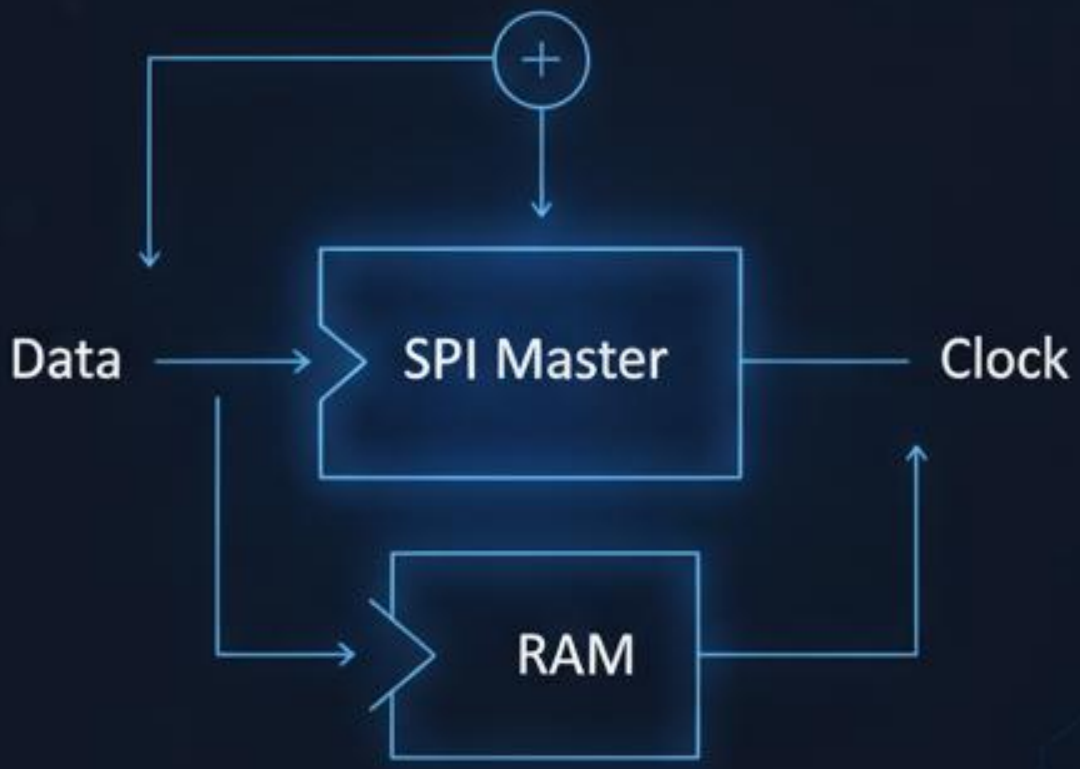
PROJECT REPORT

SPI Single Port RAM Verification

USING UVM METHOLOGY



Provided by:

Saeed Nabawy
Mostafa Ahmed Sherif
Zeyad Abdallah Shaban

Table of Contents

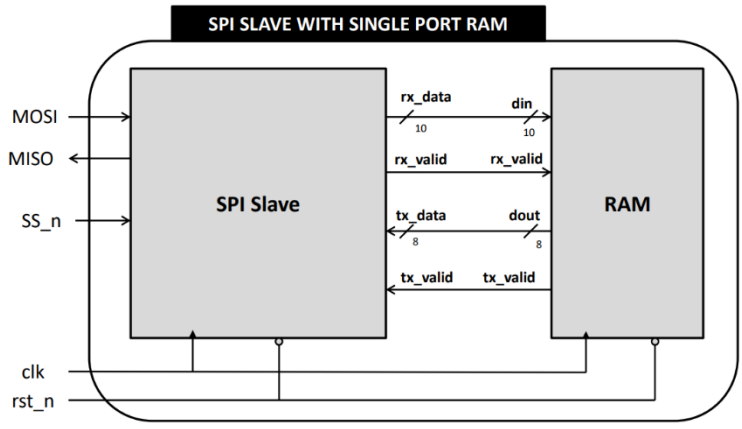
Introduction

The **SPI Slave with Single Port RAM Verification Project** focuses on verifying the functional correctness and protocol compliance of a Serial Peripheral Interface (SPI) slave module connected to a single-port RAM using the **Universal Verification Methodology (UVM)**. The SPI protocol is a widely used serial communication standard that allows synchronous data exchange between a master and multiple slave devices. In this project, the SPI slave is responsible for receiving and transmitting serial data with the master while interacting internally with a single-port RAM for data storage and retrieval.

The main objective of this verification project is to ensure that the SPI slave correctly performs all **read, write, and command decoding operations** according to the SPI protocol specifications. The slave communicates using four main signals: **MOSI (Master Out Slave In)** for data reception, **MISO (Master In Slave Out)** for data transmission, **CLK (Serial Clock)** for synchronization, and **SS_n (Slave Select)** for transaction control. The single-port RAM is used as a memory interface, where the data received through the SPI interface is written or from which data is read based on the issued commands.

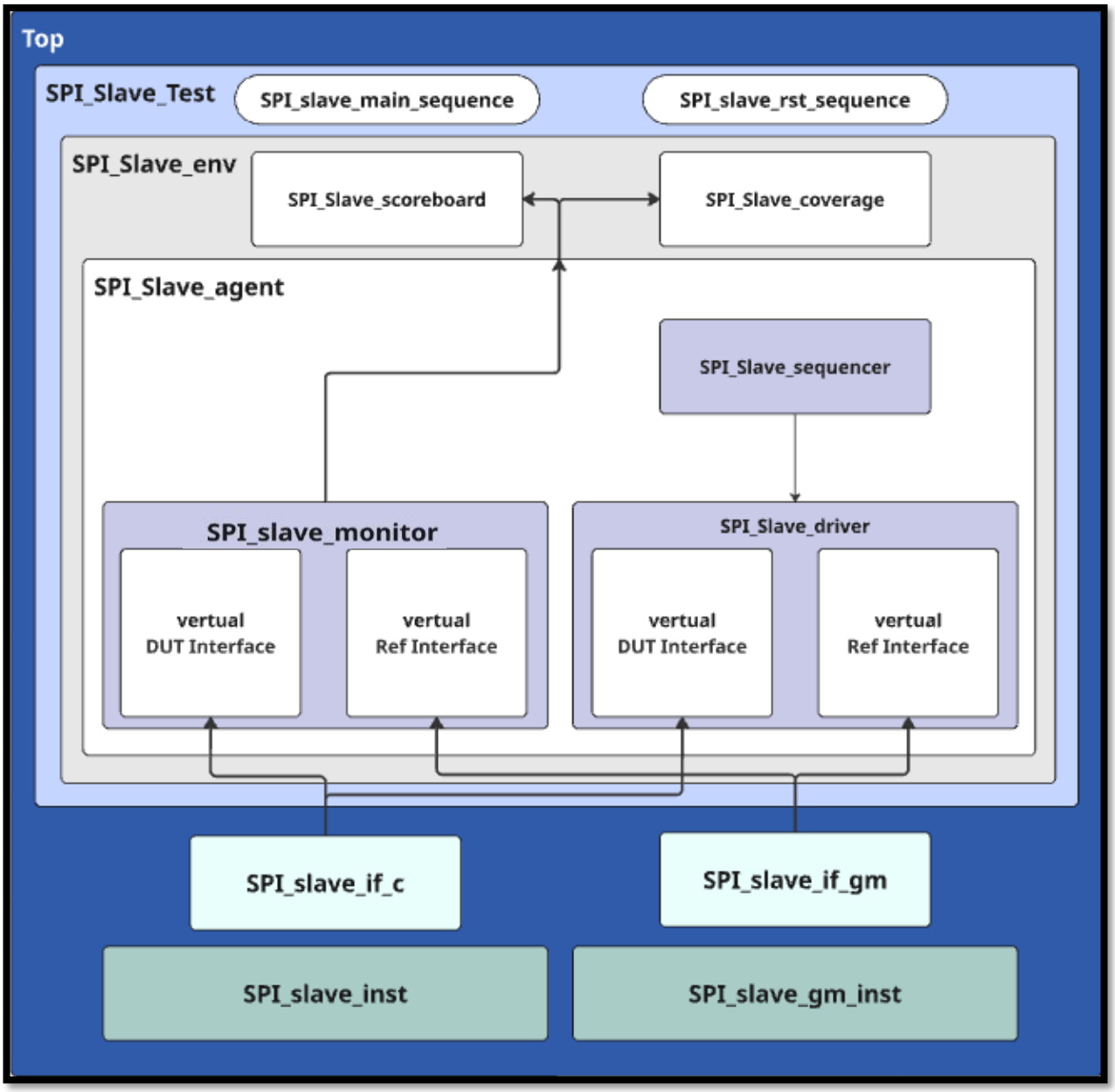
The verification environment was implemented using **SystemVerilog UVM**, providing a modular, reusable, and scalable testbench architecture. The environment includes key UVM components such as a **sequence item, sequencer, driver, monitor, agent, scoreboard, and environment**, all working together to generate stimulus, drive transactions, monitor responses, and compare expected versus actual outputs. Constrained random stimulus and functional coverage were applied to achieve high verification confidence and ensure that all SPI operations and corner cases were thoroughly exercised.

This project demonstrates the full UVM verification flow — from testbench architecture design and stimulus generation to coverage collection and result analysis — ensuring that the **SPI Slave with Single Port RAM** design meets its functional requirements and operates reliably under various conditions.



SPI Slave Verification

UVM Diagram



The top module instantiates two interfaces, one for the SPI Slave design and one for its golden reference model. In addition, it instantiates both the SPI Slave DUT and its corresponding reference model, each connected to its respective interface. These interfaces are then stored in the configuration database to be retrieved later during the verification phase. Finally, the simulation is initiated using the “run_test” function, which begins the SPI Slave test.

The SPI Slave test begins by retrieving these interfaces from the configuration database and storing them in configuration objects that contain their virtual interface handles. The test then creates the SPI Slave environment, as shown in the diagram, and runs the main and reset sequences. These sequences generate constrained sequence items that are sent from the sequencer to the drivers.

Within the SPI Slave environment, three main components are instantiated: the scoreboard, the coverage collector, and the UVM agent.
The SPI Slave scoreboard compares the results obtained from the DUT and the reference model by receiving transaction data through analysis FIFOs and analysis ports. Any mismatches between the DUT and reference transactions are reported as errors.
The SPI Slave coverage collector also connects through analysis ports to receive the same transaction data and ensures that the functional coverage metrics and test goals are met during simulation.
The SPI Slave agent retrieves the configuration object and accordingly instantiates its sequencer and two separate driver components — one for the DUT and another for the golden reference. Each driver receives its respective virtual interfaces from the configuration objects and is responsible for driving stimulus transactions directly to both the DUT and reference interfaces, effectively replacing the need for a dedicated monitor component.

During simulation, the sequencer continuously sends sequence items to both drivers, which in turn generate and apply corresponding stimulus signals to the DUT and the reference model. The resulting transactions are then sent to the scoreboard and coverage collector for comparison and coverage evaluation. Once all test sequences have completed and all verification objectives are met, the final objection is dropped from the test, marking the successful completion of the SPI Slave UVM test.

RTL Code

```
import shared_package::*;
module SPI_slave(SPI_slave_inf.dut if_c);

    localparam IDLE      = 3'b000;
    localparam CHK_CMD   = 3'b001;
    localparam WRITE     = 3'b010;
    localparam READ_ADD  = 3'b011;
    localparam READ_DATA = 3'b100;

    reg [3:0] counter;
    reg       received_address;

    reg [2:0] cs, ns;

    always @(posedge if_c.clk) begin
        if (~if_c.rst_n) begin
            cs <= IDLE;
        end
        else begin
            cs <= ns;
        end
        cs_shared <= cs; // shared variable
        ns_shared <= ns; // shared variable
    end

    always @(*) begin
        case (cs)
            IDLE : begin
                if (if_c.SS_n)
                    ns = IDLE;
                else
                    ns = CHK_CMD;
            end
            CHK_CMD : begin
                if (if_c.SS_n)
                    ns = IDLE;
                else begin
                    if (~if_c.MOSI)
                        ns = WRITE;
                    else begin
                        if (~received_address) //fix
                            ns = READ_ADD;
                        else
                            ns = READ_DATA;
                    end
                end
            end
            WRITE : begin
                if (if_c.SS_n)
                    ns = IDLE;
                else
                    ns = WRITE;
            end
            READ_ADD : begin
                if (if_c.SS_n)
                    ns = IDLE;
                else
                    ns = READ_ADD;
            end
            READ_DATA : begin
                if (if_c.SS_n)
                    ns = IDLE;
                else
                    ns = READ_DATA;
            end
        endcase
    end
```

```
always @(posedge if_c.clk) begin
    if (~if_c.rst_n) begin
        if_c.rx_data <= 0;
        if_c.rx_valid <= 0;
        received_address <= 0;
        if_c.MISO <= 0;
        counter <= 0;
    end
    else begin
        case (cs)
            IDLE : begin
                if_c.rx_data <= 1;
                if_c.rx_valid <= 0;
            end
            CHK_CMD : begin
                counter <= 10;
            end
            WRITE : begin
                if (counter > 0) begin
                    if_c.rx_data[counter-1] <= if_c.MOSI;
                    counter <= counter - 1;
                end
                else begin
                    if_c.rx_valid <= 1;
                    // counter <= 8; // IGNORE
                end
            end
            READ_ADD : begin
                if (counter > 0) begin
                    if_c.rx_data[counter-1] <= if_c.MOSI;
                    counter <= counter - 1;
                end
                else begin
                    if_c.rx_valid <= 1;
                    received_address <= 1;
                end
            end
            default : begin
                if (if_c.tx_valid) begin
                    if_c.rx_valid <= 0;
                    if (counter > 0) begin
                        if_c.MISO <= if_c.tx_data[counter-1];
                        counter <= counter - 1;
                    end
                    else begin
                        received_address <= 0;
                        if_c.rx_valid <= 1;
                    end
                end
                else begin
                    if (counter > 0) begin
                        if_c.rx_data[counter-1] <= if_c.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        if_c.rx_valid <= 1;
                        counter <= 8;
                    end
                end
            end
        endcase
    end
end
```

```

    `ifdef SIM

    property p_state_operation_1;

        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (cs == IDLE) |=> (cs == IDLE || cs == CHK_CMD)
    endproperty
    assert  property(p_state_operation_1);
    cover   property(p_state_operation_1);

    property p_state_operation_2;

        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (cs == CHK_CMD) |=> (cs == IDLE ||
        cs == WRITE || cs == READ_ADD || cs == READ_DATA)
    endproperty
    assert  property(p_state_operation_2);
    cover   property(p_state_operation_2);

    property p_state_operation_3;

        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (cs == WRITE) |=> (cs == WRITE || cs == IDLE)
    endproperty
    assert  property(p_state_operation_3);
    cover   property(p_state_operation_3);

    property p_state_operation_4;
        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (cs == READ_ADD) |=> (cs == READ_ADD || cs == IDLE)
    endproperty
    assert  property(p_state_operation_4);
    cover   property(p_state_operation_4);

    property p_state_operation_5;
        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (cs == READ_DATA) |=> (cs == READ_DATA || cs == IDLE)
    endproperty
    assert  property(p_state_operation_5);
    cover   property(p_state_operation_5);

`endif

endmodule
```

Bugs are fixed

```
CHK_CMD : begin
    if (if_c.SS_n)
        ns = IDLE;
    else begin
        if (~if_c.MOSI)
            ns = WRITE;
        else begin
            if (~received_address) //fix
                ns = READ_ADD;
            else
                ns = READ_DATA;
        end
    end
end
```

In the CHC_CMD State when deciding to the next state according to the received_address internal signal should when received_address is 0 the next state will be READ_ADD and when received_address is1 the next state will be READ_DATA

```
READ_DATA : begin
    if (if_c.tx_valid) begin
        if_c.rx_valid <= 0;
        if (counter > 0) begin
            if_c.MISO <= if_c.tx_data[counter-1];
            counter <= counter - 1;
        end
        else begin
            received_address <= 0;
            if_c.rx_valid <= 1; // fix
        end
    end
    else begin
        if (counter > 0) begin
            if_c.rx_data[counter-1] <= if_c.MOSI;
            counter <= counter - 1;
        end
        else begin
            if_c.rx_valid <= 1;
            counter <= 8;
        end
    end
end
```

In the READ_DATA State when all data in the tx_data pushed in MISO the rx_valid flag should be raise

Code Snippets

Top module

```
`include "uvm_macros.svh"
import uvm_pkg::*;
import SPI_slave_test_pkg::*;
module top;

    bit clk;
    initial begin
        clk = 0;
        forever #1 clk = ~clk;
    end

    SPI_slave_inf if_c(.clk(clk));
    SPI_slave_gm_inf if_gm(.clk(clk));
    SPI_slave SPI_slave_inst(if_c);
    SPI_Slave_golden_model gm(if_gm);
    SPI_slave_sva sva(if_c);
    // bind SPI_slave SPI_slave_sva sva(if_c);
    initial begin
        uvm_config_db #(virtual SPI_slave_inf)::set(null, "uvm_test_top", "vif", if_c);
        uvm_config_db #(virtual SPI_slave_gm_inf)::set(null, "uvm_test_top", "vif_gm", if_gm);
        run_test("SPI_slave_test");
    end
endmodule

```

Module Interface

```
interface SPI_slave_inf(input clk);
    logic MOSI, rst_n, SS_n, tx_valid;
    logic [7:0] tx_data;
    logic [9:0] rx_data;
    logic rx_valid, MISO;

    modport dut (
        input clk, MOSI, rst_n, SS_n, tx_valid,tx_data,
        output rx_data,rx_valid, MISO
    );

    modport mon (
        input clk, MOSI, rst_n, SS_n, tx_valid,tx_data,
        input rx_data,rx_valid, MISO
    );
endinterface
```

Golden Model Interface

```
interface SPI_slave_gm_inf(input clk);
    logic MOSI,SS_n,rst_n;
    logic [7:0] tx_data;
    logic tx_valid,MISO;
    logic [9:0] rx_data;
    logic rx_valid;

    modport gm(
        input clk, MOSI, rst_n, SS_n, tx_valid,tx_data,
        output rx_data,rx_valid, MISO
    );
endinterface
```

Golden Model Module

```
module SPI_Slave_golden_model(SPI_slave_gm_inf.dut if_gm);

    reg [3:0] counter;
    reg       received_address;
    reg [2:0] cs, ns;

    localparam IDLE      = 3'b000;
    localparam WRITE     = 3'b001;
    localparam CHK_CMD   = 3'b010;
    localparam READ_ADD  = 3'b011;
    localparam READ_DATA = 3'b100;


    always @(posedge if_gm.clk) begin
        if (~if_gm.rst_n) begin
            if_gm.rx_data <= 0;
            if_gm.rx_valid <= 0;
            received_address <= 0;
            if_gm.MISO <= 0;
            counter <= 0;
        end
        else begin
            case (cs)
                IDLE : begin
                    if_gm.rx_data <= 1;
                    if_gm.rx_valid <= 0;
                end
                CHK_CMD : begin
                    counter <= 10;
                end
                WRITE : begin
                    if (counter > 0) begin
                        if_gm.rx_data[counter-1] <= if_gm.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        if_gm.rx_valid <= 1;
                        // counter <= 8; // IGNORE
                    end
                end
                READ_ADD : begin
                    if (counter > 0) begin
                        if_gm.rx_data[counter-1] <= if_gm.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        if_gm.rx_valid <= 1;
                        received_address <= 1;
                    end
                end
                READ_DATA : begin
                    if (if_gm.tx_valid) begin
                        if_gm.rx_valid <= 0;
                        if (counter > 0) begin
                            if_gm.MISO <= if_gm.tx_data[counter-1];
                            counter <= counter - 1;
                        end
                        else begin
                            received_address <= 0;
                            if_gm.rx_valid <= 1;
                        end
                    end

```

```
                    else begin
                        if (counter > 0) begin
                            if_gm.rx_data[counter-1] <= if_gm.MOSI;
                            counter <= counter - 1;
                        end
                        else begin
                            if_gm.rx_valid <= 1;
                            counter <= 8;
                        end
                    end
                end
            endcase
        end
    end

    always @(*) begin
        case (cs)
            IDLE : begin
                if (if_gm.SS_n)
                    ns = IDLE;
                else
                    ns = CHK_CMD;
            end
            CHK_CMD : begin
                if (if_gm.SS_n)
                    ns = IDLE;
                else begin
                    if (~if_gm.MOSI)
                        ns = WRITE;
                    else begin
                        if (~received_address)
                            ns = READ_ADD;
                        else
                            ns = READ_DATA;
                    end
                end
            end
            WRITE : begin
                if (if_gm.SS_n)
                    ns = IDLE;
                else
                    ns = WRITE;
            end
            READ_ADD : begin
                if (if_gm.SS_n)
                    ns = IDLE;
                else
                    ns = READ_ADD;
            end
            READ_DATA : begin
                if (if_gm.SS_n)
                    ns = IDLE;
                else
                    ns = READ_DATA;
            end
        endcase
    end

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

endmodule
```

SPI_slave_sva

```
module SPI_slave_sva(SPI_slave_inf.dut if_c);

    localparam IDLE      = 3'b000;
    localparam CHK_CMD   = 3'b001;
    localparam WRITE     = 3'b010;
    localparam READ_ADD  = 3'b011;
    localparam READ_DATA = 3'b100;


    property p_reset;
        @(posedge if_c.clk)
        (~if_c.rst_n |=> (if_c.MISO == 'b0 && if_c.rx_valid == 'b0 && if_c.rx_data == 'b0))
    endproperty
    assert  property(p_reset);
    cover   property(p_reset);

    property p_rx_valid_SS_n_operation_write_add;
        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (if_c.SS_n ##1 !if_c.SS_n ##1 (!if_c.MOSI ##1 !if_c.MOSI ##1 !if_c.MOSI)) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
    endproperty
    assert  property(p_rx_valid_SS_n_operation_write_add);
    cover   property(p_rx_valid_SS_n_operation_write_add);

    property p_rx_valid_SS_n_operation_write_data;
        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (if_c.SS_n ##1 !if_c.SS_n ##1 (!if_c.MOSI ##1 !if_c.MOSI ##1 if_c.MOSI)) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
    endproperty
    assert  property(p_rx_valid_SS_n_operation_write_data);
    cover   property(p_rx_valid_SS_n_operation_write_data);

    property p_rx_valid_SS_n_operation_read_add;
        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (if_c.SS_n ##1 !if_c.SS_n ##1 if_c.MOSI ##1 if_c.MOSI ##1 !if_c.MOSI) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
    endproperty
    assert  property(p_rx_valid_SS_n_operation_read_add);
    cover   property(p_rx_valid_SS_n_operation_read_add);

    property p_rx_valid_SS_n_operation_read_data;
        @(posedge if_c.clk) disable iff(~if_c.rst_n)
        (if_c.SS_n ##1 !if_c.SS_n ##1 if_c.MOSI ##1 if_c.MOSI ##1 if_c.MOSI) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
    endproperty
    assert  property(p_rx_valid_SS_n_operation_read_data);
    cover   property(p_rx_valid_SS_n_operation_read_data);

endmodule : SPI_slave_sva
```

SPI_RAM_config

```
package SPI_RAM_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class SPI_RAM_config extends uvm_object;
        `uvm_object_utils(SPI_RAM_config)

        virtual SPI_slave_inf SPI_vif;
        virtual SPI_slave_gm_inf SPI_vif_gm;

        function new(string name = "SPI_RAM_config");
            super.new(name);
        endfunction
    endclass
endpackage
```

SPI_slave_sequenceitem

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

    class SPI_slave_sequenceitem extends uvm_sequence_item;
        `uvm_object_utils(SPI_slave_sequenceitem)
        logic MOSI;
        rand logic [10:0] MOSI_arr;
        logic [10:0] old_MOSI_arr = 3'b111;
        logic [2:0] next_mosi = 3'b000;
        rand logic SS_n = 1;
        rand logic rst_n, tx_valid;
        rand logic [7:0] tx_data;
        logic [9:0] rx_data;
        logic rx_valid, MISO;

        int counter = 0;
        function new(string name = "SPI_slave_sequenceitem");
            super.new(name);
        endfunction

        function string convert2string();
            return $sformatf("SPI_slave_sequenceitem: %s: MOSI=%b,
            rst_n=%b, SS_n=%b, tx_valid=%b, tx_data=%b, rx_data=%b,
            rx_valid=%b, MISO=%b",super.convert2string(), MOSI, rst_n,
            SS_n, tx_valid, tx_data, rx_data, rx_valid, MISO);
        endfunction

        function void pre_randomize();
            if(SS_n == 0) begin
                MOSI_arr.rand_mode(0);
            end
            else begin
                MOSI_arr.rand_mode(1);
            end
        endfunction

        constraint c1 {
            rst_n dist {0:/1 , 1:/99};

            if(counter == 23 && old_MOSI_arr[10:8] == 3'b111){
                SS_n == 1;
            }
            if(counter == 13 && old_MOSI_arr[10:8] != 3'b111){
                SS_n == 1;
            }
            if(!(counter == 23 && old_MOSI_arr[10:8] == 3'b111) && !(counter == 13 && old_MOSI_arr[10:8] != 3'b111)){
                SS_n == 0;
            }

            if(old_MOSI_arr[10:8] == 3'b111){
                tx_valid == 1;
            }
        }

        function void post_randomize();
            if(rst_n == 0) begin
                counter = 0;
                SS_n = 1;
            end
            else begin
                if(SS_n == 0) begin
                    if(counter == 0) begin
                        MOSI_arr = {next_mosi,MOSI_arr[7:0]};
                        old_MOSI_arr = MOSI_arr;
                    end
                    counter = counter + 1;
                    MOSI = MOSI_arr[10];
                    MOSI_arr = MOSI_arr << 1;
                end
                else begin
                    counter = 0;

                    if(old_MOSI_arr[10:8] == 3'b000) begin
                        next_mosi = 3'b001;
                    end
                    else if(old_MOSI_arr[10:8] == 3'b001) begin
                        next_mosi = 3'b110;
                    end
                    else if(old_MOSI_arr[10:8] == 3'b110) begin
                        next_mosi = 3'b111;
                    end
                    else if(old_MOSI_arr[10:8] == 3'b111) begin
                        next_mosi = 3'b000;
                    end
                end
            end
        endfunction
    endclass
endpackage
```

SPI_slave_sequencer

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

SPI_slave_rst_sequence

