVA	✓	Off	7182	1	Unli...	1	100%		✓

SPI RAM CODE & ASSERTION COVERAGE

SPI_RAM > RAM_Code_Assertion_Coverage_Rprt.txt

```

1 Coverage Report by instance with details
2
3 =====
4 === Instance: /\SPI_RAM_TOP#dut_inst /SVA_inst
5 === Design Unit: work.SPI_RAM_SVA
6 =====
7
8 Assertion Coverage:
9     Assertions                3          3          0    100.00%
10  -----
11 Name                        File(Line)                        Failure    Pass
12                               Count          Count
13  -----
14 /\SPI_RAM_TOP#dut_inst /SVA_inst/tx_asser_11
15                               SPI_RAM_SVA.sv(12)                0          1
16 /\SPI_RAM_TOP#dut_inst /SVA_inst/tx_asser_00_01
17                               SPI_RAM_SVA.sv(23)                0          1
18 /\SPI_RAM_TOP#dut_inst /SVA_inst/tx_asser_10
19                               SPI_RAM_SVA.sv(34)                0          1
20
21 Directive Coverage:
22     Directives                3          3          0    100.00%
23
24 DIRECTIVE COVERAGE:
25  -----
26 Name                        Design Design   Lang File(Line)      Hits Status
27                               Unit   UnitType
28  -----
29 /\SPI_RAM_TOP#dut_inst /SVA_inst/tx_chk_11
30                               SPI_RAM_SVA Verilog   SVA   SPI_RAM_SVA.sv(11)
31                               255 Covered
32 /\SPI_RAM_TOP#dut_inst /SVA_inst/tx_chk_00_01
33                               SPI_RAM_SVA Verilog   SVA   SPI_RAM_SVA.sv(22)
34                               513 Covered
35 /\SPI_RAM_TOP#dut_inst /SVA_inst/tx_chk_10
36                               SPI_RAM_SVA Verilog   SVA   SPI_RAM_SVA.sv(33)
37                               256 Covered
38
39 =====
40 === Instance: /\SPI_RAM_TOP#dut_inst
41 === Design Unit: work.SPI_RAM_AFTER
42 =====

```

ram_coverage_rpt.txt

```

1 Coverage Report by instance with details
2
3 =====
4 === Instance: /\top#dut
5 === Design Unit: work.project_ram
6 =====
7 Branch Coverage:
8   Enabled Coverage          Bins      Hits      Misses  Coverage
9   -----
10  Branches                  7        7        0    100.00%
11
12 =====Branch Details=====
13
14 Branch Coverage for instance /\top#dut
15
16   Line      Item                      Count      Source
17   ----      -
18   File ram.v
19   -----IF Branch-----
20   14                      30608      Count coming in to IF
21   14          1                1239          if (~rst_n) begin
22
23   26          1            26408          else if (rx_valid) begin
24
25                      2961      All False Count
26 Branch totals: 3 hits of 3 branches = 100.00%
27
28   -----IF Branch-----
29   28                      26408      Count coming in to IF
30   28          1                6653          if (din[9:8] == 2'b00) begin
31
32   32          1                6588          else if (din[9:8] == 2'b01) begin
33
34   38          1                6583          else if (din[9:8] == 2'b10) begin
35
36   42          1                6584          else begin
37
38 Branch totals: 4 hits of 4 branches = 100.00%
39

```

ram_coverage_rpt.txt

```

41 Condition Coverage:
42   Enabled Coverage          Bins   Covered   Misses   Coverage
43   -----
44   Conditions                3       3        0    100.00%
45
46   =====Condition Details=====
47
48 Condition Coverage for instance /\top#dut  --
49
50   File ram.v
51   -----Focused Condition View-----
52   Line      28 Item      1  (din[9:8] == 0)
53   Condition totals: 1 of 1 input term covered = 100.00%
54
55       Input Term   Covered   Reason for no coverage   Hint
56       -----
57   (din[9:8] == 0)          Y
58
59       Rows:        Hits   FEC Target          Non-masking condition(s)
60       -----
61   Row   1:          1  (din[9:8] == 0)_0      -
62   Row   2:          1  (din[9:8] == 0)_1      -
63
64   -----Focused Condition View-----
65   Line      32 Item      1  (din[9:8] == 1)
66   Condition totals: 1 of 1 input term covered = 100.00%
67
68       Input Term   Covered   Reason for no coverage   Hint
69       -----
70   (din[9:8] == 1)          Y
71
72       Rows:        Hits   FEC Target          Non-masking condition(s)
73       -----
74   Row   1:          1  (din[9:8] == 1)_0      -
75   Row   2:          1  (din[9:8] == 1)_1      -
76
77   -----Focused Condition View-----
78   Line      38 Item      1  (din[9:8] == 2)
79   Condition totals: 1 of 1 input term covered = 100.00%
80
81       Input Term   Covered   Reason for no coverage   Hint
82       -----
83   (din[9:8] == 2)          Y
84
85       Rows:        Hits   FEC Target          Non-masking condition(s)
86       -----
87   Row   1:          1  (din[9:8] == 2)_0      -
88   Row   2:          1  (din[9:8] == 2)_1      -

```

```

ram_coverage_rpt.txt
91 Statement Coverage:
92   Enabled Coverage      Bins    Hits    Misses Coverage
93   -----
94   Statements           19      19        0  100.00%
95
96 =====Statement Details=====
97
98 Statement Coverage for instance /\top#dut --
99
100   Line      Item              Count    Source
101   ----      -
102   File ram.v
103   1                                module project_ram(din, rx_valid, dout, tx_valid, clk, rst_n);
104
105   2                                parameter MEM_DEPTH = 256;
106
107   3                                parameter ADDR_SIZE = 8;
108
109   4                                input rx_valid, clk, rst_n;
110
111   5                                input [9:0] din;
112
113   6                                output reg tx_valid;
114
115   7                                output reg [7:0] dout;
116
117   8                                reg [ADDR_SIZE-1:0] addr_rd, addr_wr;
118
119   9                                reg [7:0] mem [MEM_DEPTH-1:0];
120
121  10                                reg [8:0] i ;
122
123  11                                /*
124
125  12                                bugs if rst_n activated the internal register of read/write adderesses is not cleared*/
126
127  13                                1                                30608 always @(posedge clk or negedge rst_n) begin
128
129  14                                if (~rst_n) begin
130
131  15                                1                                1239         dout <= 8'b0;
132
133  16                                1                                1239         tx_valid <= 1'b0;
134
135  17                                1                                1239         addr_rd<=0;
136
137  18                                1                                1239         addr_wr<=0;
138
139  19                                1                                1239         i=0;
140
141  20                                1                                1239         for (i = 0; i < MEM_DEPTH; i=i+1) begin
142
143  20                                2                                317184
144  21                                1                                317184         mem [i] <= 1'b0;
145
146  22                                end

```

ram_coverage_rpt.txt

```


193 Toggle Coverage:
194   Enabled Coverage      Bins      Hits      Misses  Coverage
195   -----
196   Toggles              94       94        0    100.00%
197
198   =====Toggle Details=====
199
200 Toggle Coverage for instance /\top#dut  --
201
202                               Node      1H->0L      0L->1H  "Coverage"
203                               -----
204                               addr_rd[7-0]      1          1    100.00
205                               addr_wr[7-0]      1          1    100.00
206                               clk                1          1    100.00
207                               din[0-9]          1          1    100.00
208                               dout[7-0]          1          1    100.00
209                               i[8-0]             1          1    100.00
210                               rst_n              1          1    100.00
211                               rx_valid            1          1    100.00
212                               tx_valid            1          1    100.00
213
214 Total Node Count      =      47
215 Toggled Node Count   =      47
216 Untoggled Node Count =       0
217
218 Toggle Coverage      =    100.00% (94 of 94 bins)
219
220
221 Total Coverage By Instance (filtered view): 100.00%
222
223

```

SPI RAM FUNCTIONAL COVERAGE

ram_FC_rpt.txt

2291					
2292					
2293	COVERGROUP COVERAGE:				
2294	-----				
2295	Covergroup	Metric	Goal	Bins	Status
2296					
2297	-----				
2298	TYPE /coverage_collector/ram_coverage/g	100.00%	100	-	Covered
2299	covered/total bins:	96	96	-	
2300	missing/total bins:	0	96	-	
2301	% Hit:	100.00%	100	-	
2302	Coverpoint din_cp	100.00%	100	-	Covered
2303	covered/total bins:	64	64	-	
2304	missing/total bins:	0	64	-	
2305	% Hit:	100.00%	100	-	
2306	bin auto[0:15]	485	1	-	Covered
2307	bin auto[16:31]	458	1	-	Covered
2308	bin auto[32:47]	470	1	-	Covered
2309	bin auto[48:63]	489	1	-	Covered
2310	bin auto[64:79]	477	1	-	Covered
2311	bin auto[80:95]	457	1	-	Covered
2312	bin auto[96:111]	440	1	-	Covered
2313	bin auto[112:127]	452	1	-	Covered
2314	bin auto[128:143]	495	1	-	Covered
2315	bin auto[144:159]	478	1	-	Covered
2316	bin auto[160:175]	503	1	-	Covered
2317	bin auto[176:191]	487	1	-	Covered
2318	bin auto[192:207]	460	1	-	Covered
2319	bin auto[208:223]	467	1	-	Covered
2320	bin auto[224:239]	491	1	-	Covered
2321	bin auto[240:255]	458	1	-	Covered
2322	bin auto[256:271]	474	1	-	Covered
2323	bin auto[272:287]	457	1	-	Covered
2324	bin auto[288:303]	522	1	-	Covered
2325	bin auto[304:319]	462	1	-	Covered
2326	bin auto[320:335]	447	1	-	Covered
2327	bin auto[336:351]	452	1	-	Covered
2328	bin auto[352:367]	485	1	-	Covered
2329	bin auto[368:383]	503	1	-	Covered
2330	bin auto[384:399]	485	1	-	Covered
2331	bin auto[400:415]	464	1	-	Covered
2332	bin auto[416:431]	456	1	-	Covered
2333	bin auto[432:447]	421	1	-	Covered
2334	bin auto[448:463]	459	1	-	Covered
2335	bin auto[464:479]	473	1	-	Covered
2336	bin auto[480:495]	459	1	-	Covered
2337	bin auto[496:511]	456	1	-	Covered
2338	bin auto[512:527]	462	1	-	Covered
2339	bin auto[528:543]	465	1	-	Covered
2340	bin auto[544:559]	430	1	-	Covered
2341	bin auto[560:575]	449	1	-	Covered

 ram_FC_rpt.txt

2342	bin auto[576:591]	486	1	-	Covered
2343	bin auto[592:607]	489	1	-	Covered
2344	bin auto[608:623]	475	1	-	Covered
2345	bin auto[624:639]	487	1	-	Covered
2346	bin auto[640:655]	440	1	-	Covered
2347	bin auto[656:671]	449	1	-	Covered
2348	bin auto[672:687]	451	1	-	Covered
2349	bin auto[688:703]	476	1	-	Covered
2350	bin auto[704:719]	468	1	-	Covered
2351	bin auto[720:735]	458	1	-	Covered
2352	bin auto[736:751]	504	1	-	Covered
2353	bin auto[752:767]	472	1	-	Covered
2354	bin auto[768:783]	497	1	-	Covered
2355	bin auto[784:799]	469	1	-	Covered
2356	bin auto[800:815]	505	1	-	Covered
2357	bin auto[816:831]	480	1	-	Covered
2358	bin auto[832:847]	487	1	-	Covered
2359	bin auto[848:863]	483	1	-	Covered
2360	bin auto[864:879]	455	1	-	Covered
2361	bin auto[880:895]	496	1	-	Covered
2362	bin auto[896:911]	447	1	-	Covered
2363	bin auto[912:927]	435	1	-	Covered
2364	bin auto[928:943]	470	1	-	Covered
2365	bin auto[944:959]	457	1	-	Covered
2366	bin auto[960:975]	459	1	-	Covered
2367	bin auto[976:991]	461	1	-	Covered
2368	bin auto[992:1007]	439	1	-	Covered
2369	bin auto[1008:1023]	458	1	-	Covered
2370	Coverpoint rx_valid_cp	100.00%	100	-	Covered
2371	covered/total bins:	2	2	-	
2372	missing/total bins:	0	2	-	
2373	% Hit:	100.00%	100	-	
2374	bin auto[0]	3023	1	-	Covered
2375	bin auto[1]	26978	1	-	Covered
2376	Coverpoint the_order_cp	100.00%	100	-	Covered
2377	covered/total bins:	4	4	-	
2378	missing/total bins:	0	4	-	
2379	% Hit:	100.00%	100	-	
2380	bin read_add	7461	1	-	Covered
2381	bin read_data	7498	1	-	Covered
2382	bin write_add	7567	1	-	Covered
2383	bin write_data	7475	1	-	Covered
2384	Coverpoint rst_n_cp	100.00%	100	-	Covered
2385	covered/total bins:	2	2	-	
2386	missing/total bins:	0	2	-	
2387	% Hit:	100.00%	100	-	
2388	bin auto[0]	628	1	-	Covered
2389	bin auto[1]	29373	1	-	Covered
2390	Cross the_order_with_rx_valid_cp	100.00%	100	-	Covered
2391	covered/total bins:	8	8	-	
2392	missing/total bins:	0	8	-	
2393	% Hit:	100.00%	100	-	

ram_FC_rpt.txt

```

2390      Cross the_order_with_rx_valid_cp      100.00%      100      -      Covered
2391      covered/total bins:                      8          8      -
2392      missing/total bins:                      0          8      -
2393      % Hit:                                100.00%      100      -
2394      Auto, Default and User Defined Bins:
2395      bin <write_data,auto[1]>                  6725          1      -      Covered
2396      bin <write_data,auto[0]>                  750           1      -      Covered
2397      bin <write_add,auto[1]>                  6802          1      -      Covered
2398      bin <write_add,auto[0]>                  765           1      -      Covered
2399      bin <read_data,auto[1]>                  6741          1      -      Covered
2400      bin <read_data,auto[0]>                  757           1      -      Covered
2401      bin <read_add,auto[1]>                  6710          1      -      Covered
2402      bin <read_add,auto[0]>                  751           1      -      Covered
2403      Cross rx_valid_with_rst_n_cp            100.00%      100      -      Covered
2404      covered/total bins:                      8          8      -
2405      missing/total bins:                      0          8      -
2406      % Hit:                                100.00%      100      -
2407      Auto, Default and User Defined Bins:
2408      bin <auto[1],write_data>                  6725          1      -      Covered
2409      bin <auto[0],write_data>                  750           1      -      Covered
2410      bin <auto[1],write_add>                  6802          1      -      Covered
2411      bin <auto[0],write_add>                  765           1      -      Covered
2412      bin <auto[1],read_data>                  6741          1      -      Covered
2413      bin <auto[0],read_data>                  757           1      -      Covered
2414      bin <auto[1],read_add>                  6710          1      -      Covered
2415      bin <auto[0],read_add>                  751           1      -      Covered
2416      Cross the_order_with_rst_n_cp            100.00%      100      -      Covered
2417      covered/total bins:                      8          8      -
2418      missing/total bins:                      0          8      -
2419      % Hit:                                100.00%      100      -
2420      Auto, Default and User Defined Bins:
2421      bin <write_data,auto[1]>                  7325          1      -      Covered
2422      bin <write_data,auto[0]>                  150           1      -      Covered
2423      bin <write_add,auto[1]>                  7404          1      -      Covered
2424      bin <write_add,auto[0]>                  163           1      -      Covered
2425      bin <read_data,auto[1]>                  7329          1      -      Covered
2426      bin <read_data,auto[0]>                  169           1      -      Covered
2427      bin <read_add,auto[1]>                  7315          1      -      Covered
2428      bin <read_add,auto[0]>                  146           1      -      Covered
2429
2430      TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
2431
2432      DIRECTIVE COVERAGE:
2433      -----
2434      Name                                Design Design  Lang File(Line)      Hits Status
2435      Unit  UnitType
2436      -----
2437      /top/dut/SVA_inst/tx_chk_11          SPI_RAM_SVA Verilog  SVA  SVA.sv(13)          2127 Covered
2438      /top/dut/SVA_inst/tx_chk_00_01       SPI_RAM_SVA Verilog  SVA  SVA.sv(24)          14927 Covered
2439      /top/dut/SVA_inst/tx_chk_10          SPI_RAM_SVA Verilog  SVA  SVA.sv(35)           7182 Covered
2440
2441      TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 3

