UVM Project

HU_VM

MONAYA WAEL
MENNATU-ALLAH ABD ELFATTAH
MARWAN SAMIR EZZ-ELDIN

Verification Plan

SPI Slave

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
SPI_1	When reset (rst_n) is asserted, all outputs (MISO, rx_valid, rx_data) should be low.	Directed sequence at the beginning of the simulation, then randomized with constraint that drives reset to be deasserted 98% of the time.	Cover reset bins: asserted and deasserted. Cross with SS_n and MISO to ensure outputs are inactive when reset is high.	Assertion: when reset is low, outputs must remain zero.
SPI_2	SS_n signal must be high for one cycle every 13 cycles for normal cases, and every 23 cycles for read data operations.	Constraint on SS_n inside main sequence; inline constraints switch between 13 and 23 cycle patterns.	Cover two transaction types: Normal: 1→0[*13]→1, Read Data: 1→0[*23]→1.	Assertion to ensure proper SS_n transaction duration per opcode.
SPI_3	The first 3 bits of MOSI after SS_n falls must be valid command combinations (000, 001, 110, 111).	Randomized array MOSI_arr[10:0]; constraint to generate only legal command sequences for first 3 bits.	Cover valid MOSI patterns: 000, 001, 110, 111. Add illegal bin for negative testing.	Output checked against golden model; illegal combinations should result in inactive MISO.
SPI_4	For Write Address (000), Write Data (001), Read Address (110), and Read Data (111) commands, correct response must be generated.	Main sequence generates randomized MOSI data per command; tx_valid forced high only for read data operations.	Cross coverage between MOSI opcode bins and SS_n bins to ensure all valid command–transaction pairs are covered.	Output checked against golden model.
SPI_5	tx_valid signal must be high during read data operation.	Inline constraint in main sequence when MOSI[2:0]==111.	Coverpoint on tx_valid with cross on MOSI = 111.	Assertion that tx_valid is asserted only for read data opcode.
SPI_6	Ensure that the MISO output becomes valid 10 cycles after a valid command.	Use counter-based delay check in monitor or scoreboard.	Coverpoint on rx_valid latency (number of cycles after command).	Assertion: after command (000,001,110,111), rx_valid must assert exactly after 10 cycles.
SPI_7	FSM transitions follow valid state paths.	Randomized command sequences exercising all transitions.	Cover FSM transition bins for all legal transitions.	Assertions guarded by `ifdef SIM`: IDLE→CHK_CMD, CHK_CMD→WRITE/READ_ADD/READ_DATA, WRITE→IDLE, READ_ADD→IDLE, READ_DATA→IDLE.

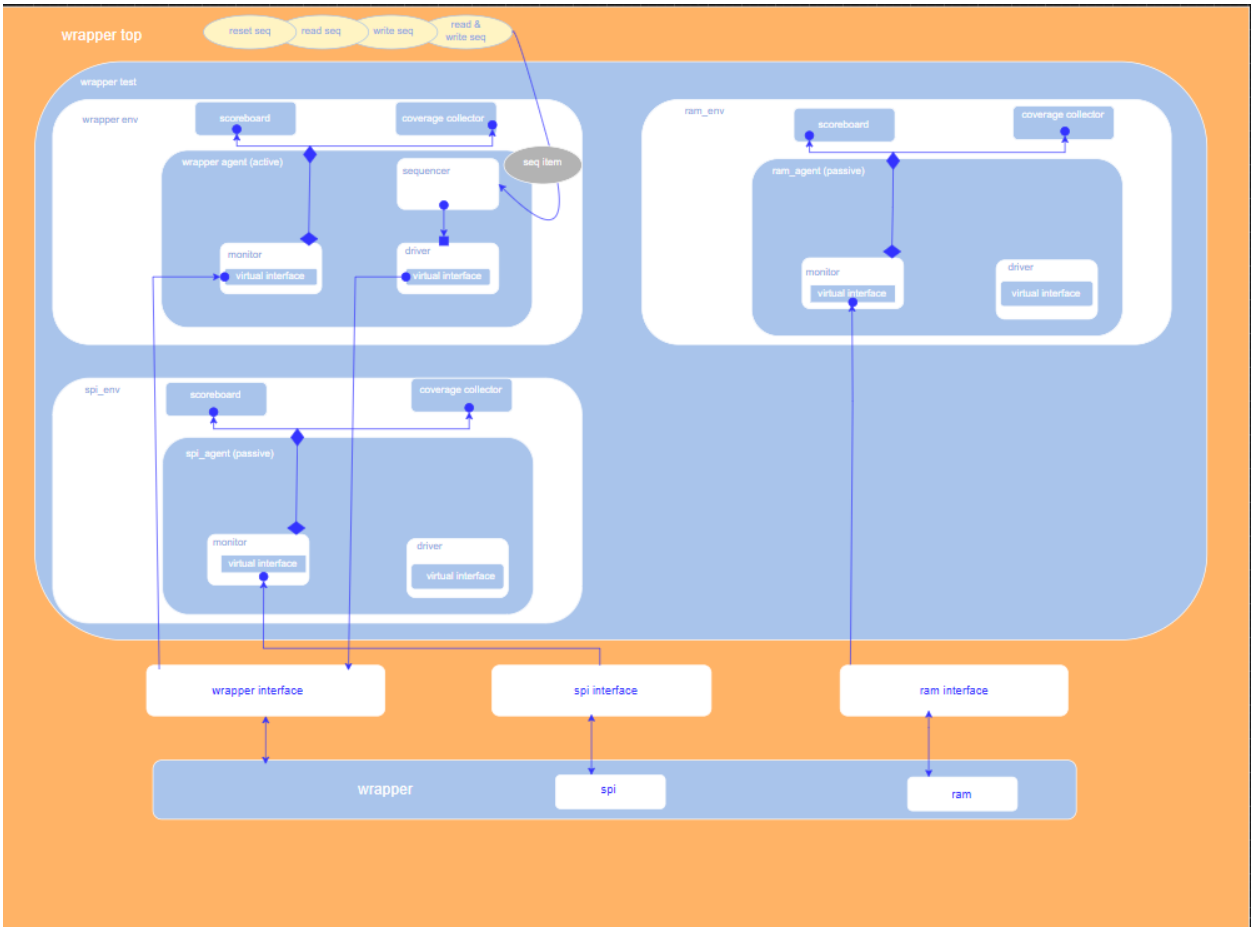
RAM

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
RAM_1	When reset (rst_n) is asserted, output signals (tx_valid and dout) must be low.	Directed reset_sequence at start of sim and occasional directed reset injections. In seq_item constrain rst_n distribution to be deasserted most of the time (95%).	Cover reset asserted / deasserted and durations. Cross reset with tx_valid and dout to ensure outputs are low during reset.	Assertion: whenever rst_n==0 then tx_valid==0 && dout==0. Checker compares golden model outputs.
RAM_2	rx_valid must be asserted most of the time (stimulus characteristic).	Constrain rx_valid distribution in seq_item (e.g., rx_valid dist {1:=95,0:=5}) or use pre_randomize to bias generation.	Coverpoint on rx_valid (high/low) with duration bins and cross with command bins.	Monitor that rx_valid is high per constraint and scoreboard tracks behavior vs golden model.
RAM_3	Write-only sequence ordering rule: every WRITE_ADDR must be followed by WRITE_ADDR or WRITE_DATA.	Implement WRITE_ONLY mode in sequence: use inline constraints or c_seq in seq_item to limit next command choices when seq_mode==WRITE_ONLY.	Coverpoint on command ordering: WRITE_ADDR→WRITE_ADDR and WRITE_ADDR→WRITE_DATA; bin the chains.	Assertion: every WRITE_ADDR eventually followed by WRITE_DATA (or another WRITE_ADDR) within N cycles; checker ensures memory write occurred for write data.
RAM_4	Read-only sequence ordering: every READ_ADDR must be followed by READ_DATA and after READ_DATA must be followed by READ_ADDR.	Implement READ_ONLY mode sequence and constraints to enforce alternation between READ_ADDR and READ_DATA.	Coverpoint on command ordering: READ_ADDR→READ_DATA and READ_DATA→READ_ADDR; cover transition sequences.	Assertion: every READ_ADDR eventually followed by READ_DATA; scoreboard checks dout and tx_valid timing.
RAM_5	Randomized read/write sequence ordering rules with specified probabilities.	In READ_WRITE mode use pre_randomize/post_randomize or inline constraints to implement probabilistic next-command selection: after WRITE_DATA -> 60% READ_ADDR, 40% WRITE_ADDR; after READ_DATA -> 60% WRITE_ADDR, 40% READ_ADDR. Ensure READ_ADDR always followed by READ_DATA; WRITE_ADDR followed by WRITE_ADDR/WRITE_DATA.	Cover sequences and probabilities by sampling many runs; cover transitions WRITE_ADDR→WRITE_DATA, WRITE_DATA→READ_ADDR, READ_DATA→WRITE_ADDR, etc.	Checker validates ordering rules, probabilities (statistical check), and that data flows correctly between writes and reads.
RAM_6	din[9:8] data-cover: check din[9:8] takes all 4 possible values and transitions.	Randomize din payload and command fields; ensure seq_item generates all four din[9:8] values via constraints or directed sequences for edge cases.	Coverpoint on din[9:8] with bins for 00,01,10,11 and transition bins between them. Also add sequence chain bin WRITE_ADDR→WRITE_DATA→READ_ADDR→READ_DATA.	Scoreboard verifies that payload used in write shows up in dout when read and that transitions follow protocol.
RAM_7	Cross coverage between din[9:8] bins and rx_valid being high.	Drive transactions with rx_valid asserted (biased) and sample din values during rx_valid windows.	Cross din[9:8] with rx_valid==1, exclude rx_valid==0. Ensure all din values are seen while rx_valid is asserted.	Checker ensures rx_valid accompanies address/data phases; assertions flag if din sampled when rx_valid==0.
RAM_8	Cross coverage between din[9:8]==READ_DATA and tx_valid==1.	Explicitly drive read_data transactions where din[9:8]==11 and observe tx_valid behavior in subsequent cycles.	Cross din[9:8]==READ_DATA with tx_valid==1 bins; measure that reads produce tx_valid high as expected.	Assertion: when READ_DATA occurs, tx_valid must pulse high (and eventually fall). Scoreboard verifies dout matches memory content.
RAM_9	tx_valid must be low during address or data input phases.	During input-phase sequences (WRITE_ADDR, WRITE_DATA, READ_ADDR) ensure rx_valid==1 and tx_valid==0 via constraints in sequences or monitor checks.	Cover that tx_valid is zero in input phases by sampling these phases and counting occurrences.	Assertion: during address/data input phases tx_valid==0. Failure report includes timestamp and sequence item context.
RAM_10	every WRITE_ADDR eventually followed by WRITE_DATA and every READ_ADDR eventually followed by READ_DATA and tx_valid pulse behavior.	Combine directed and randomized sequences to force long traces that exercise eventuality properties; use liveness checks with timeout windows.	Cover long chains such as WRITE_ADDR→WRITE_DATA→READ_ADDR→READ_DATA and transitions across multiple cycles.	Assertions for eventuality: WRITE_ADDR -> eventually WRITE_DATA; READ_ADDR -> eventually READ_DATA; READ_DATA -> tx_valid rising then falling. Bind assertions in top module and run coverage.

Wrapper

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
Wrapper_1	Reset behavior: when rst_n is asserted, MISO (and other outputs) must be inactive.	Directed reset_sequence at start of sim; bias rst_n distribution in seq_items to be deasserted most of the time (e.g., 95% deasserted).	Cover reset asserted/deasserted, durations, and cross reset with MISO, tx_valid (RAM) and rx_valid (SPI).	Assertion (bound in top): when rst_n==0, MISO==0. Also bind RAM and SPI assertions in top-level. Checker compares DUT to golden model during post-reset operation.
Wrapper_2	SS_n timing: normal transactions use 13-cycle SS low period; READ_DATA uses 23-cycle low period.	Main sequences drive SS_n using inline constraints: ss_period=13 or ss_period=23 depending on first-3 MOSI bits (111->23 else 13).	Cover SS_n timing bins: normal (13) and extended (23). Cross SS_n bins with MOSI opcode bins.	Assertion: when MOSI opcode indicates READ_DATA, SS_n must return high after 23 cycles; otherwise after 13 cycles.
Wrapper_3	MOSI command restriction: first 3 bits after SS fall must be one of {000,001,110,111}.	Randomized MOSI bit-array in seq_item; use pre_randomize/post_randomize and inline constraints to force first3bits inside allowed set.	Cover MOSI first-3-bits with bins for each opcode and transitions between opcodes across transactions.	Checker decodes MOSI and verifies legality; illegal commands flagged. Cross MOSI with RAM command mapping.
Wrapper_4	Write-only ordering: every WRITE_ADDR must be followed by WRITE_ADDR or WRITE_DATA.	Implement write_only_sequence with seq_mode flag; use inline constraints in seq_item to restrict next command choices when seq_mode==WRITE_ONLY.	Cover bins for WRITE_ADDR→WRITE_ADDR and WRITE_ADDR→WRITE_DATA chains and longer chains.	Assertion: every WRITE_ADDR eventually followed by WRITE_DATA or another WRITE_ADDR within N cycles.
Wrapper_5	Read-only ordering: READ_ADDR←→READ_DATA alternation.	Implement read_only_sequence that toggles between READ_ADDR and READ_DATA using constraints in seq_item.	Cover READ_ADDR→READ_DATA and READ_DATA→READ_ADDR transitions and frequency.	Assertion: every READ_ADDR must be followed by READ_DATA. Scoreboard checks dout and tx_valid timing for reads.
Wrapper_6	Randomized read/write ordering with probabilities.	Implement write_read_sequence (READ_WRITE mode) with pre_randomize/post_randomize or inline constraints to implement probabilistic distributions: after WRITE_DATA -> 60% READ_ADDR, 40% WRITE_ADDR; after READ_DATA -> 60% WRITE_ADDR, 40% READ_ADDR. Enforce mandatory follow-ups where required.	Collect statistical coverage: frequency of transitions and long-run distribution matching target probabilities. Cover chains like WRITE_ADDR→WRITE_DATA→READ_ADDR→READ_DATA.	Checker validates enforced order and logs statistical distribution; assertions enforce mandatory immediate-follow rules (e.g., READ_ADDR→READ_DATA).
Wrapper_7	MISO stability: MISO remains stable (unchanged) unless in READ_DATA operation.	Driver/monitor must sample MISO between transactions; sequences should create idle windows to observe stability.	Coverpoint on MISO change vs command: bins for stable-during-non-read and changes-during-read. Cross with MOSI opcode.	Assertion (bound in top): if not in READ_DATA state then MISO must not change (stable value eventually).
Wrapper_8	rx_valid and tx_valid behavior across integrated DUT.	Bias rx_valid in SPI agent sequences as required; ensure RAM agent sequences use rx_valid to accept din. Passive agents reuse monitors to check signals.	Cover rx_valid high occurrences and cross rx_valid with din[9:8] and tx_valid.	Assertion: tx_valid low during input phases; tx_valid pulses following READ_DATA and then falls. Cross-check timing between SPI and RAM outputs.
Wrapper_9	End-to-end data correctness: data written via SPI must be read back correctly through integrated RAM.	Stimulate sequences that perform WRITE_ADDR→WRITE_DATA then later READ_ADDR→READ_DATA using wrapper sequences that map SPI transactions to RAM transactions.	Cover end-to-end scenarios: write-then-read success bins, multibank checks, and data integrity across different addresses and data values.	Scoreboard compares read-back dout with original write payload; raise error on mismatch.
Wrapper_10	Passive agent integration & bindings: reuse RAM and SPI environments with passive agents in wrapper environment.	Instantiate SPI and RAM agents in the top-level UVM env; set their driver active/passive flags appropriately. Use monitors to connect to top-level scoreboard.	Cover that both agents observed expected traffic; cross agent observations to validate mapping between SPI transactions and RAM transactions.	Ensure monitors provide transactions to scoreboard; assertions for interface-level checks are bound in top module for DUT and in monitor checks.