```
package SPI_slave_rst_sequence_pkg;
    import uvm_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_rst_sequence extends uvm_sequence #(SPI_slave_sequenceitem);
        `uvm_object_utils(SPI_slave_rst_sequence)
        SPI_slave_sequenceitem item;
        function new(string name = "SPI_slave_rst_sequence");
            super.new(name);
        endfunction

        task body();
            item = SPI_slave_sequenceitem::type_id::create("item");
            start_item(item);
            item.rst_n = 0;
            item.tx_valid = 0;
            item.tx_data = 0;
            item.MISO = 0;
            finish_item(item);
        endtask
    endclass
endpackage
```

SPI_slave_random_sequence

```
package SPI_slave_random_sequence_pkg;
    import uvm_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_random_sequence extends uvm_sequence #(SPI_slave_sequenceitem);
        `uvm_object_utils(SPI_slave_random_sequence)
        SPI_slave_sequenceitem item;
        function new(string name = "SPI_slave_random_sequence");
            super.new(name);
        endfunction

        task body();
            item = SPI_slave_sequenceitem::type_id::create("item");
            for (int i = 0; i < 100000 ; i++) begin
                start_item(item);
                assert(item.randomize());
                finish_item(item);
            end
        endtask
    endclass
endpackage
```

SPI_slave_driver

```
package SPI_slave_driver_pkg;
    import uvm_pkg::*;
    import SPI_RAM_config_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_driver extends uvm_driver #(SPI_slave_sequenceitem);
        `uvm_component_utils(SPI_slave_driver)
        virtual SPI_slave_inf SPI_slave_driver_inf;
        virtual SPI_slave_gm_inf SPI_slave_driver_inf_gm;
        SPI_slave_sequenceitem item;
        function new(string name = "SPI_slave_driver", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            `uvm_info("SPI_slave_driver", "inside the SPI_slave driver", UVM_LOW);
            item = SPI_slave_sequenceitem::type_id::create("item");
            forever begin
                seq_item_port.get_next_item(item);
                SPI_slave_driver_inf.MOSI = item.MOSI;
                SPI_slave_driver_inf.rst_n = item.rst_n;
                SPI_slave_driver_inf.SS_n = item.SS_n;
                SPI_slave_driver_inf.tx_valid = item.tx_valid;
                SPI_slave_driver_inf.tx_data = item.tx_data;
                SPI_slave_driver_inf_gm.tx_data = item.tx_data;
                SPI_slave_driver_inf_gm.MOSI = item.MOSI;
                SPI_slave_driver_inf_gm.rst_n = item.rst_n;
                SPI_slave_driver_inf_gm.SS_n = item.SS_n;
                SPI_slave_driver_inf_gm.tx_valid = item.tx_valid;
                seq_item_port.item_done();
                @(negedge SPI_slave_driver_inf.clk);
                item.MISO = SPI_slave_driver_inf.MISO;
                item.rx_valid = SPI_slave_driver_inf.rx_valid;
                item.rx_data = SPI_slave_driver_inf.rx_data;
                `uvm_info("SPI_slave_driver", item.convert2string(), UVM_LOW);
            end
        endtask

    endclass
endpackage
```

SPI_slave_monitor

```
package SPI_slave_monitor_pkg;
    import uvm_pkg::*;
    import SPI_RAM_config_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_monitor extends uvm_monitor;
        `uvm_component_utils(SPI_slave_monitor)
        virtual SPI_slave_inf SPI_slave_monitor_inf;
        virtual SPI_slave_gm_inf SPI_slave_monitor_inf_gm;
        SPI_slave_sequenceitem item;
        SPI_slave_sequenceitem item_gm;
        uvm_analysis_port#(SPI_slave_sequenceitem) mon_ap;
        uvm_analysis_port#(SPI_slave_sequenceitem) mon_ap_gm;

        function new(string name = "SPI_slave_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);
            mon_ap_gm = new("mon_ap_gm", this);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            `uvm_info("SPI_slave_monitor", "inside the SPI_slave monitor", UVM_LOW);
            forever begin
                @(negedge SPI_slave_monitor_inf.clk);

                item_gm = SPI_slave_sequenceitem::type_id::create("item_gm");
                item_gm.MOSI = SPI_slave_monitor_inf_gm.MOSI;
                item_gm.rst_n = SPI_slave_monitor_inf_gm.rst_n;
                item_gm.SS_n = SPI_slave_monitor_inf_gm.SS_n;
                item_gm.tx_valid = SPI_slave_monitor_inf_gm.tx_valid;
                item_gm.tx_data = SPI_slave_monitor_inf_gm.tx_data;
                item_gm.MISO = SPI_slave_monitor_inf_gm.MISO;
                item_gm.rx_valid = SPI_slave_monitor_inf_gm.rx_valid;
                item_gm.rx_data = SPI_slave_monitor_inf_gm.rx_data;


                item = SPI_slave_sequenceitem::type_id::create("item");
                item.MOSI = SPI_slave_monitor_inf.MOSI;
                item.rst_n = SPI_slave_monitor_inf.rst_n;
                item.SS_n = SPI_slave_monitor_inf.SS_n;
                item.tx_valid = SPI_slave_monitor_inf.tx_valid;
                item.tx_data = SPI_slave_monitor_inf.tx_data;
                item.MISO = SPI_slave_monitor_inf.MISO;
                item.rx_valid = SPI_slave_monitor_inf.rx_valid;
                item.rx_data = SPI_slave_monitor_inf.rx_data;

                mon_ap_gm.write(item_gm);
                mon_ap.write(item);
                `uvm_info("SPI_slave_monitor", item.convert2string(), UVM_LOW);
            end
        endtask

    endclass
endpackage
```

SPI_slave_scoreboard

```
package SPI_slave_scoreboard_pkg;
    import uvm_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(SPI_slave_scoreboard)
        uvm_analysis_export #(SPI_slave_sequenceitem) sb_export;
        uvm_tlm_analysis_fifo #(SPI_slave_sequenceitem) sb_fifo;
        uvm_analysis_export #(SPI_slave_sequenceitem) sb_export_gm;
        uvm_tlm_analysis_fifo #(SPI_slave_sequenceitem) sb_fifo_gm;
        SPI_slave_sequenceitem item;
        SPI_slave_sequenceitem item_gm;


        int correct_count = 0 , error_count = 0;
        function new(string name = "SPI_slave_scoreboard", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            sb_fifo = new("sb_fifo", this);
            sb_export = new("sb_export", this);
            sb_fifo_gm = new("sb_fifo_gm", this);
            sb_export_gm = new("sb_export_gm", this);
            sb_export.connect(sb_fifo.analysis_export);
            sb_export_gm.connect(sb_fifo_gm.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(item);
                sb_fifo_gm.get(item_gm);

                if(item.MISO === item_gm.MISO && item.rx_valid === item_gm.rx_valid && item.rx_data === item_gm.rx_data) begin
                    correct_count = correct_count + 1;
                end
                else begin
                    `uvm_error("run_phase", $sformatf("MISO = %d, rx_data = %d, rx_valid = %d", item.MISO, item.rx_data, item.rx_valid));
                    `uvm_error("run_phase", $sformatf("MISO = %d, rx_data = %d, rx_valid = %d", item_gm.MISO, item_gm.rx_data, item_gm.rx_valid));
                    error_count = error_count + 1;
                end
            end
        endtask


        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("correct_count = %d", correct_count), UVM_LOW);
            `uvm_info("report_phase", $sformatf("error_count = %d", error_count), UVM_LOW);
        endfunction
    endclass
endpackage
```

SPI_slave_coverage

```
package SPI_slave_coverage_pkg;
    import uvm_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_coverage extends uvm_component;
        `uvm_component_utils(SPI_slave_coverage)
        SPI_slave_sequenceitem item;
        uvm_analysis_export #(SPI_slave_sequenceitem) cv_export;
        uvm_tlm_analysis_fifo #(SPI_slave_sequenceitem) cv_fifo;
        bit pervious_SS_n;

        covergroup cg;
            rx_data_cp: coverpoint item.rx_data[9:8];
            SS_n_cp: coverpoint item.SS_n{
                bins transition13 = (1'b1 => 0[*13] => 1'b1);
                bins transition23 = (1'b1 => 0[*23] => 1'b1);
            }
            MOSI_add_cp: coverpoint item.MOSI iff (item.SS_n == 1){
                option.auto_bin_max = 0;
                bins write_address = (0 => 0 => 0);
                bins write_data = (0 => 0 => 1);
                bins read_address = (1 => 1 => 0);
                bins read_data = (1 => 1 => 1);
            }

            SS_n_MOSI_cc: cross SS_n_cp,MOSI_add_cp {
                ignore_bins illegal2 = binsof(SS_n_cp.transition23) && binsof(MOSI_add_cp.write_data);
                ignore_bins illegal4 = binsof(SS_n_cp.transition13) && binsof(MOSI_add_cp.read_data);
                ignore_bins illegal5 = binsof(SS_n_cp.transition13) && (binsof(MOSI_add_cp.write_data));
                ignore_bins illegal6 = binsof(SS_n_cp.transition23) && binsof(MOSI_add_cp.read_data);
            }
        endgroup

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            cv_export = new("cv_export", this);
            cv_fifo = new("cv_fifo", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cv_export.connect(cv_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cv_fifo.get(item);
                pervious_SS_n = item.SS_n;
                cg.sample();
            end
        endtask

    endclass
endpackage
```

SPI_slave_agent

```
package SPI_slave_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import SPI_RAM_config_pkg::*;
    import SPI_slave_sequencer_pkg::*;
    import SPI_slave_driver_pkg::*;
    import SPI_slave_monitor_pkg::*;
    import SPI_slave_sequenceitem_pkg::*;
    class SPI_slave_agent extends uvm_agent;
        `uvm_component_utils(SPI_slave_agent)
        SPI_slave_driver driver;
        SPI_slave_monitor monitor;
        SPI_slave_sequencer sequencer;
        SPI_RAM_config cfg;
        uvm_analysis_port #(SPI_slave_sequenceitem) agt_ap;
        uvm_analysis_port #(SPI_slave_sequenceitem) agt_ap_gm;

        function new(string name = "SPI_slave_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#(SPI_RAM_config)::get(this, "", "SPI_slave_config_test", cfg)) begin
                `uvm_fatal("SPI_slave_agent", "virtual interface must be set for SPI_slave_agent")
            end
            sequencer = SPI_slave_sequencer::type_id::create("sequencer", this);
            driver = SPI_slave_driver::type_id::create("driver", this);
            monitor = SPI_slave_monitor::type_id::create("monitor", this);
            agt_ap = new("agt_ap", this);
            agt_ap_gm = new("agt_ap_gm", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            driver.SPI_slave_driver_inf = cfg.SPI_vif;
            driver.SPI_slave_driver_inf_gm = cfg.SPI_vif_gm;
            monitor.SPI_slave_monitor_inf = cfg.SPI_vif;
            monitor.SPI_slave_monitor_inf_gm = cfg.SPI_vif_gm;
            driver.seq_item_port.connect(sequencer.seq_item_export);
            monitor.mon_ap.connect(agt_ap);
            monitor.mon_ap_gm.connect(agt_ap_gm);
        endfunction

    endclass

endpackage
```

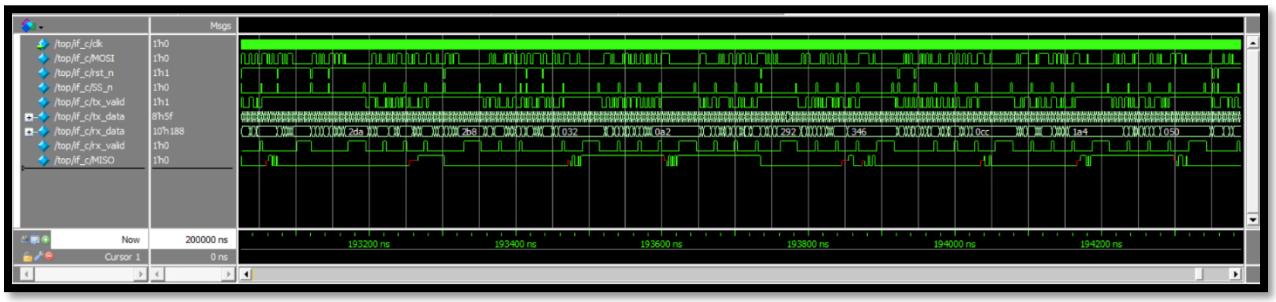
SPI_slave_env

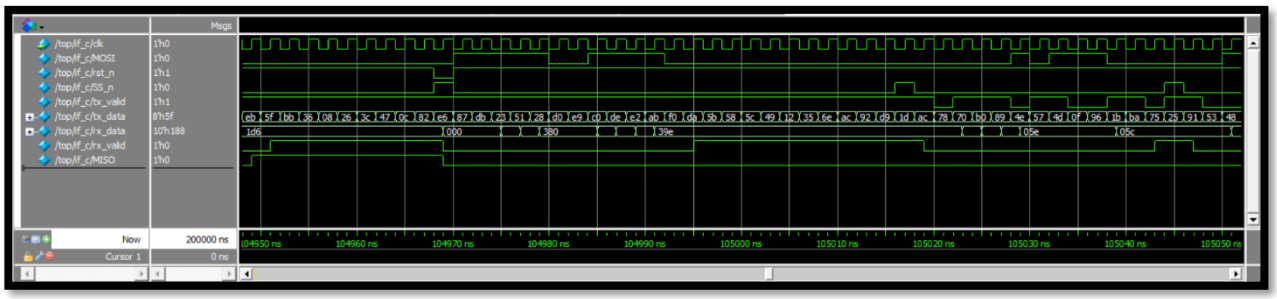
```
package SPI_slave_env_pkg;
    import uvm_pkg::*;
    import SPI_slave_agent_pkg::*;
    import SPI_slave_coverage_pkg::*;
    import SPI_slave_scoreboard_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_env extends uvm_env;
        `uvm_component_utils(SPI_slave_env)
        SPI_slave_scoreboard scoreboard;
        SPI_slave_coverage coverage;
        SPI_slave_agent agent;
        function new(string name = "SPI_slave_env", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            agent = SPI_slave_agent::type_id::create("agent", this);
            scoreboard = SPI_slave_scoreboard::type_id::create("scoreboard", this);
            coverage = SPI_slave_coverage::type_id::create("coverage", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agent.agt_ap.connect(coverage.cv_export);
            agent.agt_ap.connect(scoreboard.sb_export);
            agent.agt_ap_gm.connect(scoreboard.sb_export_gm);
        endfunction
    endclass
endpackage
```

SPI_slave_test

```
package SPI_slave_test_pkg;
    import uvm_pkg::*;
    import SPI_slave_env_pkg::*;
    import SPI_RAM_config_pkg::*;
    import SPI_slave_rst_sequence_pkg::*;
    import SPI_slave_random_sequence_pkg::*;
    `include "uvm_macros.svh"
    class SPI_slave_test extends uvm_test;
        `uvm_component_utils(SPI_slave_test)

        SPI_slave_env env;
        SPI_RAM_config cfg;
        SPI_slave_rst_sequence rst_sequence;
        SPI_slave_random_sequence random_sequence;
        function new(string name = "SPI_slave_test", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = SPI_slave_env::type_id::create("env", this);
            cfg = SPI_RAM_config::type_id::create("cfg");
            rst_sequence = SPI_slave_rst_sequence::type_id::create("rst_sequence");
            random_sequence = SPI_slave_random_sequence::type_id::create("random_sequence");
            if (!uvm_config_db#(virtual SPI_slave_inf)::get(this, "", "vif", cfg.SPI_vif)) begin
                `uvm_fatal("SPI_slave_test", "virtual interface must be set for SPI_slave_test")
            end
            if (!uvm_config_db#(virtual SPI_slave_gm_inf)::get(this, "", "vif_gm", cfg.SPI_vif_gm)) begin
                `uvm_fatal("SPI_slave_test", "virtual interface must be set for SPI_slave_test")
            end
            uvm_config_db#(SPI_RAM_config)::set(this, "*", "SPI_slave_config_test", cfg);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("SPI_slave_test", "running reset test", UVM_LOW);
            rst_sequence.start(env.agent.sequencer);
            `uvm_info("Finish_test", "reset test finished", UVM_LOW);
            #2;
            `uvm_info("SPI_slave_test", "running random test", UVM_LOW);
            random_sequence.start(env.agent.sequencer);
            `uvm_info("Finish_test", "random test finished", UVM_LOW);
            phase.drop_objection(this);
        endtask
    endclass
endpackage
```

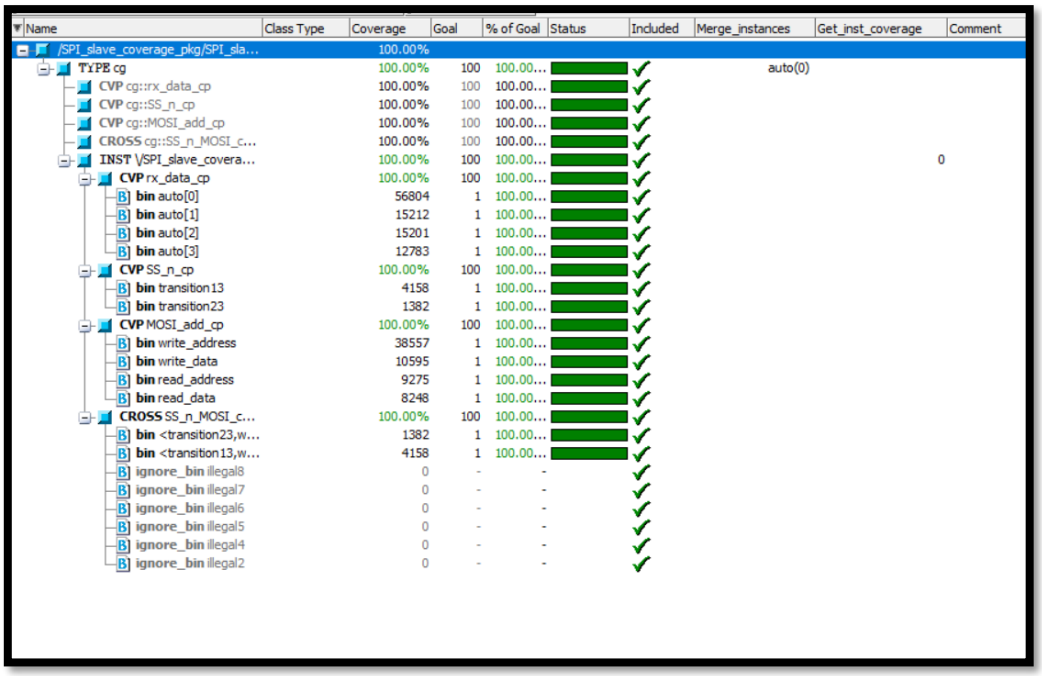
QuestaSim Snippets






```
# UVM_INFO SPI_slave_test.sv(43) @ 200000: uvm_test_top [Finish_test] random test finished
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 200000: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO SPI_slave_scoreboard.sv(50) @ 200000: uvm_test_top.env.scoreboard [report_phase] correct_count =      100000
# UVM_INFO SPI_slave_scoreboard.sv(51) @ 200000: uvm_test_top.env.scoreboard [report_phase] error_count =           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :200012
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Finish_test]     2
# [Questa UVM]     2
# [RNTST]     1
# [SPI_slave_driver] 100001
# [SPI_slave_monitor] 100001
# [SPI_slave_test]     2
# [TEST_DONE]     1
# [report_phase]     2
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 200 us  Iteration: 65  Instance: /top
# 1
# 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

VSIM 2>
```

Functional Coverage



Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/SPI_slave_coverage_pkg/SPI_sla...		100.00%							
TYPE cg		100.00%	100	100.00...			auto(0)		
CVP cg::rx_data_cp		100.00%	100	100.00...					
CVP cg::SS_n_cp		100.00%	100	100.00...					
CVP cg::MOSI_add_cp		100.00%	100	100.00...					
CROSS cg::SS_n_MOSI_c...		100.00%	100	100.00...					
INST \/SPI_slave_covera...		100.00%	100	100.00...				0	
CVP rx_data_cp		100.00%	100	100.00...					
bin auto[0]		56804	1	100.00...					
bin auto[1]		15212	1	100.00...					
bin auto[2]		15201	1	100.00...					
bin auto[3]		12783	1	100.00...					
CVP SS_n_cp		100.00%	100	100.00...					
bin transition13		4158	1	100.00...					
bin transition23		1382	1	100.00...					
CVP MOSI_add_cp		100.00%	100	100.00...					
bin write_address		38557	1	100.00...					
bin write_data		10595	1	100.00...					
bin read_address		9275	1	100.00...					
bin read_data		8248	1	100.00...					
CROSS SS_n_MOSI_c...		100.00%	100	100.00...					
bin <transition23,w...		1382	1	100.00...					
bin <transition13,w...		4158	1	100.00...					
ignore_bin illegal8		0	-	-					
ignore_bin illegal7		0	-	-					
ignore_bin illegal6		0	-	-					
ignore_bin illegal5		0	-	-					
ignore_bin illegal4		0	-	-					
ignore_bin illegal2		0	-	-					

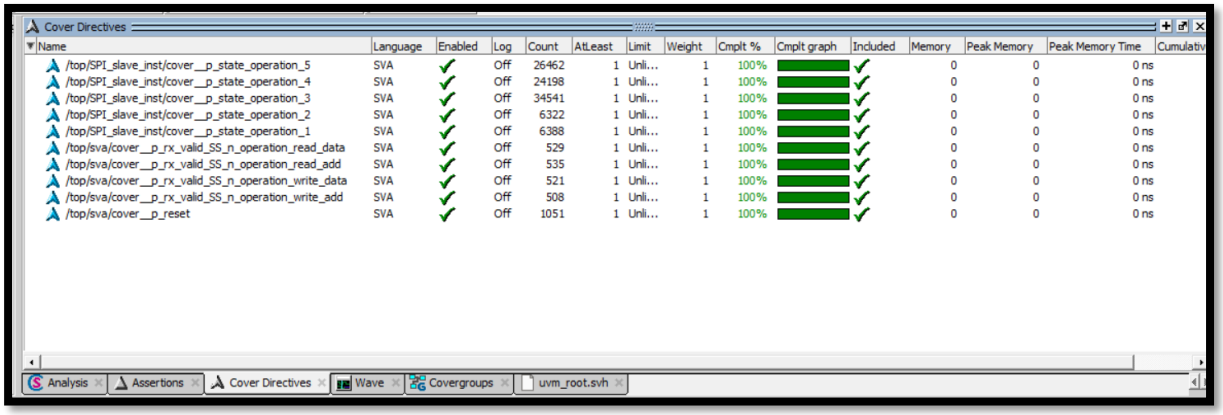
```
================================================================================
=== Instance: /SPI_slave_coverage_pkg
=== Design Unit: work.SPI_slave_coverage_pkg
================================================================================