```

SPI SLAVE WITH SINGLE PORT RAM**Full Wrapper Environment****WRAPPER RTL WITHOUT BUGS**

```
module spi_wrapper(mosi, miso, ss_n, clk, rst_n);  
input mosi, ss_n, clk, rst_n;  
output miso;  
wire rx_valid, tx_valid;  
wire [9:0] rx_data;  
wire [7:0] tx_data;  
SPI spislave(mosi, miso, ss_n, clk, rst_n, rx_valid, rx_data, tx_valid, tx_data);  
project_ram mem(rx_data, rx_valid, tx_data, tx_valid, clk, rst_n);  
endmodule
```

SIPO RTL WITHOUT BUGS

```
module SIPO (clk, SS_n , MOSI, rx_data);  
input clk, MOSI, SS_n;  
output [9:0] rx_data;  
reg [9:0] tmp;  
  
always @(posedge clk)  
begin  
    if(!SS_n)  
        tmp = {tmp[8:0], MOSI};  
end  
assign rx_data = tmp;  
endmodule
```

PISO RTL WITHOUT BUGS

```

module PISO(clk, tx_valid, counter, tx_data, dout);

output reg dout;
input [7:0] tx_data;
input clk ;
input tx_valid ;
input [3:0] counter;
reg [7:0]temp;

always @ (posedge clk) begin
    if (tx_valid) begin
        if (counter == 0)
            temp <= tx_data;
        else begin
            dout <= temp[7];
            temp <= {temp[6:0],1'b0};
        end
    end
end
endmodule

```

COUNTER RTL WITHOUT BUGS

```

module up_counter(input clk, rst_n, output[3:0] counter);
reg [3:0] counter_up;

always @(posedge clk or negedge rst_n)
begin
    if(~rst_n||counter_up == 4'd10)
        counter_up <= 4'd0;
    else
        counter_up <= counter_up + 4'd1;
    end
assign counter = counter_up;
endmodule

```

WRAPPER TOP

```
import Wrapper_test_pkg::* ;
import uvm_pkg::*;
`include "uvm_macros.svh"
module top();
  bit clk ;
  initial begin
    forever #2 clk = ~clk ;
  end
  Wrapper_if Wrap_if(clk);
  spi_wrapper DUT(Wrap_if.MOSI , Wrap_if.MISO, Wrap_if.SS_n, Wrap_if.clk, Wrap_if.rst_n);
  initial begin
    uvm_config_db#(virtual Wrapper_if)::set(null, "uvm_test_top", "Wrapper_IF", Wrap_if);
    run_test("Wrapper_test") ;
  end
endmodule
```

WRAPPER TEST

```
package Wrapper_test_pkg;
  import Wrapper_env::* ;
  import uvm_pkg::*;
  import Wrapper_config_obj::*;
  import sequences::*;
  `include "uvm_macros.svh"
  class Wrapper_test extends uvm_test ;
    `uvm_component_utils (Wrapper_test) ;
    Wrapper_env my_env ;
    virtual Wrapper_if Wrapper_test_vif ;
    Wrapper_config_obj Wrapper_config_obj_test ;
    Wrapper_main_seq main_seq ;
    Wrapper_reset_seq reset_seq;
    function new(string name = "Wrapper_test", uvm_component parent = null );
      super.new(name, parent);
    endfunction
    function void build_phase(uvm_phase phase);
      super.build_phase(phase);
      Wrapper_config_obj_test
        = Wrapper_config_obj::type_id::create("Wrapper_config_obj_test", this) ;
      my_env = Wrapper_env::type_id::create("my_env", this) ;
      main_seq = Wrapper_main_seq::type_id::create("main_seq", this) ;
      reset_seq = Wrapper_reset_seq::type_id::create("reset_seq", this) ;
```

```

if(! uvm_config_db#(virtual Wrapper_if)
    : : get(this, "", "Wrapper_IF", Wrapper_config_obj_test.Wrapper_config_vif))
    `uvm_fatal("run_phase", "ERROR in getting virtual interface");
uvm_config_db#(Wrapper_config_obj)::set(this, "
    * ", "CFG", Wrapper_config_obj_test);
endfunction
task run_phase (uvm_phase phase);
    super.run_phase(phase);
    phase.raise_objection(this);
    `uvm_info ("run_phase", "RESET_ASSERTED ", UVM_LOW);
    reset_seq.start(my_env.agt.sqr);
    `uvm_info ("run_phase", "RESET_DEASSERTED ", UVM_LOW);
    `uvm_info("run_phase", "Started generating stimulus", UVM_LOW);
    main_seq.start(my_env.agt.sqr);
    `uvm_info ("run_phase", "Stimulus generation ended ", UVM_LOW);
    phase.drop_objection(this);
endtask
endclass
endpackage

```

WRAPPER SEQUENCER

```

package sequences ;
import Wrapper_pack ::*;
import Wrapper_seq_item ::*;
import uvm_pkg ::*;
`include "uvm_macros.svh"
Wrapper_seq_item seq_item_static ;
class Wrapper_reset_seq extends uvm_sequence #(Wrapper_seq_item);
    `uvm_object_utils(Wrapper_reset_seq)
    function new( string name = "Wrapper_reset_seq");
        super.new(name);
    endfunction : new
    task body ;
        seq_item_static = Wrapper_seq_item::type_id::create("seq_item_static");
        start_item(seq_item_static);
        seq_item_static.rst_n = 0;
        seq_item_static.SS_n = 0 ;
        seq_item_static.MOSI = 0;
        finish_item(seq_item_static);
    endtask
endclass : Wrapper_reset_seq
class Wrapper_main_seq extends uvm_sequence #(Wrapper_seq_item) ;
    `uvm_object_utils(Wrapper_main_seq)

```

```

Wrapper_seq_item seq_item_main, seq_item_static;
function new(string name = "Wrapper_main_seq");
    super.new(name);
endfunction : new
task body ;
    seq_item_static = Wrapper_seq_item::type_id::create("seq_item_static");
    repeat(10000) begin
        seq_item_main = Wrapper_seq_item::type_id::create("seq_item_main");
        start_item(seq_item_main);
        assert(seq_item_static.randomize() with {SS_n == 0;});
        update_seq_item();
        finish_item(seq_item_main);
    end
endtask
task update_seq_item();
    seq_item_main.rst_n = seq_item_static.rst_n;
    seq_item_main.SS_n = seq_item_static.SS_n;
    seq_item_main.MOSI = seq_item_static.MOSI;
    seq_item_main.MISO = seq_item_static.MISO;
    seq_item_main.data_holder = seq_item_static.data_holder;
    seq_item_main.address_sent = seq_item_static.address_sent;
endtask
endclass : Wrapper_main_seq
endpackage

```

SEQUENCER ITEM

```

package Wrapper_seq_item ;
import Wrapper_pack ::*;
import uvm_pkg ::*;
`include "uvm_macros.svh"
class Wrapper_seq_item extends uvm_sequence_item ;
    `uvm_object_utils(Wrapper_seq_item)
    rand logic MOSI, SS_n, rst_n;
    rand logic [10:0] data_holder ;
    logic MISO ;
    integer rand_no ;
    bit address_sent;
    function void pre_randomize();
        if(SS_n)
            rand_no = 0;

```

```

if(!rst_n) begin
    address_sent = 0;
    rand_no = 0;
end
else begin
    rand_no = rand_no + 1;
    if(data_holder[10:8] == 3'b110)
        address_sent = 1;
    else if(data_holder[10:8] == 3'b111)
        address_sent = 0;
    end
endfunction
constraint wrapper_c
{
    rst_n dist {0:= 1 , 1:= 800};
    if(address_sent && data_holder[10])
    {
        data_holder[9:8] == 2'b11;
    }
    else
    {
        data_holder[10:8] inside{3'b110,3'b001,3'b000};
    } }
function new(string name = "Wrapper_seq_item");
    super.new(name);
    data_holder = 0;
    rand_no = 0;
endfunction
function string convert2string();
    return $sformatf("%s reset = %b , SS_n = %b , MOSI = %b , MISO
        = %b , address_sent = %b"
        , super.convert2string(), rst_n, SS_n, MOSI, MISO, address_sent);
endfunction
function string convert2string_op();
    return $sformatf("%s reset = %b , SS_n = %b , MOSI = %b , data_holder
        = %h , address_sent = %b"
        , super.convert2string(), rst_n, SS_n, MOSI, data_holder, address_sent);
endfunction
function string convert2string_stimulus();
    return $sformatf("%s reset = %b , SS_n = %b , MOSI = %b "
        , super.convert2string(), rst_n, SS_n, MOSI);
endfunction
endclass
endpackage

```

SCOREBOARD

```

package Wrapper_scoreboard ;
import Wrapper_pack ::* ;
import uvm_pkg::*;
`include "uvm_macros.svh"
import Wrapper_seq_item ::* ;
class Wrapper_scoreboard extends uvm_scoreboard ;
    `uvm_component_utils(Wrapper_scoreboard)
    uvm_analysis_export #(Wrapper_seq_item) sb_export ;
    uvm_tlm_analysis_fifo #(Wrapper_seq_item) sb_fifo ;
    Wrapper_seq_item seq_item_sb ;
    int error_count = 0 ;
    int correct_count = 0 ;
    int bits_counter = 0 ;
    bit u ;
    int counter ;
    logic [7:0] word_exp;
    logic [7:0] word_rec ;
    logic [7:0] mem [255:0];
    logic [7:0] wr_addr;
    logic [7:0] rd_addr;
    function new(string name = "Wrapper_scoreboard", uvm_component parent = null);
        super.new(name, parent) ;
    endfunction : new
    function void build_phase(uvm_phase phase) ;
        super.build_phase(phase) ;
        sb_export = new("sb_export", this) ;
        sb_fifo = new("sb_fifo", this) ;
    endfunction
    function void connect_phase(uvm_phase phase);
        super.connect_phase(phase);
        sb_export.connect(sb_fifo.analysis_export);
    endfunction
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        word_rec = 0 ;
        counter = 0 ;
        forever begin
            sb_fifo.get(seq_item_sb);
            `uvm_info("run_phase", seq_item_sb.convert2string_op(), UVM_HIGH)
            refrence_model(seq_item_sb);
            if(seq_item_sb.data_holder[10:8] == 3'b111 ) begin
                if(counter > 14) begin
                    word_rec[22 - counter] = seq_item_sb.MISO ;
                end
                counter + +;
                if(counter == 23) begin
                    counter = 0;
                end
            end
        end
    endtask
endclass

```



```

        if(word_rec != word_exp) begin
            `uvm_error("run_phase", $sformatf("Comparison Failed, Recieved: %b Expected: %h",
            , word_rec, word_exp))
            error_count ++ ;
        end
        else begin
            `uvm_info("run_phase", "Correct_output in Scoreboard", UVM_HIGH)
            correct_count ++ ;
        end
    end
end
end
else begin
    word_rec = 0 ;
    counter = 0;
end
end
endtask
task refrence_model(Wrapper_seq_item x);
    if(! x.rst_n)
        begin
            for (int i = 0; i < 256; i = i + 1)    mem [i] = 1'b0;
            word_exp = 0;
            wr_addr = 0;
            rd_addr = 0;
            u = 0;
        end
        else begin
            if(x.data_holder[10:8] == 3'b000)
                wr_addr = x.data_holder[7:0];
            if(x.data_holder[10:8] == 3'b001)
                mem[wr_addr] = x.data_holder[7:0];
            if(x.data_holder[10:8] == 3'b110)begin
                rd_addr = x.data_holder[7:0];
                u = 1;
            end
            if((x.data_holder[10:8] == 3'b111)&&(u)) begin
                word_exp = mem[rd_addr];
                u = 0;
            end
        end
    endtask
function void report_phase(uvm_phase phase);
    super.report_phase(phase);
`uvm_info ("report_phase", $sformatf("Total successful transactions: %d ", correct_count), UVM_LOW)
`uvm_info ("report_phase", $sformatf("Total failed transactions: %d ", error_count), UVM_LOW)
endfunction
endclass
endpackage : Wrapper_scoreboard

```

OBJ. CONFIG.

```
package Wrapper_config_obj ;
import uvm_pkg::*;
`include "uvm_macros.svh"
class Wrapper_config_obj extends uvm_object ;
  `uvm_object_utils(Wrapper_config_obj)
  virtual Wrapper_if Wrapper_config_vif;
  function new(string name = "Wrapper_config_obj");
    super.new(name);
  endfunction
endclass
endpackage
```

MONITOR

```
package monitor ;
import Wrapper_seq_item::* ;
import uvm_pkg::*;
`include "uvm_macros.svh"
class MyMonitor extends uvm_monitor ;
  `uvm_component_utils(MyMonitor)
  virtual Wrapper_if Wrapper_vif ;
  Wrapper_seq_item mon_seq_item ;
  uvm_analysis_port #(Wrapper_seq_item) mon_ap ;
  function new (string name = "monitor", uvm_component parent = null );
    super.new(name, parent);
  endfunction
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    mon_ap = new ("mon_ap" , this);
  endfunction
  task run_phase(uvm_phase phase) ;
    super.run_phase(phase);
    forever begin
      mon_seq_item = Wrapper_seq_item :: type_id:: create("mon_seq_item");
      @(negedge Wrapper_vif.clk);
      update_seq_item();
      mon_ap.write(mon_seq_item);
      `uvm_info("run_phase", mon_seq_item.convert2string(), UVM_HIGH)
    end
  end
```

```

endtask
task update_seq_item();
    mon_seq_item.rst_n = Wrapper_vif.rst_n ;
    mon_seq_item.SS_n = Wrapper_vif.SS_n;
    mon_seq_item.MOSI = Wrapper_vif.MOSI;
    mon_seq_item.MISO = Wrapper_vif.MISO;
    mon_seq_item.address_sent = Wrapper_vif.address_sent;
    mon_seq_item.data_holder = Wrapper_vif.data_holder;
endtask
endclass
endpackage : monitor