UVM Structure



Code Snippets

SPI Slave

- Sequence Item package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_seq_item.sv > ...

```
package spi_seq_item_pkg;

// Importing needed packages and macros
import uvm_pkg::*;
`include "uvm_macros.svh"

// Creating the class
class spi_seq_item extends uvm_sequence_item;
    `uvm_object_utils(spi_seq_item)

    // Constructor
    function new(string name = "spi_seq_item");
        super.new(name);
    endfunction

    // Randomizing stimulus inputs and outputs
    rand logic rst_n, SS_n, tx_valid;
    rand logic [7:0] tx_data;
    logic MOSI;
    // outputs
    logic MISO, rx_valid;
    logic [9:0] rx_data;
    // reference outputs
    logic MISO_ref, rx_valid_ref;
    logic [9:0] rx_data_ref;
    // for constraints
    typedef enum bit [2:0] { WRITE_ADDR = 3'b000, WRITE_DATA = 3'b001, READ_ADDR = 3'b110, READ_DATA = 3'b111 } cmd_e;
    rand cmd_e cmd;
    rand bit MOSI_arr[10:0];
    bit MOSI_arr_drv[10:0];
    int read_counter = 0, normal_counter = 0, ss_counter = 0;
    bit read_data_falg = 0;

    // Constraints
    // Reset constraint
    constraint rst_c {
        rst_n dist {0 := 2, 1 := 98};
    }

    function void post_randomize();

        {MOSI_arr[0], MOSI_arr[1], MOSI_arr[2]} = cmd;

        if (ss_counter == 0)
            SS_n = 1;
        else
            SS_n = 0;

        if ((normal_counter == 0) && (read_counter == 0) && (ss_counter == 0)) begin
            MOSI_arr_drv = MOSI_arr;
        end
        read_data_falg = (({MOSI_arr_drv[0], MOSI_arr_drv[1], MOSI_arr_drv[2]}) == READ_DATA);

        if (read_data_falg) begin
            if (ss_counter < 23)
                ss_counter++;
            else
                ss_counter = 0;
        end
        else begin
            if (ss_counter < 13)
                ss_counter++;
            else
                ss_counter = 0;
        end

        if (read_counter < 25 && !SS_n && read_data_falg && ss_counter > 2) begin
            if (read_counter < 11)
                MOSI = MOSI_arr_drv[read_counter++];
            else
                read_counter++;
        end
        else begin
            read_counter = 0;
            read_data_falg = 0;
        end

        if (normal_counter < 15 && !SS_n && !read_data_falg && ss_counter > 2) begin
            if (normal_counter < 11)
                MOSI = MOSI_arr_drv[normal_counter++];
            else
                normal_counter++;
        end
        else
            normal_counter = 0;

        tx_valid = (({MOSI_arr_drv[0], MOSI_arr_drv[1], MOSI_arr_drv[2]}) == READ_DATA);
    endfunction

    // Convert to string (full)
    function string convert2string();
        return $sformatf(
            "%s reset = %0b, SS_n = %0b, MOSI = %0b, MISO = %0b, rx_valid = %0b, tx_valid = %0b,rx_data = %0b, tx_data = %0b",
            super.convert2string(),
            rst_n, SS_n, MOSI, MISO, rx_valid, tx_valid, rx_data, tx_data
        );
    endfunction

    // Convert to string (stimulus only)
    function string convert2string_stimulus();
        return $sformatf(
            " reset = %0b, SS_n = %0b, MOSI = %0b, tx_valid = %0b, tx_data = %0b",
            rst_n, SS_n, MOSI, tx_valid, tx_data
        );
    endfunction

endclass

endpackage
```

➢ Reset Sequence package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_reset_seq.sv > ...

```
package spi_reset_seq_pkg;

// importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_seq_item_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_reset_seq extends uvm_sequence #(spi_seq_item);
        `uvm_object_utils(spi_reset_seq)

        // handling the seq_item
        spi_seq_item seq_item;

        // constructor
        function new(string name = "spi_reset_seq");
            super.new(name);
        endfunction

        // generating randomized stimulus inputs and outputs
        task body;
            seq_item = spi_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.rst_n = 0;
                seq_item.SS_n = 0;
                seq_item.MOSI = 0;
                seq_item.tx_valid = 0;
                seq_item.tx_data = 0;
                finish_item(seq_item);
        endtask

    endclass
endpackage
```

➢ Main Sequence package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_main_seq.sv > ...

```
package spi_main_seq_pkg;

// importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_seq_item_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_main_seq extends uvm_sequence #(spi_seq_item);
        `uvm_object_utils(spi_main_seq)

        // handling the seq_item
        spi_seq_item seq_item;

        // constructor
        function new(string name = "spi_main_seq");
            super.new(name);
        endfunction

        //generating randomized stimulus inputs and outputs
        task body;
            seq_item = spi_seq_item::type_id::create("seq_item");

            repeat(10000) begin
                start_item(seq_item);
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask

    endclass
endpackage
```

➢ Sequencer package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_sequencer.sv > ...

```
package spi_sequencer_pkg;
    // importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_seq_item_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_sequencer extends uvm_sequencer #(spi_seq_item);
        `uvm_component_utils(spi_sequencer)

        // constructor
        function new(string name = "spi_sequencer", uvm_component parent = null);
            super.new(name, parent);
        endfunction
    endclass
endpackage
```

➢ Design Interface

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_interface.sv > ...

```
interface SPI_IF #(
  parameter IDLE = 3'b000,
  parameter CHK_CMD = 3'b001,
  parameter WRITE = 3'b010,
  parameter READ_ADDRESS = 3'b011,
  parameter READ_DATA = 3'b100

) (input bit clk);

  logic rst_n, SS_n, MOSI, MISO, rx_valid, tx_valid ;
  logic [9:0] rx_data;
  logic [7:0] tx_data;

endinterface

```

➢ Golden Model Interface

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_GM_interface.sv > ...

```
interface SPI_GM_IF #(
  parameter IDLE = 3'b000,
  parameter CHK_CMD = 3'b001,
  parameter WRITE = 3'b010,
  parameter READ_ADDRESS = 3'b011,
  parameter READ_DATA = 3'b100

) (input bit clk);

  logic rst_n, SS_n, MOSI, MISO, rx_valid, tx_valid ;
  logic [9:0] rx_data;
  logic [7:0] tx_data;

endinterface

```

➢ Driver Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > spi_driver.sv > ...

```
package spi_driver_pkg;
// importing needed pkgs and macros
import uvm_pkg::*;
import spi_seq_item_pkg::*;
`include "uvm_macros.svh"
   class spi_driver extends uvm_driver #(spi_seq_item);
    `uvm_component_utils(spi_driver)

      //giving handles
      virtual SPI_IF SPI_vif;
      virtual SPI_GM_IF SPI_GM_vif;
      spi_seq_item stim_seq_item;

      //constructor
      function new(string name = "spi_driver", uvm_component parent = null);
          super.new(name, parent);
      endfunction //new()

      task run_phase(uvm_phase phase);
          super.run_phase(phase);
          forever begin
              stim_seq_item = spi_seq_item::type_id::create("stim_seq_item",this);
              seq_item_port.get_next_item(stim_seq_item);
              SPI_vif.rst_n= stim_seq_item.rst_n;
              SPI_vif.SS_n= stim_seq_item.SS_n;
              SPI_vif.tx_valid  = stim_seq_item.tx_valid;
              SPI_vif.tx_data = stim_seq_item.tx_data;
              SPI_vif.MOSI = stim_seq_item.MOSI;
              if (SPI_GM_vif != null) begin
                  SPI_GM_vif.rst_n= stim_seq_item.rst_n;
                  SPI_GM_vif.SS_n= stim_seq_item.SS_n;
                  SPI_GM_vif.tx_valid  = stim_seq_item.tx_valid;
                  SPI_GM_vif.tx_data = stim_seq_item.tx_data;
                  SPI_GM_vif.MOSI = stim_seq_item.MOSI;
              end

              @(negedge SPI_vif.clk);
              seq_item_port.item_done();

              `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(),UVM_HIGH)

          end
      endtask: run_phase
   endclass
endpackage

```

➢ Monitor Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_monitor.sv > ...

```
package spi_monitor_pkg;
// importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_seq_item_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_monitor extends uvm_component;
        `uvm_component_utils(spi_monitor)

        // analysis port of agent
        uvm_analysis_port #(spi_seq_item) mon_aport;

        // giving handles
        virtual SPI_IF SPI_vif;
        virtual SPI_GM_IF SPI_GM_vif;
        spi_seq_item res_seq_item;

        // constructor
        function new(string name = "spi_monitor", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        //build phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            mon_aport = new("mon_aport", this);
        endfunction

        //run phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
              res_seq_item = spi_seq_item::type_id::create("res_seq_item",this);
              @(negedge SPI_vif.clk);
              res_seq_item.rst_n= SPI_vif.rst_n;
               res_seq_item.SS_n= SPI_vif.SS_n;
               res_seq_item.MOSI = SPI_vif.MOSI;
               res_seq_item.tx_valid = SPI_vif.tx_valid;
               res_seq_item.tx_data = SPI_vif.tx_data;
              res_seq_item.MISO = SPI_vif.MISO;
              res_seq_item.rx_valid = SPI_vif.rx_valid;
              res_seq_item.rx_data = SPI_vif.rx_data;

               if (SPI_GM_vif != null) begin
                   res_seq_item.MISO_ref = SPI_GM_vif.MISO;
                   res_seq_item.rx_valid_ref = SPI_GM_vif.rx_valid;
                   res_seq_item.rx_data_ref = SPI_GM_vif.rx_data;
               end

              mon_aport.write (res_seq_item);
              `uvm_info("run_phase", res_seq_item.convert2string(),UVM_HIGH)
            end
        endtask

    endclass
endpackage

```

➢ Config Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_config_obj.sv > ...

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

     class spi_config extends uvm_object;
         `uvm_object_utils(spi_config)

         virtual SPI_IF SPI_vif;
         virtual SPI_GM_IF SPI_GM_vif;
         uvm_active_passive_enum is_active;

         function new(string name = "spi_config");
             super.new(name);
             is_active = UVM_ACTIVE;
         endfunction

     endclass
 endpackage

```

➢ Agent Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_agent.sv > ...

```
package spi_agent_pkg;
    // importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_config_pkg::*;
    import spi_driver_pkg::*;
    import spi_sequencer_pkg::*;
    import spi_seq_item_pkg::*;
    import spi_monitor_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_agent extends uvm_agent;
        `uvm_component_utils(spi_agent)
        // analysis port of agent
        uvm_analysis_port #(spi_seq_item) agent_aport;

        // giving handles
        spi_driver drv;
        spi_sequencer sqcr;
        spi_monitor mon;
        spi_config conf;

        // constructor
        function new(string name = "spi_agent", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        //build phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            if (!uvm_config_db #(spi_config)::get(this , "" , "CFG" , conf)) begin
                `uvm_fatal("build_phase", "error in getting the data");
            end
            if(conf.is_active == UVM_ACTIVE) begin
                drv = spi_driver::type_id::create("drv", this);
                sqcr = spi_sequencer::type_id::create("sqcr", this);
            end
            mon = spi_monitor::type_id::create("mon", this);
            agent_aport = new("agent_aport", this);
        endfunction

        // connect phase
        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            if (conf.is_active == UVM_ACTIVE) begin
                drv.SPI_vif = conf.SPI_vif;
                drv.SPI_GM_vif = conf.SPI_GM_vif;
                drv.seq_item_port.connect(sqcr.seq_item_export);
            end

            mon.SPI_vif = conf.SPI_vif;
            mon.SPI_GM_vif = conf.SPI_GM_vif;
            mon.mon_aport.connect(agent_aport);

        endfunction

    endclass
endpackage

```

➢ Scoreboard Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_scoreboard.sv > ...

```
package spi_scoreboard_pkg;
    // importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_seq_item_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(spi_scoreboard)

        // analysis port of scoreboard
        uvm_analysis_export #(spi_seq_item) sb_exp;
        uvm_tlm_analysis_fifo #(spi_seq_item) sb_fifo;
        int correct_count = 0, error_count = 0;

        // giving handles
        spi_seq_item seq_item_sb;

        // constructor
        function new(string name = "spi_scoreboard", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        //build phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            sb_exp = new("sb_exp", this);
            sb_fifo = new("sb_fifo", this);

        endfunction

        // connect phase
        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            sb_exp.connect(sb_fifo.analysis_export);
        endfunction
```

```
        //run phase
        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                    if (seq_item_sb.MISO != seq_item_sb.MISO_ref || seq_item_sb.rx_valid != seq_item_sb.rx_valid_ref || seq_item_sb.rx_data != seq_item_sb.rx_data_ref ) begin
                        if (seq_item_sb.MISO != seq_item_sb.MISO_ref) begin
                            `uvm_error("run_phase", $sformatf("MISO mismatch: miso = %0b, miso_ref = %0b", seq_item_sb.MISO, seq_item_sb.MISO_ref ));
                        end
                        if (seq_item_sb.rx_valid != seq_item_sb.rx_valid_ref) begin
                            `uvm_error("run_phase", $sformatf("RX_VALID mismatch: rx_valid = %0b, rx_valid_ref = %0b", seq_item_sb.rx_valid, seq_item_sb.rx_valid_ref ));
                        end
                        if (seq_item_sb.rx_data != seq_item_sb.rx_data_ref) begin
                            `uvm_error("run_phase", $sformatf("RX_DATA mismatch: rx_data = %0b, rx_data_ref = %0b", seq_item_sb.rx_data, seq_item_sb.rx_data_ref ));
                        end
                        error_count++;
                    end else begin
                        `uvm_info("run_phase", $sformatf("comparison completed"),UVM_HIGH);
                        correct_count++;
                    end
                end
            endtask

            //report phase
            function void report_phase (uvm_phase phase);
                super.report_phase(phase) ;
                `uvm_info("report_phase", $sformatf("total correct tranxactions: %0d", correct_count),UVM_MEDIUM);
                `uvm_info("report_phase", $sformatf("total failed tranxactions: %0d", error_count),UVM_MEDIUM);
            endfunction

    endclass
endpackage
```

➢ Coverage Package

```
D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_coverage.sv > ...
package spi_coverage_pkg;
    // importing needed pkgs and macros
    import uvm_pkg::*;
    import spi_seq_item_pkg::*;
    `include "uvm_macros.svh"

    // creating the class
    class spi_coverage extends uvm_component;
        `uvm_component_utils(spi_coverage)

        // analysis port of agent
        uvm_analysis_port #(spi_seq_item) cov_exp;
        uvm_tlm_analysis_fifo #(spi_seq_item) cov_fifo;

        // giving handles
        spi_seq_item cov_seq_item;

        // adding cover groups
        covergroup cvr_gp;

            // ---- rx_data coverpoints ----
            rx_data_cp: coverpoint cov_seq_item.rx_data[9:8]{
                bins all_vals[] = {2'b00,2'b01,2'b10,2'b11};
                bins wradd_2_wrdata = (2'b00 => 2'b01);
                bins rdaddr_2_rddata = (2'b10 => 2'b11);
                bins rddata_2_wradd = (2'b11 => 2'b00);
            }

            // ---- ss_n coverpoints ----
            ss_n_cp: coverpoint cov_seq_item.SS_n{
                bins normal_trans  = (1 => 0 [*13] => 1);
                bins readdata_trans = (1 => 0 [*23] => 1);
            }
            SS_N_cp:coverpoint cov_seq_item.SS_n{
                bins normal  = (1 -> 0 [*4]);
            }

            MOSI_cp : coverpoint cov_seq_item.MOSI {
                bins write_addr  = (0 => 0 => 0);
                bins write_data  = (0 => 0 => 1);
                bins read_addr   = (1 => 1 => 0);
                bins read_data   = (1 => 1 => 1);
                bins others = default;
            }

            // ---- Crosses ----
            SS_MOSI_c : cross SS_N_cp, MOSI_cp;

        endgroup : cvr_gp;

        // constructor
        function new(string name = "spi_coverage", uvm_component parent = null);
            super.new(name, parent);
            cvr_gp = new();
        endfunction

        //build phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            cov_exp = new("cov_exp", this);
            cov_fifo =  new("cov_fifo", this);
        endfunction

        // connect phase
        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cov_exp.connect(cov_fifo.analysis_export);
        endfunction

        //run phase
        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(cov_seq_item);
                cvr_gp.sample();
            end
        endtask

    endclass
endpackage
```

➢ Env Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_env.sv > ...

```
package spi_env_pkg;

//importing needed pkgs
import uvm_pkg::*;
import spi_agent_pkg::*;
import spi_scoreboard_pkg::*;
import spi_coverage_pkg::*;
`include "uvm_macros.svh"
    class spi_env extends uvm_env;
        `uvm_component_utils(spi_env)

        //giving handels
        spi_agent agent;
        spi_scoreboard score;
        spi_coverage cov;

        //constructor
        function new(string name = "spi_env", uvm_component parent = null);
        super.new(name, parent);
        endfunction

        //build_phase
        function void build_phase (uvm_phase phase);
            super.build_phase(phase);
            agent =  spi_agent::type_id::create("agent", this);
            score =  spi_scoreboard::type_id::create("score", this);
            cov =  spi_coverage::type_id::create("cov", this);
        endfunction

         //conecct phase
        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase) ;
            agent.agent_aport.connect(score.sb_exp);
            agent.agent_aport.connect(cov.cov_exp);
        endfunction

    endclass

endpackage

```

➢ Test Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_test.sv > ...