Covergroup Coverage:
    Covergroups                 1        na        na   100.00%
        Coverpoints/Crosses     4        na        na        na
            Covergroup Bins    12        12         0   100.00%
----------------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------------------------
 TYPE /SPI_slave_coverage_pkg/SPI_slave_coverage/cg
                                                   100.00%        100          -    Covered
    covered/total bins:                                 12         12          -
    missing/total bins:                                  0         12          -
    % Hit:                                         100.00%        100          -
    Coverpoint rx_data_cp                          100.00%        100          -    Covered
        covered/total bins:                              4          4          -
        missing/total bins:                              0          4          -
        % 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 MOSI_add_cp                         100.00%        100          -    Covered
        covered/total bins:                              4          4          -
        missing/total bins:                              0          4          -
        % Hit:                                     100.00%        100          -
    Cross SS_n_MOSI_cc                             100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
 Covergroup instance \/SPI_slave_coverage_pkg::SPI_slave_coverage::cg
                                                   100.00%        100          -    Covered
    covered/total bins:                                 12         12          -
    missing/total bins:                                  0         12          -
    % Hit:                                         100.00%        100          -
    Coverpoint rx_data_cp                          100.00%        100          -    Covered
        covered/total bins:                              4          4          -
        missing/total bins:                              0          4          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                  56804          1          -    Covered
        bin auto[1]                                  15212          1          -    Covered
        bin auto[2]                                  15201          1          -    Covered
        bin auto[3]                                  12783          1          -    Covered
    Coverpoint SS_n_cp                             100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin transition13                              4158          1          -    Covered
        bin transition23                              1382          1          -    Covered
```

Coverpoint MOSI_add_cp	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
bin write_address	38557	1	-	Covered
bin write_data	10595	1	-	Covered
bin read_address	9275	1	-	Covered
bin read_data	8248	1	-	Covered
Cross SS_n_MOSI_cc	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <transition23,write_address>	1382	1	-	Covered
bin <transition13,write_address>	4158	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin illegal8	0		-	ZERO
ignore_bin illegal7	0		-	ZERO
ignore_bin illegal6	0		-	ZERO
ignore_bin illegal5	0		-	ZERO
ignore_bin illegal4	0		-	ZERO
ignore_bin illegal2	0		-	ZERO

Assertion Summary

Feature	Assertion
Whenever reset is asserted, the outputs (MISO, rx_valid, and rx_data) are all low.	@(posedge if_c.clk) (~if_c.rst_n |=> (if_c.MISO == 'b0 && if_c.rx_valid == 'b0 && if_c.rx_data == 'b0))
After write_add_seq (000), the rx_valid signal must assert exactly after 10 cycles and the SS_n should eventually after the 10 cycles to close communication.	@(posedge if_c.clk) disable iff(~if_c.rst_n) (if_c.SS_n ##1 !if_c.SS_n ##1 (!if_c.MOSI ##1 !if_c.MOSI ##1 !if_c.MOSI)) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
After write_data_seq (001), the rx_valid signal must assert exactly after 10 cycles and the SS_n should eventually after the 10 cycles to close communication.	@(posedge if_c.clk) disable iff(~if_c.rst_n) (if_c.SS_n ##1 !if_c.SS_n ##1 (!if_c.MOSI ##1 !if_c.MOSI ##1 if_c.MOSI)) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
After read_add_seq(110), the rx_valid signal must assert exactly after 10 cycles and the SS_n should eventually after the 10 cycles to close communication.	@(posedge if_c.clk) disable iff(~if_c.rst_n) (if_c.SS_n ##1 !if_c.SS_n ##1 if_c.MOSI ##1 if_c.MOSI ##1 !if_c.MOSI) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
After read_data_seq(111), the rx_valid signal must assert exactly after 10 cycles and the SS_n should eventually after the 10 cycles to close communication.	@(posedge if_c.clk) disable iff(~if_c.rst_n) (if_c.SS_n ##1 !if_c.SS_n ##1 if_c.MOSI ##1 if_c.MOSI ##1 if_c.MOSI) |-> ##10 ($rose(if_c.rx_valid)) ##[1:$] if_c.SS_n;
FSM transition: IDLE → CHK_CMD	@(posedge if_c.clk) disable iff(~if_c.rst_n) (cs == IDLE) |=> (cs == IDLE || cs == CHK_CMD)
FSM transition: CHK_CMD → WRITE or READ_ADD or READ_DATA	@(posedge if_c.clk) disable iff(~if_c.rst_n) (cs == CHK_CMD) |=> (cs == IDLE || cs == WRITE || cs == READ_ADD || cs == READ_DATA)
FSM transition: WRITE → IDLE	@(posedge if_c.clk) disable iff(~if_c.rst_n) (cs == WRITE) |=> (cs == WRITE || cs == IDLE)
FSM transition: READ_ADD → IDLE	@(posedge if_c.clk) disable iff(~if_c.rst_n) (cs == READ_ADD) |=> (cs == READ_ADD || cs == IDLE)
FSM transition: READ_DATA → IDLE	@(posedge if_c.clk) disable iff(~if_c.rst_n) (cs == READ_DATA) |=> (cs == READ_DATA || cs == IDLE)

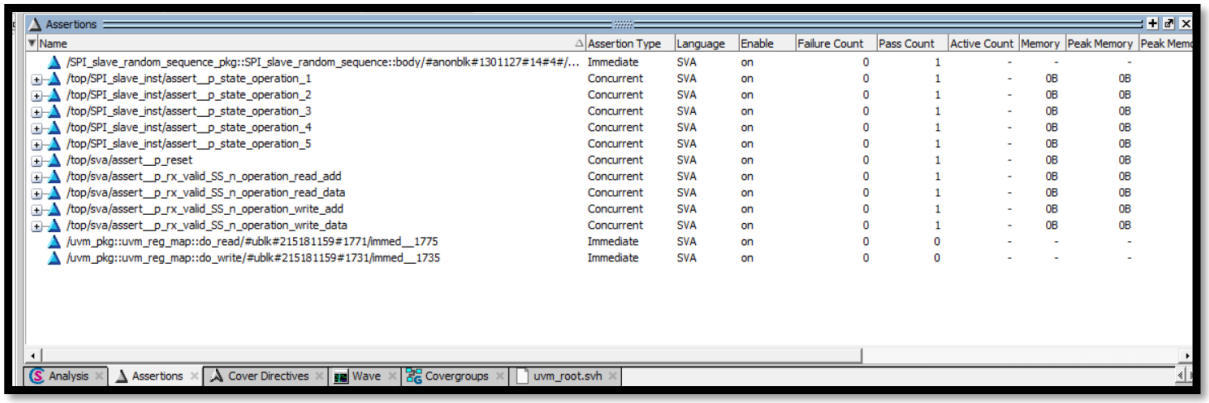
Assertion Coverage



Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulativ
/top/SPI_slave_inst/cover__p_state_operation_5	SVA	✓	Off	26462	1	Unli...	1	100%		✓	0	0	0 ns	
/top/SPI_slave_inst/cover__p_state_operation_4	SVA	✓	Off	24198	1	Unli...	1	100%		✓	0	0	0 ns	
/top/SPI_slave_inst/cover__p_state_operation_3	SVA	✓	Off	34541	1	Unli...	1	100%		✓	0	0	0 ns	
/top/SPI_slave_inst/cover__p_state_operation_2	SVA	✓	Off	6322	1	Unli...	1	100%		✓	0	0	0 ns	
/top/SPI_slave_inst/cover__p_state_operation_1	SVA	✓	Off	6388	1	Unli...	1	100%		✓	0	0	0 ns	
/top/sva/cover__p_rx_valid_SS_n_operation_read_data	SVA	✓	Off	529	1	Unli...	1	100%		✓	0	0	0 ns	
/top/sva/cover__p_rx_valid_SS_n_operation_read_add	SVA	✓	Off	535	1	Unli...	1	100%		✓	0	0	0 ns	
/top/sva/cover__p_rx_valid_SS_n_operation_write_data	SVA	✓	Off	521	1	Unli...	1	100%		✓	0	0	0 ns	
/top/sva/cover__p_rx_valid_SS_n_operation_write_add	SVA	✓	Off	508	1	Unli...	1	100%		✓	0	0	0 ns	
/top/sva/cover__p_reset	SVA	✓	Off	1051	1	Unli...	1	100%		✓	0	0	0 ns	

Analysis | Assertions | Cover Directives | Wave | Covergroups | uvm_root.svh



Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memo
/SPI_slave_random_sequence_pkg::SPI_slave_random_sequence::body/#anonblk#1301127#14#4#/...	Immediate	SVA	on	0	1	-	-	-	
/top/SPI_slave_inst/assert__p_state_operation_1	Concurrent	SVA	on	0	1	-	0B	0B	
/top/SPI_slave_inst/assert__p_state_operation_2	Concurrent	SVA	on	0	1	-	0B	0B	
/top/SPI_slave_inst/assert__p_state_operation_3	Concurrent	SVA	on	0	1	-	0B	0B	
/top/SPI_slave_inst/assert__p_state_operation_4	Concurrent	SVA	on	0	1	-	0B	0B	
/top/SPI_slave_inst/assert__p_state_operation_5	Concurrent	SVA	on	0	1	-	0B	0B	
/top/sva/assert__p_reset	Concurrent	SVA	on	0	1	-	0B	0B	
/top/sva/assert__p_rx_valid_SS_n_operation_read_add	Concurrent	SVA	on	0	1	-	0B	0B	
/top/sva/assert__p_rx_valid_SS_n_operation_read_data	Concurrent	SVA	on	0	1	-	0B	0B	
/top/sva/assert__p_rx_valid_SS_n_operation_write_add	Concurrent	SVA	on	0	1	-	0B	0B	
/top/sva/assert__p_rx_valid_SS_n_operation_write_data	Concurrent	SVA	on	0	1	-	0B	0B	
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	

Analysis | Assertions | Cover Directives | Wave | Covergroups | uvm_root.svh

```
================================================================================
=== Instance: /top/sva
=== Design Unit: work.SPI_slave_sva
================================================================================

Assertion Coverage:
    Assertions                       5         5         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/sva/assert__p_rx_valid_SS_n_operation_read_data
                     SPI_slave_sva.sv(42)               0          1
/top/sva/assert__p_rx_valid_SS_n_operation_read_add
                     SPI_slave_sva.sv(35)               0          1
/top/sva/assert__p_rx_valid_SS_n_operation_write_data
                     SPI_slave_sva.sv(28)               0          1
/top/sva/assert__p_rx_valid_SS_n_operation_write_add
                     SPI_slave_sva.sv(21)               0          1
/top/sva/assert__p_reset
                     SPI_slave_sva.sv(14)               0          1

Directive Coverage:
    Directives                       5         5         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------------
/top/sva/cover__p_rx_valid_SS_n_operation_read_data
                                         SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(43)
                                                                          529 Covered
/top/sva/cover__p_rx_valid_SS_n_operation_read_add
                                         SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(36)
                                                                          535 Covered
/top/sva/cover__p_rx_valid_SS_n_operation_write_data
                                         SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(29)
                                                                          521 Covered
/top/sva/cover__p_rx_valid_SS_n_operation_write_add
                                         SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(22)
                                                                          508 Covered
/top/sva/cover__p_reset                  SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(15)
                                                                         1051 Covered
```

Code Coverage

```
================================================================================
=== Instance: /top/if_c
=== Design Unit: work.SPI_slave_inf
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         50        50         0   100.00%

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

Toggle Coverage for instance /top/if_c --

                                          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)
```

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

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

Branch Coverage for instance /top/SPI_slave_inst

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File SPI_slave.sv
-----------------------------------IF Branch-----------------------------------
    15                                     25698     Count coming in to IF
    15              1                       1048             if (~if_c.rst_n) begin
    18              1                      24650             else begin
Branch totals: 2 hits of 2 branches = 100.00%

---------------------------------CASE Branch-----------------------------------
    24                                     65151     Count coming in to CASE
    25              1                      12941                 IDLE : begin
    31              1                       6532                 CHK_CMD : begin
    45              1                      20975                 WRITE : begin
    51              1                      13882                 READ_ADD : begin
    57              1                      10820                 READ_DATA : begin
                                               1     All False Count
Branch totals: 6 hits of 6 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    26                                     12941     Count coming in to IF
    26              1                       6481                     if (if_c.SS_n)
    28              1                       6460                     else
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    32                                      6532     Count coming in to IF
    32              1                         72                     if (if_c.SS_n)
    34              1                       6460                     else begin
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    35                                      6460     Count coming in to IF
    35              1                       3151                         if (~if_c.MOSI)
    37              1                       3309                         else begin
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    38                                      3309     Count coming in to IF
    38              1                       1788                             if (~received_address) //fix
    40              1                       1521                             else
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    46                                     20975     Count coming in to IF
    46              1                       3112                     if (if_c.SS_n)
    48              1                      17863                     else
Branch totals: 2 hits of 2 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
    58                                  10820     Count coming in to IF
    58              1                    1503                     if (if_c.SS_n)
    60              1                    9317                     else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    67                                 100000     Count coming in to IF
    67              1                    1051             if (~if_c.rst_n) begin
    74              1                   98949             else begin
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------CASE Branch-----------------------------------
    75                                  98949     Count coming in to CASE
    76              1                    6460                     IDLE : begin
    80              1                    6388                     CHK_CMD : begin
    83              1                   34919                     WRITE : begin
    93              1                   24445                     READ_ADD : begin
    103             1                   26737                     default : begin
Branch totals: 5 hits of 5 branches = 100.00%

------------------------------------IF Branch------------------------------------
    84                                  34919     Count coming in to IF
    84              1                   32149                          if (counter > 0) begin
    88              1                    2770                          else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    94                                  24445     Count coming in to IF
    94              1                   16835                          if (counter > 0) begin
    98              1                    7610                          else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    104                                 26737     Count coming in to IF
    104             1                   25019                          if (if_c.tx_valid) begin
    115             1                    1718                          else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    106                                 25019     Count coming in to IF
    106             1                   12721                               if (counter > 0) begin
    110             1                   12298                               else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    116                                  1718     Count coming in to IF
    116             1                    1525                               if (counter > 0) begin
    120             1                     193                               else begin
Branch totals: 2 hits of 2 branches = 100.00%
```

```
================================FSM Details================================

FSM Coverage for instance /top/SPI_slave_inst --

FSM_ID: cs
    Current State Object : cs
    ---------------------
    State Value MapInfo :
    ---------------------
Line           State Name               Value
----           ----------               -----
  25                 IDLE                   0
  31              CHK_CMD                   1
  57            READ_DATA                   4
  51             READ_ADD                   3
  45                WRITE                   2
    Covered States :
    ----------------
                    State           Hit_count
                    -----           ---------
                     IDLE                6528
                  CHK_CMD                6460
                READ_DATA                2982
                 READ_ADD                3531
                    WRITE                6197
    Covered Transitions :
    ---------------------
Line            Trans_ID           Hit_count          Transition
----            --------           ---------          ----------
  29                   0                6460          IDLE -> CHK_CMD
  41                   1                1503          CHK_CMD -> READ_DATA
  39                   2                1772          CHK_CMD -> READ_ADD
  36                   3                3113          CHK_CMD -> WRITE
  33                   4                  72          CHK_CMD -> IDLE
  59                   5                1503          READ_DATA -> IDLE
  53                   6                1772          READ_ADD -> IDLE
  47                   7                3112          WRITE -> IDLE


    Summary                       Bins      Hits    Misses  Coverage
    -------                       ----      ----    ------  --------
        FSM States                   5         5         0   100.00%
        FSM Transitions              8         8         0   100.00%
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      42        42         0   100.00%
```

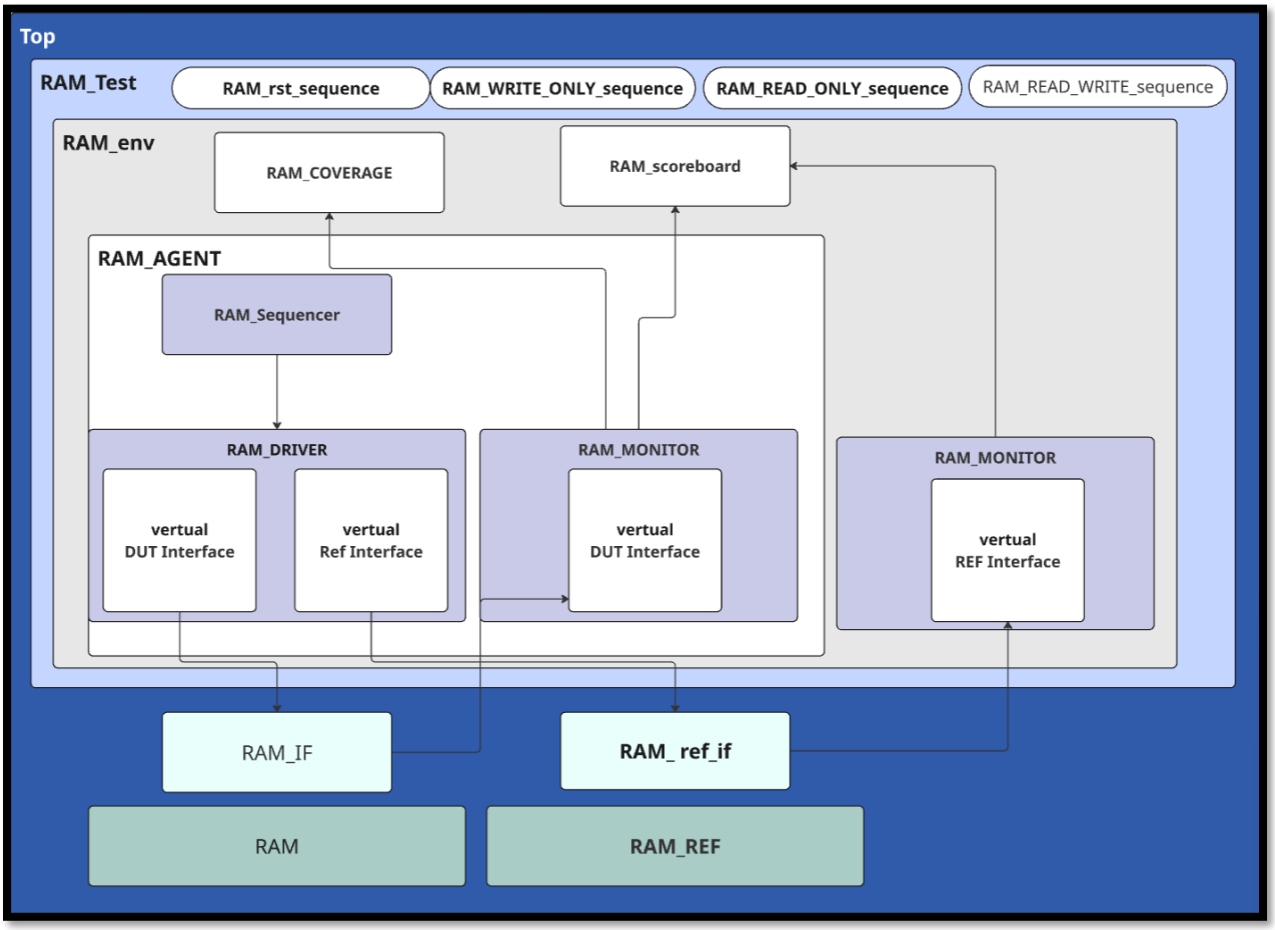
```
================================Statement Details================================

Statement Coverage for instance /top/SPI_slave_inst --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File SPI_slave.sv
    1                                                module SPI_slave(SPI_slave_inf.dut if_c);
    2
    3                                                    localparam IDLE      = 3'b000;
    4                                                    localparam CHK_CMD   = 3'b001;
    5                                                    localparam WRITE     = 3'b010;
    6                                                    localparam READ_ADD  = 3'b011;
    7                                                    localparam READ_DATA = 3'b100;
    8
    9                                                    reg [3:0] counter;
    10                                                   reg       received_address;
    11
    12                                                   reg [2:0] cs, ns;
    13
    14              1                    25698           always @(posedge if_c.clk) begin
    15                                                       if (~if_c.rst_n) begin
    16              1                     1048                   cs <= IDLE;
    17                                                       end
    18                                                       else begin
    19              1                    24650                   cs <= ns;
    20                                                       end
    21                                                   end
    22
    23              1                    65151           always @(*) begin
    24                                                       case (cs)
    25                                                           IDLE : begin
    26                                                               if (if_c.SS_n)
    27              1                     6481                          ns = IDLE;
    28                                                               else
    29              1                     6460                          ns = CHK_CMD;
    30                                                           end
    31                                                           CHK_CMD : begin
    32                                                               if (if_c.SS_n)
    33              1                       72                          ns = IDLE;
    34                                                               else begin
    35                                                                   if (~if_c.MOSI)
    36              1                     3151                              ns = WRITE;
    37                                                                   else begin
    38                                                                       if (~received_address) //fix
    39              1                     1788                                  ns = READ_ADD;
    40                                                                       else
    41              1                     1521                                  ns = READ_DATA;
    42                                                                   end
    43                                                               end
    44                                                           end
    45                                                           WRITE : begin
    46                                                               if (if_c.SS_n)
    47              1                     3112                          ns = IDLE;
    48                                                               else
    49              1                    17863                          ns = WRITE;
    50                                                           end
    51                                                           READ_ADD : begin
    52                                                               if (if_c.SS_n)
    53              1                     1772                          ns = IDLE;
```

```
     54                                                          else
     55              1                       12110                   ns = READ_ADD;
     56                                                      end
     57                                                      READ_DATA : begin
     58                                                          if (if_c.SS_n)
     59              1                        1503               ns = IDLE;
     60                                                          else
     61              1                        9317               ns = READ_DATA;
     62                                                      end
     63                                                  endcase
     64                                              end
     65
     66              1                      100000       always @(posedge if_c.clk) begin
     67                                                  if (~if_c.rst_n) begin
     68              1                        1051           if_c.rx_data <= 0;
     69              1                        1051           if_c.rx_valid <= 0;
     70              1                        1051           received_address <= 0;
     71              1                        1051           if_c.MISO <= 0;
     72              1                        1051           counter <= 10;
     73                                                  end
     74                                                  else begin
     75                                                      case (cs)
     76                                                          IDLE : begin
     77              1                        6460               if_c.rx_data <= 1;
     78              1                        6460               if_c.rx_valid <= 0;
     79                                                          end
     80                                                          CHK_CMD : begin
     81              1                        6388               counter <= 10;
     82                                                          end
     83                                                          WRITE : begin
     84                                                              if (counter > 0) begin
     85              1                       32149                   if_c.rx_data[counter-1] <= if_c.MOSI;
     86              1                       32149                   counter <= counter - 1;
     87                                                              end
     88                                                              else begin
     89              1                        2770                   if_c.rx_valid <= 1;
     90              1                        2770                   counter <= 8;
     91                                                              end
     92                                                          end
     93                                                          READ_ADD : begin
     94                                                              if (counter > 0) begin
     95              1                       16835                   if_c.rx_data[counter-1] <= if_c.MOSI;
     96              1                       16835                   counter <= counter - 1;
     97                                                              end
     98                                                              else begin
     99              1                        7610                   if_c.rx_valid <= 1;
    100              1                        7610                   received_address <= 1;
    101                                                             end
    102                                                         end
    103                                                         default : begin
    104                                                             if (if_c.tx_valid) begin
    105              1                       25019                   if_c.rx_valid <= 0;
    106                                                             if (counter > 0) begin
    107              1                       12721                       if_c.MISO <= if_c.tx_data[counter-1];
    108              1                       12721                       counter <= counter - 1;
    109                                                             end
    110                                                             else begin
    111              1                       12298                       received_address <= 0;
    112              1                       12298                       if_c.rx_valid <= 1; // fix
```

```
    112              1                       12298                       if_c.rx_valid <= 1; // fix
    113                                                             end
    114                                                         end
    115                                                         else begin
    116                                                             if (counter > 0) begin
    117              1                        1525                   if_c.rx_data[counter-1] <= if_c.MOSI;
    118              1                        1525                   counter <= counter - 1;
    119                                                             end
    120                                                             else begin
    121              1                         193                   if_c.rx_valid <= 1;
    122              1                         193                   counter <= 8;

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

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

Toggle Coverage for instance /top/SPI_slave_inst --

                                        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 Verification

UVM Diagram



The top module instantiates two interfaces, one for the RAM design and one for its golden reference model. Additionally, it instantiates both the RAM DUT and its reference model, each connected to its corresponding interface. Finally, both interfaces are stored in the configuration database to be retrieved later during the verification process, and the simulation is started by the “run_test” function for starting the RAM test.

The RAM test begins by retrieving these interfaces from the configuration database and embedding them into configuration objects that hold their respective virtual interfaces. The test then creates the RAM environment, which will be explained below, and triggers the four sequences, reset, write-only, read-only, and read & write, as shown in the diagram test

sector. Each sequence generates constrained sequence items that are sent from the sequencer to the driver.

Inside the RAM environment, three main components are created: the scoreboard, the coverage collector, and the UVM agent.

The RAM scoreboard is responsible for comparing the DUT outputs with the reference model outputs by using two analysis FIFOs and analysis ports that receive the transactions from the monitors. Any mismatches between the DUT and the reference are reported for debugging.

The RAM coverage collector operates similarly by capturing the same transactions through its analysis ports to verify that all required functional coverage points are met during the test execution.

The RAM agent receives the configuration object to determine the environment's activity mode. Since this environment is active, the agent instantiates its driver, sequencer, and monitor components. The driver sends stimulus transactions from the sequencer to the DUT through its virtual DUT interface, while the monitors observe both the DUT and reference interfaces to capture transactions and forward them to the scoreboard and coverage collector.

This process continues throughout the duration of all sequence operations. Once all test iterations and stimulus generations are completed successfully, the final objection is dropped from the test, signaling the end of the UVM test.

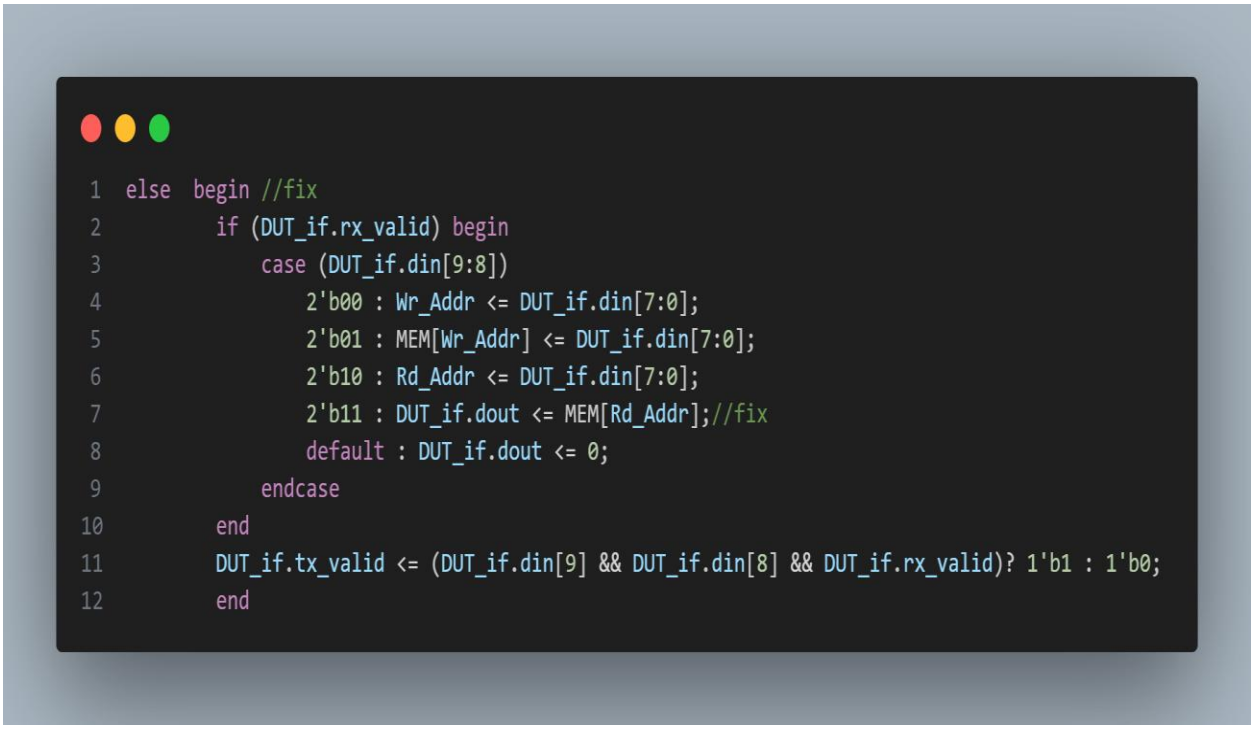
RTL Code