```

INTERFACE

```

import Wrapper_pack :: * ;
interface Wrapper_if (input clk );
    logic MOSI, SS_n, rst_n;
    logic MISO;
    logic address_sent ;
    logic [10:0] data_holder ;
endinterface

```

ENVIRONMENT

```

package Wrapper_env ;
import uvm_pkg::*;
`include "uvm_macros.svh"
import Wrapper_agent::*;
import Wrapper_config_obj::*;
import Wrapper_scoreboard::*;
import Wrapper_coverage::*;
class Wrapper_env extends uvm_env ;
    `uvm_component_utils(Wrapper_env) ;
    Wrapper_agent agt ;
    Wrapper_scoreboard sb ;
    coverage cov ;
    function new(string name = "Wrapper_env", uvm_component parent = null );
        super.new(name, parent);
    endfunction

```

```

function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    agt = Wrapper_agent::type_id::create("agt", this);
    sb = Wrapper_scoreboard::type_id::create("sb", this);
    cov = coverage::type_id::create("cov", this);
endfunction
function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    agt.agt_ap.connect(sb.sb_export);
    agt.agt_ap.connect(cov.cov_export);
endfunction
endclass
endpackage

```

DRIVER

```

package Wrapper_driver ;
import Wrapper_config_obj::*;
import Wrapper_seq_item::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class Wrapper_driver extends uvm_driver #(Wrapper_seq_item);
    `uvm_component_utils(Wrapper_driver)
    virtual Wrapper_vif Wrapper_vif;
    Wrapper_seq_item seq_item;
    function new(string name = "Wrapper_driver", uvm_component parent
        = null);
        super.new(name, parent);
    endfunction
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            seq_item = Wrapper_seq_item::type_id::create("seq_item");
            seq_item_port.get_next_item(seq_item);
            for (int i = 0; i < 11; i++) begin
                update_if(i);
                @(negedge Wrapper_vif.clk);
            end
            if(seq_item.data_holder[10:8] == 3'b111)
                repeat(12) begin
                    @(negedge Wrapper_vif.clk);
                end
        end
    endtask
endclass

```

```

    @(negedge Wrapper_vif.clk) ;
    seq_item.SS_n = 1;
    update_if(10);
    @(negedge Wrapper_vif.clk) ;
    seq_item_port.item_done();
    `uvm_info("run_phase",seq_item.convert2string(),UVM_HIGH)
end
endtask
task update_if(int i);
    Wrapper_vif.rst_n = seq_item.rst_n;
    Wrapper_vif.SS_n = seq_item.SS_n;
    Wrapper_vif.MOSI = seq_item.data_holder[10 - i];
    Wrapper_vif.address_sent = seq_item.address_sent;
    Wrapper_vif.data_holder = seq_item.data_holder;
endtask
endclass
endpackage : Wrapper_driver

```

COVERAGE

```

package Wrapper_coverage ;
import uvm_pkg::*;
`include "uvm_macros.svh"
import Wrapper_seq_item::* ;
class coverage extends uvm_component ;
    `uvm_component_utils(coverage)
    uvm_analysis_export #(Wrapper_seq_item) cov_export ;
    uvm_tlm_analysis_fifo #(Wrapper_seq_item) cov_fifo ;
    Wrapper_seq_item seq_item_cov ;
    covergroup Wrapper_cg ;
        reset_cp : coverpoint seq_item_cov.rst_n;
        SS_n_cp : coverpoint seq_item_cov.SS_n ;
        MOSI_cp : coverpoint seq_item_cov.MOSI;
        MISO_cp : coverpoint seq_item_cov.MISO ;
        operation_cp : coverpoint seq_item_cov.data_holder[10:8]
        {
            bins WRITE_add = {3'b000};
            bins WRITE_data = {3'b001};
            bins READ_add = {3'b110};
            bins READ_data = {3'b111};
            illegal_bins Invalid_op = default ;
        }
    }
endclass
endpackage

```

```

READ_op : cross operation_cp, MISO_cp
{
    ignore_bins not_read0 = binsof(operation_cp.WRITE_add);
    ignore_bins not_read1 = binsof(operation_cp.WRITE_data);
    ignore_bins not_read2 = binsof(operation_cp.READ_add);
}
endgroup
function new(string name = "coverage", uvm_component parent);
    super.new(name, parent);
    Wrapper_cg = new();
endfunction
function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    cov_export = new("cov_export", this);
    cov_fifo = new("cov_fifo", this);
endfunction
function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    cov_export.connect(cov_fifo.analysis_export);
endfunction
task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
        cov_fifo.get(seq_item_cov);
        Wrapper_cg.sample();
    end
endtask
endclass
endpackage : Wrapper_coverage

```

AGENT

```

package Wrapper_agent;
import Wrapper_config_obj ::*;
import Wrapper_seq_item ::*;
import uvm_pkg ::*;
import monitor ::*;
import Wrapper_driver ::*;
import sequencer ::*;
`include "uvm_macros.svh"
class Wrapper_agent extends uvm_agent;
    `uvm_component_utils(Wrapper_agent)
    MyMonitor mon;

```

```

MySequencer sqr ;
Wrapper_driver drv ;
Wrapper_config_obj Wrapper_cfg ;
uvm_analysis_port #(Wrapper_seq_item) agt_ap;
function new (string name = "Wrapper_agent", uvm_component parent
    = null );
    super.new(name, parent);
endfunction
function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(! uvm_config_db#(Wrapper_config_obj)::get(this, "", "CFG", Wrapper_cfg))
        `uvm_fatal("build_phase", "Error in reading configurable object")
    mon = MyMonitor :: type_id :: create ("mon", this) ;
    sqr = MySequencer :: type_id :: create ("sqr", this) ;
    drv = Wrapper_driver :: type_id :: create ("drv", this);
    agt_ap = new("agt_ap", this);
endfunction : build_phase
function void connect_phase(uvm_phase phase) ;
    drv.Wrapper_vif = Wrapper_cfg.Wrapper_config_vif ;
    mon.Wrapper_vif = Wrapper_cfg.Wrapper_config_vif ;
    drv.seq_item_port.connect(sqr.seq_item_export) ;
    mon.mon_ap.connect(agt_ap) ;
endfunction
endclass
endpackage : Wrapper_agent

```

PACKAGE

```

package Wrapper_pack ;
    typedef enum bit[2:0] {IDLE, CHK_CMD, WRITE, READ_ADD, READ_DATA} state_e ;
endpackage

```

TEST

```

import Wrapper_pack::*;
module SPI (MOSI, MISO, SS_n, clk, rst_n, rx_valid, rx_data, tx_valid, tx_data);
    //input output decleration
    input MOSI, SS_n, clk, rst_n, tx_valid;
    input [7:0]tx_data;
    output reg MISO, rx_valid;
    output reg [9:0]rx_data;
    reg g;

```

```

//additional signals needed during operation
reg flag, counter_rst_n;
state_e cs, ns;
wire dout;
wire [9:0]din;
wire [3:0]counter_out;
//SIPO instan.
SIPO shift_reg(clk, rx_valid, MOSI, din);
//up_counter piso instan.
up_counter counter(clk, counter_rst_n, counter_out);
//PISO instan.
PISO shift_regII(clk, tx_valid, counter_out, tx_data, dout);
//state memory
always @(posedge clk or negedge rst_n) begin
    if (~rst_n)
        cs <= IDLE;
    else
        cs <= ns;
end
//next state logic
always @(cs, MOSI, SS_n) begin //bug : MOSI MUST NOT BE in the sensetivity list !!
    case(cs)
        IDLE: begin
            if(!rst_n)
                flag = 0 ;
            if(SS_n == 0) begin
                ns = CHK_CMD;
            end
            else
                ns = IDLE;
            end
        CHK_CMD:
            if(SS_n == 1)
                ns = IDLE;
            else if (SS_n == 0&&MOSI == 0)
                ns = WRITE;
            else if (SS_n == 0&&MOSI == 1&&flag == 0) begin
                ns = READ_ADD;
            end else begin
                ns = READ_DATA;
                flag = 0;
            end
        WRITE:
            if(SS_n == 1) begin
                ns = IDLE;
            end
    end
end

```



```

else
    ns = WRITE;
READ_ADD:
if(SS_n == 1) begin
    ns = IDLE;
    flag = 1;
end
else
    ns = READ_ADD;
default:
if(SS_n == 1)
    ns = IDLE;
else
    ns = READ_DATA;
endcase
end
//output logic
always @(posedge clk or negedge rst_n) begin
if(! rst_n)
begin
rx_valid <= 0;
rx_data <= 0;
MISO <= 0; // the mOSI MUST RESET TOO !
end
else
begin
if (ns == CHK_CMD) begin //THE CONDITIONING MUST BE ON THE ns not cs
counter_rst_n <= 1'b0;
end
else
counter_rst_n <= 1'b1;
if(cs == IDLE)
g = 0;
if (((cs == WRITE)|| (cs == READ_ADD))&&(! SS_n)) begin
if (counter_out == 4'd9 && (g == 0)) begin
rx_valid <= 1;
rx_data <= din;
g = 1;
end
end
else if ((cs == WRITE||cs == READ_ADD)&&SS_n == 1) begin
rx_valid <= 0;
end
else if (cs == READ_DATA&&SS_n == 0) begin
if (counter_out == 4'd9 && (g == 0)) begin
rx_valid <= 1;
rx_data <= din;
g = 1;
end
end
end

```

```

MISO <= dout;
end
else if (cs == READ_DATA&&SS_n == 1) begin
rx_valid <= 0;
end
end
//rx_valid must be 0 at the SS_N = 1;
end
endmodule

```

SEQUENCER

```

package sequencer ;
import Wrapper_seq_item ::* ;
import uvm_pkg ::* ;
`include "uvm_macros.svh"

class MySequencer extends uvm_sequencer #(Wrapper_seq_item) ;
    `uvm_component_utils(MySequencer)

    function new(string name = "MySequencer", uvm_component parent = null) ;
        super.new(name, parent) ;
    endfunction
endclass
endpackage

```

DO FILE

```

vlib work
vlog -f src_files.list -sv +cover
vsim -voptargs = +acc work.top -classdebug -uvmcontrol = all -coverage
run 0
add wave -position insertpoint \
sim:/top/DUT/spislave/clock \
sim:/top/DUT/spislave/counter_out \
sim:/top/DUT/spislave/counter_rst_n \
sim:/top/DUT/spislave/cs \
sim:/top/DUT/spislave/din \
sim:/top/DUT/spislave/dout \
sim:/top/DUT/spislave/flag \
sim:/top/DUT/spislave/g \
sim:/top/DUT/spislave/MISO \
sim:/top/DUT/spislave/MOSI \

```

```

sim:/top/DUT/spislave/ns \
sim:/top/DUT/spislave/rst_n \
sim:/top/DUT/spislave/rx_data \
sim:/top/DUT/spislave/rx_valid \
sim:/top/DUT/spislave/SS_n \
sim:/top/DUT/spislave/tx_data \
sim:/top/DUT/spislave/tx_valid
add wave -- position insertpoint \
sim:/top/DUT/mem/addr_rd \
sim:/top/DUT/mem/addr_wr
add wave -- position insertpoint \
sim:/top/Wrap_if/data_holder
coverage save Wrapper_DB.ucdb -- onexit -- du spi_wrapper
run -- all

#quit -- sim
#vcover report Wrapper_DB.ucdb -- all -- annotate -- details
-- output fc_cvr_wrp_rprt.txt