```
package spi_test_pkg;

//importing needed pkgs
import uvm_pkg::*;
import spi_env_pkg::*;
import spi_config_pkg::*;
import spi_reset_seq_pkg::*;
import spi_main_seq_pkg::*;
`include "uvm_macros.svh"

    class spi_test extends uvm_test;
        `uvm_component_utils(spi_test)

        //giving handels
        spi_env env;
        spi_config conf;
        virtual SPI_if spi_test_vif;
        virtual SPI_GM_IF SPI_GM_vif;
        spi_reset_seq res_seq;
        spi_main_seq main_seq;

        // constructor
        function new(string name = "spi_test", uvm_component parent = null);
        super.new(name, parent);
        endfunction

        //build_phase
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = spi_env::type_id::create("env", this);
            conf = spi_config::type_id::create("conf", this);
            res_seq = spi_reset_seq::type_id::create("res_seq", this);
            main_seq = spi_main_seq::type_id::create("main_seq", this);
            if (!uvm_config_db #(virtual SPI_IF)::get(this , "" , "spi" , conf.SPI_vif)) begin
                `uvm_fatal("build_phase", "error in getting the data");
            end
            if (!uvm_config_db #(virtual SPI_GM_IF)::get(this , "" , "spi_ref" , conf.SPI_GM_vif)) begin
                `uvm_fatal("build_phase", "error in getting the data");
            end
            uvm_config_db #(spi_config)::set(this , "*" , "CFG" , conf);
        endfunction: build_phase

        //run_phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            // reset seq
            `uvm_info("run_phase", "reset asserted", UVM_LOW);
            res_seq.start(env.agent.sqcr);
            `uvm_info("run_phase", "reset deasserted", UVM_LOW);

            // main seq
            `uvm_info("run_phase", "main asserted", UVM_LOW);
            main_seq.start(env.agent.sqcr);
            `uvm_info("run_phase", "main deasserted", UVM_LOW);
            phase.drop_objection(this);
        endtask: run_phase
    endclass: spi_test

endpackage
```

➢ Design Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI.sv > SPI_SLAVE

```
module SPI_SLAVE (SPI_IF spi_if);

    reg [3:0] counter;
    reg       received_address;

    // state registers
    reg [2:0] cs, ns;

    always @(posedge spi_if.clk) begin
        if (~spi_if.rst_n) begin
            cs <= spi_if.IDLE;
        end
        else begin
            cs <= ns;
        end
    end

    // next state logic
    always @(*) begin
        case (cs)
            spi_if.IDLE: begin
                if (spi_if.SS_n)
                    ns = spi_if.IDLE;
                else
                    ns = spi_if.CHK_CMD;
            end

            spi_if.CHK_CMD: begin
                if (spi_if.SS_n)
                    ns = spi_if.IDLE;
                else begin
                    if (~spi_if.MOSI)
                        ns = spi_if.WRITE;
                    else begin
                        if (received_address)
                            //*****fixed this bug here, from read address to read data*****//
                            ns = spi_if.READ_DATA;
                        else
                            ns = spi_if.READ_ADDRESS;
                        end
                    end
                end

            spi_if.WRITE: begin
                if (spi_if.SS_n)
                    ns = spi_if.IDLE;
                else
                    ns = spi_if.WRITE;
                end

            spi_if.READ_ADDRESS: begin
                if (spi_if.SS_n)
                    ns = spi_if.IDLE;
                else
                    ns = spi_if.READ_ADDRESS;
                end
```

```
            spi_if.READ_ADDRESS: begin
                if (spi_if.SS_n)
                    ns = spi_if.IDLE;
                else
                    ns = spi_if.READ_ADDRESS;
                end

            spi_if.READ_DATA: begin
                if (spi_if.SS_n)
                    ns = spi_if.IDLE;
                else
                    ns = spi_if.READ_DATA;
                end
                //*****added default case to avoid latches*****//
                default: ns = spi_if.IDLE;
            endcase
    end

    always @(posedge spi_if.clk) begin
        if (~spi_if.rst_n) begin
            spi_if.rx_data <= 0;
            spi_if.rx_valid <= 0;
            received_address <= 0;
            spi_if.MISO <= 0;
            spi_if.MOSI <= 0;
            //*****fixed this bug here, counter should be 0 at reset*****//
            counter <= 0;
        end
        else begin
            case (cs)
                spi_if.IDLE: begin
                    spi_if.rx_valid <= 0;
                end

                spi_if.CHK_CMD: begin
                    counter <= 10;
                end

                spi_if.WRITE: begin
                    if (counter > 0) begin
                        spi_if.rx_data[counter-1] <= spi_if.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        spi_if.rx_valid <= 1;
                    end
                end
```

```
                spi_if.READ_ADDRESS: begin
                    if (counter > 0) begin
                        spi_if.rx_data[counter-1] <= spi_if.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        spi_if.rx_valid <= 1;
                        received_address <= 1;
                    end
                end

                spi_if.READ_DATA: begin
                    if (spi_if.tx_valid) begin
                        spi_if.rx_valid <= 0;
                        if (counter > 0) begin
                            spi_if.MISO <= spi_if.tx_data[counter-1];
                            counter <= counter - 1;
                        end
                        else begin
                            received_address <= 0;
                        end
                    end
                    else begin
                        if (counter > 0) begin
                            spi_if.rx_data[counter-1] <= spi_if.MOSI;
                            counter <= counter - 1;
                        end
                        else begin
                            spi_if.rx_valid <= 1;
                            //* fixed the counter reset to get all bits of miso*//
                            counter <= 9;
                        end
                    end
                end
                default : begin
                    spi_if.rx_data <= 0;
                    spi_if.rx_valid <= 0;
                    received_address <= 0;
                    spi_if.MISO <= 0;
                    spi_if.MOSI <= 0;
                    counter <= 0;
                end

            endcase
        end
    end
```

```
`ifdef SIM

    // sequences for command patterns
    sequence write_addr; ((spi_if.MOSI == 0) ##1 (spi_if.MOSI == 0) ##1 (spi_if.MOSI == 0) && counter == 10); endsequence
    sequence write_data; ((spi_if.MOSI == 0) ##1 (spi_if.MOSI == 0) ##1 (spi_if.MOSI == 1) && counter == 10); endsequence
    sequence read_addr;  ((spi_if.MOSI == 1) ##1 (spi_if.MOSI == 1) ##1 (spi_if.MOSI == 0) && counter == 10); endsequence
    sequence read_data;  ((spi_if.MOSI == 1) ##1 (spi_if.MOSI == 1) ##1 (spi_if.MOSI == 1) && counter == 9 && !received_address); endsequence

    // ==================================================
    //                FSM PROPERTIES
    // ==================================================

    // --- reset assertion
    property p_rst;
        @(posedge spi_if.clk)
            (!spi_if.rst_n)
            |=> (spi_if.MISO == 0 && spi_if.rx_valid == 0 && spi_if.rx_data == 0);
    endproperty

    property p_IDLE_to_chk;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n)
            (cs == spi_if.IDLE && !spi_if.SS_n) |=> (cs == spi_if.CHK_CMD);
    endproperty

    property p_chk_to_WRITE;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (cs == spi_if.CHK_CMD && !spi_if.MOSI) |=> (cs == spi_if.WRITE);
    endproperty

    property p_chk_to_READ_ADDRESS;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (cs == spi_if.CHK_CMD && spi_if.MOSI && !received_address) |=> (cs == spi_if.READ_ADDRESS);
    endproperty

    property p_chk_to_READ_DATA;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (cs == spi_if.CHK_CMD && spi_if.MOSI && received_address) |=> (cs == spi_if.READ_DATA);
    endproperty

    property p_WRITE_to_IDLE;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n)
            (cs == spi_if.WRITE && spi_if.SS_n) |=> (cs == spi_if.IDLE);
    endproperty

    property p_readadd_to_IDLE;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n)
            (cs == spi_if.READ_ADDRESS && spi_if.SS_n) |=> (cs == spi_if.IDLE);
    endproperty

    property p_readdata_to_IDLE;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n)
            (cs == spi_if.READ_DATA && spi_if.SS_n) |=> (cs == spi_if.IDLE);
    endproperty
```

```
    property p_rx_valid_wr_addr;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (write_addr) |=> ##10 (spi_if.rx_valid);
    endproperty

    property p_rx_valid_wr_data;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (write_data) |=> ##10 (spi_if.rx_valid);
    endproperty

    property p_rx_valid_rd_addr;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (read_addr) |=> ##10 (spi_if.rx_valid);
    endproperty

    property p_rx_valid_rd_data;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n)
            (read_data) |=> ##10 (spi_if.rx_valid);
    endproperty

    property p_ss_N_rd_data;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n)
            (cs == spi_if.READ_DATA) |=> ##23 (spi_if.SS_n[->1]);
    endproperty

    property p_ss_N_normal;
        @(posedge spi_if.clk) disable iff(!spi_if.rst_n)
            (cs == spi_if.WRITE || cs == spi_if.READ_ADDRESS) |=> ##13 (spi_if.SS_n[->1]);
    endproperty
```

```
    // ==================================================
    //                FSM ASSERTIONS
    // ==================================================

    assert property (p_rst)
        else $error("ASSERTION FAILED: reset didn't clear the outputs");

    assert property (p_IDLE_to_chk)
        else $error("FSM Error: Expected spi_if.IDLE → spi_if.CHK_CMD");

    assert property (p_chk_to_WRITE)
        else $error("FSM Error: spi_if.CHK_CMD must go to spi_if.WRITE");

    assert property (p_chk_to_READ_ADDRESS)
        else $error("FSM Error: spi_if.CHK_CMD must go to READ_ADD");

    assert property (p_chk_to_READ_DATA)
        else $error("FSM Error: spi_if.CHK_CMD must go to spi_if.READ_DATA");

    assert property (p_WRITE_to_IDLE)
        else $error("FSM Error: spi_if.WRITE must return to spi_if.IDLE");

    assert property (p_readadd_to_IDLE)
        else $error("FSM Error: READ_ADD must return to spi_if.IDLE");

    assert property (p_readdata_to_IDLE)
        else $error("FSM Error: spi_if.READ_DATA must return to spi_if.IDLE");

    assert property (p_rx_valid_wr_addr)
        else $error("ASSERTION FAILED: rx_valid_wr_addr did not assert after command");

    assert property (p_rx_valid_wr_data)
        else $error("ASSERTION FAILED: rx_valid_wr_data did not assert after command");

    assert property (p_rx_valid_rd_addr)
        else $error("ASSERTION FAILED: rx_valid_rd_addr did not assert after command");

    assert property (p_rx_valid_rd_data)
        else $error("ASSERTION FAILED: rx_valid_rd_data did not assert after command");

    assert property (p_ss_N_rd_data)
        else $error("ASSERTION FAILED: ss_N_rd_data did not assert after command");

    assert property (p_ss_N_normal)
        else $error("ASSERTION FAILED: ss_N_normal did not assert after command");
```

```
    // ==================================================
    //                FSM COVERS
    // ==================================================

    cover property (p_rst);
    cover property (p_IDLE_to_chk);
    cover property (p_chk_to_WRITE);
    cover property (p_chk_to_READ_ADDRESS);
    cover property (p_chk_to_READ_DATA);
    cover property (p_WRITE_to_IDLE);
    cover property (p_readadd_to_IDLE);
    cover property (p_readdata_to_IDLE);
    cover property (p_rx_valid_wr_addr);
    cover property (p_rx_valid_wr_data);
    cover property (p_rx_valid_rd_addr);
    cover property (p_rx_valid_rd_data);
    cover property (p_ss_N_rd_data);
    cover property (p_ss_N_normal);

`endif

endmodule
```

➢ Golden Model Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_GM.sv > ...

```
module SPI_REF (SPI_GM_IF spi_gm_if);

reg [3:0] counter;
reg       received_address;
// state registers
reg [2:0] cs, ns;
always @(posedge spi_gm_if.clk) begin
    if (~spi_gm_if.rst_n) begin
        cs <= spi_gm_if.IDLE;
    end
    else begin
        cs <= ns;
    end
end

// next state logic
always @(*) begin
    case (cs)
        spi_gm_if.IDLE : begin
            if (spi_gm_if.SS_n)
                ns = spi_gm_if.IDLE;
            else
                ns = spi_gm_if.CHK_CMD;
        end
        spi_gm_if.CHK_CMD : begin
            if (spi_gm_if.SS_n)
                ns = spi_gm_if.IDLE;
            else begin
                if (~spi_gm_if.MOSI)
                    ns = spi_gm_if.WRITE;
                else begin
                    if (received_address)
                    //*****fixed this bug here, from read address to read data*****//
                        ns = spi_gm_if.READ_DATA;
                    else
                        ns = spi_gm_if.READ_ADDRESS;
                end
            end
        end
        spi_gm_if.WRITE : begin
            if (spi_gm_if.SS_n)
                ns = spi_gm_if.IDLE;
            else
                ns = spi_gm_if.WRITE;
        end
        spi_gm_if.READ_ADDRESS : begin
            if (spi_gm_if.SS_n)
                ns = spi_gm_if.IDLE;
            else
                ns = spi_gm_if.READ_ADDRESS;
        end
        spi_gm_if.READ_DATA : begin
            if (spi_gm_if.SS_n)
                ns = spi_gm_if.IDLE;
            else
                ns = spi_gm_if.READ_DATA;
        end
        //*****added default case to avoid latches*****//
        default : ns = spi_gm_if.IDLE;
    endcase
end


always @(posedge spi_gm_if.clk) begin
    if (~spi_gm_if.rst_n) begin
        spi_gm_if.rx_data <= 0;
        spi_gm_if.rx_valid <= 0;
        received_address <= 0;
        spi_gm_if.MISO <= 0;
        //*****fixed this bug here, counter should be 0 at reset*****//
        counter <= 0;
    end
    else begin
        case (cs)
            spi_gm_if.IDLE : begin
                spi_gm_if.rx_valid <= 0;
            end
            spi_gm_if.CHK_CMD : begin
                counter <= 10;
            end
            spi_gm_if.WRITE : begin
                if (counter > 0) begin
                    spi_gm_if.rx_data[counter-1] <= spi_gm_if.MOSI;
                    counter <= counter - 1;
                end
                else begin
                    spi_gm_if.rx_valid <= 1;
                end
            end
            spi_gm_if.READ_ADDRESS : begin
                if (counter > 0) begin
                    spi_gm_if.rx_data[counter-1] <= spi_gm_if.MOSI;
                    counter <= counter - 1;
                end
                else begin
                    spi_gm_if.rx_valid <= 1;
                    received_address <= 1;
                end
            end
            spi_gm_if.READ_DATA : begin
```

```
            spi_gm_if.READ_DATA : begin

                if (spi_gm_if.tx_valid) begin
                    spi_gm_if.rx_valid <= 0;
                    if (counter > 0) begin
                        spi_gm_if.MISO <= spi_gm_if.tx_data[counter-1];
                        counter <= counter - 1;
                    end
                    else begin
                        received_address <= 0;
                    end
                end
                else begin
                    if (counter > 0) begin
                        spi_gm_if.rx_data[counter-1] <= spi_gm_if.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        spi_gm_if.rx_valid <= 1;
                        counter <= 9;
                    end
                end
            end
        endcase
    end
end

endmodule

```

Top Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > SPI_top.sv > ...

```
import uvm_pkg::*;
`include "uvm_macros.svh"
import spi_test_pkg::*;
module TOP_MODULE();
    bit clk;

    // clock generation
    initial begin
    forever
    #1 clk = ~clk;
  end

  //instantioation of DUT, TEST and interfce
  SPI_IF if_inst(clk);
  SPI_GM_IF gm_if_inst(clk);

  SPI_SLAVE DUT (if_inst);
  SPI_REF GM_DUT(gm_if_inst);

  // run the testbench using global task run_test
  initial begin

    // Set the virtual interface for the uvm test
    uvm_config_db#(virtual SPI_IF)::set(null , "uvm_test_top" , "spi" , if_inst);
    uvm_config_db#(virtual SPI_GM_IF)::set(null , "uvm_test_top" , "spi_ref" , gm_if_inst);
    run_test("spi_test");
  end
endmodule

```

RAM

➢ Sequence Item package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_seq_item.sv > ...

```
package ram_seq_item_pkg;

import uvm_pkg::*;
`include "uvm_macros.svh"

typedef enum bit [1:0] { WRITE_ADDR, WRITE_DATA, READ_ADDR, READ_DATA } cmd_e ;
typedef enum bit [1:0] { WRITE_ONLY , READ_ONLY , READ_WRITE} seq_mode_e ;

class ram_seq_item extends uvm_sequence_item;

    `uvm_object_utils(ram_seq_item)

    rand logic rst_n;
    rand logic rx_valid;
    rand cmd_e command;
    rand logic [7:0] data;

    logic tx_valid ;
    logic [7:0] dout ;

    logic tx_valid_ref ;
    logic [7:0] dout_ref ;

    rand seq_mode_e seq_mode ;
    cmd_e last_command ;


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


    constraint c_reset { rst_n dist {0 := 5 , 1 := 95}; }
    constraint c_rx_valid { rx_valid dist {0 := 5 , 1:= 95}; }

    constraint c_seq {
        if (seq_mode == WRITE_ONLY)
        {
            if (last_command == WRITE_ADDR)
                command inside {WRITE_ADDR, WRITE_DATA};
            else
                command == WRITE_ADDR;
        }
        else if (seq_mode == READ_ONLY)
        {
            if (last_command == READ_ADDR)
                command == READ_DATA ;
            else if (last_command == READ_DATA)
                command == READ_ADDR;
        }
        else if (seq_mode == READ_WRITE)
        {
            if (last_command == WRITE_ADDR)
                command inside {WRITE_ADDR, WRITE_DATA};
            else if (last_command == WRITE_DATA)
                command dist {READ_ADDR := 60 , WRITE_ADDR := 40};
            else if (last_command == READ_ADDR)
                command inside {READ_ADDR, READ_DATA};
            else if (last_command == READ_DATA)
                command dist {WRITE_ADDR := 60 , READ_ADDR := 40};
        }
    }


    function void post_randomize();
        last_command = command ;
    endfunction

    function string convert2string();
        return $sformatf("%s  rx_valid = 0b%0b , command = 0b%0b  , data = 0b%0b " ,
                         super.convert2string() ,  rx_valid, command, data ) ;
    endfunction

    function string convert2string_stimulus();
        return $sformatf(" rx_valid = 0b%0b , command = 0b%0b  , data = 0b%0b " ,
                            rx_valid, command, data ) ;
    endfunction

endclass

endpackage
```

➢ Reset Sequence package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_rst_seq.sv > ...