```
module RAM (RAM_if.DUT DUT_if);

logic[7:0] MEM [255:0];
logic [7:0] Rd_Addr, Wr_Addr;

always @(posedge DUT_if.clk) begin
    if (~DUT_if.rst_n) begin
        DUT_if.dout <= 0;
        DUT_if.tx_valid <= 0;
        Rd_Addr <= 0;
        Wr_Addr <= 0;
    end
    else  begin //fix
        if (DUT_if.rx_valid) begin
            case (DUT_if.din[9:8])
                2'b00 : Wr_Addr <= DUT_if.din[7:0];
                2'b01 : MEM[Wr_Addr] <= DUT_if.din[7:0];
                2'b10 : Rd_Addr <= DUT_if.din[7:0];
                2'b11 : DUT_if.dout <= MEM[Rd_Addr];//fix
                default : DUT_if.dout <= 0;
            endcase
        end
        DUT_if.tx_valid <= (DUT_if.din[9] && DUT_if.din[8] && DUT_if.rx_valid)? 1'b1 : 1'b0;
        end
end

endmodule
```

Bugs are fixed

1_ read data from write data address

2_there is no begin _end in else so the tx_valid condition can be high when the reset is asserted



```
else  begin //fix
        if (DUT_if.rx_valid) begin
            case (DUT_if.din[9:8])
                2'b00 : Wr_Addr <= DUT_if.din[7:0];
                2'b01 : MEM[Wr_Addr] <= DUT_if.din[7:0];
                2'b10 : Rd_Addr <= DUT_if.din[7:0];
                2'b11 : DUT_if.dout <= MEM[Rd_Addr];//fix
                default : DUT_if.dout <= 0;
            endcase
        end
        DUT_if.tx_valid <= (DUT_if.din[9] && DUT_if.din[8] && DUT_if.rx_valid)? 1'b1 : 1'b0;
        end
```

Code Snippets

Top module

```
import uvm_pkg::*;
import RAM_test::*;
`include "uvm_macros.svh"
module RAM_top();
bit clk;
initial begin
      clk = 0;
      forever #1 clk = ~clk;
end
RAM_if RAM_if_inst(clk);
RAM_if_ref RAM_if_ref_inst(clk);
RAM DUT(RAM_if_inst);
RAM_ref REF(RAM_if_ref_inst);
assign RAM_if_ref_inst.din = RAM_if_inst.din;
assign RAM_if_ref_inst.rx_valid = RAM_if_inst.rx_valid;
assign RAM_if_ref_inst.rst_n = RAM_if_inst.rst_n;
bind RAM RAM_sva RAM_sva_inst(RAM_if_inst.DUT);

initial begin
      uvm_config_db#(virtual RAM_if)::set(null, "uvm_test_top", "vif_DUT", RAM_if_inst);
      uvm_config_db#(virtual RAM_if_ref)::set(null, "uvm_test_top", "vif_REF", RAM_if_ref_inst);
      run_test("RAM_test");
end

endmodule
```

Module Interface

```
interface RAM_if(clk);
input clk;
logic [9:0] din;
logic  rst_n, rx_valid;
logic [7:0] dout;
logic tx_valid;
modport DUT (input din, clk, rst_n, rx_valid, output dout, tx_valid);
endinterface
```

Golden Model Interface

```
interface RAM_if_ref(clk);
input clk ;
logic [9:0] din;
logic  rst_n, rx_valid;
logic [7:0] dout;
logic tx_valid;
modport REF (input din, clk, rst_n, rx_valid, output dout, tx_valid);
endinterface
```

Golden Model Module

```
module RAM_ref (RAM_if_ref.REF REF_if);

logic [7:0] MEM [255:0];
logic [7:0] Rd_Addr, Wr_Addr;

always @(posedge REF_if.clk) begin
    if (~REF_if.rst_n) begin
        REF_if.dout <= 0;
        REF_if.tx_valid <= 0;
        Rd_Addr <= 0;
        Wr_Addr <= 0;
    end
    else  begin
        if (REF_if.rx_valid) begin
            case (REF_if.din[9:8])
                2'b00 : Wr_Addr <= REF_if.din[7:0];
                2'b01 : MEM[Wr_Addr] <= REF_if.din[7:0];
                2'b10 : Rd_Addr <= REF_if.din[7:0];
                2'b11 : REF_if.dout <= MEM[Rd_Addr];
                default : REF_if.dout <= 0;
            endcase
        end
        REF_if.tx_valid <= (REF_if.din[9] && REF_if.din[8] && REF_if.rx_valid)? 1'b1 : 1'b0;
        end
end

endmodule
```

RAM_sva

```
module RAM_sva (RAM_if.dut DUT_if);

//==reset assertion
property rst_check;
@(posedge DUT_if.clk) !DUT_if.rst_n |=> ((DUT_if.dout)==8'b00000000 &&DUT_if.tx_valid==0);
endproperty
assert property (rst_check) else $error("Reset assertion failed");
cover property (rst_check);

// ===tx_valid_low assertion====
property tx_valid_low_check;
@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) (DUT_if.din[9:8]!=2'b11) |=> (DUT_if.tx_valid)==0;
endproperty
assert property (tx_valid_low_check) else $error("tx_valid_low assertion failed");
cover property (tx_valid_low_check);

// ===tx_valid_high assertion====
property tx_valid_high_check;
@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) ($past(DUT_if.din[9:8])==2'b10&& DUT_if.din[9:8]==2'b11) |=> ($rose(DUT_if.tx_valid))|=>##[1:$]($fell(DUT_if.tx_valid));
endproperty
assert property (tx_valid_high_check) else $error("tx_valid_high assertion failed");
cover property (tx_valid_high_check);

// ====write address then write data assertion====
property write_addr_data_check;
@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) (DUT_if.din[9:8]==2'b00) |=> ##[1:$] (DUT_if.din[9:8]==2'b01);
endproperty
assert property (write_addr_data_check) else $error("write_addr_data assertion failed");
cover property (write_addr_data_check);

// ====read address then read data assertion====
property read_addr_data_check;
@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) (DUT_if.din[9:8]==2'b10) |=> ##[1:$] (DUT_if.din[9:8]==2'b11);
endproperty
assert property (read_addr_data_check) else $error("read_addr_data assertion failed");
cover property (read_addr_data_check);
endmodule
```

RAM_config

```
package RAM_config;
import uvm_pkg::*;
`include "uvm_macros.svh"
class RAM_config extends uvm_object;
`uvm_object_utils(RAM_config)
virtual RAM_if vif;
virtual RAM_if_ref vif_ref;
 function new(string name="RAM_config");
     super.new(name);
 endfunction
 endclass
 endpackage
```

RAM_sequenceitem

```
package RAM_seq_item;
import uvm_pkg::*;
`include "uvm_macros.svh"
class RAM_seq_item extends uvm_sequence_item;
`uvm_object_utils(RAM_seq_item)
rand logic  [9:0] din;
rand logic rx_valid;
rand logic rst_n;
rand logic [7:0] dout;
rand logic tx_valid;
logic [1:0]old_op=2'b00;

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

function string convert2string();
    return $sformatf("din=%0h, rx_valid=%0b, dout=%0h, tx_valid=%0b", din, rx_valid, dout, tx_valid);
endfunction

// ====constraints====
constraint reset_cn {
      rst_n dist {0:=1, 1:=9};
}

constraint rx_valid_cn {
      rx_valid dist {0:=1, 1:=9};
}

constraint write_cn {
      din[9:8] inside {[0:1]};

}
constraint read_cn {
       if (old_op==2){

            din[9:8]==3;
       }
       else if (old_op==3) {
            din[9:8]==2;
       }
}
constraint read_write_cn {
      if (old_op==2'b00) {
            din[9:8] inside {[0:1]};
      }
      else if (old_op==2'b01) {
           din[9:8] dist {0:=40, 2:=60};
      }
      else if (old_op==2'b10) {
            din[9:8] inside {[2:3]};

      }
      else if (old_op==2'b11) {
               din[9:8] dist {0:=60, 2:=40};

      }


}
function void post_randomize();
old_op=din[9:8];

endfunction


endclass

endpackage
```

RAM_sequencer

```
package RAM_sequencer;
import uvm_pkg::*;
import RAM_seq_item::*;
`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
```

RAM_rst_sequence

```
package RAM_reset_sequence;
import uvm_pkg::*;
`include "uvm_macros.svh"
import RAM_seq_item::*;
class reset_sequence extends uvm_sequence #(RAM_seq_item);
`uvm_object_utils(reset_sequence)
RAM_seq_item seq_item;
function new(string name = "reset_sequence");
    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.din = 0;
seq_item.dout = 0;
seq_item.tx_valid = 0;
finish_item(seq_item);

endtask
endclass
endpackage
```

RAM_write_only_sequence

```
1  package RAM_write_only_sequence;
2  import uvm_pkg::*;
3  `include "uvm_macros.svh"
4  import RAM_seq_item::*;
5  class write_only_sequence extends uvm_sequence #(RAM_seq_item);
6  `uvm_object_utils(write_only_sequence)
7  RAM_seq_item seq_item;
8  function new(string name = "write_only_sequence");
9      super.new(name);
10 endfunction
11
12 task body();
13 repeat(10000) begin
14 seq_item = RAM_seq_item::type_id::create("seq_item");
15 start_item(seq_item);
16 seq_item.constraint_mode(0);
17 seq_item.reset_cn.constraint_mode(1);
18 seq_item.rx_valid_cn.constraint_mode(1);
19 seq_item.write_cn.constraint_mode(1);
20  assert(seq_item.randomize());
21 finish_item(seq_item);
22 end
23
24
25 endtask
26 endclass
27 endpackage
```

RAM_write_only_sequence

```
package RAM_read_only_sequence;
import uvm_pkg::*;
`include "uvm_macros.svh"
import RAM_seq_item::*;
class read_only_sequence extends uvm_sequence #(RAM_seq_item);
`uvm_object_utils(read_only_sequence)
RAM_seq_item seq_item;
function new(string name = "read_only_sequence");
    super.new(name);
endfunction

task body();
seq_item = RAM_seq_item::type_id::create("seq_item");
repeat(100) begin
start_item(seq_item);
seq_item.constraint_mode(0);
seq_item.reset_cn.constraint_mode(1);
seq_item.rx_valid_cn.constraint_mode(1);
seq_item.read_cn.constraint_mode(1);
 assert(seq_item.randomize());
finish_item(seq_item);
end


endtask
endclass
endpackage
```

RAM_read_write_sequence

```
package RAM_read_write_sequence;
import uvm_pkg::*;
`include "uvm_macros.svh"
import RAM_seq_item::*;
class read_write_sequence extends uvm_sequence #(RAM_seq_item);
`uvm_object_utils(read_write_sequence)
RAM_seq_item seq_item;
function new(string name = "read_write_sequence");
    super.new(name);
endfunction

task body();
seq_item = RAM_seq_item::type_id::create("seq_item");
repeat(100) begin
start_item(seq_item);
seq_item.constraint_mode(0);
seq_item.reset_cn.constraint_mode(1);
seq_item.rx_valid_cn.constraint_mode(1);
seq_item.read_write_cn.constraint_mode(1);
 assert(seq_item.randomize());
finish_item(seq_item);
end


endtask
endclass
endpackage
```

RAM_driver

```
package RAM_driver;
import uvm_pkg::*;
import RAM_seq_item::*;
`include "uvm_macros.svh"
class RAM_driver extends uvm_driver #(RAM_seq_item);
`uvm_component_utils(RAM_driver)
virtual RAM_if vif;
RAM_seq_item 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
      seq_item = RAM_seq_item::type_id::create("seq_item");
      seq_item_port.get_next_item(seq_item);
      vif.rst_n = seq_item.rst_n;
      vif.rx_valid = seq_item.rx_valid;
      vif.din = seq_item.din;
      @(negedge vif.clk);
      seq_item_port.item_done();
end

endtask
endclass

endpackage
```

RAM_monitor

```
package RAM_monitor;
import uvm_pkg::*;
import RAM_seq_item::*;
`include "uvm_macros.svh"
class RAM_monitor extends uvm_monitor;
`uvm_component_utils(RAM_monitor)
virtual RAM_if vif;
RAM_seq_item 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
      seq_item = RAM_seq_item::type_id::create("seq_item");
      @(negedge vif.clk);
      seq_item.rst_n = vif.rst_n;
      seq_item.rx_valid = vif.rx_valid;
      seq_item.din = vif.din;
      seq_item.dout = vif.dout;
      seq_item.tx_valid = vif.tx_valid;
      mon_ap.write(seq_item);
end
endtask
endclass


endpackage
```

RAM_monitor_golden_model

```
package RAM_monitor_ref;
import uvm_pkg::*;
`include "uvm_macros.svh"
import RAM_seq_item::*;
import RAM_config::*;
class RAM_monitor_ref extends uvm_monitor;
`uvm_component_utils(RAM_monitor_ref)
virtual RAM_if_ref vif_ref;
RAM_seq_item seq_item;
uvm_analysis_port #(RAM_seq_item) mon_ref_ap;
RAM_config cfg;
function new(string name="RAM_monitor_ref", uvm_component parent=null);
      super.new(name, parent);
endfunction

function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    mon_ref_ap = new("mon_ref_ap", this);
      if(!uvm_config_db#(RAM_config)::get(this, "", "cfg", cfg)) begin
            `uvm_fatal("NOVIF","Virtual interface reference must be set for: via configuration database");
      end
endfunction
function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
      vif_ref = cfg.vif_ref;
endfunction

task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
      seq_item = RAM_seq_item::type_id::create("seq_item");
      @(negedge vif_ref.clk);
      seq_item.rst_n = vif_ref.rst_n;
      seq_item.rx_valid = vif_ref.rx_valid;
      seq_item.din = vif_ref.din;
      seq_item.dout = vif_ref.dout;
      seq_item.tx_valid = vif_ref.tx_valid;
      mon_ref_ap.write(seq_item);
end
endtask

endclass
endpackage
```

RAM_scoreboard

```
package RAM_scoreboard;
import uvm_pkg::*;
import RAM_seq_item::*;
import RAM_config::*;
`include "uvm_macros.svh"
class RAM_scoreboard extends uvm_scoreboard;
`uvm_component_utils(RAM_scoreboard)
int correct_count=0;
int error_count=0;
uvm_analysis_export #(RAM_seq_item) sb_export;
uvm_tlm_analysis_fifo #(RAM_seq_item) sb_fifo;
uvm_analysis_export #(RAM_seq_item) sb_export_ref;
uvm_tlm_analysis_fifo #(RAM_seq_item) sb_fifo_ref;
RAM_seq_item seq_item;
RAM_seq_item seq_item_ref;
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);
      sb_export_ref = new("sb_export_ref", this);
      sb_fifo_ref = new("sb_fifo_ref", this);
endfunction

function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    sb_export.connect(sb_fifo.analysis_export);
      sb_export_ref.connect(sb_fifo_ref.analysis_export);
endfunction

task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
        sb_fifo.get(seq_item);
       sb_fifo_ref.get(seq_item_ref);

      if(seq_item.dout!==seq_item_ref.dout&&seq_item.tx_valid!== seq_item_ref.tx_valid) begin
           `uvm_error("SCOREBOARD", $sformatf("Mismatch Detected! Expected Dout: %0h, Actual Dout: %0h, Expected Tx_Valid: %0b, Actual Tx_Valid: %0b", seq_item.dout, seq_item_ref.dout, seq_item.tx_valid, seq_item_ref.tx_valid));
           error_count++;
      end
      else begin
           correct_count++;
      end
    end
endtask
function void report_phase(uvm_phase phase);
    super.report_phase(phase);
    `uvm_info("SCOREBOARD", $sformatf("Correct Transactions: %0d, Error Transactions: %0d", correct_count, error_count), UVM_LOW);
endfunction

endclass
endpackage
```

RAM_coverage

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

// =====covergroups=====

covergroup COV;
c1:coverpoint seq_item.din[9:8]{
      bins write_addr = {2'b00};
      bins write_data = {2'b01};
      bins read_addr  = {2'b10};
      bins read_data  = {2'b11};
      bins write_addr_data =(0=>1);
      bins read_addr_data  =(2=>3);
      bins write_read = (0[*14]=>1[*14]=>2[*14]=>3[*14]);

}
rx_valid_cp:coverpoint seq_item.rx_valid{
      bins rx_valid_0 = {0};
      bins rx_valid_1 = {1};
}
tx_valid_cp:coverpoint seq_item.tx_valid{
      bins tx_valid_0 = {0};
      bins tx_valid_1 = {1};
}
cross rx_valid_cp,c1{
      option.cross_auto_bin_max=0;
      bins b1 = binsof(rx_valid_cp.rx_valid_0) && binsof(c1.write_addr);
      bins b2 = binsof(rx_valid_cp.rx_valid_1) && binsof(c1.write_data);
      bins b3 = binsof(rx_valid_cp.rx_valid_1) && binsof(c1.read_addr);
      bins b4 = binsof(rx_valid_cp.rx_valid_1) && binsof(c1.read_data);
}
cross tx_valid_cp,c1{
      option.cross_auto_bin_max=0;
      bins read_data_valid = binsof(tx_valid_cp.tx_valid_1) && binsof(c1.read_data);

}

endgroup



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




endclass

endpackage
```

RAM_agent

```
package RAM_agent;
import uvm_pkg::*;
import RAM_driver::*;
import RAM_sequencer::*;
import RAM_monitor::*;
import RAM_config::*;
import RAM_seq_item::*;
`include "uvm_macros.svh"
class RAM_agent extends uvm_agent;
`uvm_component_utils(RAM_agent)
RAM_driver driver;
RAM_sequencer sequencer;
RAM_monitor monitor;
RAM_config cfg;
uvm_analysis_port #(RAM_seq_item) agent_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", cfg)) begin
        `uvm_fatal("NOVIF","Virtual interface must be set for: via configuration database");
    end
    driver = RAM_driver::type_id::create("driver", this);
    sequencer = RAM_sequencer::type_id::create("sequencer", this);
    monitor = RAM_monitor::type_id::create("monitor", this);
      agent_ap = new("agent_ap", this);
endfunction

function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    driver.vif = cfg.vif;
    monitor.vif = cfg.vif;
    monitor.mon_ap.connect(agent_ap);
      driver.seq_item_port.connect(sequencer.seq_item_export);
endfunction
endclass


endpackage
```

SPI_slave_env

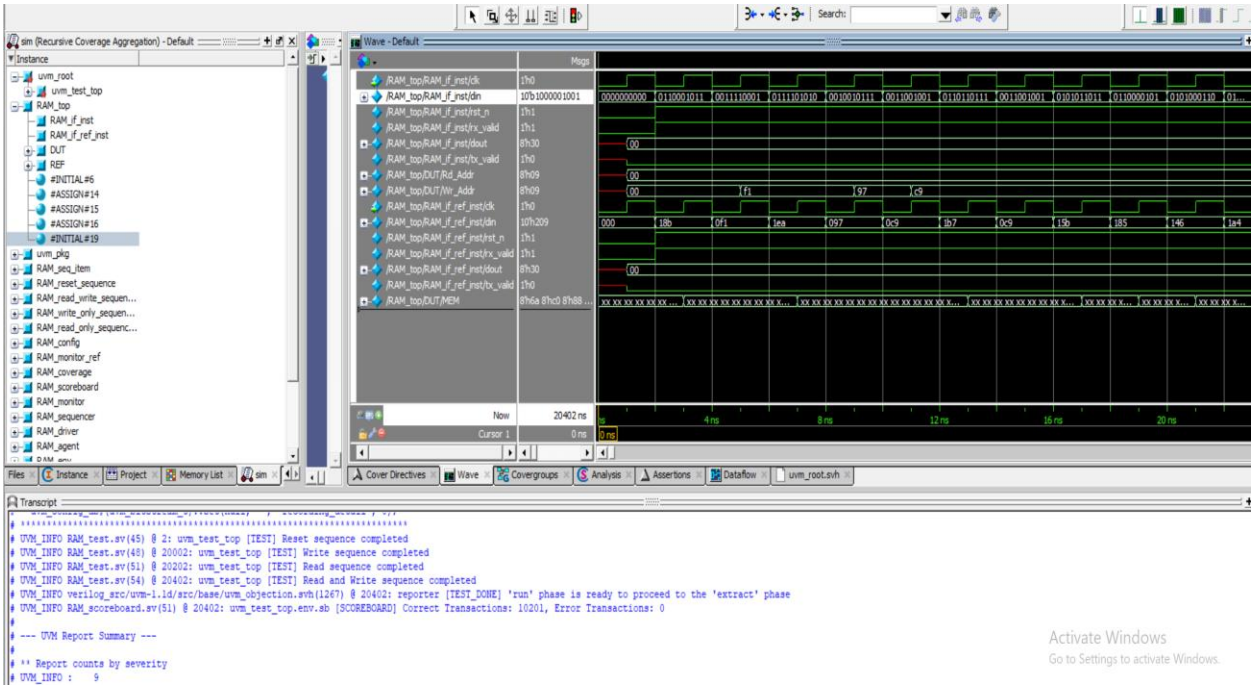
```
package RAM_env;
import uvm_pkg::*;
import RAM_seq_item::*;
import RAM_agent::*;
import RAM_scoreboard::*;
import RAM_coverage::*;
import RAM_monitor_ref::*;
`include "uvm_macros.svh"
class RAM_env extends uvm_env;
`uvm_component_utils(RAM_env)
RAM_agent agent;
RAM_scoreboard sb;
RAM_monitor_ref mon_ref;
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);
    agent = RAM_agent::type_id::create("agent", this);
    sb = RAM_scoreboard::type_id::create("sb", this);
    cov = RAM_coverage::type_id::create("cov", this);
    mon_ref = RAM_monitor_ref::type_id::create("mon_ref", this);
endfunction
function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    agent.agent_ap.connect(sb.sb_export);
    agent.agent_ap.connect(cov.cov_export);
    mon_ref.mon_ref_ap.connect(sb.sb_export_ref);
endfunction

endclass
endpackage
```