```

Part_2 >  src_files.list

```

1  wrap_p.sv
2  Wrapper_if.sv
3  counter.v
4  piso.v
5  sipo.v
6  SPI_under_test.sv
7  ram.v
8  spi_wrapper.v
9  Wrapper_obj_conf.sv
10 Wrapper_seq_item.sv
11 Wrapper_seq.sv
12 sequencer.sv
13 Wrapper_driver.sv
14 Wrapper_scoreboard.sv
15 Wrapper_monitor.sv
16 Wrapper_cov.sv
17 Wrapper_agent.sv
18 Wrapper_env.sv
19 Wrapper_test.sv
20 Wrapper_top.sv

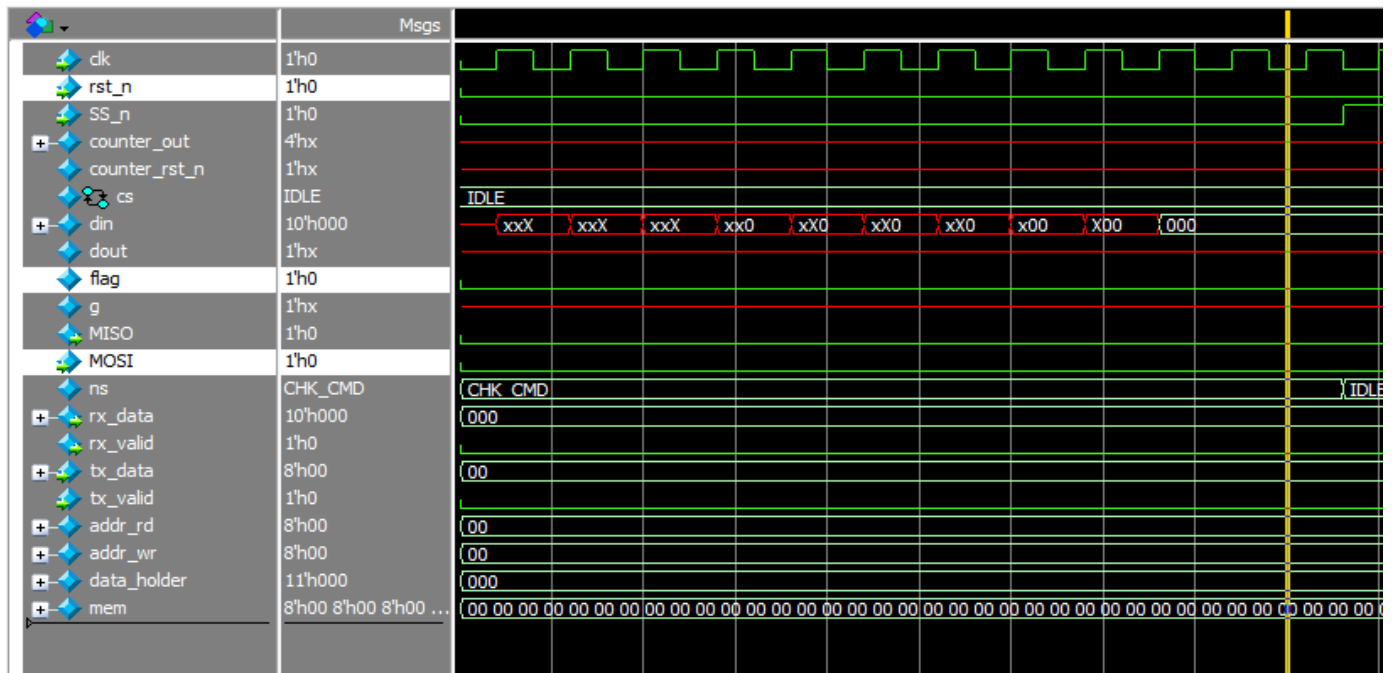
```

```

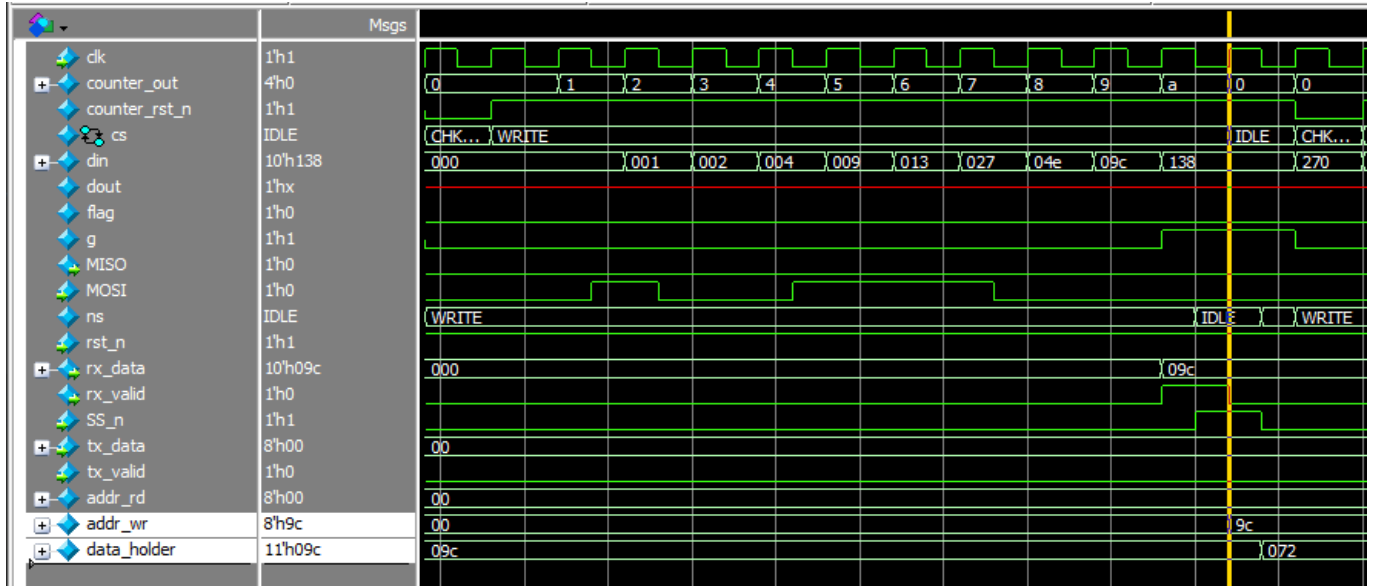
# UVM_INFO Wrapper_scoreboard.sv(101) @ 600212: uvm_test_top.my_env.sb [report_phase] Total successful transactions: 1670
# UVM_INFO Wrapper_scoreboard.sv(102) @ 600212: uvm_test_top.my_env.sb [report_phase] Total failed transactions: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 10
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# ** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [report_phase] 2
# [run_phase] 4
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 600212 ns Iteration: 61 Instance: /top

```

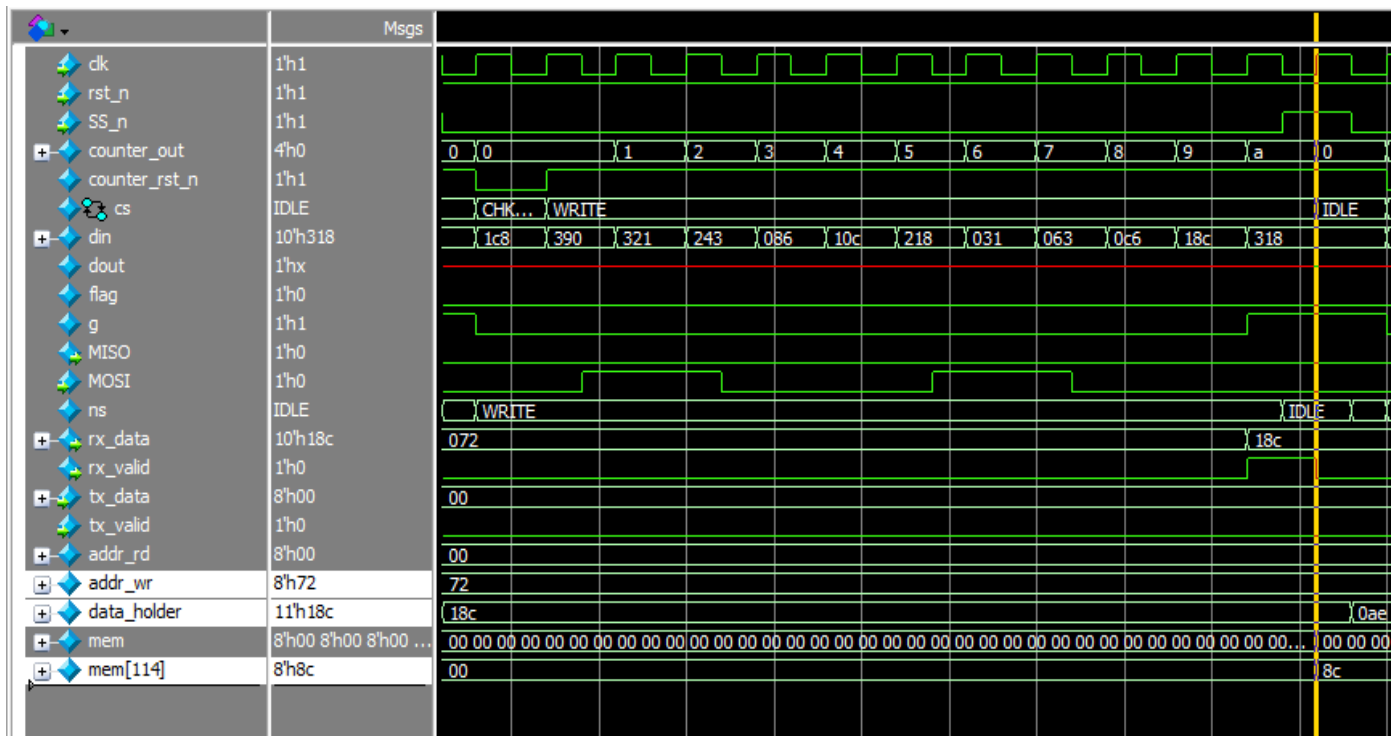
WAVEFORM



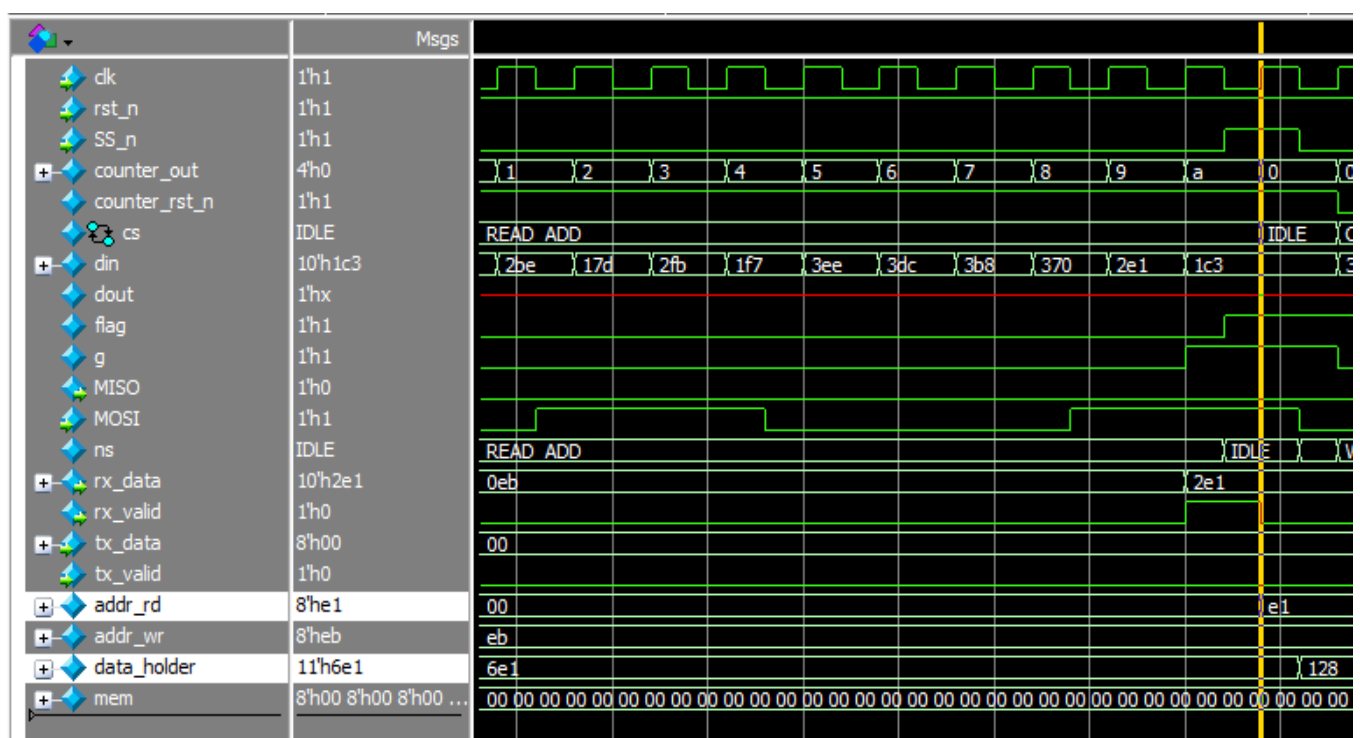
Waveform shows the Reset functionality.



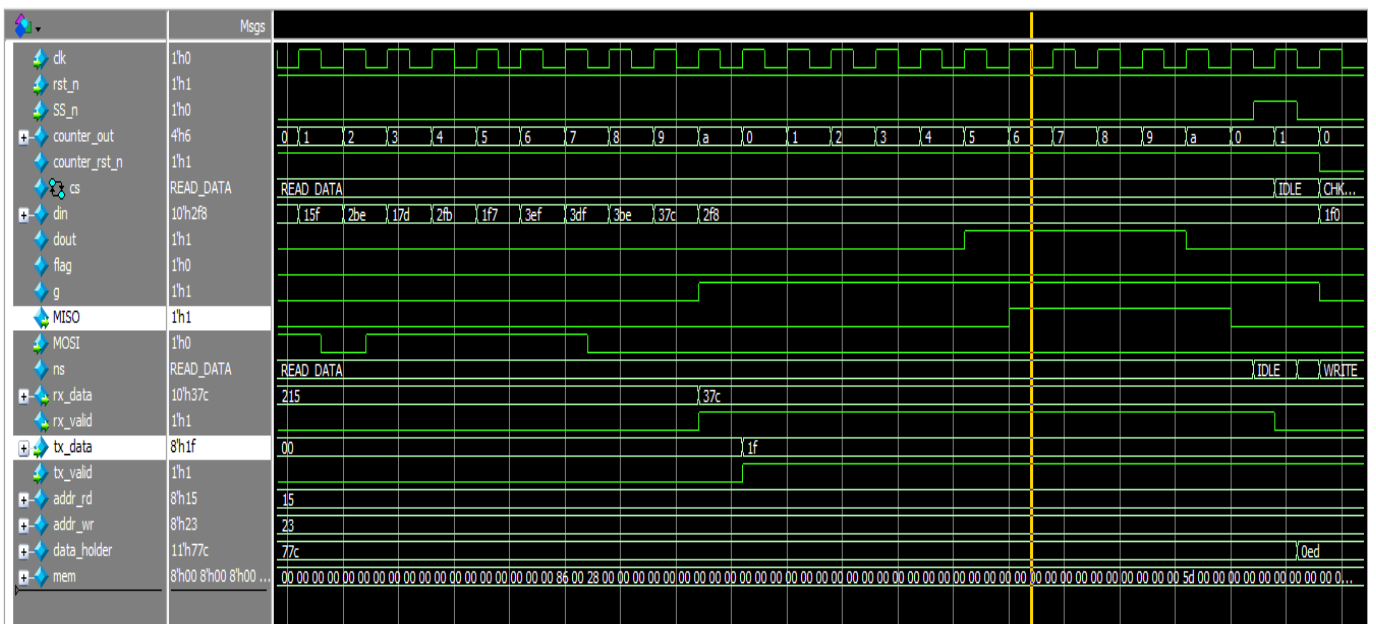
Waveform shows the write address operation.



Waveform shows the write data operation.



Waveform shows the read address operation.