```
package ram_reset_seq_pkg;

import ram_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_reset_seq extends uvm_sequence #(ram_seq_item);

    `uvm_object_utils(ram_reset_seq)

    ram_seq_item seq_item ;

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

    task body ;

        seq_item = ram_seq_item::type_id::create("seq_item");

        start_item (seq_item)  ;

        seq_item.rst_n = 0 ;
        seq_item.rx_valid = 0 ;
        seq_item.command = cmd_e'(0) ;
        seq_item.data = 0 ;

        finish_item(seq_item);

    endtask

endclass

endpackage
```

➢ Write Only Sequence package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_write_only_seq.sv > ...

```
package ram_write_seq_pkg;

import ram_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_write_only_seq extends uvm_sequence #(ram_seq_item);

    `uvm_object_utils(ram_write_only_seq)

    ram_seq_item seq_item ;

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

    task body;

        seq_item = ram_seq_item::type_id::create("seq_item");
        repeat (5000) begin

            start_item(seq_item);

            assert(seq_item.randomize() with { seq_mode == WRITE_ONLY ;} ) ;

            finish_item(seq_item);

        end

    endtask

endclass

endpackage
```

➢ Read Only Sequence package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_read_only_seq.sv > ...

```
package ram_read_seq_pkg;

import ram_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_read_only_seq extends uvm_sequence #(ram_seq_item);

    `uvm_object_utils(ram_read_only_seq)

    ram_seq_item seq_item ;

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

    task body;

        seq_item = ram_seq_item::type_id::create("seq_item");
        repeat (5000) begin

            start_item(seq_item);

            assert(seq_item.randomize() with { seq_mode == READ_ONLY ; } ) ;

            finish_item(seq_item);

        end

    endtask

endclass

endpackage
```

➢ Read/Write Sequence package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_read_write_seq.sv > ...

```
package ram_read_write_seq_pkg;

import ram_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_read_write_seq extends uvm_sequence #(ram_seq_item);

    `uvm_object_utils(ram_read_write_seq)

    ram_seq_item seq_item ;

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

    task body;

        seq_item = ram_seq_item::type_id::create("seq_item");
        repeat (5000) begin

            start_item(seq_item);

            assert(seq_item.randomize() with { seq_mode == READ_WRITE ;} ) ;

            finish_item(seq_item);
     end

    endtask

endclass

endpackage
```

➢ Sequencer package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_sequencer.sv > ...

```
package ram_sequencer_pkg ;

import ram_seq_item_pkg::*;
import uvm_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

```

➢ Design Interface

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_if.sv > ...

```
interface ram_if (clk);

 input clk ;
 logic rst_n , rx_valid , tx_valid ;
 logic [9:0] din ;
 logic [7:0] dout ;

 endinterface : ram_if

```

➢ Golden Model Interface

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_GM_if.sv > ...

```
interface ram_GM_if (clk);

input clk ;
logic rst_n , rx_valid , tx_valid ;
logic [9:0] din ;
logic [7:0] dout ;

endinterface : ram_GM_if

```

➢ Driver Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_driver.sv > ...

```
package ram_driver_pkg;

import ram_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_driver extends uvm_driver #(ram_seq_item) ;

    `uvm_component_utils(ram_driver)

    virtual ram_if ram_driver_vif;
    virtual ram_GM_if ram_GM_driver_vif;
    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) ;

            ram_driver_vif.rst_n = stim_seq_item.rst_n ;
            ram_driver_vif.rx_valid = stim_seq_item.rx_valid ;
            ram_driver_vif.din[9:8] = stim_seq_item.command ;
            ram_driver_vif.din[7:0] = stim_seq_item.data ;

            if (ram_GM_driver_vif != null) begin
                ram_GM_driver_vif.rst_n = stim_seq_item.rst_n ;
                ram_GM_driver_vif.rx_valid = stim_seq_item.rx_valid ;
                ram_GM_driver_vif.din[9:8] = stim_seq_item.command ;
                ram_GM_driver_vif.din[7:0] = stim_seq_item.data ;
            end

            @(negedge ram_driver_vif.clk ) ;

            seq_item_port.item_done() ;
            `uvm_info("run_phase", stim_seq_item.convert2string_stimulus() , UVM_HIGH)
        end

    endtask

endclass

endpackage
```

➢ Monitor Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_monitor.sv > ...

```
package ram_monitor_pkg;

import ram_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_monitor extends uvm_monitor;
    `uvm_component_utils(ram_monitor)

    virtual ram_if ram_monitor_vif;
    virtual ram_GM_if ram_GM_monitor_vif;
    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);
    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
            rsp_seq_item = ram_seq_item::type_id::create("rsp_seq_item");

            @(negedge ram_monitor_vif.clk ) ;
            rsp_seq_item.rst_n = ram_monitor_vif.rst_n ;
            rsp_seq_item.rx_valid = ram_monitor_vif.rx_valid ;
            rsp_seq_item.command = cmd_e'(ram_monitor_vif.din[9:8]) ;
            rsp_seq_item.data = ram_monitor_vif.din[7:0] ;
            rsp_seq_item.tx_valid = ram_monitor_vif.tx_valid ;
            rsp_seq_item.dout = ram_monitor_vif.dout ;

            if (ram_GM_monitor_vif != null) begin
                rsp_seq_item.tx_valid_ref = ram_GM_monitor_vif.tx_valid ;
                rsp_seq_item.dout_ref = ram_GM_monitor_vif.dout ;
            end

            mon_ap.write(rsp_seq_item) ;
            `uvm_info("run_phase", rsp_seq_item.convert2string() , UVM_HIGH)
        end
    endtask

endclass

endpackage
```

➢ Config Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_config_obj.sv > ...

```
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 ram_config_vif;
    virtual ram_GM_if ram_GM_config_vif;

    uvm_active_passive_enum is_active;

    function new(string name = "ram_config");
        super.new(name);
        is_active = UVM_ACTIVE;
    endfunction

endclass

endpackage

```

➢ Agent Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_agent.sv > ...

```
package ram_agent_pkg;

import ram_seq_item_pkg::*;
import ram_sequencer_pkg::*;
import ram_driver_pkg::*;
import ram_monitor_pkg::*;
import ram_config_pkg::*;

import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_agent extends uvm_agent ;
    `uvm_component_utils(ram_agent)

    ram_sequencer sqr ;
    ram_driver drv ;
    ram_monitor mon ;
    ram_config cfg ;

    uvm_analysis_port #(ram_seq_item) agt_ap ;

    function new(string name = "ram_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 #(ram_config)::get(this , "" , "CFG_RAM" , cfg )) begin
            `uvm_fatal("build_phase", "Unable to get configuration object");
        end

        if (cfg.is_active == UVM_ACTIVE) begin
            sqr = ram_sequencer::type_id::create("sqr", this);
            drv = ram_driver::type_id::create("drv", this);
        end

        mon =  ram_monitor::type_id::create("mon",this);
        agt_ap = new("agt_ap",this) ;

    endfunction

    function void connect_phase (uvm_phase phase);
        super.connect_phase(phase);
        if (cfg.is_active == UVM_ACTIVE) begin
            drv.ram_driver_vif = cfg.ram_config_vif;
            drv.ram_GM_driver_vif = cfg.ram_GM_config_vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
        end

        mon.ram_monitor_vif =  cfg.ram_config_vif;
        mon.ram_GM_monitor_vif = cfg.ram_GM_config_vif;
        mon.mon_ap.connect(agt_ap) ;

    endfunction

endclass

endpackage

```

➢ Scoreboard Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_scoreboard.sv > ...

```
package ram_scoreboard_pkg;

import ram_seq_item_pkg::*;
import uvm_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;

    int error_count = 0;
    int correct_count = 0;

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

    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);
        forever begin
            sb_fifo.get(seq_item_sb);
            if ((seq_item_sb.dout != seq_item_sb.dout_ref ) || (seq_item_sb.tx_valid != seq_item_sb.tx_valid_ref)) begin
                `uvm_error("run_phase", $sformatf("Comparsion failed, Transaction received by the DUT:%s While the reference out: 0b%0b" ,
                seq_item_sb.convert2string(),seq_item_sb.dout_ref));
                error_count++;
            end
            else begin
                `uvm_info("run_phase", $sformatf("Correct ram out: %s ", seq_item_sb.convert2string()), UVM_HIGH);
                correct_count++;
            end
        end
    endtask

    function void report_phase(uvm_phase phase);
        super.report_phase(phase);
        `uvm_info("report_phase", $sformatf("Total successful transactions: %0d", correct_count), UVM_MEDIUM);
        `uvm_info("report_phase", $sformatf("Total failed transactions: %0d",error_count), UVM_MEDIUM);
    endfunction

endclass

endpackage
```

➢ Coverage Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_coverage.sv > ...

```
package ram_coverage_pkg;

import ram_seq_item_pkg::*;
import uvm_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 declaration
    covergroup cvr_gp;

        command_cp: coverpoint seq_item_cov.command {
            bins write_address = {WRITE_ADDR};
            bins write_data = {WRITE_DATA};
            bins read_address = {READ_ADDR};
            bins read_data = {READ_DATA};
            bins write_data_after_address = ( WRITE_ADDR => WRITE_DATA );
            bins read_data_after_address = ( READ_ADDR => READ_DATA );
            bins transition = (WRITE_ADDR => WRITE_DATA => READ_ADDR => READ_DATA );
        }

        rx_valid_cp: coverpoint seq_item_cov.rx_valid ;

        tx_valid_cp: coverpoint seq_item_cov.tx_valid ;


        rx_valid_with_commands : cross command_cp , rx_valid_cp {
            ignore_bins rx_valid_zero = binsof (rx_valid_cp) intersect {0} ;
        }

        tx_valid_with_read_data : cross command_cp, tx_valid_cp {
            option.cross_auto_bin_max = 0 ;
            bins tx_valid_rd = binsof(command_cp.read_data) && binsof(tx_valid_cp) intersect {1};
        }

    endgroup

    function new(string name = "ram_coverage", uvm_component parent = null) ;
        super.new(name, parent);
        cvr_gp = 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);
            cvr_gp.sample();
        end
    endtask

endclass

endpackage
```

➢ Env Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_env.sv > ...

```
package ram_env_pkg;

import ram_agent_pkg::*;
import ram_scoreboard_pkg::*;
import ram_coverage_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_env extends uvm_env;
   `uvm_component_utils(ram_env)

   ram_agent agt;
   ram_scoreboard sb;
   ram_coverage cov;

   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: build_phase

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

endclass

endpackage
```

➢ Test Package

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_test.sv > ...

```
package ram_test_pkg;

import ram_config_pkg::*;
import ram_env_pkg::*;
import ram_reset_seq_pkg::*;
import ram_write_seq_pkg::*;
import ram_read_seq_pkg::*;
import ram_read_write_seq_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class ram_test extends uvm_test;

    `uvm_component_utils(ram_test)
    ram_env env;
    ram_config ram_cfg;
    virtual ram_if ram_test_vif;
    virtual ram_GM_if ram_GM_test_vif;

    ram_reset_seq rst_seq ;
    ram_write_only_seq write_seq ;
    ram_read_only_seq read_seq ;
    ram_read_write_seq read_write_seq ;

    function new(string name = "ram_test", 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("ram_cfg");

        rst_seq = ram_reset_seq::type_id:: create("rst_seq");
        write_seq = ram_write_only_seq::type_id::create("write_seq");
        read_seq = ram_read_only_seq::type_id:: create("read_seq");
        read_write_seq = ram_read_write_seq::type_id::create("read_write_seq");


        if (!uvm_config_db #(virtual ram_if)::get(this , "" , "RAM_IF" , ram_cfg.ram_config_vif )) begin
            `uvm_fatal("build_phase", "Virtual interface not found ");
        end

        //FOR GOLDEN MODEL
        if (!uvm_config_db #(virtual ram_GM_if)::get(this , "" , "RAM_GM_IF" , ram_cfg.ram_GM_config_vif )) begin
            `uvm_fatal("build_phase", "Virtual interface not found ");
        end

        uvm_config_db#(ram_config)::set(this, "*", "CFG_RAM", ram_cfg);

    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);
        //reset sequence
        `uvm_info("run_phase", "Reset Asserted", UVM_LOW)
        rst_seq.start(env.agt.sqr);
        `uvm_info("run_phase", "Reset Deasserted", UVM_LOW)

        //main sequence
        `uvm_info("run_phase", "Stimulus Generation Started", UVM_LOW)
        write_seq.start(env.agt.sqr);
        read_seq.start(env.agt.sqr);
        read_write_seq.start(env.agt.sqr);
        `uvm_info("run_phase", "Stimulus Generation Ended", UVM_LOW)
        phase.drop_objection(this);
    endtask

endclass: ram_test

endpackage
```

➢ Design Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM.v > ...

```
module RAM (din,clk,rst_n,rx_valid,dout,tx_valid);

input      [9:0] din;
input            clk, rst_n, rx_valid;

output reg [7:0] dout;
output reg       tx_valid;

reg [7:0] mem  [255:0];  //mem not MEM//

reg [7:0] Rd_Addr, Wr_Addr;

always @(posedge clk) begin
    if (~rst_n) begin
        dout <= 0;
        tx_valid <= 0;
        Rd_Addr <= 0;
        Wr_Addr <= 0;
    end
    else  begin   //begin..end//
        if (rx_valid) begin
            tx_valid <= 0 ;
            case (din[9:8])
                2'b00 : begin
                    Wr_Addr <= din[7:0];
                    tx_valid <= 0 ;
                end
                2'b01 :begin
                    mem [Wr_Addr] <= din[7:0];
                    tx_valid <= 0 ;
                end
                2'b10 :begin
                    Rd_Addr <= din[7:0];
                    tx_valid <= 0 ;
                end
                2'b11 : begin
                    dout <= mem [Rd_Addr]; //*****fixed this bug here, from write address to read address*****//
                    tx_valid <= 1 ;     //**put the tx_valid in case statement**//
                end
                default : dout<= 0 ;

            endcase
        end
    end
end

endmodule

```

➢ Golden Model Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_GM.v > ...

```
module ram_ref ( clk , rst_n , rx_valid , din , tx_valid , dout );

parameter MEM_DEPTH = 256 ;
parameter ADDR_SIZE = 8 ;

input clk , rst_n , rx_valid ;
input [9:0] din ;
output reg tx_valid ;
output reg [7:0] dout ;

reg [7:0] mem [MEM_DEPTH-1:0] ;

reg [ADDR_SIZE-1:0] addr_wr , addr_rd ;

wire [1:0] command = din [9:8] ;
wire [7:0] data = din [7:0] ;


always @( posedge clk ) begin
    if (!rst_n) begin
        addr_wr <= 0 ;
        addr_rd <= 0 ;
        dout <= 0 ;
        tx_valid <= 0 ;
    end
    else begin
        if (rx_valid) begin
            tx_valid <= 0 ;
            case (command)
                2'b00: addr_wr <= data ;
                2'b01: mem[addr_wr] <= data ;
                2'b10: addr_rd <= data ;
                default : begin
                    dout <= mem[addr_rd] ;
                    tx_valid <= 1 ;
                end
            endcase
        end
    end
end

endmodule

```

➢ Assertions Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_sva.sv > ...

```
module ram_sva( clk , rst_n , rx_valid , din , tx_valid , dout );


input clk , rst_n , rx_valid ;
input [9:0] din ;
input tx_valid ;
input [7:0] dout ;


property reset_outputs_low;
    @(posedge clk)
        (!rst_n) |=> (tx_valid == 0 && dout == 0);
endproperty
assert property(reset_outputs_low)
    else $error("ERROR : reset operation");
cover property(reset_outputs_low);

property tx_valid_low ;
    @(posedge clk) disable iff(!rst_n)
        (rx_valid && (din[9:8]== 2'b00 || din[9:8]== 2'b01 || din[9:8]== 2'b10) ) |=> (tx_valid == 0);
endproperty
assert property(tx_valid_low)
    else $error("ERROR : tx valid during input command phase");
cover property(tx_valid_low);


property tx_valid_high ;
    @(posedge clk) disable iff(!rst_n)
        (din[9:8] == 2'b11) |=> $rose(tx_valid)[->1] ##1 $fell(tx_valid)[->1] ;
endproperty
assert property(tx_valid_high)
    else $error("ERROR : tx valid after read data");
cover property(tx_valid_high);


property write_data_after_address ;
    @(posedge clk) disable iff(!rst_n)
        (din[9:8]== 2'b00) |=> (din[9:8]== 2'b01 [->1] );
endproperty
assert property(write_data_after_address)
    else $error("ERROR : write data after address");
cover property(write_data_after_address);


property read_data_after_address ;
    @(posedge clk) disable iff(!rst_n)
        (din[9:8]== 2'b10) |=> (din[9:8]== 2'b11 [->1] );
endproperty
assert property(read_data_after_address)
    else $error("ERROR : read data fter addressa");
cover property(read_data_after_address);

endmodule
```

➢ Top Code

D: > DigitalCourse > DigitalVerification > SYSTEM_2 > RAM_top.sv > ...