SPI_slave_test

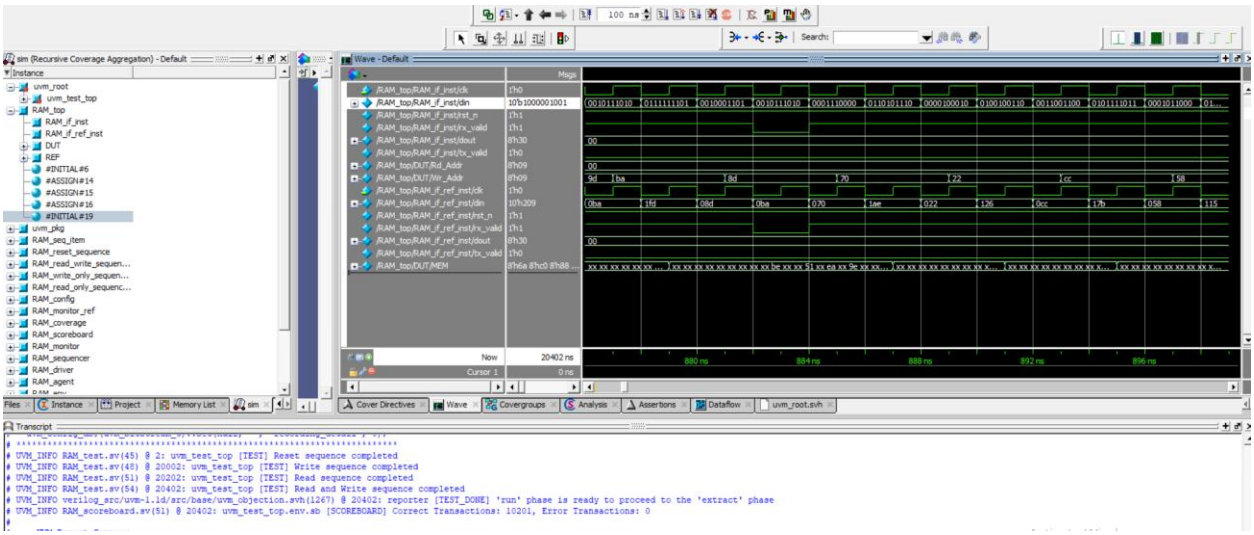
```
package RAM_test;
import uvm_pkg::*;
import RAM_env::*;
import RAM_config::*;
import RAM_read_only_sequence::*;
import RAM_write_only_sequence::*;
import RAM_read_write_sequence::*;
import RAM_reset_sequence::*;
`include "uvm_macros.svh"
class RAM_test extends uvm_test;

`uvm_component_utils(RAM_test)
RAM_env env;
RAM_config cfg;
read_only_sequence read_seq;
write_only_sequence write_seq;
read_write_sequence read_write_seq;
reset_sequence reset_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);
    cfg = RAM_config::type_id::create("cfg", this);
    read_seq = read_only_sequence::type_id::create("read_seq", this);
    write_seq = write_only_sequence::type_id::create("write_seq", this);
    read_write_seq = read_write_sequence::type_id::create("read_write_seq", this);
    reset_seq = reset_sequence::type_id::create("reset_seq", this);
    if(!uvm_config_db#(virtual RAM_if)::get(this, "", "vif_DUT", cfg.vif)) begin
        `uvm_fatal("NOVIF","Virtual interface must be set for:  via configuration database");
    end
    if(!uvm_config_db#(virtual RAM_if_ref)::get(this, "", "vif_REF", cfg.vif_ref)) begin
        `uvm_fatal("NOVIF","Virtual interface reference must be set for: via configuration database");
    end
    uvm_config_db#(RAM_config)::set(this, "*", "cfg", cfg);
endfunction
task run_phase(uvm_phase phase);
    super.run_phase(phase);
    phase.raise_objection(this);
    //reset sequence
    reset_seq.start(env.agent.sequencer);
    `uvm_info("TEST", "Reset sequence completed", UVM_LOW);
    //write sequence
    write_seq.start(env.agent.sequencer);
    `uvm_info("TEST", "Write sequence completed", UVM_LOW);
    //read sequence
    read_seq.start(env.agent.sequencer);
    `uvm_info("TEST", "Read sequence completed", UVM_LOW);
    //read and write sequence
    read_write_seq.start(env.agent.sequencer);
    `uvm_info("TEST", "Read and Write sequence completed", UVM_LOW);

    phase.drop_objection(this);
endtask


endclass
endpackage
```

QuestaSim Snippets

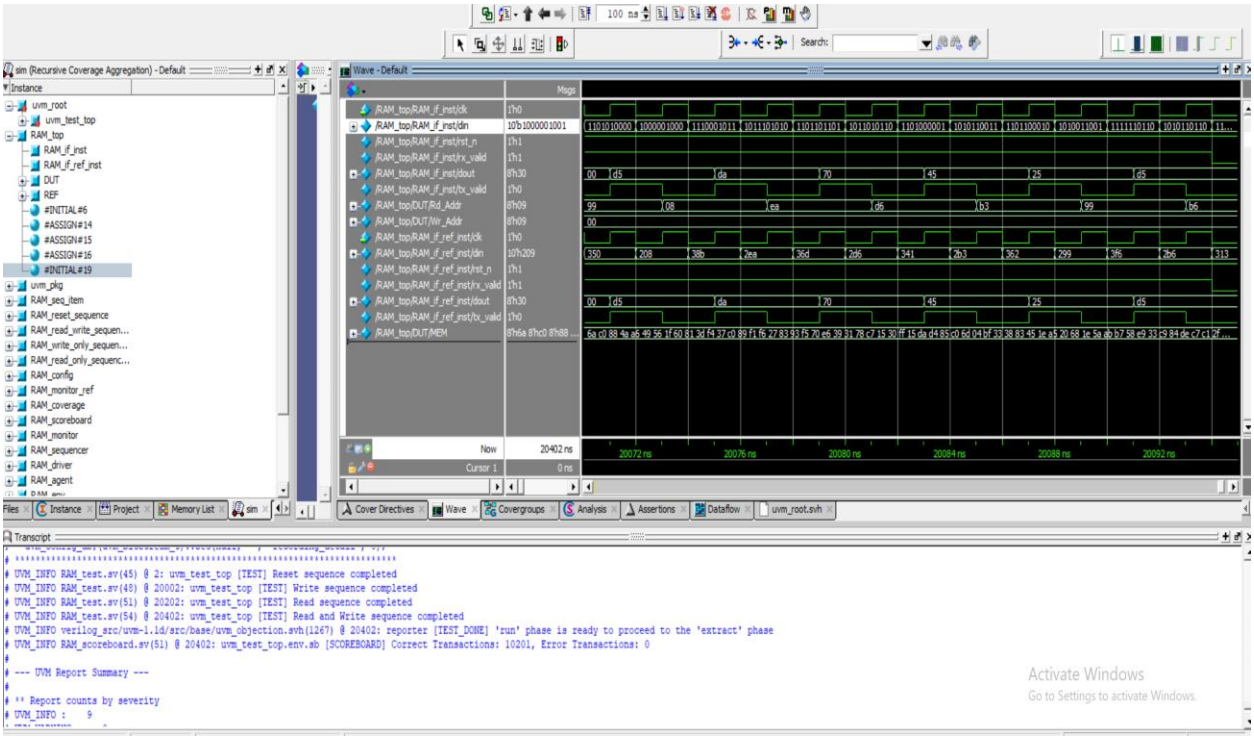
Reset_sequence



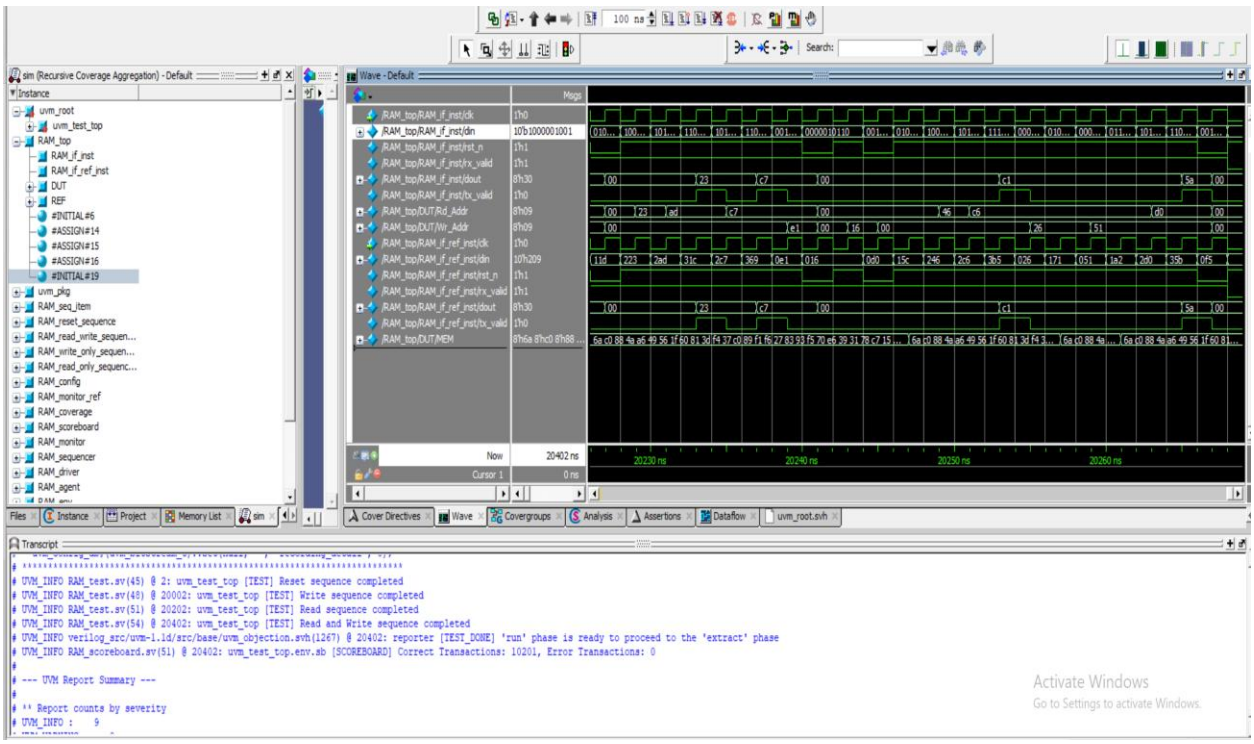
Write_only_sequence



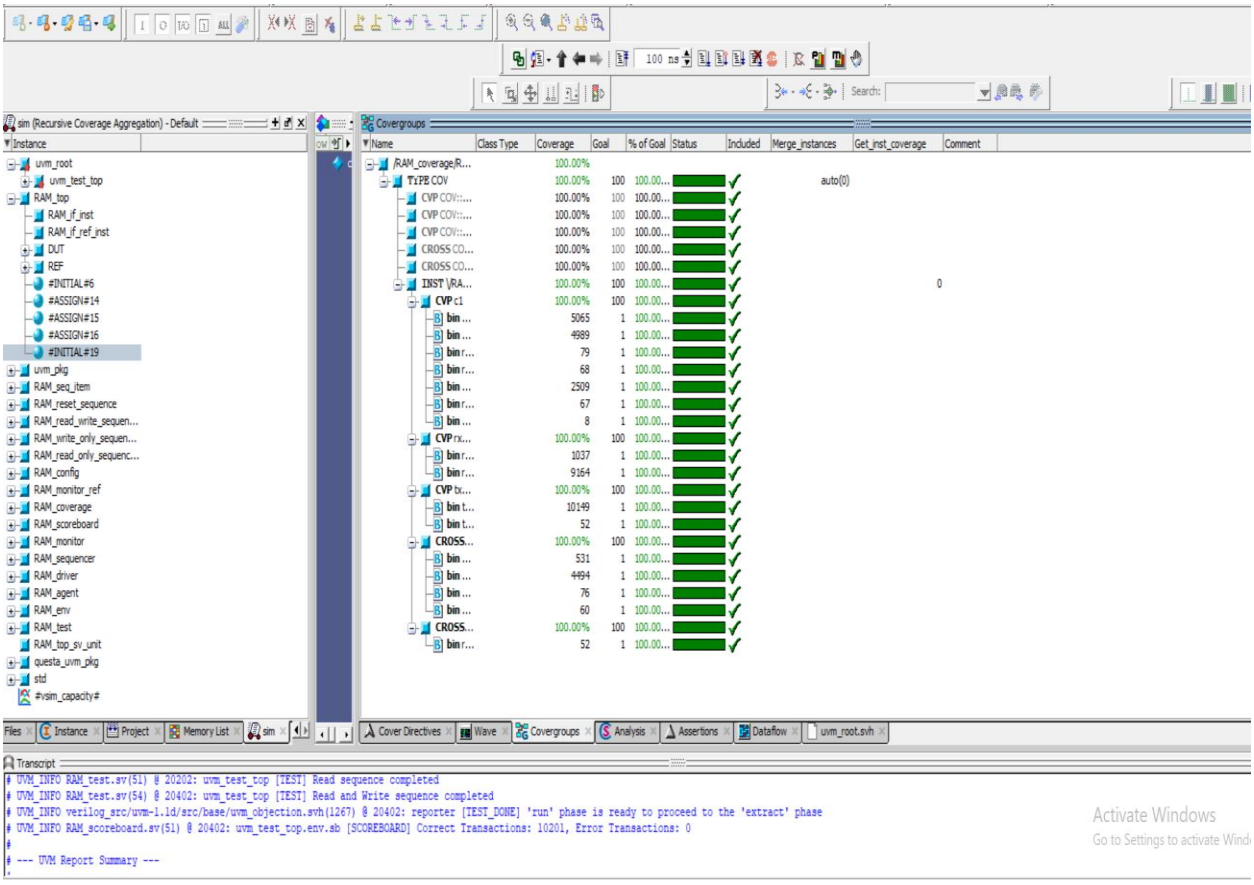
Read_only_sequence



Read_write_sequence



Functional Coverage

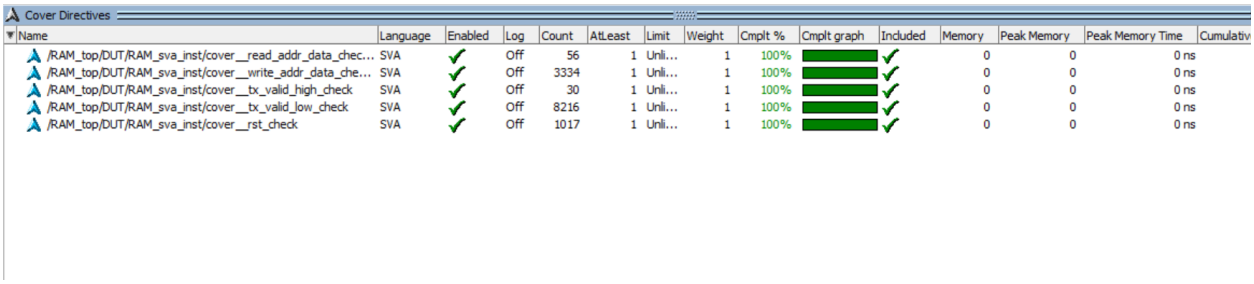


```
=== Instance: /RAM_coverage
=== Design Unit: work.RAM_coverage
================================================================================

Covergroup Coverage:
    Covergroups                 1        na        na   100.00%
        Coverpoints/Crosses     5        na        na        na
            Covergroup Bins    16        16         0   100.00%
----------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------------------
 TYPE /RAM_coverage/RAM_coverage/COV                100.00%        100          -    Covered
    covered/total bins:                                 16         16          -
    missing/total bins:                                  0         16          -
    % Hit:                                          100.00%        100          -
    Coverpoint c1                                   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 #cross__0#                                100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                      100.00%        100          -
    Cross #cross__1#                                100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
 Covergroup instance \/RAM_coverage::RAM_coverage::COV
                                                    100.00%        100          -    Covered
    covered/total bins:                                 16         16          -
    missing/total bins:                                  0         16          -
    % Hit:                                          100.00%        100          -
    Coverpoint c1                                   100.00%        100          -    Covered
        covered/total bins:                               7          7          -
        missing/total bins:                               0          7          -
        % Hit:                                      100.00%        100          -
        bin write_addr                                 5065          1          -    Covered
        bin write_data                                 4989          1          -    Covered
        bin read_addr                                    79          1          -    Covered
        bin read_data                                    68          1          -    Covered
        bin write_addr_data                            2509          1          -    Covered
        bin read_addr_data                               67          1          -    Covered
        bin write_read                                    8          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 rx_valid_0                                 1037          1          -    Covered
        bin rx_valid_1                                 9164          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 tx_valid_0                                10149          1          -    Covered
        bin tx_valid_1                                   52          1          -    Covered
    Cross #cross__0#                                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 b1                                      531          1          -    Covered
            bin b2                                     4494          1          -    Covered
            bin b3                                       76          1          -    Covered
            bin b4                                       60          1          -    Covered
    Cross #cross__1#                                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 read_data_valid                          52          1          -    Covered
```

Assertion Summary

Feature	Assertion
Whenever reset is asserted, the output signals (tx_valid and dout) are low	@(posedge DUT_if.clk) !DUT_if.rst_n |=> ((DUT_if.dout)==8'b00000000 &&DUT_if.tx_valid==0);
During address or data input phases (write_add_seq, write_data_seq, read_add_seq), the tx_valid signal must remain deasserted.	@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) (DUT_if.din[9:8]!=2'b11) |=> (DUT_if.tx_valid)==0;
After a read_data_seq occurs, the tx_valid signal must rise to indicate valid output and after it rises by one clock cycle, it should eventually fall.	@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) ($past(DUT_if.din[9:8])==2'b10&& DUT_if.din[9:8]==2'b11) \|=> ($rose(DUT_if.tx_valid))|=>##[1:$]($fell(DUT_if.tx_valid));
Every Write Address operation must be eventually followed by a Write Data operation.	@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) (DUT_if.din[9:8]==2'b00) |=> ##[1:$] (DUT_if.din[9:8]==2'b01);
Every Read Address operation must be eventually followed by a Read Data operation.	@(posedge DUT_if.clk) disable iff (!DUT_if.rst_n) (DUT_if.din[9:8]==2'b10) |=> ##[1:$] (DUT_if.din[9:8]==2'b11);

Assertion Coverage



Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulativ
/RAM_top/DUT/RAM_sva_inst/cover__read_addr_data_chec...	SVA	✓	Off	56	1	Unli...	1	100%		✓	0	0	0 ns	
/RAM_top/DUT/RAM_sva_inst/cover__write_addr_data_che...	SVA	✓	Off	3334	1	Unli...	1	100%		✓	0	0	0 ns	
/RAM_top/DUT/RAM_sva_inst/cover__tx_valid_high_check	SVA	✓	Off	30	1	Unli...	1	100%		✓	0	0	0 ns	
/RAM_top/DUT/RAM_sva_inst/cover__tx_valid_low_check	SVA	✓	Off	8216	1	Unli...	1	100%		✓	0	0	0 ns	
/RAM_top/DUT/RAM_sva_inst/cover__rst_check	SVA	✓	Off	1017	1	Unli...	1	100%		✓	0	0	0 ns	

```
================================================================================
=== Instance: /RAM_top/DUT/RAM_sva_inst
=== Design Unit: work.RAM_sva
================================================================================

Directive Coverage:
    Directives                       5         5         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                 Design Design   Lang File(Line)      Hits Status
                                     Unit   UnitType
--------------------------------------------------------------------------------------------------
/RAM_top/DUT/RAM_sva_inst/cover__read_addr_data_check
                                     RAM_sva Verilog  SVA  RAM_sva.sv(35)    56 Covered
/RAM_top/DUT/RAM_sva_inst/cover__write_addr_data_check
                                     RAM_sva Verilog  SVA  RAM_sva.sv(28)  3334 Covered
/RAM_top/DUT/RAM_sva_inst/cover__tx_valid_high_check
                                     RAM_sva Verilog  SVA  RAM_sva.sv(21)    30 Covered
/RAM_top/DUT/RAM_sva_inst/cover__tx_valid_low_check
                                     RAM_sva Verilog  SVA  RAM_sva.sv(15)  8216 Covered
/RAM_top/DUT/RAM_sva_inst/cover__rst_check
                                     RAM_sva Verilog  SVA  RAM_sva.sv(8)   1017 Covered

================================================================================
=== Instance: /RAM_coverage
=== Design Unit: work.RAM_coverage
================================================================================
```

Code Coverage

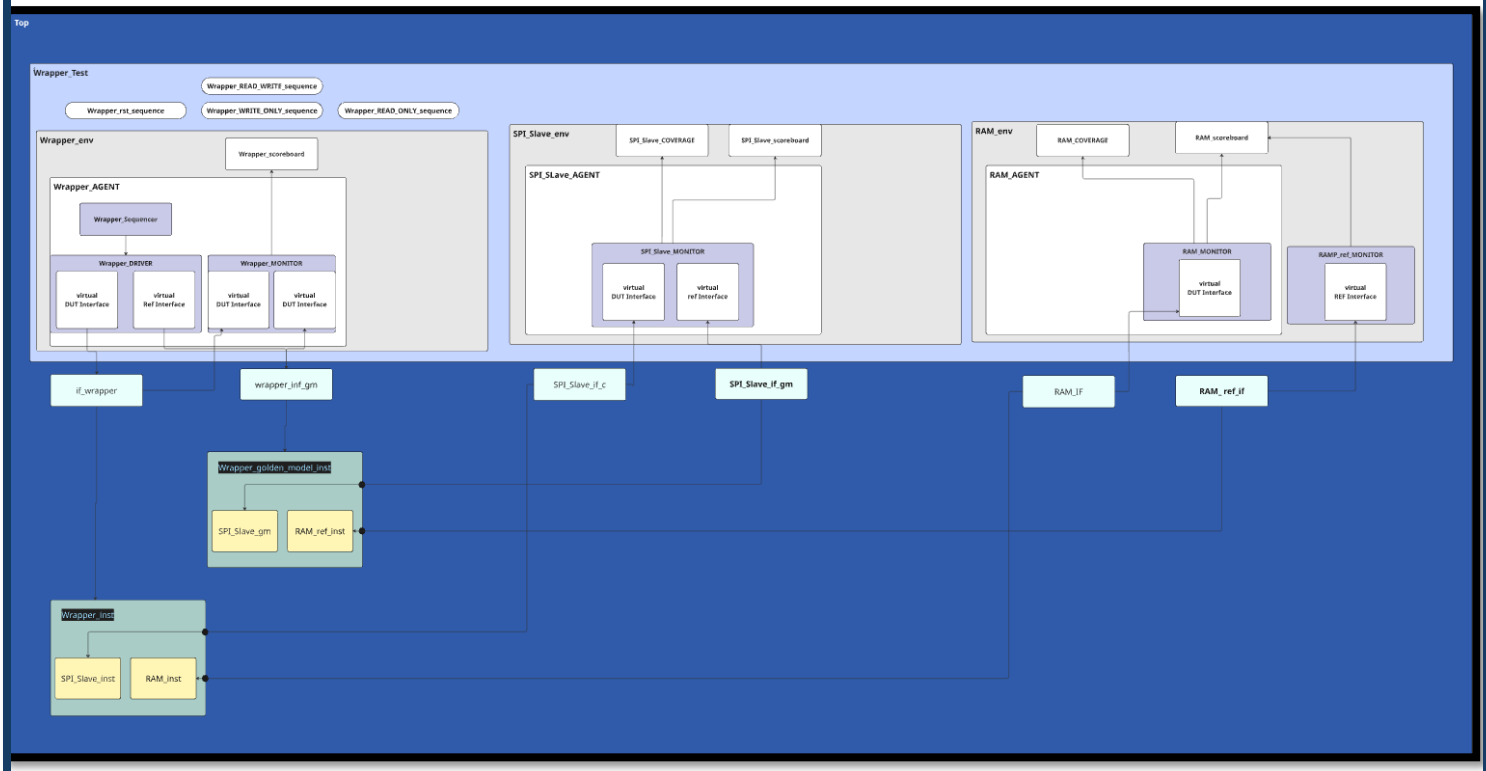
```
=== Instance: /RAM_top/DUT
=== Design Unit: work.RAM
================================================================================
Branch Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Branches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

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

Branch Coverage for instance /RAM_top/DUT

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File RAM.sv
-----------------------------------IF Branch-----------------------------------
    7                                    10201     Count coming in to IF
    7              1                      1017         if (~DUT_if.rst_n) begin

    13             1                      9184         else  begin //fix

Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    14                                   9184     Count coming in to IF
    14             1                     8241             if (DUT_if.rx_valid) begin

                                          943     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

---------------------------------CASE Branch-----------------------------------
    15                                   8241     Count coming in to CASE
    16             1                     4090                      2'b00 : Wr_Addr <= DUT_if.din[7:0];

    17             1                     4071                      2'b01 : MEM[Wr_Addr] <= DUT_if.din[7:0];

    18             1                       46                      2'b10 : Rd_Addr <= DUT_if.din[7:0];

    19             1                       34                      2'b11 : DUT_if.dout <= MEM[Rd_Addr];//fix

Branch totals: 4 hits of 4 branches = 100.00%


Expression Coverage:
    Enabled Coverage             Bins   Covered    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Expressions                     3         3         0   100.00%

```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      10        10         0   100.00%

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

Statement Coverage for instance /RAM_top/DUT --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File RAM.sv
    1                                                module RAM (RAM_if.DUT DUT_if);

    2

    3                                                logic[7:0] MEM [255:0];

    4                                                logic [7:0] Rd_Addr, Wr_Addr;

    5

    6              1                     10201     always @(posedge DUT_if.clk) begin

    7                                                  if (~DUT_if.rst_n) begin

    8              1                      1017            DUT_if.dout <= 0;

    9              1                      1017            DUT_if.tx_valid <= 0;

    10             1                      1017            Rd_Addr <= 0;

    11             1                      1017            Wr_Addr <= 0;

    12                                                 end

    13                                                 else  begin //fix

    14                                                     if (DUT_if.rx_valid) begin

    15                                                         case (DUT_if.din[9:8])

    16             1                      4090                 2'b00 : Wr_Addr <= DUT_if.din[7:0];

    17             1                      4071                 2'b01 : MEM[Wr_Addr] <= DUT_if.din[7:0];

    18             1                        46                 2'b10 : Rd_Addr <= DUT_if.din[7:0];

    19             1                        34                 2'b11 : DUT_if.dout <= MEM[Rd_Addr];//fix

    20                                                         default : DUT_if.dout <= 0;

    21                                                     endcase

    22                                                 end

    23             1                      9184            DUT_if.tx_valid <= (DUT_if.din[9] && DUT_if.din[8] && DUT_if.rx_valid)? 1'b1 : 1'b
0;

```