Waveform shows the read data operation and the delays caused by the submodules (*dout* & MISO)

CODE COVERAGE

Part 2 > cvr_wrp_rprt.txt

```

1 Coverage Report by instance with details
2
3 =====
4 === Instance: /\top#DUT /spislave/shift_reg
5 === Design Unit: work.SIPO
6 =====
7 Branch Coverage:
8   Enabled Coverage      Bins      Hits      Misses      Coverage
9   -----
10  Branches              2        2          0      100.00%
11
12 =====Branch Details=====
13
14 Branch Coverage for instance /\top#DUT /spislave/shift_reg
15
16   Line      Item      Count      Source
17   ----      -
18   File sipo.v
19   -----IF Branch-----
20   8          129841      Count coming in to IF
21   8          119852      if(!SS_n)
22
23          9989      All False Count
24 Branch totals: 2 hits of 2 branches = 100.00%
25

```

Part_2 > cvr_wrp_rprt.txt

```

26
27 Statement Coverage:
28   Enabled Coverage          Bins    Hits    Misses  Coverage
29   -----
30   Statements                2      2      0    100.00%
31
32 =====Statement Details=====
33
34 Statement Coverage for instance /\top#DUT /spislave/shift_reg --
35
36   Line      Item              Count    Source
37   ----      -
38   File sipo.v
39   1
36          module SIPO (clk,SS_n ,MOSI, rx_data);
40
41   2
36          input  clk, MOSI,SS_n;
42
43   3
36          output [9:0] rx_data;
44
45   4
36          reg [9:0] tmp;
46
47   5
48
49   6          1                129841    always @(posedge clk)
50
51   7
36          begin
52
53   8
36          if(!SS_n)
54
55   9          1                119852    tmp = {tmp[8:0],MOSI};
56
57
58 Toggle Coverage:
59   Enabled Coverage          Bins    Hits    Misses  Coverage
60   -----
61   Toggles                46      46      0    100.00%
62
63 =====Toggle Details=====
64
65 Toggle Coverage for instance /\top#DUT /spislave/shift_reg --
66
67           Node      1H->0L    0L->1H              "Coverage"
68   -----
69           MOSI        1         1             100.00
70           SS_n        1         1             100.00
71           clk         1         1             100.00
72           rx_data[0-9] 1         1             100.00
73           tmp[9-0]     1         1             100.00
74
75 Total Node Count      =      23
76 Toggled Node Count   =      23
77 Untoggled Node Count =      0
78
79 Toggle Coverage      =    100.00% (46 of 46 bins)
80

```

Part_2 > cvr_wrp_rprt.txt

```

81 =====
82 === Instance: /\top#DUT /spislave/counter
83 === Design Unit: work.up_counter
84 =====
85 Branch Coverage:
86     Enabled Coverage          Bins      Hits      Misses  Coverage
87     -----
88     Branches                  2        2        0    100.00%
89
90 =====Branch Details=====
91
92 Branch Coverage for instance /\top#DUT /spislave/counter
93
94     Line      Item              Count      Source
95     ----      -
96     File counter.v
97     -----IF Branch-----
98     6              160042      Count coming in to IF
99     6              31648      if(~rst_n||counter_up==4'd10)
100
101     8              128394      else
102
103 Branch totals: 2 hits of 2 branches = 100.00%
104
105
106 Condition Coverage:
107     Enabled Coverage          Bins    Covered    Misses  Coverage
108     -----
109     Conditions                2        2        0    100.00%
110
111 =====Condition Details=====
112
113 Condition Coverage for instance /\top#DUT /spislave/counter --
114
115     File counter.v
116     -----Focused Condition View-----
117     Line      6 Item      1 (~rst_n || (counter_up == 10))
118 Condition totals: 2 of 2 input terms covered = 100.00%
119
120     Input Term    Covered    Reason for no coverage    Hint
121     -----
122     rst_n         Y
123     (counter_up == 10)    Y
124
125     Rows:        Hits    FEC Target          Non-masking condition(s)
126     -----
127     Row 1:        1    rst_n_0              -
128     Row 2:        1    rst_n_1              ~(counter_up == 10)
129     Row 3:        1    (counter_up == 10)_0  rst_n
130     Row 4:        1    (counter_up == 10)_1  rst_n
131

```


Part 2 > cvr_wrp_rprt.txt

```

133 Statement Coverage:
134   Enabled Coverage          Bins    Hits    Misses  Coverage
135   -----
136   Statements                3      3      0    100.00%
137
138   =====Statement Details=====
139
140 Statement Coverage for instance /\top#DUT /spislave/counter --
141
142   Line      Item              Count    Source
143   ----      -
144   File counter.v
145   1                                module up_counter(input clk, rst_n, output[3:0] counter);
146
147   2                                reg [3:0] counter_up;
148
149   3
150
151   4              1              160042    always @(posedge clk or negedge rst_n)
152
153   5                                begin
154
155   6                                if(~rst_n||counter_up==4'd10)
156
157   7              1              31648      counter_up <= 4'd0;
158
159   8                                else
160
161   9              1              128394     counter_up <= counter_up + 4'd1;
162
163
164 Toggle Coverage:
165   Enabled Coverage          Bins    Hits    Misses  Coverage
166   -----
167   Toggles                  20     20      0    100.00%
168
169   =====Toggle Details=====
170
171 Toggle Coverage for instance /\top#DUT /spislave/counter --
172
173                                Node      1H->0L      0L->1H      "Coverage"
174                                -----
175                                clk          1          1          100.00
176                                counter[0-3]  1          1          100.00
177                                counter_up[3-0] 1          1          100.00
178                                rst_n          1          1          100.00
179
180 Total Node Count      =      10
181 Toggled Node Count   =      10
182 Untoggled Node Count =      0
183
184 Toggle Coverage      =    100.00% (20 of 20 bins)

```

Part 2 > cvr_wrp_rprt.txt

```

186 =====
187 === Instance: /\top#DUT /spislave/shift_regII
188 === Design Unit: work.PISO
189 =====
190 Branch Coverage:
191   Enabled Coverage          Bins      Hits      Misses  Coverage
192   -----
193   Branches                  4        4        0    100.00%
194
195 =====Branch Details=====
196
197 Branch Coverage for instance /\top#DUT /spislave/shift_regII
198
199   Line      Item                      Count      Source
200   ----      -
201   File piso.v
202   -----IF Branch-----
203   11                      140697      Count coming in to IF
204   11              1          40606          if (tx_valid) begin
205
206                      100091      All False Count
207 Branch totals: 2 hits of 2 branches = 100.00%
208
209   -----IF Branch-----
210   12                      40606      Count coming in to IF
211   12              1          5643          if (counter==0)
212
213   14              1          34963          else begin
214
215 Branch totals: 2 hits of 2 branches = 100.00%
216
217
218 Condition Coverage:
219   Enabled Coverage          Bins   Covered   Misses  Coverage
220   -----
221   Conditions                  1       1        0    100.00%
222
223 =====Condition Details=====
224
225 Condition Coverage for instance /\top#DUT /spislave/shift_regII --
226
227   File piso.v
228   -----Focused Condition View-----
229   Line      12 Item      1 (counter == 0)
230 Condition totals: 1 of 1 input term covered = 100.00%
231
232   Input Term   Covered Reason for no coverage   Hint
233   -----
234   (counter == 0)      Y
235
236   Rows:         Hits  FEC Target          Non-masking condition(s)
237   -----
238   Row  1:         1 (counter == 0)_0      -
239   Row  2:         1 (counter == 0)_1      -
240

```

```

Part_2 > cvr_wrp_rprt.txt
242 Statement Coverage:
243   Enabled Coverage          Bins    Hits    Misses Coverage
244   -----
245   Statements                4      4      0    100.00%
246
247 =====Statement Details=====
248
249 Statement Coverage for instance /\top#DUT /spislave/shift_regII --
250
251   Line      Item                      Count    Source
252   ----      -
253   File piso.v
254   1                      module PISO(clk,tx_valid,counter,tx_data,dout);
255
256   2
257
258   3                      output reg dout;
259
260   4                      input [7:0] tx_data;
261
262   5                      input clk ;
263
264   6                      input tx_valid ;
265
266   7                      input [3:0] counter;
267
268   8                      reg [7:0]temp;
269
270   9
271
272   10             1          140697    always @ (posedge clk) begin
273
274   11                      if (tx_valid) begin
275
276   12                      if (counter==0)
277
278   13             1          5643          temp <= tx_data;
279
280   14                      else begin
281
282   15             1          34963          dout <= temp[7];
283
284   16             1          34963          temp <= {temp[6:0],1'b0};
285
286
287 Toggle Coverage:
288   Enabled Coverage          Bins    Hits    Misses Coverage
289   -----
290   Toggles                46      46      0    100.00%
291
292 =====Toggle Details=====
293
294 Toggle Coverage for instance /\top#DUT /spislave/shift_regII --
295
296                      Node      1H->0L      0L->1H                      "Coverage"
297                      -----
298                      clk          1          1                      100.00
299                      counter[0-3] 1          1                      100.00
300                      dout          1          1                      100.00
301                      temp[7-0]    1          1                      100.00
302                      tx_data[0-7] 1          1                      100.00
303                      tx_valid     1          1                      100.00
304
305 Total Node Count   =      23
306 Toggled Node Count =      23
307 Untoggled Node Count =      0
308
309 Toggle Coverage   =    100.00% (46 of 46 bins)

```

Part_2 > cvr_wrp_rprt.txt

```

312 === Instance: /\top#DUT /spislave
313 === Design Unit: work.SPI
314 =====
315 Branch Coverage:
316   Enabled Coverage      Bins      Hits      Misses  Coverage
317   -----
318   Branches              36       35       1     97.22%
319
320 =====Branch Details=====
321
322 Branch Coverage for instance /\top#DUT /spislave
323
324   Line      Item              Count      Source
325   ----      -
326   File SPI_under_test.sv
327   -----IF Branch-----
328   29              39992      Count coming in to IF
329   29              36        if (~rst_n)
330
331   31              39956      else
332
333 Branch totals: 2 hits of 2 branches = 100.00%
334
335   -----CASE Branch-----
336   36              95069      Count coming in to CASE
337   37              20048      IDLE:begin
338
339   46              9989       CHK_CMD:
340
341   58              43349      WRITE:
342
343   64              11670      READ_ADD:
344
345   72              10013      default:
346
347 Branch totals: 5 hits of 5 branches = 100.00%
348
349   -----IF Branch-----
350   38              20048      Count coming in to IF
351   38              70        if(!rst_n)
352
353              19978      All False Count
354 Branch totals: 2 hits of 2 branches = 100.00%
355
356   -----IF Branch-----
357   40              20048      Count coming in to IF
358   40              10047      if(SS_n==0) begin
359
360   43              10001      else
361
362 Branch totals: 2 hits of 2 branches = 100.00%

```

Part_2 > cvr_wrp_rprt.txt

```

920
921 Toggle Coverage:
922   Enabled Coverage      Bins    Hits    Misses Coverage
923   -----
924   Toggles              96      96      0    100.00%
925
926 =====Toggle Details=====
927
928 Toggle Coverage for instance /\top#DUT /spislave --
929
930           Node      1H->0L      0L->1H      "Coverage"
931   -----
932           MISO              1          1      100.00
933           MOSI              1          1      100.00
934           SS_n              1          1      100.00
935           clk               1          1      100.00
936           counter_out[0-3]  1          1      100.00
937           counter_rst_n    1          1      100.00
938           cs
939           ENUM type      Value      Count
940           IDLE           16      100.00
941           CHK_CMD        17      100.00
942           WRITE          15      100.00
943           READ_ADD       1       100.00
944           READ_DATA      1       100.00
945           din[0-9]        1          1      100.00
946           dout            1          1      100.00
947           flag            1          1      100.00
948           g               1          1      100.00
949           ns
950           ENUM type      Value      Count
951           IDLE           17      100.00
952           CHK_CMD        18      100.00
953           WRITE          15      100.00
954           READ_ADD       1       100.00
955           READ_DATA      1       100.00
956           rst_n           1          1      100.00
957           rx_data[9-0]    1          1      100.00
958           rx_valid        1          1      100.00
959           tx_data[0-7]    1          1      100.00
960           tx_valid        1          1      100.00
961
962 Total Node Count      =      53
963 Toggled Node Count   =      53
964 Untoggled Node Count =      0
965
966 Toggle Coverage      =    100.00% (96 of 96 bins)
967
968 =====
969 === Instance: /\top#DUT /mem
970 === Design Unit: work.project_ram
971 =====
972 Branch Coverage:
973   Enabled Coverage      Bins    Hits    Misses Coverage
974   -----
975   Branches              7      7      0    100.00%
976
977 =====Branch Details=====
978
979 Branch Coverage for instance /\top#DUT /mem
980
981   Line      Item      Count      Source
982   ----      -
983   File ram.v
984   -----IF Branch-----
985   14              21669      Count coming in to IF
986   14              24        if (~rst_n) begin
987   25              11656      else if (rx_valid) begin
988   9989           9989      All False Count
989
990 Branch totals: 3 hits of 3 branches = 100.00%

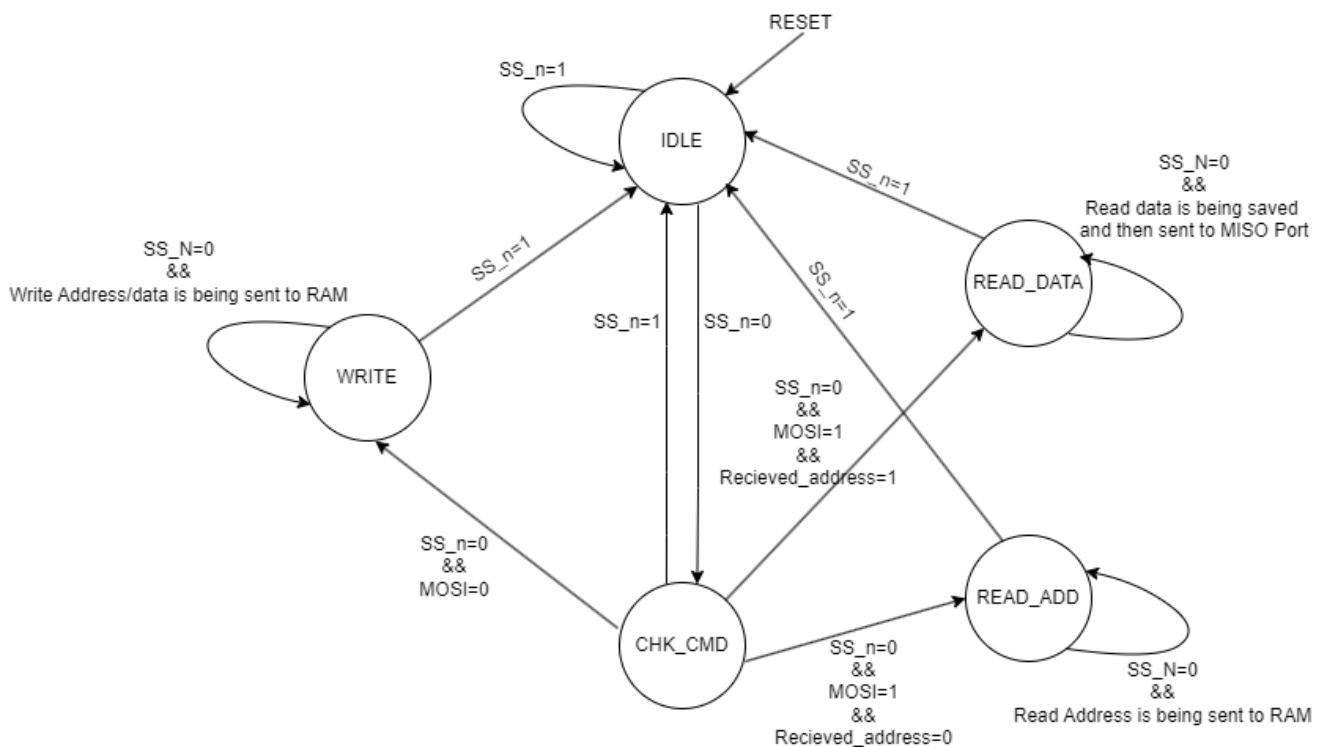
```