```
import uvm_pkg::*;
`include "uvm_macros.svh"
import ram_test_pkg::*;

module ram_top();

    bit clk, reset;
    initial begin
        forever
        #1 clk = ~clk;
    end

    // Instantiate the interface and DUT
    ram_if ramif (clk);
    RAM DUT (.din(ramif.din) , .clk(ramif.clk) , .rst_n(ramif.rst_n) , .rx_valid(ramif.rx_valid)  , .dout(ramif.dout) , .tx_valid(ramif.tx_valid) );

    ram_GM_if ramGMif (clk);
    ram_ref REF (ramGMif.clk , ramGMif.rst_n , ramGMif.rx_valid , ramGMif.din , ramGMif.tx_valid , ramGMif.dout);

    //for assertion file
    bind RAM ram_sva ram_sva_inst (ramif.clk , ramif.rst_n , ramif.rx_valid , ramif.din , ramif.tx_valid , ramif.dout) ;

    // run test using run_test task
    initial begin
        // Set the virtual interface for the uvm test
        uvm_config_db#(virtual ram_if)::set(null , "uvm_test_top" , "RAM_IF" , ramif ) ;
        uvm_config_db#(virtual ram_GM_if)::set(null , "uvm_test_top" , "RAM_GM_IF" , ramGMif ) ;

        run_test("ram_test");
    end

endmodule
```

Wrapper

➢ Sequence Item package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_seq_item.sv > { } wrapper_seq_item_pkg > wrapper_seq_item

```
package wrapper_seq_item_pkg;

import uvm_pkg::*;
`include "uvm_macros.svh"

    typedef enum bit [1:0] { WRITE_ONLY , READ_ONLY , READ_WRITE} seq_mode_e;
    typedef enum bit [2:0] { WRITE_ADDR = 3'b000, WRITE_DATA = 3'b001, READ_ADDR = 3'b110, READ_DATA = 3'b111 } cmd_e;

    class wrapper_seq_item extends uvm_sequence_item;
    `uvm_object_utils(wrapper_seq_item)

        rand  logic SS_n;
        rand  logic rst_n;
        rand  cmd_e command;
        rand  seq_mode_e seq_mode;
        rand  bit MOSI_arr[10:0];
        bit   MOSI_arr_drv[10:0];
        logic MOSI;
        logic MISO;
        cmd_e last_command;
        logic MISO_ref;

        int read_counter    = 0;
        int normal_counter  = 0;
        int ss_counter      = 0;
        bit read_data_falg  = 0;

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


        constraint c_reset { rst_n dist {0 := 2 , 1 := 98}; }

        constraint c_seq {
            if (seq_mode == WRITE_ONLY)
            {
                if (last_command == WRITE_ADDR)
                    command inside {WRITE_ADDR, WRITE_DATA};
                else
                    command == WRITE_ADDR;
            }
            else if (seq_mode == READ_ONLY)
            {
                if (last_command == READ_ADDR)
                    command == READ_DATA ;
                else if (last_command == READ_DATA)
                    command == READ_ADDR;
            }
            else if (seq_mode == READ_WRITE)
            {
                if (last_command == WRITE_ADDR)
                    command inside {WRITE_ADDR, WRITE_DATA};
                else if (last_command == WRITE_DATA)
                    command dist {READ_ADDR := 60 , WRITE_ADDR := 40};
                else if (last_command == READ_ADDR)
                    command inside {READ_ADDR, READ_DATA};
                else if (last_command == READ_DATA)
                    command dist {WRITE_ADDR := 60 , READ_ADDR := 40};
            }
        }
```

```

        function void post_randomize();
            last_command = command;
            {MOSI_arr[0], MOSI_arr[1], MOSI_arr[2]} = command;

            if (ss_counter == 0)
                SS_n = 1;
            else
                SS_n = 0;

            if ((normal_counter == 0) && (read_counter == 0) && (ss_counter == 0)) begin
                MOSI_arr_drv = MOSI_arr;
            end

            read_data_falg = (({MOSI_arr_drv[0], MOSI_arr_drv[1], MOSI_arr_drv[2]}) == READ_DATA);

            if (read_data_falg) begin
                if (ss_counter < 23)
                    ss_counter++;
                else
                    ss_counter = 0;
            end
            else begin
                if (ss_counter < 13)
                    ss_counter++;
                else
                    ss_counter = 0;
            end

            if (read_counter < 25 && !SS_n && read_data_falg && ss_counter > 2) begin
                if (read_counter < 11)
                    MOSI = MOSI_arr_drv[read_counter++];
                else
                    read_counter++;
            end
            else begin
                read_counter   = 0;
                read_data_falg = 0;
            end

            if (normal_counter < 15 && !SS_n && !read_data_falg && ss_counter > 2) begin
                if (normal_counter < 11)
                    MOSI = MOSI_arr_drv[normal_counter++];
                else
                    normal_counter++;
            end
            else
                normal_counter = 0;

        endfunction

```

```
115
116         function string convert2string();
117             return $sformatf(
118                 "%s reset = %0b, SS_n = %0b, MOSI = %0b, MISO = %0b, command = 0b%0b , MISO_ref = %0b",
119                 super.convert2string(),
120                 rst_n, SS_n, MOSI, MISO, command, MISO_ref
121             );
122         endfunction
123
124
125         function string convert2string_stimulus();
126             return $sformatf(
127                 " reset = %0b, SS_n = %0b, MOSI = %0b, command = %0b,  MISO_ref = %0b",
128                 rst_n, SS_n, MOSI, command, MISO_ref
129             );
130         endfunction
131     endclass
132 endpackage
133
```

➢ Reset Sequence package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_reset_seq.sv > ...

```
1  package wrapper_reset_seq_pkg;
2
3  import wrapper_seq_item_pkg::*;
4  import uvm_pkg::*;
5  `include "uvm_macros.svh"
6
7      class wrapper_reset_seq extends uvm_sequence #(wrapper_seq_item);
8      `uvm_object_utils(wrapper_reset_seq)
9
10         wrapper_seq_item seq_item;
11
12         function new(string name = "wrapper_reset_seq");
13             super.new(name);
14         endfunction
15
16         task body;
17             seq_item = wrapper_seq_item::type_id::create("seq_item");
18             start_item (seq_item);
19             seq_item.rst_n          = 0;
20             seq_item.SS_n           = 0;
21             seq_item.MOSI           = 0;
22             seq_item.MISO           = 0;
23             finish_item(seq_item);
24         endtask
25     endclass
26 endpackage
27
```

➢ Write Only Sequence package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_write_only_seq.sv > ...

```
1  package wrapper_write_seq_pkg;
2
3  import wrapper_seq_item_pkg::*;
4  import uvm_pkg::*;
5  `include "uvm_macros.svh"
6
7      class wrapper_write_only_seq extends uvm_sequence #(wrapper_seq_item);
8      `uvm_object_utils(wrapper_write_only_seq)
9
10         wrapper_seq_item seq_item;
11
12         function new(string name = "wrapper_write_only_seq");
13             super.new(name);
14         endfunction
15
16         task body;
17             seq_item = wrapper_seq_item::type_id::create("seq_item");
18             repeat (5000) begin
19                 start_item(seq_item);
20                 assert(seq_item.randomize() with { seq_mode == WRITE_ONLY; } );
21                 finish_item(seq_item);
22             end
23         endtask
24     endclass
25 endpackage
26
```

➢ Read Only Sequence package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_read_only_seq.sv > ...

```
1  package wrapper_read_seq_pkg;
2
3  import wrapper_seq_item_pkg::*;
4  import uvm_pkg::*;
5  `include "uvm_macros.svh"
6
7      class wrapper_read_only_seq extends uvm_sequence #(wrapper_seq_item);
8      `uvm_object_utils(wrapper_read_only_seq)
9
10         wrapper_seq_item seq_item;
11
12         function new(string name = "wrapper_read_only_seq");
13             super.new(name);
14         endfunction
15
16         task body;
17             seq_item = wrapper_seq_item::type_id::create("seq_item");
18             repeat (5000) begin
19                 start_item(seq_item);
20                 assert(seq_item.randomize() with { seq_mode == READ_ONLY; } );
21                 finish_item(seq_item);
22             end
23         endtask
24     endclass
25 endpackage
26
```

➢ Read/Write Sequence package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_read_write_seq.sv > ...

```
package wrapper_read_write_seq_pkg;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    class wrapper_read_write_seq extends uvm_sequence #(wrapper_seq_item);
    `uvm_object_utils(wrapper_read_write_seq)

        wrapper_seq_item seq_item;

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

        task body;
            seq_item = wrapper_seq_item::type_id::create("seq_item");
            repeat (5000) begin
                start_item(seq_item);
                assert(seq_item.randomize() with { seq_mode == READ_WRITE; } );
                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

➢ Sequencer package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_sequencer.sv > ...

```
package wrapper_sequencer_pkg ;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    class wrapper_sequencer extends uvm_sequencer #(wrapper_seq_item) ;
    `uvm_component_utils(wrapper_sequencer)

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

➢ Design Interface

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_if.sv > ...

```
interface wrapper_if(clk);

  input clk;
  logic rst_n;
  logic MOSI;
  logic MISO;
  logic SS_n;

endinterface
```

➢ Golden Model Interface

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_ref_if.sv > ...

```
interface wrapper_ref_if(clk);

  input clk;
  logic rst_n;
  logic MOSI_ref;
  logic MISO_ref;
  logic SS_n_ref;

endinterface
```

➢ Driver Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_driver.sv > ...

```
package wrapper_driver_pkg;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    class wrapper_driver extends uvm_driver #(wrapper_seq_item);
    `uvm_component_utils(wrapper_driver)

        virtual wrapper_if     wrapper_driver_vif;
        virtual wrapper_ref_if wrapper_ref_driver_vif;
        wrapper_seq_item       stim_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
                stim_seq_item =  wrapper_seq_item::type_id::create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);

                wrapper_driver_vif.rst_n               = stim_seq_item.rst_n;
                wrapper_driver_vif.SS_n                = stim_seq_item.SS_n;
                wrapper_driver_vif.MOSI                = stim_seq_item.MOSI;


                if (wrapper_ref_driver_vif != null) begin
                    wrapper_ref_driver_vif.rst_n               = stim_seq_item.rst_n;
                    wrapper_ref_driver_vif.SS_n_ref            = stim_seq_item.SS_n;
                    wrapper_ref_driver_vif.MOSI_ref            = stim_seq_item.MOSI;
                end
                @(negedge wrapper_driver_vif.clk);
                seq_item_port.item_done();
                `uvm_info("run_phase", stim_seq_item.convert2string_stimulus() , UVM_HIGH)
            end
        endtask
    endclass
endpackage

```

➢ Monitor Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_monitor.sv > ...

```
package wrapper_monitor_pkg;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    class wrapper_monitor extends uvm_monitor;
    `uvm_component_utils(wrapper_monitor)

        virtual wrapper_if      wrapper_monitor_vif;
        virtual wrapper_ref_if  wrapper_ref_monitor_vif;
        wrapper_seq_item        seq_item;

        uvm_analysis_port #(wrapper_seq_item) mon_ap;

        function new(string name = "wrapper_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
                seq_item = wrapper_seq_item::type_id::create("seq_item");

                @(negedge wrapper_monitor_vif.clk);
                seq_item.rst_n        = wrapper_monitor_vif.rst_n;
                seq_item.SS_n         = wrapper_monitor_vif.SS_n;
                seq_item.MOSI         = wrapper_monitor_vif.MOSI;
                seq_item.MISO         = wrapper_monitor_vif.MISO;

                if (wrapper_ref_monitor_vif  != null) begin
                    seq_item.MISO_ref         = wrapper_ref_monitor_vif.MISO_ref;
                end

                mon_ap.write(seq_item);
                `uvm_info("run_phase", seq_item.convert2string() , UVM_HIGH)
            end
        endtask
    endclass
endpackage

```

➢ Config Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_config_obj.sv > ...

```
package wrapper_config_pkg;

 import uvm_pkg::*;
 `include "uvm_macros.svh"

    class wrapper_config extends uvm_object;
     `uvm_object_utils(wrapper_config)

     virtual wrapper_if     wrapper_config_vif;
     virtual wrapper_ref_if wrapper_ref_config_vif;

     uvm_active_passive_enum is_active;

        function new(string name = "wrapper_config");
            super.new(name);
            is_active = UVM_ACTIVE;
        endfunction
    endclass
endpackage

```

➢ Agent Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_agent.sv > ...

```
package wrapper_agent_pkg;

import wrapper_seq_item_pkg::*;
import wrapper_sequencer_pkg::*;
import wrapper_driver_pkg::*;
import wrapper_monitor_pkg::*;
import wrapper_config_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    class wrapper_agent extends uvm_agent;
    `uvm_component_utils(wrapper_agent)

        wrapper_sequencer sqr;
        wrapper_driver    drv;
        wrapper_monitor   mon;
        wrapper_config    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)::get(this , "" , "CFG" , cfg )) begin
                `uvm_fatal("build_phase", "Unable to get configuration object");
            end
            if (cfg.is_active == UVM_ACTIVE) begin
                sqr = wrapper_sequencer::type_id::create("sqr", this);
                drv = wrapper_driver::type_id::create("drv", this);
            end
            mon    =  wrapper_monitor::type_id::create("mon",this);
            agt_ap = new("agt_ap",this);
        endfunction

        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase);
            if (cfg.is_active == UVM_ACTIVE) begin
                drv.wrapper_driver_vif     = cfg.wrapper_config_vif;
                drv.wrapper_ref_driver_vif = cfg.wrapper_ref_config_vif;
                drv.seq_item_port.connect(sqr.seq_item_export);
            end
            mon.wrapper_monitor_vif     = cfg.wrapper_config_vif;
            mon.wrapper_ref_monitor_vif = cfg.wrapper_ref_config_vif;
            mon.mon_ap.connect(agt_ap);
        endfunction
    endclass
endpackage

```

➢ Scoreboard Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_scoreboard.sv > ...

```
package wrapper_scoreboard_pkg;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    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;

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

        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);
            forever begin
                sb_fifo.get(seq_item_sb);
                if ((seq_item_sb.MISO_ref != seq_item_sb.MISO)) begin
                    `uvm_error("run_phase", $sformatf("Comparsion failed, MISO received by the DUT:%0b While the reference MISO: %0b" ,
                    seq_item_sb.MISO, seq_item_sb.MISO_ref));
                    error_count++;
                end
                else begin
                    `uvm_info("run_phase", $sformatf("Correct MISO: %0b  == MISO_ref %0b", seq_item_sb.MISO, seq_item_sb.MISO_ref), UVM_HIGH);
                    correct_count++;
                end
            end
        endtask

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Total successful transactions: %0d", correct_count), UVM_MEDIUM);
            `uvm_info("report_phase", $sformatf("Total failed transactions: %0d",error_count), UVM_MEDIUM);
        endfunction
    endclass
endpackage

```

➢ Coverage Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_coverage.sv > ...

```
package wrapper_coverage_pkg;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

    class wrapper_coverage extends uvm_component;
        `uvm_component_utils(wrapper_coverage)

        uvm_analysis_export #(wrapper_seq_item) cov_export;
        uvm_tlm_analysis_fifo #(wrapper_seq_item) cov_fifo;
        wrapper_seq_item seq_item_cov;

        function new(string name = "wrapper_coverage", uvm_component parent = null) ;
            super.new(name, parent);

        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);
            end
        endtask
    endclass
endpackage

```

➢ Env Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_env.sv > ...

```
package wrapper_env_pkg;

  import wrapper_agent_pkg::*;
  import wrapper_scoreboard_pkg::*;
  import wrapper_coverage_pkg::*;
  import uvm_pkg::*;
  `include "uvm_macros.svh"


  class wrapper_env extends uvm_env;
    `uvm_component_utils(wrapper_env)

    wrapper_agent      agt;
    wrapper_scoreboard sb;
    wrapper_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 = wrapper_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


    function void report_phase(uvm_phase phase);
      super.report_phase(phase);

      `uvm_info("WRAPPER_ENV", "Wrapper Environment completed successfully", UVM_LOW)
    endfunction
  endclass
endpackage

```

➢ Test Package

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_test.sv > ...