```
================================================================================
=== Instance: /RAM_top/RAM_if_inst
=== Design Unit: work.RAM_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         44        44         0   100.00%

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

Toggle Coverage for instance /RAM_top/RAM_if_inst --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                           clk           1           1      100.00
                                      din[9-0]           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     =         22
Toggled Node Count   =         22
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (44 of 44 bins)
```

Part 3 SPI-Wrapper Verification

UVM Diagram



The top module instantiates six interfaces, one for each design unit and one for their golden models. Additionally, it instantiates the main SPI wrapper design and its golden “reference” model where they instantiate the SPI & RAM or their golden models, being passed their respective interfaces corresponding to its wrapper. Finally, all interfaces are sent to the configuration database to be retrieved later in the verification process, and it finishes its operation by running the wrapper test using the “run_test” function.

In turn, the wrapper test gets the interfaces from the config db and sends them back in different configuration objects containing their respective virtual interfaces and the “is_active” enum variable that later controls the environment type. The test also creates three environments for each design, which will be elaborated on later, and starts each sequence from the four sequences stated above and shown in the diagram, and sends the constrained sequence items through the sequencer to the driver.

Inside each and every environment, three main things occur, which are creating the scoreboard, the coverage collector, and the UVM Agent.

The scoreboard is mainly responsible for comparing the outputs from the sequence items sent to it with the outputs from reference sequence items for each module through two TLM FIFOs and analysis ports, and prints any mismatches found.

The coverage collector does pretty much the same with taking these sequence items from the TLM FIFOs and through analysis ports, and checking the functional coverage requirements.

The UVM agent of each environment should then get the config objects and see whether these environments are active or passive. If active as the wrapper, then the driver, sequencer, & monitor are created, or if passive, so it is only monitored by them as the SPI & the RAM.

Each agent assigns their existing virtual interfaces to the interfaces from the configuration objects, so the driver can drive the stimulus to the interface, and the monitor can send the response to the scoreboard and the coverage collector.

Finally, until all repetitions end for each sequence, the final objection is dropped from the test, and the UVM test finishes successfully.

RTL Code

```
module Wrapper (Wrapper_inf.dut if_wrapper , SPI_slave_inf.dut if_c , RAM_if.dut if_ram);

    RAM   RAM_instance   (if_ram);
    SPI_slave SLAVE_instance (if_c);

    always @(*) begin
        if_c.MOSI = if_wrapper.MOSI;
        if_c.SS_n = if_wrapper.SS_n;
        if_c.rst_n = if_wrapper.rst_n;
        if_wrapper.MISO = if_c.MISO;
        if_c.rst_n = if_wrapper.rst_n;

        if_ram.din = if_c.rx_data;
        if_ram.rx_valid = if_c.rx_valid;
        if_c.tx_data = if_ram.dout;
        if_c.tx_valid = if_ram.tx_valid;
        if_ram.rst_n = if_wrapper.rst_n;
    end

endmodule
```

No major design changes are made, just some design adjustments to suit the whole verification process.

Code Snippets

Top module

```
`include "uvm_macros.svh"
import uvm_pkg::*;
import Wrapper_test_pkg::*;
module top;

    bit clk;
    initial begin
        clk = 0;
        forever #1 clk = ~clk;
    end

    SPI_slave_inf if_c(.clk(clk));
    SPI_slave_gm_inf if_gm(.clk(clk));
    Wrapper_inf if_wrapper(.clk(clk));
    Wrapper_gm_inf wrapper_inf_gm(.clk(clk));
    RAM_if if_ram(.clk(clk));
    RAM_if_ref RAM_if_ref_inst(.clk(clk));

    Wrapper_golden_model Wrapper_golden_model_inst(wrapper_inf_gm, if_gm, RAM_if_ref_inst);

    Wrapper Wrapper_inst(if_wrapper, if_c, if_ram);

    Wrapper_sva     sva_Wrapper(if_wrapper);
    RAM_sva         sva_RAM(if_ram);
    SPI_slave_sva   sva_SPI_slave(if_c);

    initial begin
        uvm_config_db #(virtual SPI_slave_inf)::set(null, "uvm_test_top", "vif", if_c);
        uvm_config_db #(virtual SPI_slave_gm_inf)::set(null, "uvm_test_top", "vif_gm", if_gm);

        uvm_config_db#(virtual RAM_if)::set(null, "uvm_test_top", "vif_DUT", if_ram);
        uvm_config_db#(virtual RAM_if_ref)::set(null, "uvm_test_top", "vif_REF", RAM_if_ref_inst);

        uvm_config_db#(virtual Wrapper_inf)::set(null, "uvm_test_top", "wrap_vif", if_wrapper);
        uvm_config_db#(virtual Wrapper_gm_inf)::set(null, "uvm_test_top", "wrap_ref", wrapper_inf_gm);

        run_test("Wrapper_test");
    end
endmodule

```

Module Interface

```
interface Wrapper_inf(input clk);
    logic MOSI, SS_n, rst_n;
    logic MISO;

    modport dut (input clk, MOSI, rst_n, SS_n, output MISO);
endinterface
```

Golden Model Interface

```
interface Wrapper_gm_inf(input clk);
    logic MOSI, SS_n,rst_n;
    logic MISO;

    modport dut (input clk, MOSI, rst_n, SS_n, output MISO);
endinterface
```

Golden Model Module

```
module Wrapper_golden_model (Wrapper_gm_inf.dut wrapper_inf_gm , SPI_slave_gm_inf.dut if_gm , RAM_if_ref.dut RAM_if_ref_inst);

    RAM_ref   RAM_instance   (RAM_if_ref_inst);
    SPI_Slave_golden_model SLAVE_instance (if_gm);

    always @(*) begin
        if_gm.MOSI = wrapper_inf_gm.MOSI;
        if_gm.SS_n = wrapper_inf_gm.SS_n;
        if_gm.rst_n = wrapper_inf_gm.rst_n;
        wrapper_inf_gm.MISO = if_gm.MISO;
        if_gm.rst_n = wrapper_inf_gm.rst_n;

        RAM_if_ref_inst.din = if_gm.rx_data;
        RAM_if_ref_inst.rx_valid = if_gm.rx_valid;
        if_gm.tx_data = RAM_if_ref_inst.dout;
        if_gm.tx_valid = RAM_if_ref_inst.tx_valid;
        RAM_if_ref_inst.rst_n = wrapper_inf_gm.rst_n;
    end

endmodule
```

Wrapper_sva

```
module Wrapper_sva (Wrapper_inf.dut if_wrapper);

    property p_reset;
        @(posedge if_wrapper.clk)
        (~if_wrapper.rst_n |=> (if_wrapper.MISO == 'b0))
    endproperty
    assert  property(p_reset);
    cover   property(p_reset);

    property MISO_stable_when_not_read;
        @(posedge if_wrapper.clk)
        disable iff (!if_wrapper.rst_n)    // disable when reset active or slave not selected
        (if_wrapper.SS_n ## 1 !if_wrapper.SS_n ## 1 (if_wrapper.MOSI ## 1 if_wrapper.MOSI ## 1 if_wrapper.MOSI)) |=> $stable(if_wrapper.MISO) throughout (!if_wrapper.SS_n);
    endproperty

    assert  property(MISO_stable_when_not_read);
    cover   property(MISO_stable_when_not_read);
endmodule
```

SPI_RAM_config “Changed to suit the whole verification process”

```
package SPI_RAM_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class SPI_RAM_config extends uvm_object;
        `uvm_object_utils(SPI_RAM_config)

        virtual SPI_slave_inf SPI_vif;
        virtual SPI_slave_gm_inf SPI_vif_gm;
        virtual RAM_if vif;
        virtual RAM_if_ref vif_ref;
        virtual Wrapper_inf wrapper_inf;
        virtual Wrapper_gm_inf wrapper_inf_gm;
        uvm_active_passive_enum is_active;
        function new(string name = "SPI_RAM_config");
            super.new(name);
        endfunction
    endclass
endpackage
```

Wrapper_sequenceitem

```
package Wrapper_sequenceitem_pkg;
    import uvm_pkg::*;
    import shared_package::*;
    `include "uvm_macros.svh"
    typedef enum bit [2:0] {
        WRITE_ADD  = 3'b000,
        WRITE_DATA = 3'b001,
        READ_ADD   = 3'b110,
        READ_DATA  = 3'b111
    } cmd_t;

    class Wrapper_sequenceitem extends uvm_sequence_item;
        `uvm_object_utils(Wrapper_sequenceitem)
        // number of transactions
        cmd_t cmd_seq_old = WRITE_ADD;
        rand cmd_t cmd_seq = WRITE_DATA;
        rand bit [10:0] MOSI_arr;
        logic [10:0] old_MOSI_arr = 11'b11100000000;
        rand logic SS_n = 1;
        rand logic rst_n;
        logic MOSI;
        logic MISO;

        int counter = 0;
        bit [10:0] active_MOSI_word;

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

        function string convert2string();
            return $sformatf("Wrapper_sequenceitem: %s: MOSI=%b, rst_n=%b, SS_n=%b",super.convert2string(), MOSI, rst_n, SS_n);
        endfunction

        function void pre_randomize();
            if(SS_n == 0) begin
                MOSI_arr.rand_mode(0);
                ordering_rules.constraint_mode(0);
                cmd_seq.rand_mode(0);
            end
            else begin
                MOSI_arr.rand_mode(1);
                ordering_rules.constraint_mode(1);
                cmd_seq.rand_mode(1);
            end
        endfunction
```

```
        constraint ordering_rules {

            if(cmd_seq_old == WRITE_ADD){
                cmd_seq inside {WRITE_ADD, WRITE_DATA};
            }
            else if(cmd_seq_old == WRITE_DATA){
                cmd_seq dist {READ_ADD:= 60,WRITE_ADD:= 40};
            }
            else if(cmd_seq_old == READ_ADD){
                cmd_seq == READ_DATA;
            }
            else if(cmd_seq_old == READ_DATA){
                cmd_seq dist {WRITE_ADD:= 60,READ_ADD:= 40};
            }
        }


        // Random payload for remaining bits
        constraint payload {
            MOSI_arr[7:0] inside {[8'h00 : 8'hFF]};
        }

        constraint c1 {
            rst_n dist {0 := 1, 1 := 99};

            if(counter == 23 && old_MOSI_arr[10:8] == 3'b111){
                SS_n == 1;
            }
            if(counter == 13 && old_MOSI_arr[10:8] != 3'b111){
                SS_n == 1;
            }
            if(!(counter == 23 && old_MOSI_arr[10:8] == 3'b111) && !(counter == 13 && old_MOSI_arr[10:8] != 3'b111)){
                SS_n == 0;
            }
        }

        function void post_randomize();
            if (rst_n == 0) begin
                counter = 0;
                SS_n = 1;
            end
            else begin
                // === SPI active (chip selected) ===
                if (SS_n == 0) begin
                    if(counter == 0) begin
                        MOSI_arr = {cmd_seq, MOSI_arr[7:0]};
                        active_MOSI_word = MOSI_arr;
                        old_MOSI_arr = active_MOSI_word;
                        cmd_seq_old = cmd_seq;
                    end
                    else if(counter > 0) begin
                        MOSI = active_MOSI_word[10];

                        active_MOSI_word = active_MOSI_word << 1;
                    end
                    counter++;
                end
                else begin
                    counter = 0;
                end
            end
            $display("MOSI : %b",MOSI);
            $display("MOSI_word : %b",old_MOSI_arr);
            $display("cmd_seq : %b",cmd_seq);
        endfunction

    endclass
endpackage
```

Wrapper_sequencer

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

Wrapper_rst_sequence

```
package Wrapper_rst_sequence_pkg;
    import uvm_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_rst_sequence extends uvm_sequence #(Wrapper_sequenceitem);
        `uvm_object_utils(Wrapper_rst_sequence)
        Wrapper_sequenceitem item;
        function new(string name = "Wrapper_rst_sequence");
            super.new(name);
        endfunction

        task body();
            item = Wrapper_sequenceitem::type_id::create("item");
            start_item(item);
            item.rst_n = 0;
            item.MISO = 0;
            item.SS_n = 1;
            finish_item(item);
        endtask
    endclass
endpackage
```

Wrapper_read_only_sequence

```
package Wrapper_read_only_sequence_pkg;
    import uvm_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_read_only_sequence extends uvm_sequence #(Wrapper_sequenceitem);
        `uvm_object_utils(Wrapper_read_only_sequence)
        Wrapper_sequenceitem item;
        function new(string name = "Wrapper_read_only_sequence");
            super.new(name);
        endfunction

        task body();
            item = Wrapper_sequenceitem::type_id::create("item");

            for (int i = 0; i < 500 ; i++) begin
                start_item(item);
                assert(item.randomize());
                if(item.SS_n == 1) begin
                    if(i % 2 == 0) begin
                        item.active_MOSI_word = {3'b110, item.active_MOSI_word[7:0]};
                    end
                    else begin
                        item.active_MOSI_word = {3'b111, item.active_MOSI_word[7:0]};
                    end
                end
                finish_item(item);
            end
        endtask
    endclass
endpackage
```

Wrapper_write_only_sequence

```
package Wrapper_write_only_sequence_pkg;
    import uvm_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_write_only_sequence extends uvm_sequence #(Wrapper_sequenceitem);
        `uvm_object_utils(Wrapper_write_only_sequence)
        Wrapper_sequenceitem item;
        function new(string name = "Wrapper_write_only_sequence");
            super.new(name);
        endfunction

        task body();
            item = Wrapper_sequenceitem::type_id::create("item");
            for (int i = 0; i < 500 ; i++) begin
                start_item(item);
                assert(item.randomize());
                if(item.SS_n == 1) begin
                    if(i % 2 == 0) begin
                        item.active_MOSI_word = {3'b000, item.active_MOSI_word[7:0]};
                    end
                    else begin
                        item.active_MOSI_word = {3'b001, item.active_MOSI_word[7:0]};
                    end
                end
                finish_item(item);
            end
        endtask
    endclass
endpackage
```

Wrapper_read_write_sequence

```
package Wrapper_read_write_sequence_pkg;
    import uvm_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_read_write_sequence extends uvm_sequence #(Wrapper_sequenceitem);
        `uvm_object_utils(Wrapper_read_write_sequence)
        Wrapper_sequenceitem item;
        function new(string name = "Wrapper_read_write_sequence");
            super.new(name);
        endfunction

        task body();
            item = Wrapper_sequenceitem::type_id::create("item");
            for (int i = 0; i < 10000 ; i++) begin
                start_item(item);
                assert(item.randomize());
                finish_item(item);
            end
        endtask
    endclass
endpackage
```

Wrapper_driver

```
package Wrapper_driver_pkg;
    import uvm_pkg::*;
    import SPI_RAM_config_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_driver extends uvm_driver #(Wrapper_sequenceitem);
        `uvm_component_utils(Wrapper_driver)
        virtual Wrapper_inf Wrapper_driver_inf;
        virtual Wrapper_gm_inf Wrapper_driver_inf_gm;
        Wrapper_sequenceitem 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);
            `uvm_info("Wrapper_driver", "inside the Wrapper driver", UVM_LOW);
            item = Wrapper_sequenceitem::type_id::create("item");
            forever begin
                seq_item_port.get_next_item(item);
                Wrapper_driver_inf.MOSI = item.MOSI;
                Wrapper_driver_inf.rst_n = item.rst_n;
                Wrapper_driver_inf.SS_n = item.SS_n;

                Wrapper_driver_inf_gm.MOSI = item.MOSI;
                Wrapper_driver_inf_gm.rst_n = item.rst_n;
                Wrapper_driver_inf_gm.SS_n = item.SS_n;

                seq_item_port.item_done();
                @(negedge Wrapper_driver_inf.clk);
                item.MISO = Wrapper_driver_inf.MISO;
                `uvm_info("Wrapper_driver", item.convert2string(), UVM_LOW);
            end
        endtask

    endclass
endpackage
```

Wrapper_monitor

```
package Wrapper_monitor_pkg;
    import uvm_pkg::*;
    import SPI_RAM_config_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_monitor extends uvm_monitor;
        `uvm_component_utils(Wrapper_monitor)
        virtual Wrapper_inf Wrapper_monitor_inf;
        virtual Wrapper_gm_inf Wrapper_monitor_inf_gm;
        Wrapper_sequenceitem item;
        Wrapper_sequenceitem item_gm;
        uvm_analysis_port#(Wrapper_sequenceitem) mon_ap;
        uvm_analysis_port#(Wrapper_sequenceitem) mon_ap_gm;

        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);
            mon_ap_gm = new("mon_ap_gm", this);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            `uvm_info("Wrapper_monitor", "inside the Wrapper monitor", UVM_LOW);
            forever begin
                @(negedge Wrapper_monitor_inf.clk);

                item_gm = Wrapper_sequenceitem::type_id::create("item_gm");
                item_gm.MOSI = Wrapper_monitor_inf_gm.MOSI;
                item_gm.rst_n = Wrapper_monitor_inf_gm.rst_n;
                item_gm.SS_n = Wrapper_monitor_inf_gm.SS_n;
                item_gm.MISO = Wrapper_monitor_inf_gm.MISO;


                item = Wrapper_sequenceitem::type_id::create("item");
                item.MOSI = Wrapper_monitor_inf.MOSI;
                item.rst_n = Wrapper_monitor_inf.rst_n;
                item.SS_n = Wrapper_monitor_inf.SS_n;
                item.MISO = Wrapper_monitor_inf.MISO;

                mon_ap_gm.write(item_gm);
                mon_ap.write(item);
                `uvm_info("Wrapper_monitor", item.convert2string(), UVM_LOW);
            end
        endtask

    endclass
endpackage
```

Wrapper_scoreboard

```
package Wrapper_scoreboard_pkg;
    import uvm_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(Wrapper_scoreboard)
        uvm_analysis_export #(Wrapper_sequenceitem) sb_export;
        uvm_tlm_analysis_fifo #(Wrapper_sequenceitem) sb_fifo;
        uvm_analysis_export #(Wrapper_sequenceitem) sb_export_gm;
        uvm_tlm_analysis_fifo #(Wrapper_sequenceitem) sb_fifo_gm;
        Wrapper_sequenceitem item;
        Wrapper_sequenceitem item_gm;

        int correct_count = 0 , error_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_fifo = new("sb_fifo", this);
            sb_export = new("sb_export", this);
            sb_fifo_gm = new("sb_fifo_gm", this);
            sb_export_gm = new("sb_export_gm", this);
            sb_export.connect(sb_fifo.analysis_export);
            sb_export_gm.connect(sb_fifo_gm.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(item);
                sb_fifo_gm.get(item_gm);

                if(item.MISO === item_gm.MISO) begin
                    correct_count = correct_count + 1;
                end
                else begin
                    `uvm_error("run_phase", $sformatf("MISO = %d", item.MISO));
                    `uvm_error("run_phase", $sformatf("MISO = %d", item_gm.MISO));
                    error_count = error_count + 1;
                end
            end
        endtask

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("correct_count = %d", correct_count), UVM_LOW);
            `uvm_info("report_phase", $sformatf("error_count = %d", error_count), UVM_LOW);
        endfunction
    endclass
endpackage
```

Wrapper_agent

```
package Wrapper_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import SPI_RAM_config_pkg::*;
    import Wrapper_sequencer_pkg::*;
    import Wrapper_driver_pkg::*;
    import Wrapper_monitor_pkg::*;
    import Wrapper_sequenceitem_pkg::*;
    class Wrapper_agent extends uvm_agent;
        `uvm_component_utils(Wrapper_agent)
        Wrapper_driver driver;
        Wrapper_monitor monitor;
        Wrapper_sequencer sequencer;
        SPI_RAM_config cfg;
        uvm_analysis_port #(Wrapper_sequenceitem) agt_ap;
        uvm_analysis_port #(Wrapper_sequenceitem) agt_ap_gm;

        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#(SPI_RAM_config)::get(this, "", "wrapper_cfg", cfg)) begin
                `uvm_fatal("Wrapper_agent", "virtual interface must be set for Wrapper_agent")
            end
            if(cfg.is_active == UVM_ACTIVE) begin
                sequencer = Wrapper_sequencer::type_id::create("sequencer", this);
                driver = Wrapper_driver::type_id::create("driver", this);
            end
            monitor = Wrapper_monitor::type_id::create("monitor", this);
            agt_ap = new("agt_ap", this);
            agt_ap_gm = new("agt_ap_gm", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            driver.Wrapper_driver_inf = cfg.wrapper_inf;
            driver.Wrapper_driver_inf_gm = cfg.wrapper_inf_gm;
            monitor.Wrapper_monitor_inf = cfg.wrapper_inf;
            monitor.Wrapper_monitor_inf_gm = cfg.wrapper_inf_gm;
            driver.seq_item_port.connect(sequencer.seq_item_export);
            monitor.mon_ap.connect(agt_ap);
            monitor.mon_ap_gm.connect(agt_ap_gm);
        endfunction

    endclass

endpackage
```

Wrapper_env

```
package Wrapper_env_pkg;
    import uvm_pkg::*;
    import Wrapper_agent_pkg::*;
    import Wrapper_scoreboard_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_env extends uvm_env;
        `uvm_component_utils(Wrapper_env)
        Wrapper_scoreboard scoreboard;
        Wrapper_agent agent;
        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);
            agent = Wrapper_agent::type_id::create("agent", this);
            scoreboard = Wrapper_scoreboard::type_id::create("scoreboard", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agent.agt_ap.connect(scoreboard.sb_export);
            agent.agt_ap_gm.connect(scoreboard.sb_export_gm);
        endfunction
    endclass
endpackage
```

Wrapper_test

```
package Wrapper_test_pkg;
    import uvm_pkg::*;
    import SPI_slave_env_pkg::*;
    import SPI_RAM_config_pkg::*;
    import Wrapper_write_only_sequence_pkg::*;
    import Wrapper_read_only_sequence_pkg::*;
    import Wrapper_rst_sequence_pkg::*;
    import Wrapper_read_write_sequence_pkg::*;
    import RAM_env::*;
    import Wrapper_env_pkg::*;
    `include "uvm_macros.svh"
    class Wrapper_test extends uvm_test;
        `uvm_component_utils(Wrapper_test)

        SPI_slave_env spi_env;
        RAM_env env;
        Wrapper_env wrapper_env;
        SPI_RAM_config spi_cfg;
        SPI_RAM_config ram_cfg;
        SPI_RAM_config wrapper_cfg;
        Wrapper_write_only_sequence write_only_seq;
        Wrapper_read_only_sequence read_only_seq;
        Wrapper_rst_sequence rst_seq;
        Wrapper_read_write_sequence read_write_seq;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            spi_env = SPI_slave_env::type_id::create("spi_env", this);
            env = RAM_env::type_id::create("env", this);
            wrapper_env = Wrapper_env::type_id::create("wrapper_env", this);
            spi_cfg = SPI_RAM_config::type_id::create("spi_cfg");
            ram_cfg = SPI_RAM_config::type_id::create("ram_cfg");
            wrapper_cfg = SPI_RAM_config::type_id::create("wrapper_cfg");
            // create the rst and main sequence
            rst_seq = Wrapper_rst_sequence::type_id::create("rst_seq");
            write_only_seq = Wrapper_write_only_sequence::type_id::create("write_only_seq");
            read_only_seq = Wrapper_read_only_sequence::type_id::create("read_only_seq");
            read_write_seq = Wrapper_read_write_sequence::type_id::create("read_write_seq");
```

```
            if (!uvm_config_db#(virtual SPI_slave_inf)::get(this, "", "vif", spi_cfg.SPI_vif)) begin
                `uvm_fatal("Wrapper_test", "virtual interface must be set for Wrapper_test")
            end
            if (!uvm_config_db#(virtual SPI_slave_gm_inf)::get(this, "", "vif_gm", spi_cfg.SPI_vif_gm)) begin
                `uvm_fatal("Wrapper_test", "virtual interface must be set for Wrapper_test")
            end

            if (!uvm_config_db#(virtual Wrapper_inf)::get(this, "", "wrap_vif", wrapper_cfg.wrapper_inf)) begin
                `uvm_fatal("Wrapper_test", "virtual interface must be set for Wrapper_test")
            end
            if (!uvm_config_db#(virtual Wrapper_gm_inf)::get(this, "", "wrap_ref", wrapper_cfg.wrapper_inf_gm)) begin
                `uvm_fatal("Wrapper_test", "virtual interface must be set for Wrapper_test")
            end

            if(!uvm_config_db#(virtual RAM_if)::get(this, "", "vif_DUT", ram_cfg.vif)) begin
                `uvm_fatal("NOVIF","Virtual interface must be set for:  via configuration database");
            end
            if(!uvm_config_db#(virtual RAM_if_ref)::get(this, "", "vif_REF", ram_cfg.vif_ref)) begin
                `uvm_fatal("NOVIF","Virtual interface reference must be set for: via configuration database");
            end

            spi_cfg.is_active = UVM_PASSIVE;
            ram_cfg.is_active = UVM_PASSIVE;
            wrapper_cfg.is_active = UVM_ACTIVE;

            uvm_config_db#(SPI_RAM_config)::set(this, "*", "SPI_slave_config_test", spi_cfg);
            uvm_config_db#(SPI_RAM_config)::set(this, "*", "cfg", ram_cfg);
            uvm_config_db#(SPI_RAM_config)::set(this, "*", "wrapper_cfg", wrapper_cfg);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("Wrapper_test", "rst_sequence start", UVM_LOW);
            rst_seq.start(wrapper_env.agent.sequencer);
            `uvm_info("Wrapper_test", "rst_sequence end", UVM_LOW);
            `uvm_info("Wrapper_test", "write_only_sequence start", UVM_LOW);
            write_only_seq.start(wrapper_env.agent.sequencer);
            `uvm_info("Wrapper_test", "write_only_sequence end", UVM_LOW);
            `uvm_info("Wrapper_test", "read_only_sequence start", UVM_LOW);
            read_only_seq.start(wrapper_env.agent.sequencer);
            `uvm_info("Wrapper_test", "read_only_sequence end", UVM_LOW);
            `uvm_info("Wrapper_test", "read_write_sequence start", UVM_LOW);
            read_write_seq.start(wrapper_env.agent.sequencer);
            `uvm_info("Wrapper_test", "read_write_sequence end", UVM_LOW);
            phase.drop_objection(this);
        endtask
    endclass
endpackage
```

Previous File changes to accommodate for the full environment

```
function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            if(!uvm_config_db#(SPI_RAM_config)::get(this, "", "SPI_slave_config_test", cfg)) begin
                `uvm_fatal("SPI_slave_agent", "virtual interface must be set for SPI_slave_agent")
            end
            if(cfg.is_active == UVM_ACTIVE) begin
                sequencer = SPI_slave_sequencer::type_id::create("sequencer", this);
                driver = SPI_slave_driver::type_id::create("driver", this);
            end
            monitor = SPI_slave_monitor::type_id::create("monitor", this);
            agt_ap = new("agt_ap", this);
            agt_ap_gm = new("agt_ap_gm", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            monitor.SPI_slave_monitor_inf = cfg.SPI_vif;
            monitor.SPI_slave_monitor_inf_gm = cfg.SPI_vif_gm;
            if(cfg.is_active == UVM_ACTIVE) begin
                driver.seq_item_port.connect(sequencer.seq_item_export);
                driver.SPI_slave_driver_inf = cfg.SPI_vif;
                driver.SPI_slave_driver_inf_gm = cfg.SPI_vif_gm;
            end
            monitor.mon_ap.connect(agt_ap);
            monitor.mon_ap_gm.connect(agt_ap_gm);
        endfunction
```

A section that shows the changes performed on the SPI_SLAVE agent to accommodate for the passive creation of these evnvironments.