Part 2 > cvr_wrp_rprt.txt

```

1052
1053
1054 Statement Coverage:
1055     Enabled Coverage          Bins    Hits    Misses  Coverage
1056     -----
1057     Statements                19      19      0    100.00%
1058
1059 =====Statement Details=====
1060
1061 Statement Coverage for instance /\top#DUT /mem --
1062
1063     Line      Item          Count    Source
1064     ----      -
1065     File ram.v
1066     1
1067     module project_ram(din, rx_valid, dout, tx_valid, clk, rst_n);
1068     2
1069     parameter MEM_DEPTH = 256;
1070     3
1071     parameter ADDR_SIZE = 8;
1072     4
1073     input rx_valid, clk, rst_n;
1074     5
1075     input [9:0] din;
1076     6
1077     output reg tx_valid;
1078     7
1079     output reg [7:0] dout;
1080     8
1081     reg [ADDR_SIZE-1:0] addr_rd, addr_wr;
1082     9
1083     reg [7:0] mem [MEM_DEPTH-1:0];
1084     10
1085     reg [8:0] i ;

```



Part_2 > cvr_wrp_rprt.txt

```

1154 Toggle Coverage:
1155   Enabled Coverage      Bins    Hits    Misses  Coverage
1156   -----
1157   Toggles                94      94        0   100.00%
1158
1159   =====Toggle Details=====
1160
1161 Toggle Coverage for instance /\top#DUT /mem --
1162
1163           Node          1H->0L    0L->1H          "Coverage"
1164   -----
1165           addr_rd[7-0]          1          1          100.00
1166           addr_wr[7-0]          1          1          100.00
1167           clk                    1          1          100.00
1168           din[0-9]              1          1          100.00
1169           dout[7-0]             1          1          100.00
1170           i[8-0]                1          1          100.00
1171           rst_n                  1          1          100.00
1172           rx_valid               1          1          100.00
1173           tx_valid               1          1          100.00
1174
1175 Total Node Count    =          47
1176 Toggled Node Count =          47
1177 Untoggled Node Count =          0
1178
1179 Toggle Coverage    =    100.00% (94 of 94 bins)
1180
1181   =====
1182   === Instance: /\top#DUT
1183   === Design Unit: work.spi_wrapper
1184   =====
1185 Toggle Coverage:
1186   Enabled Coverage      Bins    Hits    Misses  Coverage
1187   -----
1188   Toggles                50      50        0   100.00%
1189
1190   =====Toggle Details=====
1191
1192 Toggle Coverage for instance /\top#DUT --
1193
1194           Node          1H->0L    0L->1H          "Coverage"
1195   -----
1196           clk                    1          1          100.00
1197           miso                    1          1          100.00
1198           mosi                    1          1          100.00
1199           rst_n                    1          1          100.00
1200           rx_data[0-9]            1          1          100.00
1201           rx_valid                1          1          100.00
1202           ss_n                    1          1          100.00
1203           tx_data[0-7]            1          1          100.00
1204           tx_valid                1          1          100.00
1205
1206 Total Node Count    =          25
1207 Toggled Node Count =          25
1208 Untoggled Node Count =          0
1209
1210 Toggle Coverage    =    100.00% (50 of 50 bins)

```

FUNCTIONAL COVERAGE

Part 2 > fc_cvr_wrp_rprt.txt

3361	COVERGROUP COVERAGE:				
3362	-----				
3363	Covergroup	Metric	Goal	Bins	Status
3364	-----				
3365					
3366	TYPE /Wrapper_coverage/coverage/Wrapper_cg	100.00%	100	-	Covered
3367	covered/total bins:	14	14	-	
3368	missing/total bins:	0	14	-	
3369	% Hit:	100.00%	100	-	
3370	Coverpoint reset_cp	100.00%	100	-	Covered
3371	covered/total bins:	2	2	-	
3372	missing/total bins:	0	2	-	
3373	% Hit:	100.00%	100	-	
3374	bin auto[0]	192	1	-	Covered
3375	bin auto[1]	149861	1	-	Covered
3376	Coverpoint SS_n_cp	100.00%	100	-	Covered
3377	covered/total bins:	2	2	-	
3378	missing/total bins:	0	2	-	
3379	% Hit:	100.00%	100	-	
3380	bin auto[0]	140052	1	-	Covered
3381	bin auto[1]	10001	1	-	Covered
3382	Coverpoint MOSI_cp	100.00%	100	-	Covered
3383	covered/total bins:	2	2	-	
3384	missing/total bins:	0	2	-	
3385	% Hit:	100.00%	100	-	
3386	bin auto[0]	78503	1	-	Covered
3387	bin auto[1]	71550	1	-	Covered
3388	Coverpoint MISO_cp	100.00%	100	-	Covered
3389	covered/total bins:	2	2	-	
3390	missing/total bins:	0	2	-	
3391	% Hit:	100.00%	100	-	
3392	bin auto[0]	147492	1	-	Covered
3393	bin auto[1]	2548	1	-	Covered
3394	Coverpoint operation_cp	100.00%	100	-	Covered
3395	covered/total bins:	4	4	-	
3396	missing/total bins:	0	4	-	
3397	% Hit:	100.00%	100	-	
3398	bin WRITE_add	43758	1	-	Covered
3399	bin WRITE_data	42783	1	-	Covered
3400	bin READ_add	21762	1	-	Covered
3401	bin READ_data	41750	1	-	Covered
3402	illegal_bin Invalid_op	0		-	ZERO
3403	Cross READ_op	100.00%	100	-	Covered
3404	covered/total bins:	2	2	-	
3405	missing/total bins:	0	2	-	
3406	% Hit:	100.00%	100	-	
3407	Auto, Default and User Defined Bins:				
3408	bin <READ_data,auto[1]>	2548	1	-	Covered
3409	bin <READ_data,auto[0]>	39189	1	-	Covered
3410	Illegal and Ignore Bins:				
3411	ignore_bin not_read2	21762		-	Occurred
3412	ignore_bin not_read1	42783		-	Occurred
3413	ignore_bin not_read0	43758		-	Occurred
3414					
3415	TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1				
3416					
3417	ASSERTION RESULTS:				
3418	-----				
3419	Name	File(Line)	Failure	Pass	
3420			Count	Count	
3421	-----				
3422	/sequences/Wrapper_main_seq/body/#ublk#146550787#35/immed__38				
3423		Wrapper_seq.sv(38)	0	1	

SPI SLAVE WITH SINGLE PORT RAM

RAM + Wrapper Environment

LINKAGE WRAPPER TEST

```

package Wrapper_test_pkg;
import ram_env_pkg::*;
import Wrapper_env::*;
import uvm_pkg::*;
import Wrapper_config_obj::*;
import ram_config_pkg::*;
import sequences::*;
import sequence_pkg::*;
`include "uvm_macros.svh"
class Wrapper_test extends uvm_test ;
    `uvm_component_utils (Wrapper_test) ;
    Wrapper_env my_env ;
    ram_env R_env ;
    virtual Wrapper_if Wrapper_test_vif ;
    virtual ram_if ramif;
    Wrapper_config_obj Wrapper_config_obj_test ;
    ram_config ram_cfg;
    Wrapper_main_seq main_seq ;
    Wrapper_reset_seq reset_seq_w;
    ram_read_and_write_sequence read_and_write_seq;
    ram_reset_sequence reset_seq;
    ram_read_sequence read_seq;
    ram_write_sequence write_seq;
    function new(string name = "Wrapper_test",uvm_component parent = null );
        super.new(name,parent);
    endfunction
    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        Wrapper_config_obj_test
= Wrapper_config_obj::type_id::create("Wrapper_config_obj_test",this) ;
        my_env = Wrapper_env::type_id::create("my_env",this) ;
        main_seq = Wrapper_main_seq::type_id::create("main_seq",this) ;
        reset_seq_w = Wrapper_reset_seq::type_id::create("reset_seq_w",this) ;

```

```

R_env = ram_env::type_id::create("env",this);
ram_cfg = ram_config::type_id::create("ram_cfg",this);
read_and_write_seq
= ram_read_and_write_sequence::type_id::create("read_and_write_seq",this);
read_seq = ram_read_sequence::type_id::create("read_seq",this);
write_seq = ram_write_sequence::type_id::create("write_seq",this);
reset_seq = ram_reset_sequence::type_id::create("reset_seq",this);
if(!uvm_config_db#(virtual Wrapper_if)
::get(this,"","Wrapper_IF",Wrapper_config_obj_test.Wrapper_config_vif))
`uvm_fatal("build_phase","ERROR in getting virtual interface");
uvm_config_db#(Wrapper_config_obj)::set(this,"
    *","CFG_Wrapper",Wrapper_config_obj_test);
Wrapper_config_obj_test.active = UVM_ACTIVE;
if(!uvm_config_db#(virtual ram_if)::get(this,"","ram_IF",ram_cfg.ramif))
`uvm_fatal("build_phase","test
    – unable to get the virtual interface of alu from uvm_config_db");
uvm_config_db#(ram_config)::set(this,"*","CFG",ram_cfg);
ram_cfg.active = UVM_PASSIVE;
endfunction
task run_phase (uvm_phase phase);
    super.run_phase(phase);
    phase.raise_objection(this);
    `uvm_info ("run_phase","RESET_ASSERTED ",UVM_LOW);
    reset_seq.w.start(my_env.agt.sqr);
    `uvm_info ("run_phase","RESET_DEASSERTED ",UVM_LOW);
    `uvm_info("run_phase","Started generating stimulus",UVM_LOW);
    main_seq.start(my_env.agt.sqr);
    `uvm_info ("run_phase","Stimulus generation ended ",UVM_LOW);
    phase.drop_objection(this);
endtask
endclass
endpackage

```

LINKAGE WRAPPER TOP

```

import Wrapper_test_pkg::*;
import ram_test_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
module top();
    bit clk;
    initial begin
        forever #2 clk = ~clk;
    end
endmodule

```

```

end
Wrapper_if Wrap_if(clk);
ram_if ramif(clk);
spi_wrapper DUT(Wrap_if.MOSI , Wrap_if.MISO, Wrap_if.SS_n, Wrap_if.clk, Wrap_if.rst_n);
initial begin
    uvm_config_db#(virtual Wrapper_if)::set(null, "uvm_test_top", "Wrapper_IF", Wrap_if);
    uvm_config_db#(virtual ram_if)::set(null, "uvm_test_top", "ram_IF", ramif);
    run_test("Wrapper_test");
end
assign ramif.rst_n = Wrap_if.rst_n;
assign ramif.rx_valid = DUT.rx_valid;
assign ramif.din = DUT.rx_data;
assign ramif.dout = DUT.tx_data;
assign ramif.tx_valid = DUT.tx_valid;
endmodule

```

LINKAGE RAM AGENT

```

package agent_pkg;
`include "uvm_macros.svh"
import uvm_pkg::*;
import sequencer_pkg::*;
import ram_config_pkg::*;
import monitor_pkg::*;
import sequence_item_pkg::*;
import ram_driver_pkg::*;
class ram_agent extends uvm_agent;
`uvm_component_utils(ram_agent)
    ram_driver driver;
    ram_monitor mon;
    ram_sequencer sqr;
    ram_config ram_cfg;
    uvm_analysis_port #(ram_seq_item) agt_ap;
    function new(string name = "ram_agent", uvm_component parent = null);
        super.new(name, parent);
        agt_ap = new("agt_ap", this);
    endfunction
    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        if(! uvm_config_db#(ram_config)::get(this, "", "CFG", ram_cfg))
            `uvm_fatal("build_phase", "unable to get configuration object")
        if(ram_cfg.active == UVM_ACTIVE) begin
            driver = ram_driver::type_id::create("driver", this);
            sqr = ram_sequencer::type_id::create("sqr", this);

```

```

end
mon = ram_monitor::type_id::create("mon", this);
endfunction
function void connect_phase(uvm_phase phase);
  if(ram_cfg.active == UVM_ACTIVE) begin
    driver.ramif = ram_cfg.ramif;
    driver.seq_item_port.connect(sqr.seq_item_export);
  end
  mon.ramif = ram_cfg.ramif;
  mon.mon_ap.connect(agt_ap);
endfunction
endclass
endpackage

```

LINKAGE WRAPPER AGENT

```

package Wrapper_agent;
import Wrapper_config_obj::*;
import Wrapper_seq_item::*;
import uvm_pkg::*;
import monitor::*;
import Wrapper_driver::*;
import sequencer::*;
`include "uvm_macros.svh"
class Wrapper_agent extends uvm_agent;
`uvm_component_utils(Wrapper_agent)
  MyMonitor mon;
  MySequencer sqr;
  Wrapper_driver drv;
  Wrapper_config_obj Wrapper_cfg;
  uvm_analysis_port #(Wrapper_seq_item) agt_ap;

  function new (string name = "Wrapper_agent", uvm_component parent = null);
    super.new(name, parent);
  endfunction
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db #(Wrapper_config_obj)::get(this, "", "CFG_Wrapper", Wrapper_cfg))
      `uvm_fatal("build_phase", "Error in reading configurable object")
    if(Wrapper_cfg.active == UVM_ACTIVE) begin
      sqr = MySequencer::type_id::create("sqr", this);
      drv = Wrapper_driver::type_id::create("drv", this);
    end
  end

```

```

mon = MyMonitor :: type_id :: create ("mon", this) ;
agt_ap = new("agt_ap", this);
endfunction : build_phase
function void connect_phase(uvm_phase phase) ;
if(Wrapper_cfg.active == UVM_ACTIVE) begin
drv.Wrapper_vif = Wrapper_cfg.Wrapper_config_vif ;
drv.seq_item_port.connect(sqr.seq_item_export) ;
end
mon.Wrapper_vif = Wrapper_cfg.Wrapper_config_vif ;
mon.mon_ap.connect(agt_ap) ;
endfunction
endclass
endpackage : Wrapper_agent