```
package wrapper_test_pkg;


  import wrapper_config_pkg::*;
  import wrapper_env_pkg::*;
  import wrapper_reset_seq_pkg::*;
  import wrapper_write_seq_pkg::*;
  import wrapper_read_seq_pkg::*;
  import wrapper_read_write_seq_pkg::*;
  import spi_config_pkg::*;
  import ram_config_pkg::*;
  import spi_env_pkg::*;
  import ram_env_pkg::*;
  import uvm_pkg::*;
  `include "uvm_macros.svh"


  class wrapper_test extends uvm_test;
    `uvm_component_utils(wrapper_test)

    wrapper_env            env;
    wrapper_config         wrapper_cfg;
    virtual wrapper_if     wrapper_test_vif;
    virtual wrapper_ref_if wrapper_ref_test_vif;
    wrapper_reset_seq      rst_seq ;
    wrapper_write_only_seq write_seq ;
    wrapper_read_only_seq  read_seq ;
    wrapper_read_write_seq read_write_seq ;

    ram_config             ram_conf;
    spi_config             spi_conf;
    spi_env                s_env;
    ram_env                r_env;


    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);

      env                = wrapper_env::type_id::create("env",this);
      r_env              = ram_env::type_id::create("r_env",this);
      s_env              = spi_env::type_id::create("s_env",this);
      wrapper_cfg        = wrapper_config::type_id::create("wrapper_cfg");

      rst_seq            = wrapper_reset_seq::type_id:: create("rst_seq");
      write_seq          = wrapper_write_only_seq::type_id::create("write_seq");
      read_seq           = wrapper_read_only_seq::type_id:: create("read_seq");
      read_write_seq     = wrapper_read_write_seq::type_id::create("read_write_seq");

      ram_conf           = ram_config::type_id::create("ram_conf");
      spi_conf           = spi_config::type_id::create("spi_conf");

      spi_conf.is_active    = UVM_PASSIVE;
      ram_conf.is_active    = UVM_PASSIVE;
      wrapper_cfg.is_active = UVM_ACTIVE;


      if (!uvm_config_db #(virtual wrapper_if)::get(this , "" , "wifv" , wrapper_cfg.wrapper_config_vif )) begin
       `uvm_fatal("build_phase", "Virtual interface not found ");
      end

      if (!uvm_config_db #(virtual wrapper_ref_if)::get(this , "" , "wrifv" , wrapper_cfg.wrapper_ref_config_vif )) begin
       `uvm_fatal("build_phase", "Virtual interface not found ");
      end

      uvm_config_db #(wrapper_config)::set(this, "*", "CFG", wrapper_cfg);

      if (!uvm_config_db #(virtual ram_if)::get(this , "" , "RAM_IF" , ram_conf.ram_config_vif )) begin
        `uvm_fatal("build_phase", "Virtual interface not found ");
      end


      if (!uvm_config_db #(virtual ram_GM_if)::get(this , "" , "RAM_GM_IF" , ram_conf.ram_GM_config_vif )) begin
       `uvm_fatal("build_phase", "Virtual interface not found ");
      end

      uvm_config_db#(ram_config)::set(this, "*", "CFG_RAM", ram_conf);

      if (!uvm_config_db #(virtual SPI_IF)::get(this , "" , "spi" , spi_conf.SPI_vif)) begin
        `uvm_fatal("build_phase", "error in getting the data");
      end
      if (!uvm_config_db #(virtual SPI_GM_IF)::get(this , "" , "spi_ref" , spi_conf.SPI_GM_vif)) begin
       `uvm_fatal("build_phase", "error in getting the data");
      end
      uvm_config_db #(spi_config)::set(this , "*" , "CFG_SPI" , spi_conf);
    endfunction


    task run_phase(uvm_phase phase);
      super.run_phase(phase);
      phase.raise_objection(this);

      `uvm_info("wrapper_test", "Starting all sequences in parallel", UVM_LOW)

      //reset sequence
      `uvm_info("run_phase", "Reset Asserted", UVM_LOW)
      rst_seq.start(env.agt.sqr);
      `uvm_info("run_phase", "Reset Deasserted", UVM_LOW)

      //main sequence
      `uvm_info("run_phase", "Stimulus Generation Started", UVM_LOW)
      write_seq.start(env.agt.sqr);
      read_seq.start(env.agt.sqr);
      read_write_seq.start(env.agt.sqr);
      `uvm_info("run_phase", "Stimulus Generation Ended", UVM_LOW)
      `uvm_info("wrapper_test", "All sequences completed successfully", UVM_LOW)

      phase.drop_objection(this);
    endtask
  endclass
endpackage
```

➢ Design Code

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper.sv > ...

```
module WRAPPER (clk, rst_n, MOSI, MISO, SS_n);


  input  bit   clk;
  input  logic rst_n;
  input  logic MOSI;
  output logic MISO;
  input  logic SS_n;

  logic tx_valid;
  logic rx_valid;
  logic [9:0] rx_data;
  logic [7:0] tx_data;


  SPI_SLAVE        SLAVE_instance (
    .clk           (clk),
    .MOSI          (MOSI),
    .MISO          (MISO),
    .SS_n          (SS_n),
    .rst_n         (rst_n),
    .rx_data       (rx_data),
    .rx_valid      (rx_valid),
    .tx_data       (tx_data),
    .tx_valid      (tx_valid)
  );


  RAM              RAM_instance (
    .clk           (clk),
    .rst_n         (rst_n),
    .tx_valid      (tx_valid),
    .dout          (tx_data),
    .din           (rx_data),
    .rx_valid      (rx_valid)
  );


  `ifdef SIM

  property reset_output_low;
      @(posedge clk)
          (!rst_n) |=> (MISO== 0);
  endproperty
  assert property(reset_output_low)
      else $error("ERROR : reset operation");
  cover property(reset_output_low);


  property MISO_stable;
      @(posedge clk) disable iff (!rst_n)
        (!tx_valid) |=> $stable(MISO)[->1];
  endproperty
  assert property(MISO_stable)
      else $error("ERROR : MISO stability");
  cover property(MISO_stable);



  `endif

endmodule
```

➢ Golden Model Code

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_ref.sv > ...

```
module wrapper_ref (clk, rst_n_ref,MOSI_ref, MISO_ref, SS_n_ref);


  input  bit   clk;
  input  logic rst_n_ref;
  input  logic MOSI_ref;
  output logic MISO_ref;
  input  logic SS_n_ref;


  wire  tx_valid_ref;
  logic rx_valid_ref;
  logic [9:0] rx_data_ref;
  logic [7:0] tx_data_ref;

  SPI_REF          SLAVE_ref_instance (
    .clk           (clk),
    .MISO          (MISO_ref),
    .MOSI          (MOSI_ref),
    .SS_n          (SS_n_ref),
    .rst_n         (rst_n_ref),
    .tx_data       (tx_data_ref),
    .tx_valid      (tx_valid_ref),
    .rx_data       (rx_data_ref),
    .rx_valid      (rx_valid_ref)
  );


  ram_ref          RAM_ref_instance   (
    .clk           (clk),
    .rst_n         (rst_n_ref),
    .din           (rx_data_ref),
    .tx_valid      (tx_valid_ref),
    .dout          (tx_data_ref),
    .rx_valid      (rx_valid_ref)
  );
endmodule

```

➢ Top Code

D: > DigitalCourse > DigitalVerification > UVM_Project > Wrapper > wrapper_top.sv > wrapper_top > if_inst

```
import uvm_pkg::*;
`include "uvm_macros.svh"
import wrapper_test_pkg::*;

module wrapper_top(clk);

  input bit clk;
  initial begin
   forever #1 clk = ~clk;
  end



  wrapper_if            wif        (clk);
  wrapper_ref_if        wrif       (clk);
  ram_GM_if             ramGMif    (clk);
  ram_if                ramif      (clk);
  SPI_IF                if_inst    (clk);
  SPI_GM_IF             gm_if_inst (clk);

  WRAPPER               dut (
    .clk                (clk),
    .MISO               (wif.MISO),
    .rst_n              (wif.rst_n),
    .SS_n               (wif.SS_n),
    .MOSI               (wif.MOSI)
  );

  wrapper_ref           dut_ref (
    .clk                (clk),
    .MISO_ref           (wrif.MISO_ref),
    .rst_n_ref          (wrif.rst_n),
    .SS_n_ref           (wrif.SS_n_ref),
    .MOSI_ref           (wrif.MOSI_ref)
  );
```

```
  /*bind WRAPPER wrapper_sva u_wrapper_sva (
    .clk                (clk),
    .rst_n              (wif.rst_n),
    .MISO               (wif.MISO),
    .rx_data            (wif.rx_data),
    .MOSI               (wif.MOSI),
    .SS_n               (wif.SS_n)
  );

  bind SPI_SLAVE SPI_ASSIRTION assert_inst (
    .clk                (wif.clk),
    .rst_n              (wif.rst_n),
    .MOSI               (wif.MOSI),
    .MISO               (wif.MISO),
    .rx_valid           (wif.rx_valid_spi),
    .rx_data            (wif.rx_data),
    .SS_n               (wif.SS_n)
  );
 *///already in spi design cause thay depend on internal sigals

  bind RAM ram_sva      ram_sva_inst (
    .clk                (dut.clk),
    .rst_n              (dut.rst_n),
    .rx_valid           (dut.rx_valid),
    .din                (dut.rx_data),
    .tx_valid           (dut.tx_valid),
    .dout               (dut.tx_data)
  );

  assign if_inst.rst_n              = dut.rst_n;
  assign ramif.rst_n                = dut.rst_n;
  assign gm_if_inst.rst_n           = dut_ref.rst_n_ref;
  assign ramGMif.rst_n              = dut_ref.rst_n_ref;

  assign if_inst.SS_n               = dut.SS_n;
  assign if_inst.MOSI               = dut.MOSI;
  assign gm_if_inst.MOSI            = dut_ref.MOSI_ref;
  assign gm_if_inst.SS_n            = dut_ref.SS_n_ref;
  assign if_inst.rx_data            = dut.rx_data;

  assign if_inst.tx_valid           = dut.tx_valid;
  assign if_inst.tx_data            = dut.tx_data;
  assign gm_if_inst.tx_valid        = dut.tx_valid;
  assign gm_if_inst.tx_data         = dut.tx_data;
  assign ramif.rx_valid             = dut.rx_valid;
  assign ramif.din                  = dut.rx_data;
  assign ramGMif.rx_valid           = dut.rx_valid;
  assign ramGMif.din                = dut.rx_data;
  assign ramif.tx_valid             = dut.tx_valid;
```

```


  initial begin
    uvm_config_db#(virtual wrapper_ref_if)::set(null, "uvm_test_top", "wrifv", wrif);
    uvm_config_db#(virtual wrapper_if)::set(null, "uvm_test_top", "wifv", wif);
    uvm_config_db#(virtual ram_if)::set(null, "uvm_test_top", "RAM_IF", ramif);
    uvm_config_db#(virtual ram_GM_if)::set(null, "uvm_test_top", "RAM_GM_IF", ramGMif);
    uvm_config_db#(virtual SPI_IF)::set(null, "uvm_test_top", "spi", if_inst);
    uvm_config_db#(virtual SPI_GM_IF)::set(null, "uvm_test_top", "spi_ref", gm_if_inst);
    run_test("wrapper_test");
  end
endmodule
```

Code Coverage Report

SPI Slave

```
================================================================================
=== Instance: /TOP_MODULE/DUT
=== Design Unit: work.SPI_SLAVE
================================================================================

Assertion Coverage:
    Assertions                      14        14         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/TOP_MODULE/DUT/assert__p_ss_N_normal
                     SPI.sv(264)                         0          1
/TOP_MODULE/DUT/assert__p_ss_N_rd_data
                     SPI.sv(261)                         0          1
/TOP_MODULE/DUT/assert__p_rx_valid_rd_data
                     SPI.sv(258)                         0          1
/TOP_MODULE/DUT/assert__p_rx_valid_rd_addr
                     SPI.sv(255)                         0          1
/TOP_MODULE/DUT/assert__p_rx_valid_wr_data
                     SPI.sv(252)                         0          1
/TOP_MODULE/DUT/assert__p_rx_valid_wr_addr
                     SPI.sv(249)                         0          1
/TOP_MODULE/DUT/assert__p_readdata_to_IDLE
                     SPI.sv(246)                         0          1
/TOP_MODULE/DUT/assert__p_readadd_to_IDLE
                     SPI.sv(243)                         0          1
/TOP_MODULE/DUT/assert__p_WRITE_to_IDLE
                     SPI.sv(240)                         0          1
/TOP_MODULE/DUT/assert__p_chk_to_READ_DATA
                     SPI.sv(237)                         0          1
/TOP_MODULE/DUT/assert__p_chk_to_READ_ADDRESS
                     SPI.sv(234)                         0          1
/TOP_MODULE/DUT/assert__p_chk_to_WRITE
                     SPI.sv(231)                         0          1
/TOP_MODULE/DUT/assert__p_IDLE_to_chk
                     SPI.sv(228)                         0          1
/TOP_MODULE/DUT/assert__p_rst
                     SPI.sv(225)                         0          1
                     SPI.sv(254)                         0          1
```

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        39        39         0   100.00%

================================Branch Details================================

Branch Coverage for instance /TOP_MODULE/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File SPI.sv
------------------------------------IF Branch------------------------------------
    10                                     3033     Count coming in to IF
    10             1                        208             if (~spi_if.rst_n) begin
    13             1                       2825             else begin
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------CASE Branch-----------------------------------
    20                                     6809     Count coming in to CASE
    21             1                       1489                 spi_if.IDLE: begin
    28             1                       1121                 spi_if.CHK_CMD: begin
    44             1                       1843                 spi_if.WRITE: begin
    51             1                       1346                 spi_if.READ_ADDRESS: begin
    58             1                       1009                 spi_if.READ_DATA: begin
    65             1                          1                 default: ns = spi_if.IDLE;
Branch totals: 6 hits of 6 branches = 100.00%

------------------------------------IF Branch------------------------------------
    22                                     1489     Count coming in to IF
    22             1                        608                     if (spi_if.SS_n)
    24             1                        881                     else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    29                                     1121     Count coming in to IF
    29             1                          9                     if (spi_if.SS_n)
    31             1                       1112                     else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    32                                     1112     Count coming in to IF
    32             1                        534                         if (~spi_if.MOSI)
    34             1                        578                         else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    35                                      578     Count coming in to IF
```

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                       4         4         0   100.00%

================================Condition Details================================

Condition Coverage for instance /TOP_MODULE/DUT --

  File SPI.sv
----------------Focused Condition View-------------------
Line       90 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  ----------------------  --------------
  (counter > 0)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (counter > 0)_0       -
  Row   2:          1  (counter > 0)_1       -

----------------Focused Condition View-------------------
Line       100 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  ----------------------  --------------
  (counter > 0)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (counter > 0)_0       -
  Row   2:          1  (counter > 0)_1       -

----------------Focused Condition View-------------------
Line       113 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  ----------------------  --------------
  (counter > 0)         Y
```

```
Directive Coverage:
    Directives                      14        14         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                 Design Design   Lang File(Line)      Hits Status
                                     Unit   UnitType
--------------------------------------------------------------------------------------------
/TOP_MODULE/DUT/cover__p_ss_N_normal     SPI_SLAVE Verilog  SVA  SPI.sv(293)     3965 Covered
/TOP_MODULE/DUT/cover__p_ss_N_rd_data    SPI_SLAVE Verilog  SVA  SPI.sv(292)     1026 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_rd_data
                                     SPI_SLAVE Verilog  SVA  SPI.sv(291)       63 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_rd_addr
                                     SPI_SLAVE Verilog  SVA  SPI.sv(290)        1 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_wr_data
                                     SPI_SLAVE Verilog  SVA  SPI.sv(289)        2 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_wr_addr
                                     SPI_SLAVE Verilog  SVA  SPI.sv(288)        2 Covered
/TOP_MODULE/DUT/cover__p_readdata_to_IDLE
                                     SPI_SLAVE Verilog  SVA  SPI.sv(287)      110 Covered
/TOP_MODULE/DUT/cover__p_readadd_to_IDLE SPI_SLAVE Verilog  SVA  SPI.sv(286)      181 Covered
/TOP_MODULE/DUT/cover__p_WRITE_to_IDLE   SPI_SLAVE Verilog  SVA  SPI.sv(285)      266 Covered
/TOP_MODULE/DUT/cover__p_chk_to_READ_DATA
                                     SPI_SLAVE Verilog  SVA  SPI.sv(284)      156 Covered
/TOP_MODULE/DUT/cover__p_chk_to_READ_ADDRESS
                                     SPI_SLAVE Verilog  SVA  SPI.sv(283)      226 Covered
/TOP_MODULE/DUT/cover__p_chk_to_WRITE    SPI_SLAVE Verilog  SVA  SPI.sv(282)      336 Covered
/TOP_MODULE/DUT/cover__p_IDLE_to_chk     SPI_SLAVE Verilog  SVA  SPI.sv(281)      756 Covered
/TOP_MODULE/DUT/cover__p_rst             SPI_SLAVE Verilog  SVA  SPI.sv(280)      208 Covered
Statement Coverage:
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      41        41         0   100.00%

================================Statement Details================================

Statement Coverage for instance /TOP_MODULE/DUT --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File SPI.sv
    1                                                module SPI_SLAVE (SPI_IF spi_if);
    2
    3                                                    reg [3:0] counter;
    4                                                    reg       received_address;
    5
    6                                                    // state registers
    7                                                    reg [2:0] cs, ns;
    8
    9               1                     3033           always @(posedge spi_if.clk) begin
    10                                                       if (~spi_if.rst_n) begin
    11              1                      208                   cs <= spi_if.IDLE;
    12                                                       end
    13                                                       else begin
    14              1                     2825                   cs <= ns;
    15                                                       end
    16                                                   end
    17
    18                                                   // next state logic
    19              1                        1           always @(*) begin
    20                                                       case (cs)
    21                                                           spi_if.IDLE: begin
    22                                                               if (spi_if.SS_n)
    23              1                      608                           ns = spi_if.IDLE;
    24                                                               else
    25              1                      881                           ns = spi_if.CHK_CMD;
    26                                                           end
    27
    28                                                           spi_if.CHK_CMD: begin
    29                                                               if (spi_if.SS_n)
    30              1                        9                           ns = spi_if.IDLE;
    31                                                               else begin
    32                                                                   if (~spi_if.MOSI)

Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         22        22         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /TOP_MODULE/DUT --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                   counter[3-0]           1           1      100.00
                                        cs[2-0]           1           1      100.00
                                        ns[2-0]           1           1      100.00
                                received_address          1           1      100.00

Total Node Count     =         11
Toggled Node Count   =         11
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (22 of 22 bins)
```

RAM

```
================================================================================
=== Instance: /\ram_top#DUT
=== Design Unit: work.RAM
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         8         0   100.00%

================================Branch Details================================

Branch Coverage for instance /\ram_top#DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File RAM.v
------------------------------------IF Branch------------------------------------
    14                                    15001     Count coming in to IF
    14              1                       764         if (~rst_n) begin
    20              1                     14237         else  begin   //begin..end
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    22                                    14237     Count coming in to IF
    22              1                     13566             if (rx_valid) begin
                                            671     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------CASE Branch----------------------------------
    23                                    13566     Count coming in to CASE
    24              1                      4578                     2'b00 : begin
    27              1                      2256                     2'b01 :begin
    30              1                      3743                     2'b10 :begin
    33              1                      2989                     2'b11 : begin
Branch totals: 4 hits of 4 branches = 100.00%
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      11        11         0   100.00%

================================Statement Details================================

Statement Coverage for instance /\ram_top#DUT  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File RAM.v
    1                                                module RAM (din,clk,rst_n,rx_valid,dout,tx_valid);
    2
    3                                                input      [9:0] din;
    4                                                input            clk, rst_n, rx_valid;
    5
    6                                                output reg [7:0] dout;
    7                                                output reg       tx_valid;
    8
    9                                                reg [7:0] mem  [255:0];  //mem not MEM
    10
    11                                               reg [7:0] Rd_Addr, Wr_Addr;
    12
    13              1                        15001   always @(posedge clk) begin
    14                                                   if (~rst_n) begin
    15              1                          764           dout <= 0;
    16              1                          764           tx_valid <= 0;
    17              1                          764           Rd_Addr <= 0;
    18              1                          764           Wr_Addr <= 0;
    19                                                   end
    20                                                   else  begin   //begin..end
    21              1                        14237           tx_valid <= 0 ;
    22                                                       if (rx_valid) begin
    23                                                           case (din[9:8])
    24                                                               2'b00 : begin
    25              1                         4578                       Wr_Addr <= din[7:0];
    26                                                               end
    27                                                               2'b01 :begin
    28              1                         2256                       mem [Wr_Addr] <= din[7:0];
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         76        76         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /\ram_top#DUT  --

                                      Node      1H->0L      0L->1H  "Coverage"
                                     ----------------------------------------
                                Rd_Addr[7-0]           1           1      100.00
                                Wr_Addr[7-0]           1           1      100.00
                                         clk           1           1      100.00
                                    din[0-9]           1           1      100.00
                                   dout[7-0]           1           1      100.00
                                       rst_n           1           1      100.00
                                    rx_valid           1           1      100.00
                                    tx_valid           1           1      100.00

Total Node Count     =         38
Toggled Node Count   =         38
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (76 of 76 bins)
```

Wrapper

```
================================Toggle Details================================

Toggle Coverage for instance /wrapper_top/dut --

                                         Node      1H->0L      0L->1H  "Coverage"
                                        ----------------------------------------
                                        MISO           1           1      100.00
                                        MOSI           1           1      100.00
                                        SS_n           1           1      100.00
                                         clk           1           1      100.00
                                       rst_n           1           1      100.00
                                rx_data[9-0]           1           1      100.00
                                    rx_valid           1           1      100.00
                                tx_data[7-0]           1           1      100.00
                                    tx_valid           1           1      100.00

Total Node Count     =         25
Toggled Node Count   =         25
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (50 of 50 bins)
```

Functional/Sequential Domain Coverage Report

SPI Slave

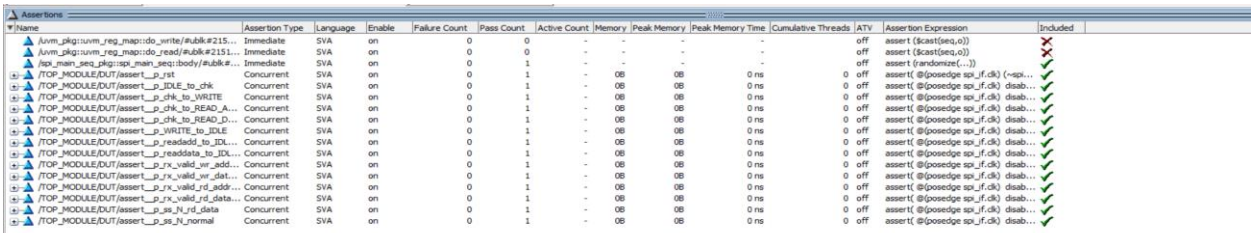
Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/spi_coverage_pkg...		100.00%							
TYPE cvr_gp		100.00%	100	100.00...		✓	auto(0)		
CVP cvr_g...		100.00%	100	100.00...		✓			
CVP cvr_g...		100.00%	100	100.00...		✓			
CVP cvr_g...		100.00%	100	100.00...		✓			
CVP cvr_g...		100.00%	100	100.00...		✓			
CROSS cvr...		100.00%	100	100.00...		✓			
INST \/spi...		100.00%	100	100.00...		✓		0	

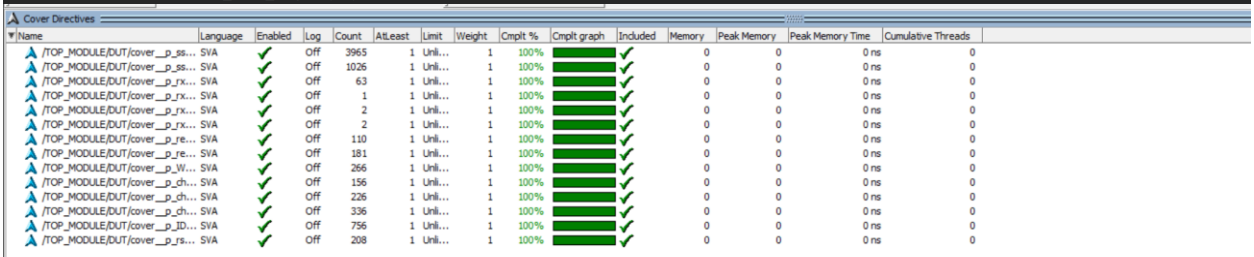
```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------------------
 TYPE /spi_coverage_pkg/spi_coverage/cvr_gp        100.00%        100          -    Covered
    covered/total bins:                                 18         18          -
    missing/total bins:                                  0         18          -
    % Hit:                                         100.00%        100          -
    Coverpoint rx_data_cp                          100.00%        100          -    Covered
        covered/total bins:                               7          7          -
        missing/total bins:                               0          7          -
        % Hit:                                     100.00%        100          -
    Coverpoint ss_n_cp                             100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint SS_N_cp                             100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                     100.00%        100          -
    Coverpoint MOSI_cp                             100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                     100.00%        100          -
    Cross SS_MOSI_c                                100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                     100.00%        100          -
 Covergroup instance \/spi_coverage_pkg::spi_coverage::cvr_gp
                                                   100.00%        100          -    Covered
    covered/total bins:                                 18         18          -
    missing/total bins:                                  0         18          -
    % Hit:                                         100.00%        100          -
    Coverpoint rx_data_cp                          100.00%        100          -    Covered
        covered/total bins:                               7          7          -
        missing/total bins:                               0          7          -
        % Hit:                                     100.00%        100          -
        bin all_vals[0]                                2854          1          -    Covered
        bin all_vals[1]                                2002          1          -    Covered
        bin all_vals[2]                                2837          1          -    Covered
        bin all_vals[3]                                2308          1          -    Covered
        bin wradd_2_wrdata                               99          1          -    Covered
        bin rdaddr_2_rddata                             114          1          -    Covered
        bin rddata_2_wradd                               46          1          -    Covered
    Coverpoint ss_n_cp                             100.00%        100          -    Covered
        covered/total bins:                               2          2          -
    Coverpoint ss_n_cp                             100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                     100.00%        100          -
        bin normal_trans                                458          1          -    Covered
        bin readdata_trans                              149          1          -    Covered
    Coverpoint SS_N_cp                             100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                     100.00%        100          -
        bin normal                                      608          1          -    Covered
    Coverpoint MOSI_cp                             100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                     100.00%        100          -
        bin write_addr                                 2417          1          -    Covered
        bin write_data                                  912          1          -    Covered
        bin read_addr                                   961          1          -    Covered
        bin read_data                                  2555          1          -    Covered
        default bin others                            10001                     -    Occurred
    Cross SS_MOSI_c                                100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                     100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <normal,read_data>                      142          1          -    Covered
            bin <normal,read_addr>                      170          1          -    Covered
            bin <normal,write_data>                     142          1          -    Covered
            bin <normal,write_addr>                     143          1          -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```



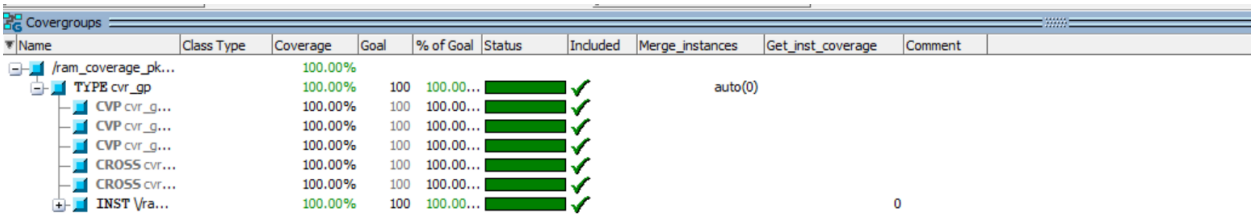

```
ASSERTION RESULTS:
---------------------------------------------------------------------
Name                File(Line)                    Failure      Pass
                                                  Count        Count
---------------------------------------------------------------------
/TOP_MODULE/DUT/assert__p_ss_N_normal
                    SPI.sv(273)                          0          1
/TOP_MODULE/DUT/assert__p_ss_N_rd_data
                    SPI.sv(270)                          0          1
/TOP_MODULE/DUT/assert__p_rx_valid_rd_data
                    SPI.sv(267)                          0          1
/TOP_MODULE/DUT/assert__p_rx_valid_rd_addr
                    SPI.sv(264)                          0          1
/TOP_MODULE/DUT/assert__p_rx_valid_wr_data
                    SPI.sv(261)                          0          1
/TOP_MODULE/DUT/assert__p_rx_valid_wr_addr
                    SPI.sv(258)                          0          1
/TOP_MODULE/DUT/assert__p_readdata_to_IDLE
                    SPI.sv(255)                          0          1
/TOP_MODULE/DUT/assert__p_readadd_to_IDLE
                    SPI.sv(252)                          0          1
/TOP_MODULE/DUT/assert__p_WRITE_to_IDLE
                    SPI.sv(249)                          0          1
/TOP_MODULE/DUT/assert__p_chk_to_READ_DATA
                    SPI.sv(246)                          0          1
/TOP_MODULE/DUT/assert__p_chk_to_READ_ADDRESS
                    SPI.sv(243)                          0          1
/TOP_MODULE/DUT/assert__p_chk_to_WRITE
                    SPI.sv(240)                          0          1
/TOP_MODULE/DUT/assert__p_IDLE_to_chk
                    SPI.sv(237)                          0          1
/TOP_MODULE/DUT/assert__p_rst
                    SPI.sv(234)                          0          1
/spi_main_seq_pkg/spi_main_seq/body/#ublk#267846023#24/immed__26
                    SPI_main_seq.sv(26)                  0          1
```




```
DIRECTIVE COVERAGE:
---------------------------------------------------------------------------------------------
Name                                  Design Design    Lang File(Line)      Hits Status
                                      Unit   UnitType
---------------------------------------------------------------------------------------------
/TOP_MODULE/DUT/cover__p_ss_N_normal     SPI_SLAVE Verilog  SVA  SPI.sv(293)     3965 Covered
/TOP_MODULE/DUT/cover__p_ss_N_rd_data    SPI_SLAVE Verilog  SVA  SPI.sv(292)     1026 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_rd_data
                                      SPI_SLAVE Verilog  SVA  SPI.sv(291)       63 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_rd_addr
                                      SPI_SLAVE Verilog  SVA  SPI.sv(290)        1 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_wr_data
                                      SPI_SLAVE Verilog  SVA  SPI.sv(289)        2 Covered
/TOP_MODULE/DUT/cover__p_rx_valid_wr_addr
                                      SPI_SLAVE Verilog  SVA  SPI.sv(288)        2 Covered
/TOP_MODULE/DUT/cover__p_readdata_to_IDLE
                                      SPI_SLAVE Verilog  SVA  SPI.sv(287)      110 Covered
/TOP_MODULE/DUT/cover__p_readadd_to_IDLE SPI_SLAVE Verilog  SVA  SPI.sv(286)      181 Covered
/TOP_MODULE/DUT/cover__p_WRITE_to_IDLE   SPI_SLAVE Verilog  SVA  SPI.sv(285)      266 Covered
/TOP_MODULE/DUT/cover__p_chk_to_READ_DATA
                                      SPI_SLAVE Verilog  SVA  SPI.sv(284)      156 Covered
/TOP_MODULE/DUT/cover__p_chk_to_READ_ADDRESS
                                      SPI_SLAVE Verilog  SVA  SPI.sv(283)      226 Covered
/TOP_MODULE/DUT/cover__p_chk_to_WRITE    SPI_SLAVE Verilog  SVA  SPI.sv(282)      336 Covered
/TOP_MODULE/DUT/cover__p_IDLE_to_chk     SPI_SLAVE Verilog  SVA  SPI.sv(281)      756 Covered
/TOP_MODULE/DUT/cover__p_rst             SPI_SLAVE Verilog  SVA  SPI.sv(280)      208 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 14
```

RAM



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/ram_coverage_pk...		100.00%							
TYPE cvr_gp		100.00%	100	100.00...			auto(0)		
CVP cvr_g...		100.00%	100	100.00...					
CVP cvr_g...		100.00%	100	100.00...					
CVP cvr_g...		100.00%	100	100.00...					
CROSS cvr...		100.00%	100	100.00...					
CROSS cvr...		100.00%	100	100.00...					
INST \/ra...		100.00%	100	100.00...				0	

```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------------
Covergroup                                                Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------------------------
 TYPE /ram_coverage_pkg/ram_coverage/cvr_gp              100.00%        100          -    Covered
    covered/total bins:                                       19         19          -
    missing/total bins:                                        0         19          -
    % Hit:                                                100.00%        100          -
    Coverpoint command_cp                                 100.00%        100          -    Covered
        covered/total bins:                                    7          7          -
        missing/total bins:                                    0          7          -
        % Hit:                                            100.00%        100          -
    Coverpoint rx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint tx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
    Cross rx_valid_with_commands                          100.00%        100          -    Covered
        covered/total bins:                                    7          7          -
        missing/total bins:                                    0          7          -
        % Hit:                                            100.00%        100          -
    Cross tx_valid_with_read_data                         100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
 Covergroup instance \/ram_coverage_pkg::ram_coverage::cvr_gp
                                                          100.00%        100          -    Covered
    covered/total bins:                                       19         19          -
    missing/total bins:                                        0         19          -
    % Hit:                                                100.00%        100          -
    Coverpoint command_cp                                 100.00%        100          -    Covered
        covered/total bins:                                    7          7          -
        missing/total bins:                                    0          7          -
        % Hit:                                            100.00%        100          -
        bin write_address                                    5035          1          -    Covered
        bin write_data                                       2493          1          -    Covered
        bin read_address                                     4161          1          -    Covered
        bin read_data                                        3312          1          -    Covered
        bin write_data_after_address                         2492          1          -    Covered
        bin read_data_after_address                          3312          1          -    Covered
        bin transition                                        235          1          -    Covered
        bin transition                                        235          1          -    Covered
    Coverpoint rx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
        bin auto[0]                                          712          1          -    Covered
        bin auto[1]                                        14289          1          -    Covered
    Coverpoint tx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
        bin auto[0]                                        12012          1          -    Covered
        bin auto[1]                                         2989          1          -    Covered
    Cross rx_valid_with_commands                          100.00%        100          -    Covered
        covered/total bins:                                    7          7          -
        missing/total bins:                                    0          7          -
        % Hit:                                            100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <transition,auto[1]>                         227          1          -    Covered
            bin <read_data_after_address,auto[1]>           3162          1          -    Covered
            bin <write_data_after_address,auto[1]>          2377          1          -    Covered
            bin <read_data,auto[1]>                         3162          1          -    Covered
            bin <write_data,auto[1]>                        2378          1          -    Covered
            bin <read_address,auto[1]>                      3956          1          -    Covered
            bin <write_address,auto[1]>                     4793          1          -    Covered
        Illegal and Ignore Bins:
            ignore_bin rx_valid_zero                         712                     -    Occurred
    Cross tx_valid_with_read_data                         100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
        Auto, Default and User Defined Bins:
            bin tx_valid_rd                                 2989          1          -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk#215...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#2151...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/ram_read_write_seq_pkg::ram_read_write_seq...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/ram_read_seq_pkg::ram_read_only_seq::body/...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/ram_write_seq_pkg::ram_write_only_seq::body...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/ram_top/DUT/ram_sva_inst/assert__reset_outp...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=...	✓
/ram_top/DUT/ram_sva_inst/assert__tx_valid_lo...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/ram_top/DUT/ram_sva_inst/assert__tx_valid_hi...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/ram_top/DUT/ram_sva_inst/assert__write_data...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/ram_top/DUT/ram_sva_inst/assert__read_data...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓

```
ASSERTION RESULTS:
---------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
---------------------------------------------------------------------
/ram_top/DUT/ram_sva_inst/assert__read_data_after_address
                     RAM_sva.sv(49)                     0          1
/ram_top/DUT/ram_sva_inst/assert__write_data_after_address
                     RAM_sva.sv(40)                     0          1
/ram_top/DUT/ram_sva_inst/assert__tx_valid_high
                     RAM_sva.sv(31)                     0          1
/ram_top/DUT/ram_sva_inst/assert__tx_valid_low
                     RAM_sva.sv(22)                     0          1
/ram_top/DUT/ram_sva_inst/assert__reset_outputs_low
                     RAM_sva.sv(14)                     0          1
/ram_read_write_seq_pkg/ram_read_write_seq/body/#ublk#31146023#20/immed__24
                     RAM_read_write_seq.sv(24)          0          1
/ram_read_seq_pkg/ram_read_only_seq/body/#ublk#256611655#20/immed__24
                     RAM_read_only_seq.sv(24)           0          1
/ram_write_seq_pkg/ram_write_only_seq/body/#ublk#114248983#20/immed__24
                     RAM_write_only_seq.sv(24)          0          1
```

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/ram_top/DUT/ram_sva_inst/cov...	SVA	✓	Off	3643	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/ram_sva_inst/cov...	SVA	✓	Off	4381	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/ram_sva_inst/cov...	SVA	✓	Off	2781	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/ram_sva_inst/cov...	SVA	✓	Off	10545	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/ram_sva_inst/cov...	SVA	✓	Off	764	1	Unli...	1	100%		✓	0	0	0 ns	0

```
DIRECTIVE COVERAGE:
---------------------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
                                      Unit   UnitType
---------------------------------------------------------------------------------------------
/ram_top/DUT/ram_sva_inst/cover__read_data_after_address
                                      ram_sva Verilog  SVA  RAM_sva.sv(51)  3643 Covered
/ram_top/DUT/ram_sva_inst/cover__write_data_after_address
                                      ram_sva Verilog  SVA  RAM_sva.sv(42)  4381 Covered
/ram_top/DUT/ram_sva_inst/cover__tx_valid_high
                                      ram_sva Verilog  SVA  RAM_sva.sv(33)  2781 Covered
/ram_top/DUT/ram_sva_inst/cover__tx_valid_low
                                      ram_sva Verilog  SVA  RAM_sva.sv(24)  10545 Covered
/ram_top/DUT/ram_sva_inst/cover__reset_outputs_low
                                      ram_sva Verilog  SVA  RAM_sva.sv(16)   764 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 5
```

Wrapper

Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/spi_coverage_pkg...		100.00%							
TYPE cvr_gp		100.00%	100	100.00...	▇	✓	auto(0)		
/ram_coverage_pk...		100.00%							
TYPE cvr_gp		100.00%	100	100.00...	▇	✓	auto(0)		

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk...	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/wrapper_read_write_seq_pkg::wrapper_r...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/wrapper_read_seq_pkg::wrapper_read_o...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/wrapper_write_seq_pkg::wrapper_write_...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/wrapper_top/dut/assert__reset_output_l...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=...	✓
/wrapper_top/dut/assert__MISO_stable	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/SLAVE_instance/assert_...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/RAM_instance/ram_sva...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=...	✓
/wrapper_top/dut/RAM_instance/ram_sva...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/RAM_instance/ram_sva...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/RAM_instance/ram_sva...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓
/wrapper_top/dut/RAM_instance/ram_sva...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...	✓

```
=================================================================================
=== Instance: /wrapper_top/dut
=== Design Unit: work.WRAPPER
=================================================================================

Assertion Coverage:
    Assertions                       2         2         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/wrapper_top/dut/assert__MISO_stable
                     wrapper.sv(54)                     0          1
/wrapper_top/dut/assert__reset_output_low
                     wrapper.sv(45)                     0          1

Directive Coverage:
    Directives                       2         2         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------
/wrapper_top/dut/cover__MISO_stable      WRAPPER Verilog  SVA  wrapper.sv(56)  12442 Covered
/wrapper_top/dut/cover__reset_output_low WRAPPER Verilog  SVA  wrapper.sv(47)    317 Covered
```

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/wrapper_top/dut/cover__MISO_stable	SVA	✓	Off	12442	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/cover__reset_output_low	SVA	✓	Off	317	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_ss...	SVA	✓	Off	6008	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_ss...	SVA	✓	Off	1484	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_rx...	SVA	✓	Off	14	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_rx...	SVA	✓	Off	1	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_rx...	SVA	✓	Off	1	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_re...	SVA	✓	Off	210	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_re...	SVA	✓	Off	242	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_W...	SVA	✓	Off	513	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_ch...	SVA	✓	Off	279	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_ch...	SVA	✓	Off	287	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_ch...	SVA	✓	Off	638	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_ID...	SVA	✓	Off	1280	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/SLAVE_instance/cover__p_rs...	SVA	✓	Off	317	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/RAM_instance/ram_sva_inst/...	SVA	✓	Off	2552	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/RAM_instance/ram_sva_inst/...	SVA	✓	Off	2743	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/RAM_instance/ram_sva_inst/...	SVA	✓	Off	621	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/RAM_instance/ram_sva_inst/...	SVA	✓	Off	1321	1	Unli...	1	100%	▇	✓	0	0	0 ns	0
/wrapper_top/dut/RAM_instance/ram_sva_inst/...	SVA	✓	Off	317	1	Unli...	1	100%	▇	✓	0	0	0 ns	0

Bug Report

SPI Slave

1.
Design Input for Bug to Appear: we are in the check_cmd case, ss_n is low, mosi is high and the received address is high
Expected Behavior: ns should be read data
Observed Behavior: ns was read address

2.
Design Input for Bug to Appear: when reset is asserted
Expected Behavior: all signals and internal signals should be low
Observed Behavior: counter didn't change to be 0

3.
Design Input for Bug to Appear: at the read data case after entering dummy data
Expected Behavior: reset the counter to start read data on the miso to get all bits
Observed Behavior: counter didn't cover all bits

RAM

1.
Design Input for Bug to Appear: at the case of read data of the din[9:8]
Expected Behavior: read data from the place that read address refers to
Observed Behavior: read the data from the place the write address refers to

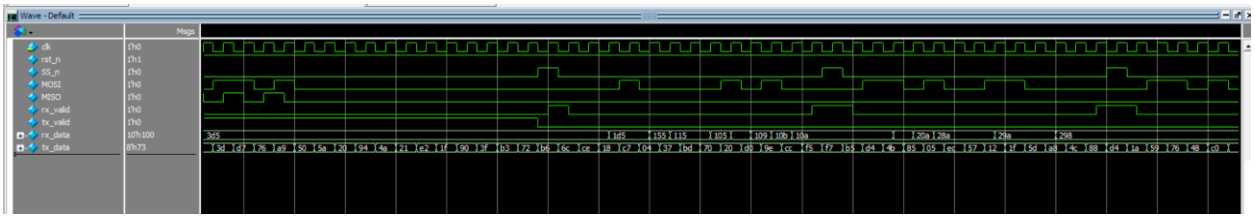
2.
Design Input for Bug to Appear: at the case of read data of the din[9:8]
Expected Behavior: spi should take bit by bit of the tx data when tx valid is high
Observed Behavior: tx valid is high at just one cycle so it takes only one bit

3.
the syntax errors: use mem instead of MEM, and add "begin ... end" in the else condition

QuestaSim Snippets

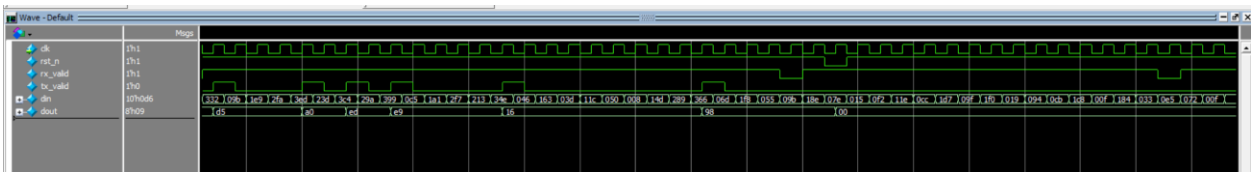
SPI Slave

```
# ****************************************************************************
# UVM_INFO SPI_test.sv(50) @ 2: uvm_test_top [run_phase] reset deasserted
# UVM_INFO SPI_test.sv(53) @ 2: uvm_test_top [run_phase] main asserted
# UVM_INFO SPI_test.sv(55) @ 20002: uvm_test_top [run_phase] main deasserted
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO SPI_scoreboard.sv(64) @ 20002: uvm_test_top.env.score [report_phase] total correct tranxactions: 10001
# UVM_INFO SPI_scoreboard.sv(65) @ 20002: uvm_test_top.env.score [report_phase] total failed tranxactions: 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: 20002 ns  Iteration: 61  Instance: /TOP_MODULE
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```



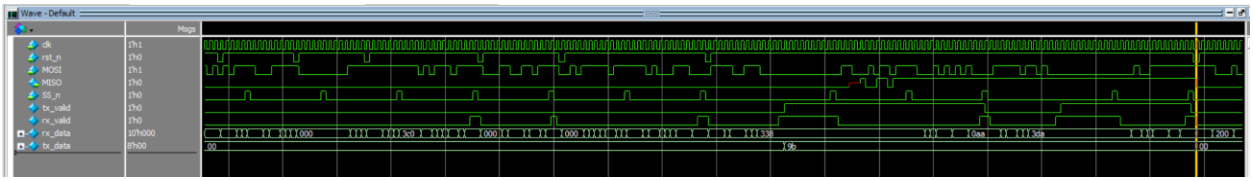
RAM

```
# ****************************************************************************
# UVM_INFO RAM_test.sv(60) @ 2: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO RAM_test.sv(63) @ 2: uvm_test_top [run_phase] Stimulus Generation Started
# UVM_INFO RAM_test.sv(67) @ 30002: uvm_test_top [run_phase] Stimulus Generation Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 30002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO RAM_scoreboard.sv(50) @ 30002: uvm_test_top.env.sb [report_phase] Total successful transactions: 15001
# UVM_INFO RAM_scoreboard.sv(51) @ 30002: uvm_test_top.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: 30002 ns  Iteration: 61  Instance: /ram_top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```



Wrapper

```
# ****************************************************************************
# UVM_INFO wrapper_test.sv(102) @ 2: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO wrapper_test.sv(105) @ 2: uvm_test_top [run_phase] Stimulus Generation Started
# UVM_INFO wrapper_test.sv(109) @ 30002: uvm_test_top [run_phase] Stimulus Generation Ended
# UVM_INFO wrapper_test.sv(110) @ 30002: uvm_test_top [wrapper_test] All sequences completed successfully
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 30002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO wrapper_scoreboard.sv(53) @ 30002: uvm_test_top.env.sb [report_phase] Total successful transactions: 15001
# UVM_INFO wrapper_scoreboard.sv(54) @ 30002: uvm_test_top.env.sb [report_phase] Total failed transactions: 0
# UVM_INFO wrapper_env.sv(40) @ 30002: uvm_test_top.env [WRAPPER_ENV] Wrapper Environment completed successfully
# UVM_INFO RAM_scoreboard.sv(50) @ 30002: uvm_test_top.r_env.sb [report_phase] Total successful transactions: 15001
# UVM_INFO RAM_scoreboard.sv(51) @ 30002: uvm_test_top.r_env.sb [report_phase] Total failed transactions: 0
# UVM_INFO SPI_scoreboard.sv(64) @ 30002: uvm_test_top.s_env.score [report_phase] total correct tranxactions: 15001
# UVM_INFO SPI_scoreboard.sv(65) @ 30002: uvm_test_top.s_env.score [report_phase] total failed tranxactions: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   17
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [WRAPPER_ENV]     1
# [report_phase]     6
# [run_phase]     4
# [wrapper_test]     2
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 30002 ns  Iteration: 61  Instance: /wrapper_top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```



Assertions table

SPI Slave

Feature	Assertion
When the rst is asserted, MISO , rx_valid & rx_data are low	@(posedge spi_if.clk) (!spi_if.rst_n) |=> (spi_if.MISO == 0 && spi_if.rx_valid == 0 && spi_if.rx_data == 0);
When the cs is idle & SS_n is low , cs is chk_cmd in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n) (cs == spi_if.IDLE && !spi_if.SS_n) |=> (cs == spi_if.CHK_CMD);
When the cs is chk_cmd & MOSI is low , cs is write in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (cs == spi_if.CHK_CMD && !spi_if.MOSI) |=> (cs == spi_if.WRITE);
When the cs is chk_cmd , MOSI is high & received address is low , cs is read address in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (cs == spi_if.CHK_CMD && spi_if.MOSI && !received_address) |=> (cs == spi_if.READ_ADDRESS);
When the cs is chk_cmd , MOSI is high & received address is high , cs is read data in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (cs == spi_if.CHK_CMD && spi_if.MOSI && received_address) |=> (cs == spi_if.READ_DATA);
When the cs is write & SS_n is high, cs is idle in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n) (cs == spi_if.WRITE && spi_if.SS_n) |=> (cs == spi_if.IDLE);
When the cs is read address & SS_n is high, cs is idle in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n) (cs == spi_if.READ_ADDRESS && spi_if.SS_n) |=> (cs == spi_if.IDLE);
When the cs is read data & SS_n is high, cs is idle in next clock cycle	@(posedge spi_if.clk) disable iff(!spi_if.rst_n) (cs == spi_if.READ_DATA && spi_if.SS_n) |=> (cs == spi_if.IDLE);
When the write address is high ,the rx valid is high after 11 clock cycles	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (write_addr) |=> ##10 (spi_if.rx_valid);
When the write data is high ,the rx valid is high after 11 clock cycles	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (write_data) |=> ##10 (spi_if.rx_valid);
When the read address is high ,the rx valid is high after 11 clock cycles	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (read_addr) |=> ##10 (spi_if.rx_valid);

When the read data is high ,the rx valid is high after 11 clock cycles	@(posedge spi_if.clk) disable iff(!spi_if.rst_n || spi_if.SS_n) (read_data) |=> ##10 (spi_if.rx_valid);
When the cs is read data , the SS_n is high after 24 clock cycles	@(posedge spi_if.clk) disable iff(!spi_if.rst_n) (cs == spi_if.READ_DATA) |=> ##23 (spi_if.SS_n[->1]);
When the cs is write or read address , the SS_n is high after 14 clock cycles	@(posedge spi_if.clk) disable iff(!spi_if.rst_n) (cs == spi_if.WRITE || cs == spi_if.READ_ADDRESS) |=> ##13 (spi_if.SS_n[->1]);

RAM

Feature	Assertion
When the rst is asserted, tx_valid & dout are low	@(posedge clk) (!rst_n) |=> (tx_valid == 0 && dout == 0);
When the rx_valid is high & din[9:8] is 0 (write address) or 1 (write data) or 2 (read address) , tx_valid is low	@(posedge clk) disable iff(!rst_n) (rx_valid && (din[9:8]== 2'b00 || din[9:8]== 2'b01 || din[9:8]== 2'b10)) |=> (tx_valid == 0);
When din[9:8] is 3 (read data) , tx_valid is rising and then it is falling	@(posedge clk) disable iff(!rst_n) (din[9:8] == 2'b11) |=> $rose(tx_valid)[->1] ##1 $fell(tx_valid)[->1] ;
When din[9:8] is 0 (write address) , din[9:8] is 1 (write data) in next clock cycle	@(posedge clk) disable iff(!rst_n) (din[9:8]== 2'b00) |=> (din[9:8]== 2'b01 [->1]);
When din[9:8] is 2 (read address) , din[9:8] is 3 (read data) in next clock cycle	@(posedge clk) disable iff(!rst_n) (din[9:8]== 2'b10) |=> (din[9:8]== 2'b11 [->1]);

Wrapper

Feature	Assertion
When the rst is asserted, MISO is low	@(posedge clk) (!rst_n) |=> (MISO== 0);
When the cs isn't read data , the MISO is stable	@(posedge clk) disable iff (!rst_n) (!tx_valid) |=> $stable(MISO)[->1];