```
function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db#(SPI_RAM_config)::get(this, "", "cfg", cfg)) begin
        `uvm_fatal("NOVIF","Virtual interface must be set for: via configuration database");
    end
    if(cfg.is_active == UVM_ACTIVE) begin
        sequencer = RAM_sequencer::type_id::create("sequencer", this);
        driver = RAM_driver::type_id::create("driver", this);
    end
    monitor = RAM_monitor::type_id::create("monitor", this);
      agent_ap = new("agent_ap", this);
endfunction

function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    monitor.vif = cfg.vif;
    monitor.mon_ap.connect(agent_ap);
    if(cfg.is_active == UVM_ACTIVE) begin
        driver.vif = cfg.vif;
        driver.seq_item_port.connect(sequencer.seq_item_export);
    end
endfunction
endclass
```

A section that shows the changes performed on the RAM agent to accommodate for the passive creation of these evnvironments.

Source Files List

run.do | src_files.list

```
SPI_slave.sv
RAM.sv
Wrapper.sv
SPI_Slave_golden_model.sv
RAM_ref.sv
Wrapper_golden_model.sv
SPI_slave_inf.sv
RAM_if.sv
Wrapper_inf.sv
SPI_slave_gm_inf.sv
RAM_if_ref.sv
Wrapper_gm_inf.sv
shared_package.sv
SPI_slave_sva.sv
RAM_sva.sv
Wrapper_sva.sv
SPI_RAM_config.sv
SPI_slave_sequenceitem.sv
RAM_seq_item.sv
Wrapper_sequenceitem.sv
SPI_slave_sequencer.sv
RAM_sequencer.sv
Wrapper_sequencer.sv
SPI_slave_driver.sv
RAM_driver.sv
Wrapper_driver.sv
SPI_slave_monitor.sv
RAM_monitor.sv
RAM_monitor_ref.sv
Wrapper_monitor.sv
SPI_slave_scoreboard.sv
RAM_scoreboard.sv
Wrapper_scoreboard.sv
SPI_slave_coverage.sv
RAM_coverage.sv
SPI_slave_agent.sv
RAM_agent.sv
Wrapper_agent.sv
SPI_slave_env.sv
RAM_env.sv
Wrapper_env.sv
top.sv
```

DO File

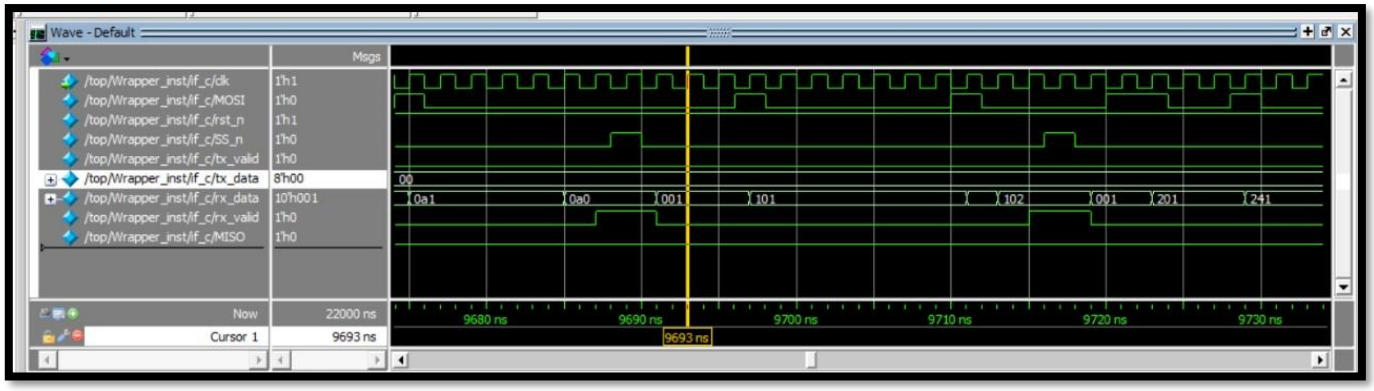
run.do | src_files.list

```
vlib work
vlog -f src_files.list +define+SIM -cover bcesf +cover +acc -covercells
vsim -coverage -voptargs=+acc work.top -classdebug -uvmcontrol=all -sv_seed random
coverage save top.ucdb -onexit
#add wave -position insertpoint sim:/top/if_wrapper/*
add wave -position insertpoint sim:/top/Wrapper_inst/if_c/*
run 0
run -all
```

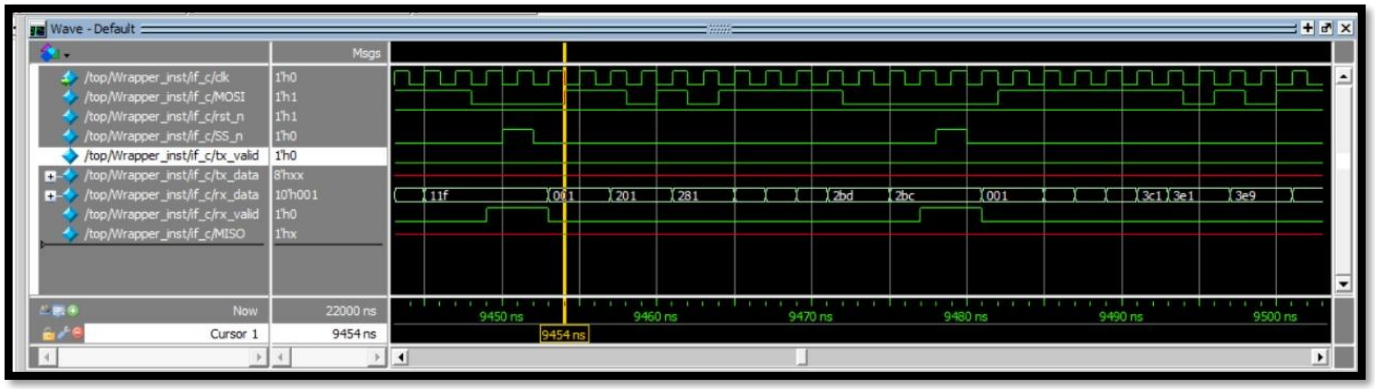
QuestaSim Snippets

Showcasing the sequence “000”

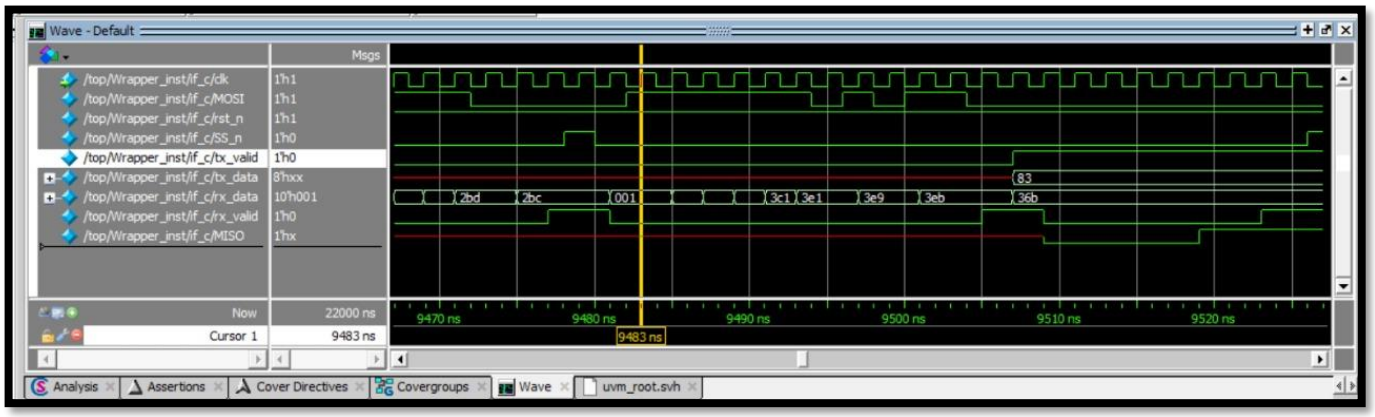
Showcasing the sequence “001”



Showcasing the sequence “110”

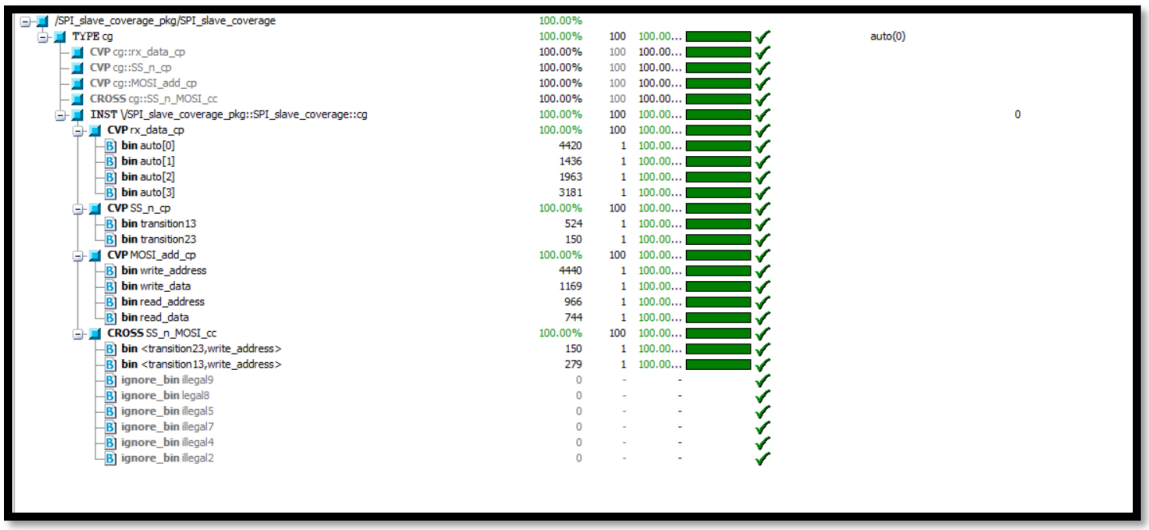


Showcasing the sequence “111”

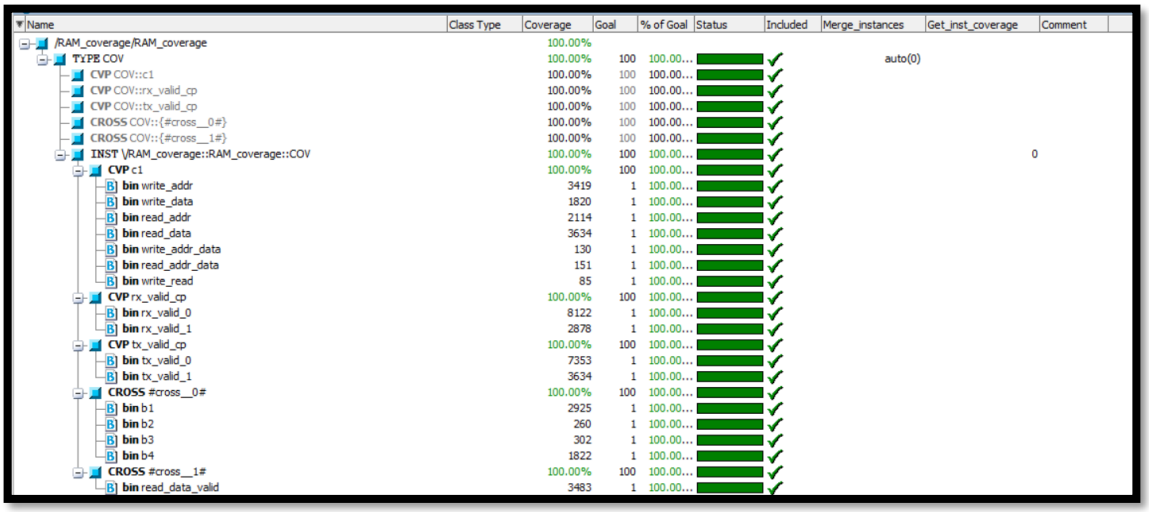



```
# UVM_INFO SPI_slave_monitor.sv(54) @ 22000: uvm_test_top.spi_env.agent.monitor [SPI_slave_monitor] SPI_slave_sequenceitem: : MOSI=1,
#             rst_n=1, SS_n=0, tx_valid=1, tx_data=11010110, rx_data=0010110001,
#             rx_valid=0, MISO=0
# UVM_INFO Wrapper_driver.sv(32) @ 22000: uvm_test_top.wrapper_env.agent.driver [Wrapper_driver] Wrapper_sequenceitem: : MOSI=1, rst_n=1, SS_n=0
# UVM_INFO Wrapper_monitor.sv(46) @ 22000: uvm_test_top.wrapper_env.agent.monitor [Wrapper_monitor] Wrapper_sequenceitem: : MOSI=1, rst_n=1, SS_n=0
# MOSI : 1
# MOSI_word : 00010110000
# cmd_seq : 000
# UVM_INFO D:/Summer2025/Digital_Verifcation/Saeed Nabawy_Project2/Wrapper/Wrapper_test.sv(87) @ 22000: uvm_test_top [Wrapper_test] read_write_sequence end
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 22000: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO RAM_scoreboard.sv(51) @ 22000: uvm_test_top.env.sb [report_phase] correct_count =       11000
# UVM_INFO RAM_scoreboard.sv(52) @ 22000: uvm_test_top.env.sb [report_phase] error_count =           0
# UVM_INFO SPI_slave_scoreboard.sv(50) @ 22000: uvm_test_top.spi_env.scoreboard [report_phase] correct_count =       11000
# UVM_INFO SPI_slave_scoreboard.sv(51) @ 22000: uvm_test_top.spi_env.scoreboard [report_phase] error_count =           0
# UVM_INFO Wrapper_scoreboard.sv(50) @ 22000: uvm_test_top.wrapper_env.scoreboard [report_phase] correct_count =       11000
# UVM_INFO Wrapper_scoreboard.sv(51) @ 22000: uvm_test_top.wrapper_env.scoreboard [report_phase] error_count =           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :33021
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SPI_slave_monitor] 11001
# [TEST_DONE]     1
# [Wrapper_driver] 11001
# [Wrapper_monitor] 11001
# [Wrapper_test]     8
# [report_phase]     6
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 22 us  Iteration: 65  Instance: /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
```

Functional Coverage



Name	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/SPI_slave_coverage_pkg/SPI_slave_coverage	100.00%							
TYPE cg	100.00%	100	100.00...			auto(0)		
CVP cg::rx_data_cp	100.00%	100	100.00...					
CVP cg::SS_n_cp	100.00%	100	100.00...					
CVP cg::MOSI_add_cp	100.00%	100	100.00...					
CROSS cg::SS_n_MOSI_cc	100.00%	100	100.00...					
INST \SPI_slave_coverage_pkg::SPI_slave_coverage::cg	100.00%	100	100.00...					0
CVP rx_data_cp	100.00%	100	100.00...					
bin auto[0]	4420	1	100.00...					
bin auto[1]	1436	1	100.00...					
bin auto[2]	1963	1	100.00...					
bin auto[3]	3181	1	100.00...					
CVP SS_n_cp	100.00%	100	100.00...					
bin transition13	524	1	100.00...					
bin transition23	150	1	100.00...					
CVP MOSI_add_cp	100.00%	100	100.00...					
bin write_address	4440	1	100.00...					
bin write_data	1169	1	100.00...					
bin read_address	966	1	100.00...					
bin read_data	744	1	100.00...					
CROSS SS_n_MOSI_cc	100.00%	100	100.00...					
bin <transition23,write_address>	150	1	100.00...					
bin <transition13,write_address>	279	1	100.00...					
ignore_bin illegal9	0	-	-					
ignore_bin legal8	0	-	-					
ignore_bin illegal5	0	-	-					
ignore_bin illegal7	0	-	-					
ignore_bin illegal4	0	-	-					
ignore_bin illegal2	0	-	-					



Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/RAM_coverage/RAM_coverage		100.00%							
TYPE COV		100.00%	100	100.00...			auto(0)		
CVP COV::c1		100.00%	100	100.00...					
CVP COV::rx_valid_cp		100.00%	100	100.00...					
CVP COV::tx_valid_cp		100.00%	100	100.00...					
CROSS COV::{#cross__0#}		100.00%	100	100.00...					
CROSS COV::{#cross__1#}		100.00%	100	100.00...					
INST \RAM_coverage::RAM_coverage::COV		100.00%	100	100.00...					0
CVP c1		100.00%	100	100.00...					
bin write_addr		3419	1	100.00...					
bin write_data		1820	1	100.00...					
bin read_addr		2114	1	100.00...					
bin read_data		3634	1	100.00...					
bin write_addr_data		130	1	100.00...					
bin read_addr_data		151	1	100.00...					
bin write_read		85	1	100.00...					
CVP rx_valid_cp		100.00%	100	100.00...					
bin rx_valid_0		8122	1	100.00...					
bin rx_valid_1		2878	1	100.00...					
CVP tx_valid_cp		100.00%	100	100.00...					
bin tx_valid_0		7353	1	100.00...					
bin tx_valid_1		3634	1	100.00...					
CROSS #cross__0#		100.00%	100	100.00...					
bin b1		2925	1	100.00...					
bin b2		260	1	100.00...					
bin b3		302	1	100.00...					
bin b4		1822	1	100.00...					
CROSS #cross__1#		100.00%	100	100.00...					
bin read_data_valid		3483	1	100.00...					

```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------
Covergroup                                              Metric       Goal       Bins    Status
----------------------------------------------------------------------------------------------------
 TYPE /RAM_coverage/RAM_coverage/COV                   100.00%        100          -    Covered
    covered/total bins:                                     16         16          -
    missing/total bins:                                      0         16          -
    % Hit:                                              100.00%        100          -
    Coverpoint c1                                       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 #cross__0#                                    100.00%        100          -    Covered
        covered/total bins:                                  4          4          -
        missing/total bins:                                  0          4          -
        % Hit:                                          100.00%        100          -
    Cross #cross__1#                                    100.00%        100          -    Covered
        covered/total bins:                                  1          1          -
        missing/total bins:                                  0          1          -
        % Hit:                                          100.00%        100          -
 Covergroup instance \/RAM_coverage::RAM_coverage::COV
                                                        100.00%        100          -    Covered
    covered/total bins:                                     16         16          -
    missing/total bins:                                      0         16          -
    % Hit:                                              100.00%        100          -
    Coverpoint c1                                       100.00%        100          -    Covered
        covered/total bins:                                  7          7          -
        missing/total bins:                                  0          7          -
        % Hit:                                          100.00%        100          -
        bin write_addr                                    4590          1          -    Covered
        bin write_data                                    1934          1          -    Covered
        bin read_addr                                     1603          1          -    Covered
        bin read_data                                     2860          1          -    Covered
        bin write_addr_data                                129          1          -    Covered
        bin read_addr_data                                  96          1          -    Covered
        bin write_read                                      57          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 rx_valid_0                                    9200          1          -    Covered
        bin rx_valid_1                                    1800          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 tx_valid_0                                    8155          1          -    Covered
        bin tx_valid_1                                    2832          1          -    Covered
    Cross #cross__0#                                    100.00%        100          -    Covered
        covered/total bins:                                  4          4          -
        missing/total bins:                                  0          4          -
        % Hit:                                          100.00%        100          -
```

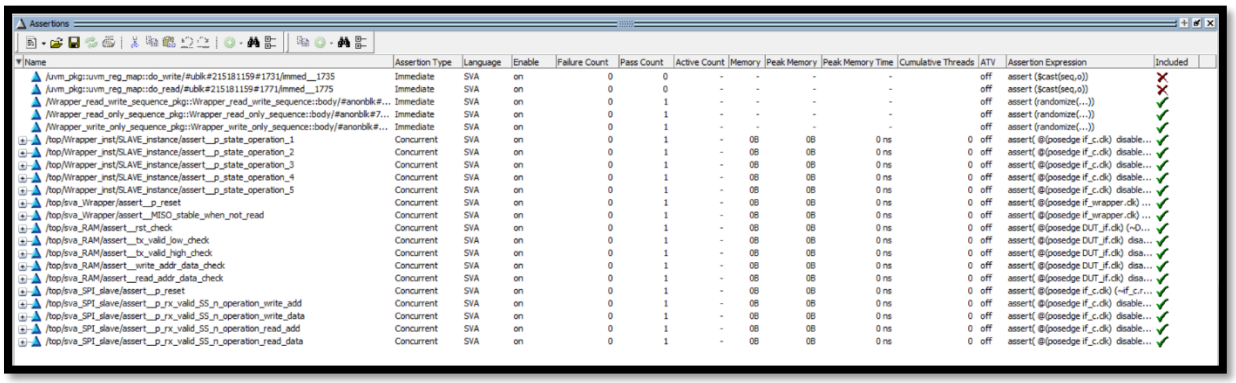
```
        covered/total bins:                                  4          4          -
        missing/total bins:                                  0          4          -
        % Hit:                                          100.00%        100          -
        Auto, Default and User Defined Bins:
            bin b1                                        3960          1          -    Covered
            bin b2                                         264          1          -    Covered
            bin b3                                         214          1          -    Covered
            bin b4                                         692          1          -    Covered
    Cross #cross__1#                                    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 read_data_valid                           2725          1          -    Covered
 TYPE /SPI_slave_coverage_pkg/SPI_slave_coverage/cg
                                                        100.00%        100          -    Covered
    covered/total bins:                                     12         12          -
    missing/total bins:                                      0         12          -
    % Hit:                                              100.00%        100          -
    Coverpoint rx_data_cp                               100.00%        100          -    Covered
        covered/total bins:                                  4          4          -
        missing/total bins:                                  0          4          -
        % 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 MOSI_add_cp                              100.00%        100          -    Covered
        covered/total bins:                                  4          4          -
        missing/total bins:                                  0          4          -
        % Hit:                                          100.00%        100          -
    Cross SS_n_MOSI_cc                                  100.00%        100          -    Covered
        covered/total bins:                                  2          2          -
        missing/total bins:                                  0          2          -
        % Hit:                                          100.00%        100          -
```

```
Covergroup instance \/SPI_slave_coverage_pkg::SPI_slave_coverage::cg
                                                  100.00%        100          -    Covered
    covered/total bins:                                12         12          -
    missing/total bins:                                 0         12          -
    % Hit:                                         100.00%        100          -
    Coverpoint rx_data_cp                          100.00%        100          -    Covered
        covered/total bins:                             4          4          -
        missing/total bins:                             0          4          -
        % Hit:                                     100.00%        100          -
        bin auto[0]                                  5581          1          -    Covered
        bin auto[1]                                  1517          1          -    Covered
        bin auto[2]                                  1475          1          -    Covered
        bin auto[3]                                  2427          1          -    Covered
    Coverpoint SS_n_cp                             100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                     100.00%        100          -
        bin transition13                              550          1          -    Covered
        bin transition23                              110          1          -    Covered
    Coverpoint MOSI_add_cp                         100.00%        100          -    Covered
        covered/total bins:                             4          4          -
        missing/total bins:                             0          4          -
        % Hit:                                     100.00%        100          -
        bin write_address                            4345          1          -    Covered
        bin write_data                               1188          1          -    Covered
        bin read_address                              951          1          -    Covered
        bin read_data                                 815          1          -    Covered
    Cross SS_n_MOSI_cc                             100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                     100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <transition23,write_address>          110          1          -    Covered
            bin <transition13,write_address>          270          1          -    Covered
        Illegal and Ignore Bins:
            ignore_bin illegal9                         0                     -    ZERO
            ignore_bin legal8                           0                     -    ZERO
            ignore_bin illegal5                         0                     -    ZERO
            ignore_bin illegal7                         0                     -    ZERO
            ignore_bin illegal4                         0                     -    ZERO
            ignore_bin illegal2                         0                     -    ZERO