```

DO FILE

```

vlib work
vlog -f src_files.list -sv +cover
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -coverage
run 0
add wave -position insertpoint sim:/top/DUT/*
run -all

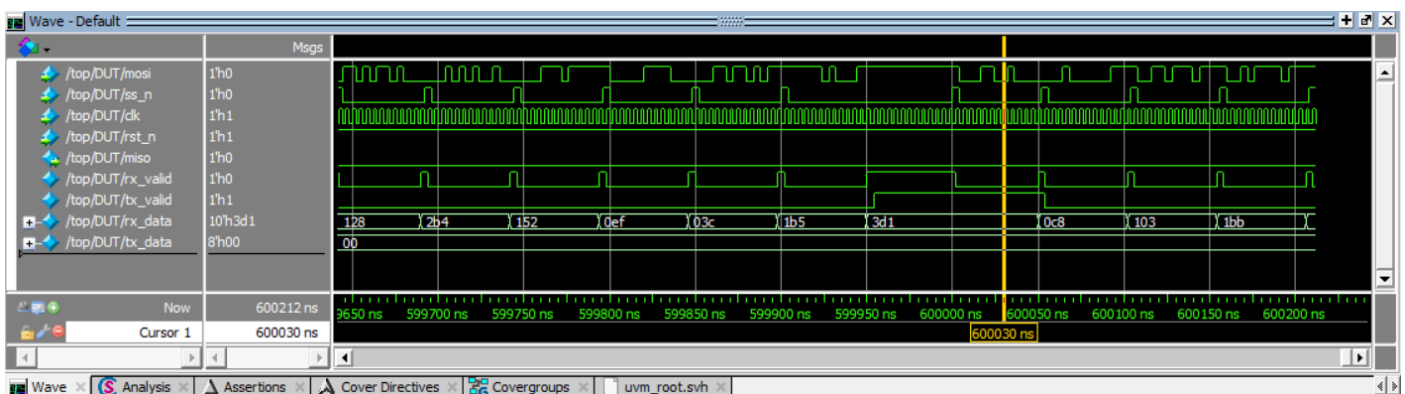
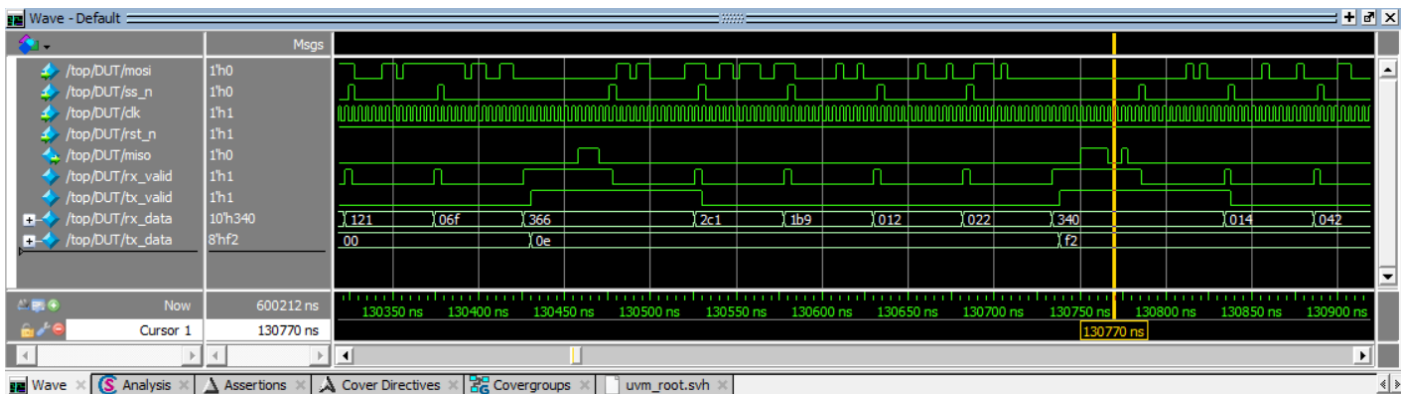
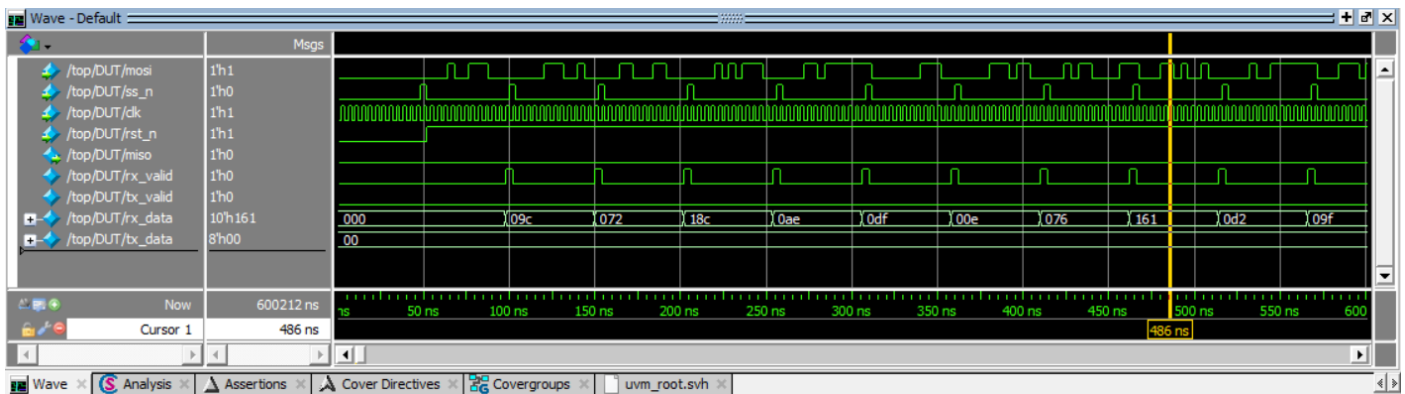
```

```

# UVM_INFO scoreboard.sv(84) @ 600212: uvm_test_top.env.sb [report phase] total successful transactions is 150053
# UVM_INFO scoreboard.sv(86) @ 600212: uvm_test_top.env.sb [report phase] total wrong transactions is 0
# UVM_INFO Wrapper_scoreboard.sv(101) @ 600212: uvm_test_top.my_env.sb [report_phase] Total successful transactions: 1670
# UVM_INFO Wrapper_scoreboard.sv(102) @ 600212: uvm_test_top.my_env.sb [report_phase] Total failed transactions: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 12
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# ** Report counts by id
# [Questa UVM] 2
# [RNIST] 1
# [TEST_DONE] 1
# [report_phase] 2
# [report_phase] 2
# [run_phase] 4
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 600212 ns Iteration: 61 Instance: /top

```

WAVEFORM



Assertions			
Name	Assertion Type	Language	Enable
▲ /uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on
▲ /uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on
▲ /sequence_pkg::ram_write_sequence::body/#ublk#81915607#28/immed__34	Immediate	SVA	on
▲ /sequence_pkg::ram_read_sequence::body/#ublk#81915607#46/immed__52	Immediate	SVA	on
▲ /sequence_pkg::ram_read_and_write_sequence::body/#ublk#81915607#64/immed__70	Immediate	SVA	on
▲ /sequences::Wrapper_main_seq::body/#ublk#146550787#35/immed__38	Immediate	SVA	on

Covergroups							
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances
[-] /Wrapper_coverag...		100.00%					
[+] TYPE Wrapper...		100.00%	100	100.00...	<div style="width: 100%; height: 10px; background-color: green;"></div>		auto(1)
[+] /coverage_collecto...		100.00%					

CODE & ASSERTION COVERAGE

```

fc_cvr_wrp_rprt.txt
59
60 =====
61 === Instance: /top/DUT/spislave/shift_reg
62 === Design Unit: work.SIPO0
63 =====
64 Branch Coverage:
65   Enabled Coverage      Bins    Hits    Misses  Coverage
66   -----
67   Branches              2      2      0    100.00%
68
69 =====Branch Details=====
70
71 Branch Coverage for instance /top/DUT/spislave/shift_reg
72
73   Line      Item              Count    Source
74   ----      -
75   File sipo.v
76   -----IF Branch-----
77   8              129841    Count coming in to IF
78   8              119852    if(!SS_n)
79
80              9989    All False Count
81 Branch totals: 2 hits of 2 branches = 100.00%
82
83
84 Statement Coverage:
85   Enabled Coverage      Bins    Hits    Misses  Coverage
86   -----
87   Statements            2      2      0    100.00%
88
89 =====Statement Details=====
90
91 Statement Coverage for instance /top/DUT/spislave/shift_reg --
92
93   Line      Item              Count    Source
94   ----      -
95   File sipo.v
96   1              module SIPO (clk,SS_n ,MOSI, rx_data);
97
98   2              input  clk, MOSI,SS_n;
99
100  3              output [9:0] rx_data;
101
102  4              reg [9:0] tmp;
103
104  5
105  6              1          129841    always @(posedge clk)
106
107  7              begin
108
109  8              if(!SS_n)
110
111  9              1          119852    tmp = {tmp[8:0],MOSI};
112
113
114
115 Toggle Coverage:
116   Enabled Coverage      Bins    Hits    Misses  Coverage
117   -----
118   Toggles              46      46      0    100.00%
119
120 =====Toggle Details=====
121

```



```

fc_cvr_wrp_rprt.txt
139 === Instance: /top/DUT/spislave/counter
140 === Design Unit: work.up_counter
141 =====
142 Branch Coverage:
143   Enabled Coverage          Bins      Hits      Misses  Coverage
144   -----
145   Branches                  2        2        0    100.00%
146
147 =====Branch Details=====
148
149 Branch Coverage for instance /top/DUT/spislave/counter
150
151   Line      Item                      Count      Source
152   ----      -
153   File counter.v
154   -----IF Branch-----
155   6                      160042    Count coming in to IF
156   6              1          31648    if(~rst_n || counter_up==4'd10)
157
158   8              1          128394    else
159
160 Branch totals: 2 hits of 2 branches = 100.00%
161
162
163 Condition Coverage:
164   Enabled Coverage          Bins  Covered  Misses  Coverage
165   -----
166   Conditions                2      2        0    100.00%
167
168 =====Condition Details=====
169
170 Condition Coverage for instance /top/DUT/spislave/counter --
171
172   File counter.v
173   -----Focused Condition View-----
174   Line      6 Item      1 (~rst_n || (counter_up == 10))
175   Condition totals: 2 of 2 input terms covered = 100.00%
176
177   Input Term  Covered  Reason for no coverage  Hint
178   -----
179   rst_n      Y
180   (counter_up == 10)  Y
181
182   Rows:      Hits  FEC Target          Non-masking condition(s)
183   -----
184   Row  1:      1  rst_n_0          -
185   Row  2:      1  rst_n_1          ~(counter_up == 10)
186   Row  3:      1  (counter_up == 10)_0  rst_n
187   Row  4:      1  (counter_up == 10)_1  rst_n
188
189
190 Statement Coverage:
191   Enabled Coverage          Bins      Hits      Misses  Coverage
192   -----
193   Statements                3        3        0    100.00%
194
195 =====Statement Details=====
196
197 Statement Coverage for instance /top/DUT/spislave/counter --
198
199   Line      Item                      Count      Source
200   ----      -

```


fc_cvr_wrp_rprt.txt

```

219
220
221 Toggle Coverage:
222   Enabled Coverage      Bins    Hits    Misses  Coverage
223   -----
224   Toggles                20      20        0  100.00%
225
226 =====Toggle Details=====
227
228 Toggle Coverage for instance /top/DUT/spislave/counter --
229
230                               Node      1H->0L    0L->1H                "Coverage"
231                               -----
232                               clk          1         1             100.00
233                               counter[0-3]  1         1             100.00
234                               counter_up[3-0] 1         1             100.00
235                               rst_n          1         1             100.00
236
237 Total Node Count      =      10
238 Toggled Node Count   =      10
239 Untoggled Node Count =       0
240
241 Toggle Coverage      =    100.00% (20 of 20 bins)
242
243 =====
244 === Instance: /top/DUT/spislave/shift_regII
245 === Design Unit: work.PISO
246 =====
247 Branch Coverage:
248   Enabled Coverage      Bins    Hits    Misses  Coverage
249   -----
250   Branches                4      4        0  100.00%
251
252 =====Branch Details=====
253
254 Branch Coverage for instance /top/DUT/spislave/shift_regII
255
256   Line      Item                Count    Source
257   ----      -
258   File piso.v
259   -----IF Branch-----
260   11                140697    Count coming in to IF
261   11                40606      if (tx_valid) begin
262
263                               100091    All False Count
264 Branch totals: 2 hits of 2 branches = 100.00%
265
266 -----IF Branch-----
267   12                40606      Count coming in to IF
268   12                5643       if (counter==0)
269
270   14                34963      else begin
271
272 Branch totals: 2 hits of 2 branches = 100.00%
273
274
275 Condition Coverage:
276   Enabled Coverage      Bins    Covered    Misses  Coverage
277   -----
278   Conditions            1         1         0  100.00%

```

fc_cvr_wrp_rprt.txt

```

299 Statement Coverage:
300   Enabled Coverage          Bins    Hits    Misses Coverage
301   -----
302   Statements                4      4      0    100.00%
303
304   =====Statement Details=====
305
306   Statement Coverage for instance /top/DUT/spislave/shift_regII --
307
308   Line      Item              Count    Source
309   ----      -
310   File piso.v
311   1                                module PISO(clk,tx_valid,counter,tx_data,dout);
312
313   2
314
315   3                                output reg dout;
316
317   4                                input [7:0] tx_data;
318
319   5                                input clk ;
320
321   6                                input tx_valid ;
322
323   7                                input [3:0] counter;
324
325   8                                reg [7:0]temp;
326
327   9
328
329   10           1                140697    always @ (posedge clk) begin
330
331   11                                if (tx_valid) begin
332
333   12                                if (counter==0)
334
335   13           1                5643        temp <= tx_data;
336
337   14                                else begin
338
339   15           1                34963        dout <= temp[7];
340
341   16           1                34963        temp <= {temp[6:0],1'b0};
342
343
344 Toggle Coverage:
345   Enabled Coverage          Bins    Hits    Misses Coverage
346   -----
347   Toggles                46      46      0    100.00%
348
349   =====Toggle Details=====
350
351   Toggle Coverage for instance /top/DUT/spislave/shift_regII --
352
353                                Node      1H->0L      0L->1H      "Coverage"
354                                -----
355                                clk          1          1          100.00
356                                counter[0-3] 1          1          100.00
357                                dout          1          1          100.00
358                                temp[7-0]    1          1          100.00
359                                tx_data[0-7] 1          1          100.00
360                                tx_valid      1          1          100.00
361
362 Total Node Count      =      23
363 Toggled Node Count   =      23
364 Untoggled Node Count =      0
365
366 Toggle Coverage      =    100.00% (46 of 46 bins)
367

```

```

fc_cvr_wrp_rprt.txt
=====
369 === Instance: /top/DUT/spislave
370 === Design Unit: work.SPI
371 =====
372 Branch Coverage:
373   Enabled Coverage      Bins      Hits      Misses  Coverage
374   -----
375   Branches              36       35        1    97.22%
376
377 =====Branch Details=====
378
379 Branch Coverage for instance /top/DUT/spislave
380
381   Line      Item              Count      Source
382   ----      -
383   File SPI_under_test.sv
384   -----IF Branch-----
385   29              39992      Count coming in to IF
386   29              36         if (~rst_n)
387
388   31              39956      else
389
390 Branch totals: 2 hits of 2 branches = 100.00%
391
392   -----CASE Branch-----
393   36              95069      Count coming in to CASE
394   37              20048      IDLE:begin
395
396   46              9989       CHK_CMD:
397
398   58              43349      WRITE:
399
400   64              11670      READ_ADD:
401
402   72              10013      default:
403
404 Branch totals: 5 hits of 5 branches = 100.00%
405
406   -----IF Branch-----
407   38              20048      Count coming in to IF
408   38              70         if(!rst_n)
409
410              19978      All False Count
411 Branch totals: 2 hits of 2 branches = 100.00%
412
413   -----IF Branch-----
414   40              20048      Count coming in to IF
415   40              10047      if(SS_n==0) begin
416
417   43              10001      else
418
419 Branch totals: 2 hits of 2 branches = 100.00%

```

```

fc_cvr_wrp_rprt.txt
978 Toggle Coverage:
979   Enabled Coverage      Bins    Hits    Misses Coverage
980   -----
981   Toggles              96      96      0    100.00%
982
983 =====Toggle Details=====
984
985 Toggle Coverage for instance /top/DUT/spislave --
986
987           Node      1H->0L      0L->1H      "Coverage"
988   -----
989           MISO        1          1          100.00
990           MOSI        1          1          100.00
991           SS_n        1          1          100.00
992           clk         1          1          100.00
993           counter_out[0-3] 1          1          100.00
994           counter_rst_n 1          1          100.00
995           cs          ENUM type  Value    Count
996                   IDLE        16      100.00
997                   CHK_CMD      17      100.00
998                   WRITE        15      100.00
999                   READ_ADD     1       100.00
1000                   READ_DATA    1       100.00
1001           din[0-9]     1          1          100.00
1002           dout         1          1          100.00
1003           flag         1          1          100.00
1004           g            1          1          100.00
1005           ns          ENUM type  Value    Count
1006                   IDLE        17      100.00
1007                   CHK_CMD      18      100.00
1008                   WRITE        15      100.00
1009                   READ_ADD     1       100.00
1010                   READ_DATA    1       100.00
1011           rst_n        1          1          100.00
1012           rx_data[9-0] 1          1          100.00
1013           rx_valid     1          1          100.00
1014           tx_data[0-7] 1          1          100.00
1015           tx_valid     1          1          100.00
1016
1017 Total Node Count      =      53
1018 Toggled Node Count    =      53
1019 Untoggled Node Count  =       0
1020
1021 Toggle Coverage      =    100.00% (96 of 96 bins)
1022
1023 =====
1024 === Instance: /top/DUT/mem
1025 === Design Unit: work.project_ram
1026 =====
1027 Branch Coverage:
1028   Enabled Coverage      Bins    Hits    Misses Coverage
1029   -----
1030   Branches              7      7      0    100.00%
1031
1032 =====Branch Details=====
1033
1034 Branch Coverage for instance /top/DUT/mem
1035
1036   Line      Item      Count    Source
1037   ----      -
1038   File ram.v
1039   -----IF Branch-----
1040   14          21669    Count coming in to IF
1041   14          24      if (~rst_n) begin
1042
1043   25          11656    else if (rx_valid) begin
1044
1045          9989      All False Count
1046 Branch totals: 3 hits of 3 branches = 100.00%
1047
1048 -----IF Branch-----
1049   27          11656    Count coming in to IF
1050   27          3365      if (din[9:8] == 2'b00) begin
1051

```

FUNCTIONAL COVERAGE

fc_cvr_wrp_rprt.txt

6045	COVERGROUP COVERAGE:				
6047	-----				
6048	Covergroup	Metric	Goal	Bins	Status
6049					
6050	-----				
6051	TYPE /coverage_collector/ram_coverage/g	100.00%	100	-	Covered
6052	covered/total bins:	409	409	-	
6053	missing/total bins:	0	409	-	
6054	% Hit:	100.00%	100	-	
6055	Coverpoint din_cp	100.00%	100	-	Covered
6056	covered/total bins:	64	64	-	
6057	missing/total bins:	0	64	-	
6058	% Hit:	100.00%	100	-	
6059	bin auto[0:15]	2950	1	-	Covered
6060	bin auto[16:31]	2691	1	-	Covered
6061	bin auto[32:47]	2643	1	-	Covered
6062	bin auto[48:63]	2407	1	-	Covered
6063	bin auto[64:79]	2615	1	-	Covered
6064	bin auto[80:95]	3081	1	-	Covered
6065	bin auto[96:111]	2862	1	-	Covered
6066	bin auto[112:127]	2535	1	-	Covered
6067	bin auto[128:143]	2782	1	-	Covered
6068	bin auto[144:159]	2756	1	-	Covered
6069	bin auto[160:175]	3029	1	-	Covered
6070	bin auto[176:191]	2522	1	-	Covered
6071	bin auto[192:207]	2821	1	-	Covered
6072	bin auto[208:223]	2847	1	-	Covered
6073	bin auto[224:239]	2808	1	-	Covered
6074	bin auto[240:255]	2665	1	-	Covered
6075	bin auto[256:271]	2912	1	-	Covered
6076	bin auto[272:287]	2342	1	-	Covered
6077	bin auto[288:303]	2574	1	-	Covered
6078	bin auto[304:319]	3146	1	-	Covered
6079	bin auto[320:335]	2873	1	-	Covered
6080	bin auto[336:351]	2509	1	-	Covered
6081	bin auto[352:367]	2847	1	-	Covered
6082	bin auto[368:383]	2550	1	-	Covered
6083	bin auto[384:399]	2652	1	-	Covered
6084	bin auto[400:415]	2808	1	-	Covered
6085	bin auto[416:431]	2600	1	-	Covered
6086	bin auto[432:447]	2678	1	-	Covered
6087	bin auto[448:463]	2665	1	-	Covered
6088	bin auto[464:479]	2600	1	-	Covered
6089	bin auto[480:495]	2678	1	-	Covered
6090	bin auto[496:511]	2249	1	-	Covered
6091	bin auto[512:527]	1250	1	-	Covered
6092	bin auto[528:543]	1716	1	-	Covered
6093	bin auto[544:559]	1404	1	-	Covered
6094	bin auto[560:575]	1079	1	-	Covered
6095	bin auto[576:591]	1404	1	-	Covered
6096	bin auto[592:607]	1391	1	-	Covered
6097	bin auto[608:623]	1456	1	-	Covered
6098	bin auto[624:639]	1326	1	-	Covered
6099	bin auto[640:655]	1378	1	-	Covered
6100	bin auto[656:671]	1248	1	-	Covered
6101	bin auto[672:687]	1404	1	-	Covered
6102	bin auto[688:703]	1469	1	-	Covered
6103	bin auto[704:719]	1352	1	-	Covered
6104	bin auto[720:735]	1248	1	-	Covered
6105	bin auto[736:751]	1339	1	-	Covered
6106	bin auto[752:767]	1261	1	-	Covered
6107	bin auto[768:783]	2714	1	-	Covered
6108	bin auto[784:799]	2450	1	-	Covered
6109	bin auto[800:815]	2650	1	-	Covered
6110	bin auto[816:831]	2825	1	-	Covered

fc_cvr_wrp_rprt.txt

```

6771      bin <nonactive,high,auto[0:3],read_add>
6772                                     21          1          -    Covered
6773  TYPE /Wrapper_coverage/coverage/Wrapper_cg      100.00%      100      -    Covered
6774      covered/total bins:                14          14          -
6775      missing/total bins:                  0          14          -
6776      % Hit:                               100.00%      100          -
6777      Coverpoint reset_cp                    100.00%      100      -    Covered
6778      covered/total bins:                  2           2          -
6779      missing/total bins:                  0           2          -
6780      % Hit:                               100.00%      100          -
6781      bin auto[0]                          192           1          -    Covered
6782      bin auto[1]                        149861           1          -    Covered
6783      Coverpoint SS_n_cp                    100.00%      100      -    Covered
6784      covered/total bins:                  2           2          -
6785      missing/total bins:                  0           2          -
6786      % Hit:                               100.00%      100          -
6787      bin auto[0]                        140052           1          -    Covered
6788      bin auto[1]                        10001           1          -    Covered
6789      Coverpoint MOSI_cp                    100.00%      100      -    Covered
6790      covered/total bins:                  2           2          -
6791      missing/total bins:                  0           2          -
6792      % Hit:                               100.00%      100          -
6793      bin auto[0]                        78503           1          -    Covered
6794      bin auto[1]                        71550           1          -    Covered
6795      Coverpoint MISO_cp                    100.00%      100      -    Covered
6796      covered/total bins:                  2           2          -
6797      missing/total bins:                  0           2          -
6798      % Hit:                               100.00%      100          -
6799      bin auto[0]                        147492           1          -    Covered
6800      bin auto[1]                        2548            1          -    Covered
6801      Coverpoint operation_cp                100.00%      100      -    Covered
6802      covered/total bins:                  4           4          -
6803      missing/total bins:                  0           4          -
6804      % Hit:                               100.00%      100          -
6805      bin WRITE_add                       43758            1          -    Covered
6806      bin WRITE_data                     42783            1          -    Covered
6807      bin READ_add                       21762            1          -    Covered
6808      bin READ_data                      41750            1          -    Covered
6809      illegal_bin Invalid_op                0              -    ZERO
6810      Cross READ_op                        100.00%      100      -    Covered
6811      covered/total bins:                  2           2          -
6812      missing/total bins:                  0           2          -
6813      % Hit:                               100.00%      100          -
6814      Auto, Default and User Defined Bins:
6815      bin <READ_data,auto[1]>                2548            1          -    Covered
6816      bin <READ_data,auto[0]>              39189            1          -    Covered
6817      Illegal and Ignore Bins:
6818      ignore_bin not_read2                 21762            -    Occurred
6819      ignore_bin not_read1                 42783            -    Occurred
6820      ignore_bin not_read0                 43758            -    Occurred
6821
6822  TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 2
6823
6824  ASSERTION RESULTS:
6825  -----
6826  Name                               File(Line)          Failure    Pass
6827                               Count              Count
6828  -----
6829  /sequences/Wrapper_main_seq/body/#ublk#146550787#35/immed__38
6830  Wrapper_seq.sv(38)                  0                1

```

BUG REPORT

❖ RAM Bugs Fixing:

- In Reset Operation `addr_rd` & `addr_wr` are $\neq 0$
→ Soln: Put `addr_rd <= 0 & addr_wr <= 0` When `rst_n`.
- integer `i = 0`; isn't Synthesizable, $\neq 100\%$ Toggle.
→ Soln: Make `(int i = 0)` in the Scope of For Loop Only.

❖ Slave Bugs Fixing:

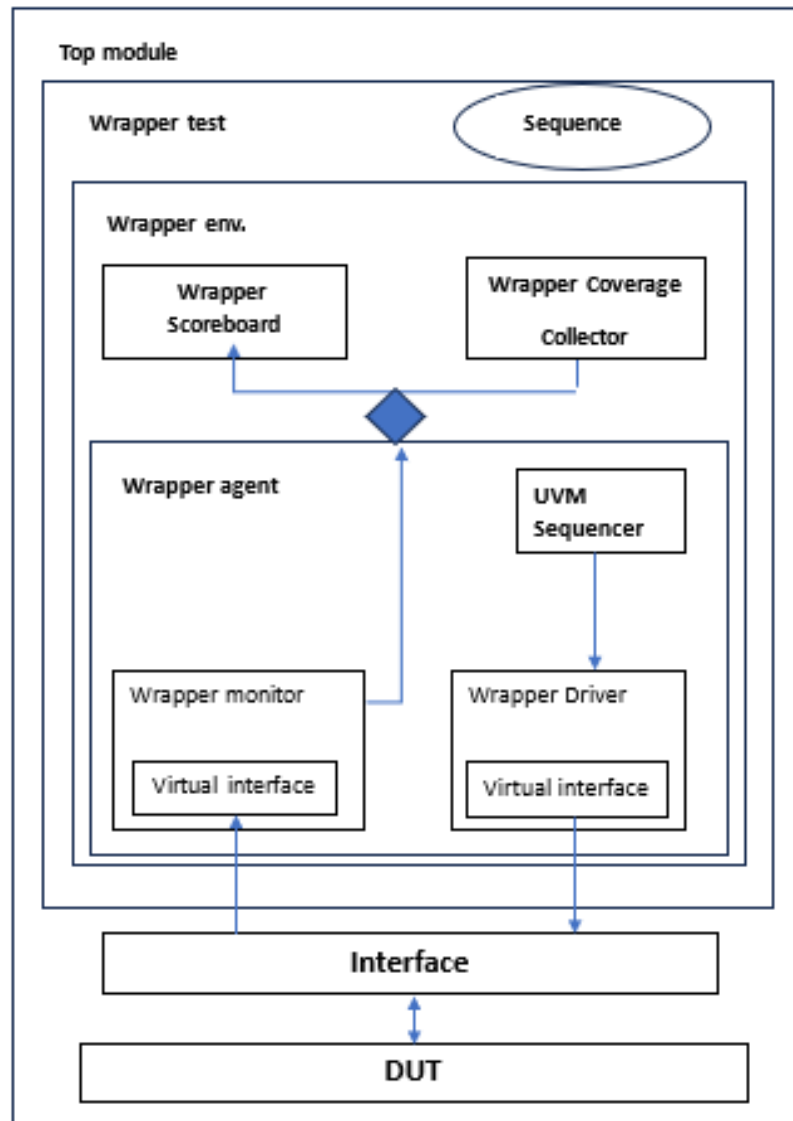
- Bug 1: MOSI Can't be included in the Sensitivity list.
- Bug 2: We should have reset default for the `rx_valid` & `rx_data` otherwise always 'x'.
- Bug 3: The output logic is related by the 'ns' not the 'cs'.
- Bug 4: `rx_valid` doesn't fall by the end of communication → Soln: Assign a new flag that checks on `rx_valid` when high (`rx_flag`).

VERIFICATION PLAN

❖ SPI Wrapper Plan:

- Reset functionality should clear the flag, counter, `rx_valid` and MISO.
- Check Starting communication phase when `SS_n` gets low.
- Testing next state choice after current state is check command according to flag & MOSI.
- Verify the serial to parallel conversion operation after 10 clock cycles after its start.
- Checking the address is updated after the `rx_valid` is asserted in WRITE and READ_ADD cases.
- Verifying the Read_data process.
- Check End communication transition when master makes `SS_n` high.

UVM DIAGRAM



UVM Structure diagram

- In Stimulus Generation the word (in 11 or 18 clock cycles) sent to the wrapper by the master is randomized only once then in the driver class the MOSI is assigned from this word one by one each negative edge of the clock.
- In Scoreboard, checking output takes place only in READ_DATA case where the MISO is collected in a variable called word_rec then it is compared at the end with the word expected.

ASSERTION TABLE

Feature	Assertion
<pre>sequence A; ((din[9] == 1'b1) && (din[8] == 1'b1)); endsequence</pre> <p>Check That in the Read Data Operation Check When rx_valid is High then After One Cycle tx_valid is Low.</p>	<pre>property tx_valid_chk_11; @(posedge clk) A → ##1 \$rose(tx_valid) → ##1 !tx_valid; endproperty</pre>
<pre>sequence B; ((din[9] == 1'b0) && (din[8] == 1'b0)) ((din[9] == 1'b0) && (din[8] == 1'b1)); endsequence</pre> <p>Check That in The Write Address & Write Data Operations After One Cycle tx_valid is Low.</p>	<pre>property tx_valid_chk_00_01; @(posedge clk) B => !(tx_valid); endproperty</pre>
<pre>sequence C; ((din[9] == 1'b1) && (din[8] == 1'b0)); endsequence</pre> <p>Check That in The Read Address Operation After One Cycle tx_valid is Low.</p>	<pre>property tx_valid_chk_10; @(posedge clk) C → ##1 (tx_valid == 0); endproperty</pre>



Eng. Kareem Waseem