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 2
```

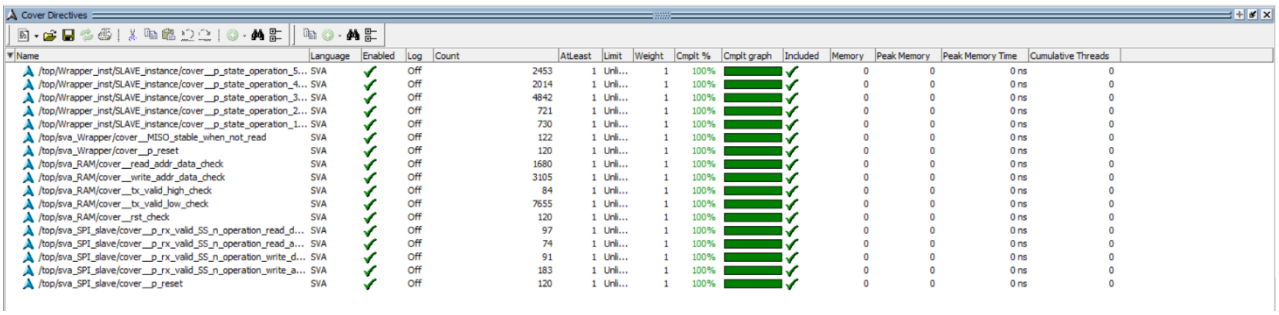
Assertion Summary

Feature	Assertion
Whenever reset is asserted, the outputs (MISO, rx_valid, and rx_data) are all inactive.	@(posedge if_wrapper.clk) (~if_wrapper.rst_n |=> (if_wrapper.MISO == 'b0))
The MISO remains with a stable value as long as it is not a read data operation	@(posedge if_wrapper.clk) disable iff (!if_wrapper.rst_n) // disable when reset active or slave not selected (if_wrapper.SS_n ## 1 !if_wrapper.SS_n ## 1 (if_wrapper.MOSI ## 1 if_wrapper.MOSI ## 1 if_wrapper.MOSI)) |=> $stable(if_wrapper.MISO) throughout (!if_wrapper.SS_n);

Assertion Coverage



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#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/Wrapper_read_write_sequence_pkg::Wrapper_read_write_sequence::body/#anonblk#...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_read_only_sequence_pkg::Wrapper_read_only_sequence::body/#anonblk#7...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_write_only_sequence_pkg::Wrapper_write_only_sequence::body/#anonblk#...	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_1	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_2	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_3	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_4	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_5	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/sva_Wrapper/assert__p_reset	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_wrapper.clk) ...	✓
/top/sva_Wrapper/assert__MISO_stable_when_not_read	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_wrapper.clk) ...	✓
/top/sva_RAM/assert__rst_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge DUT_if.clk) (~0...	✓
/top/sva_RAM/assert__tx_valid_low_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge DUT_if.clk) disa...	✓
/top/sva_RAM/assert__tx_valid_high_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge DUT_if.clk) disa...	✓
/top/sva_RAM/assert__write_addr_data_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge DUT_if.clk) disa...	✓
/top/sva_RAM/assert__read_addr_data_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge DUT_if.clk) disa...	✓
/top/sva_SPI_slave/assert__p_reset	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) (~if_c.r...	✓
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_write_add	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_write_data	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_read_add	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_read_data	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge if_c.clk) disable...	✓

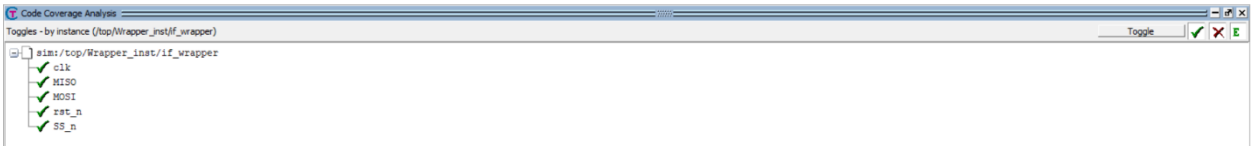


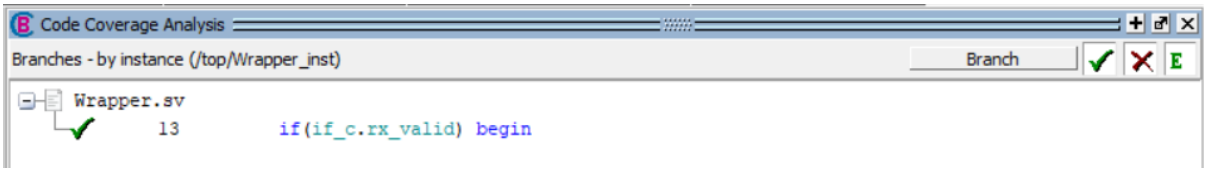
Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_5...	SVA	✓	Off	2453	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_4...	SVA	✓	Off	2014	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_3...	SVA	✓	Off	4842	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_2...	SVA	✓	Off	721	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_1...	SVA	✓	Off	730	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_Wrapper/cover__MISO_stable_when_not_read	SVA	✓	Off	122	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_Wrapper/cover__p_reset	SVA	✓	Off	120	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_RAM/cover__read_addr_data_check	SVA	✓	Off	1680	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_RAM/cover__write_addr_data_check	SVA	✓	Off	3105	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_RAM/cover__tx_valid_high_check	SVA	✓	Off	84	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_RAM/cover__tx_valid_low_check	SVA	✓	Off	7655	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_RAM/cover__rst_check	SVA	✓	Off	120	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_read_d...	SVA	✓	Off	97	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_read_a...	SVA	✓	Off	74	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_write_d...	SVA	✓	Off	91	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_write_a...	SVA	✓	Off	183	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/sva_SPI_slave/cover__p_reset	SVA	✓	Off	120	1	Unli...	1	100%		✓	0	0	0 ns	0

```
ASSERTION RESULTS:
---------------------------------------------------------------------
Name                File(Line)                  Failure      Pass
                                                Count        Count
---------------------------------------------------------------------
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_5
                    SPI_slave.sv(172)           0            1
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_4
                    SPI_slave.sv(165)           0            1
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_3
                    SPI_slave.sv(158)           0            1
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_2
                    SPI_slave.sv(150)           0            1
/top/Wrapper_inst/SLAVE_instance/assert__p_state_operation_1
                    SPI_slave.sv(141)           0            1
/top/sva_Wrapper/assert__MISO_stable_when_not_read
                    Wrapper_sva.sv(16)          0            1
/top/sva_Wrapper/assert__p_reset
                    Wrapper_sva.sv(7)           0            1
/top/sva_RAM/assert__read_addr_data_check
                    RAM_sva.sv(35)              0            1
/top/sva_RAM/assert__write_addr_data_check
                    RAM_sva.sv(28)              0            1
/top/sva_RAM/assert__tx_valid_high_check
                    RAM_sva.sv(21)              0            1
/top/sva_RAM/assert__tx_valid_low_check
                    RAM_sva.sv(14)              0            1
/top/sva_RAM/assert__rst_check
                    RAM_sva.sv(7)               0            1
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_read_data
                    SPI_slave_sva.sv(42)        0            1
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_read_add
                    SPI_slave_sva.sv(35)        0            1
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_write_data
                    SPI_slave_sva.sv(28)        0            1
/top/sva_SPI_slave/assert__p_rx_valid_SS_n_operation_write_add
                    SPI_slave_sva.sv(21)        0            1
/top/sva_SPI_slave/assert__p_reset
                    SPI_slave_sva.sv(14)        0            1
/Wrapper_read_write_sequence_pkg/Wrapper_read_write_sequence/body/#anonblk#235939735#14#4#/#ublk#235939735#14/immed__16
                    D:/Summer2025/Digital_Verifcation/Saeed Nabawy_Project2/Wrapper/Wrapper_read_write_sequence.sv(16)
                                                0            1
/Wrapper_read_only_sequence_pkg/Wrapper_read_only_sequence/body/#anonblk#757463#15#4#/#ublk#757463#15/immed__17
                    D:/Summer2025/Digital_Verifcation/Saeed Nabawy_Project2/Wrapper/Wrapper_read_only_sequence.sv(17)
                                                0            1
/Wrapper_write_only_sequence_pkg/Wrapper_write_only_sequence/body/#anonblk#63355255#14#4#/#ublk#63355255#14/immed__16
                    D:/Summer2025/Digital_Verifcation/Saeed Nabawy_Project2/Wrapper/Wrapper_write_only_sequence.sv(16)
                                                0            1
```

```
DIRECTIVE COVERAGE:
---------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
---------------------------------------------------------------------------------------------
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_5
                                        SPI_slave Verilog  SVA  SPI_slave.sv(173)
                                                                          2261 Covered
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_4
                                        SPI_slave Verilog  SVA  SPI_slave.sv(166)
                                                                          1692 Covered
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_3
                                        SPI_slave Verilog  SVA  SPI_slave.sv(159)
                                                                          5415 Covered
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_2
                                        SPI_slave Verilog  SVA  SPI_slave.sv(151)
                                                                           725 Covered
/top/Wrapper_inst/SLAVE_instance/cover__p_state_operation_1
                                        SPI_slave Verilog  SVA  SPI_slave.sv(142)
                                                                           728 Covered
/top/sva_Wrapper/cover__MISO_stable_when_not_read
                                        Wrapper_sva Verilog  SVA  Wrapper_sva.sv(17)
                                                                           108 Covered
/top/sva_Wrapper/cover__p_reset         Wrapper_sva Verilog  SVA  Wrapper_sva.sv(8)
                                                                            90 Covered
/top/sva_RAM/cover__read_addr_data_check RAM_sva Verilog  SVA  RAM_sva.sv(36)  1421 Covered
/top/sva_RAM/cover__write_addr_data_check
                                        RAM_sva Verilog  SVA  RAM_sva.sv(29)  3691 Covered
/top/sva_RAM/cover__tx_valid_high_check  RAM_sva Verilog  SVA  RAM_sva.sv(22)    74 Covered
/top/sva_RAM/cover__tx_valid_low_check   RAM_sva Verilog  SVA  RAM_sva.sv(15)  8006 Covered
/top/sva_RAM/cover__rst_check           RAM_sva Verilog  SVA  RAM_sva.sv(8)     90 Covered
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_read_data
                                        SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(43)
                                                                            94 Covered
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_read_add
                                        SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(36)
                                                                            81 Covered
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_write_data
                                        SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(29)
                                                                           113 Covered
/top/sva_SPI_slave/cover__p_rx_valid_SS_n_operation_write_add
                                        SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(22)
                                                                           236 Covered
/top/sva_SPI_slave/cover__p_reset       SPI_slave_sva Verilog  SVA  SPI_slave_sva.sv(15)
                                                                            90 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 17
```

Code Coverage





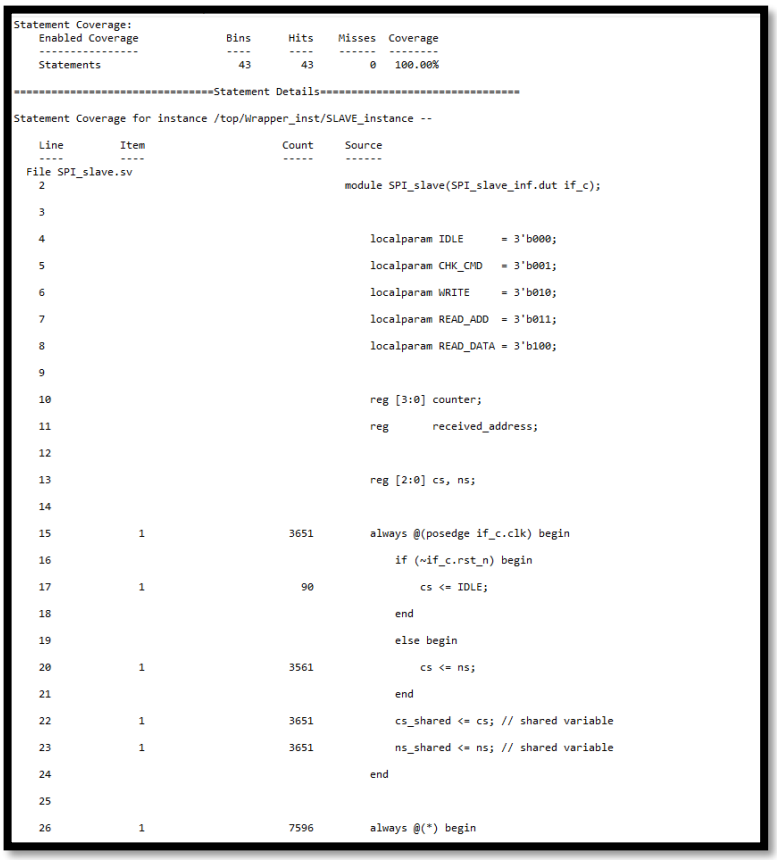

```
================================================================================
=== Instance: /top/if_c
=== Design Unit: work.SPI_slave_inf
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         50        50         0   100.00%

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

Toggle Coverage for instance /top/if_c --

                                          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)
```

```
================================================================================
=== Instance: /top/if_ram
=== Design Unit: work.RAM_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         44        44         0   100.00%

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

Toggle Coverage for instance /top/if_ram --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                           clk           1           1      100.00
                                      din[9-0]           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     =         22
Toggled Node Count   =         22
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (44 of 44 bins)
```

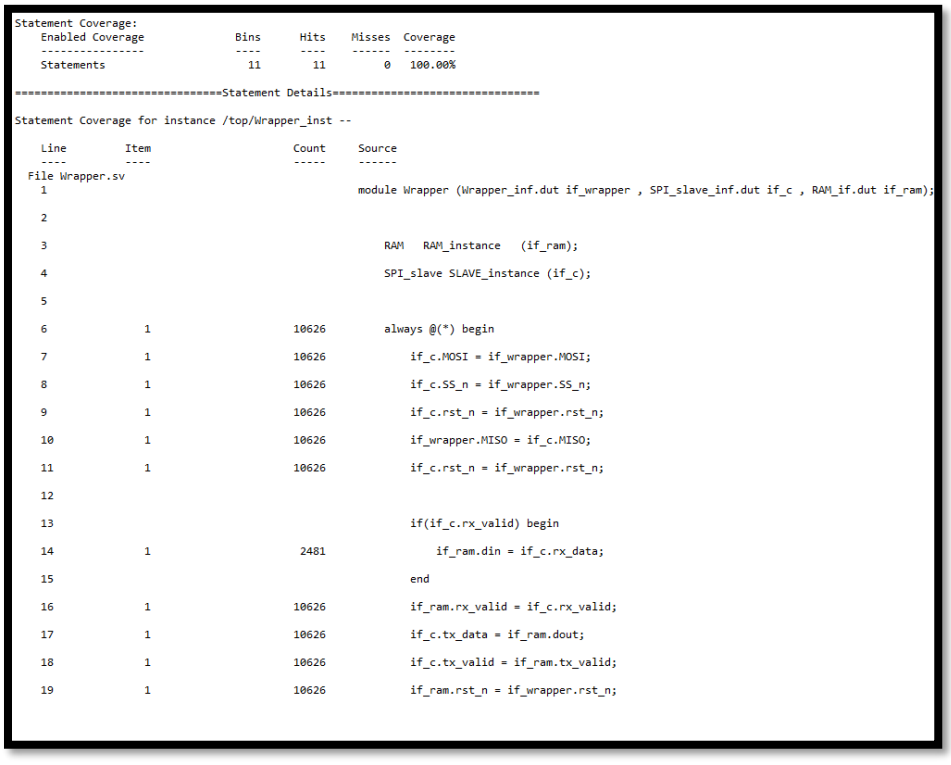
```
================================================================================
=== Instance: /top/if_wrapper
=== Design Unit: work.Wrapper_inf
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         10        10         0   100.00%

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

Toggle Coverage for instance /top/if_wrapper --

                                          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

Total Node Count     =          5
Toggled Node Count   =          5
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (10 of 10 bins)
```

```
================================================================================
=== Instance: /top/Wrapper_inst/RAM_instance
=== Design Unit: work.RAM
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         8         0    100%

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

Branch Coverage for instance /top/Wrapper_inst/RAM_instance

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File RAM.sv
------------------------------------IF Branch------------------------------------
    7                                     11000     Count coming in to IF
    7              1                        90         if (~DUT_if.rst_n) begin

    13             1                     10910         else  begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    14                                    10910     Count coming in to IF
    14             1                       1785             if (DUT_if.rx_valid) begin

                                           9125     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------CASE Branch----------------------------------
    15                                     1785     Count coming in to CASE
    16             1                        623                     2'b00 : Wr_Addr <= DUT_if.din[7:0];

    17             1                        261                     2'b01 : MEM[Wr_Addr] <= DUT_if.din[7:0];

    18             1                        214                     2'b10 : Rd_Addr <= DUT_if.din[7:0];

    19             1                        687                     2'b11 : DUT_if.dout <= MEM[Rd_Addr];

Branch totals: 4 hits of 4 branches = 100.00%
```

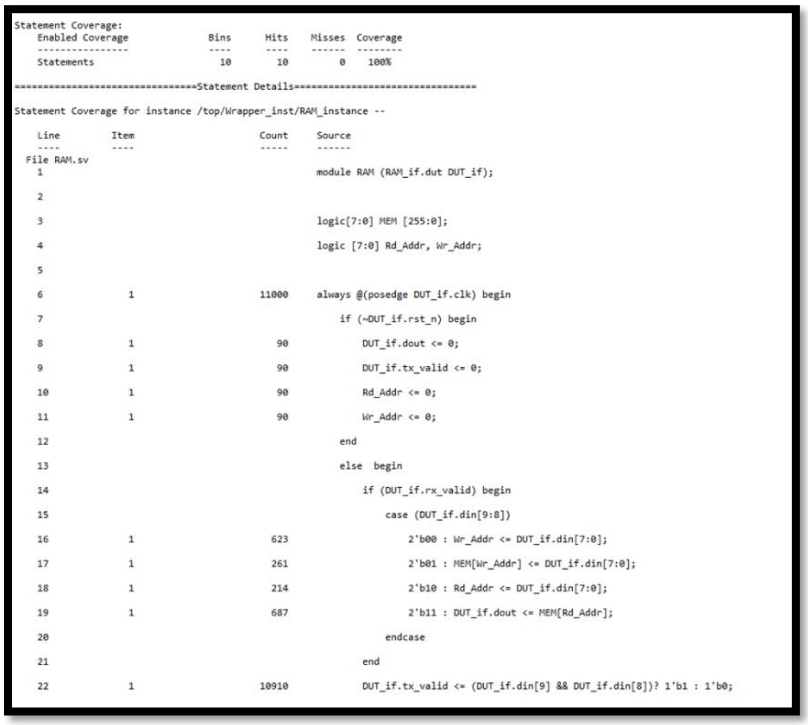
```
FSM Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    FSM States                       5         5         0   100.00%
    FSM Transitions                  8         8         0   100.00%

================================FSM Details================================

FSM Coverage for instance /top/Wrapper_inst/SLAVE_instance --

FSM_ID: cs
    Current State Object : cs
    ---------------------
    State Value MapInfo :
    ---------------------
Line          State Name               Value
----          ----------               -----
  28               IDLE                   0
  34             CHK_CMD                  1
  60           READ_DATA                  4
  54            READ_ADD                  3
  48               WRITE                  2
    Covered States :
    ----------------
                   State          Hit_count
                   -----          ---------
                    IDLE                743
                  CHK_CMD               737
                READ_DATA               354
                 READ_ADD               398
                   WRITE               1419
    Covered Transitions :
    ---------------------
Line           Trans_ID          Hit_count         Transition
----           --------          ---------         ----------
  32                  0                737         IDLE -> CHK_CMD
  44                  1                119         CHK_CMD -> READ_DATA
  42                  2                133         CHK_CMD -> READ_ADD
  39                  3                476         CHK_CMD -> WRITE
  36                  4                  9         CHK_CMD -> IDLE
  62                  5                119         READ_DATA -> IDLE
  56                  6                132         READ_ADD -> IDLE
  50                  7                476         WRITE -> IDLE


    Summary                       Bins      Hits    Misses  Coverage
    -------                       ----      ----    ------  --------
        FSM States                   5         5         0   100.00%
        FSM Transitions              8         8         0   100.00%
```



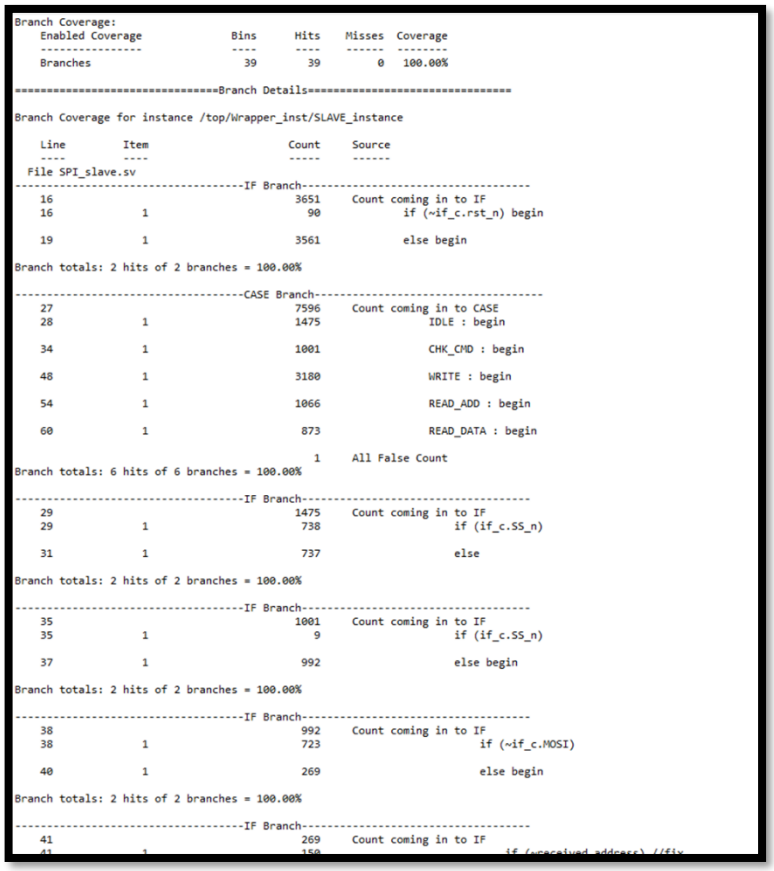
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      43        43         0   100.00%

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

Statement Coverage for instance /top/Wrapper_inst/SLAVE_instance --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File SPI_slave.sv
    2                                               module SPI_slave(SPI_slave_inf.dut if_c);

    3

    4                                                   localparam IDLE      = 3'b000;

    5                                                   localparam CHK_CMD   = 3'b001;

    6                                                   localparam WRITE     = 3'b010;

    7                                                   localparam READ_ADD  = 3'b011;

    8                                                   localparam READ_DATA = 3'b100;

    9

    10                                                  reg [3:0] counter;

    11                                                  reg       received_address;

    12

    13                                                  reg [2:0] cs, ns;

    14

    15              1                        3651       always @(posedge if_c.clk) begin

    16                                                      if (~if_c.rst_n) begin

    17              1                          90               cs <= IDLE;

    18                                                      end

    19                                                      else begin

    20              1                        3561               cs <= ns;

    21                                                      end

    22              1                        3651           cs_shared <= cs; // shared variable

    23              1                        3651           ns_shared <= ns; // shared variable

    24                                                  end

    25

    26              1                        7596       always @(*) begin
```



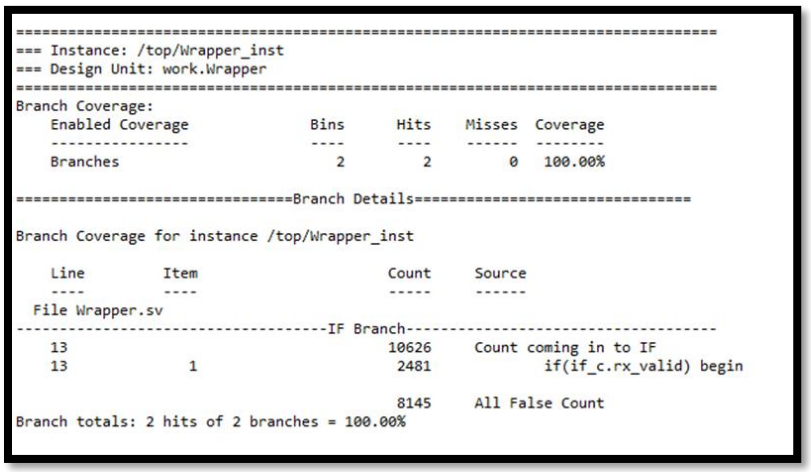
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      11        11         0   100.00%

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

Statement Coverage for instance /top/Wrapper_inst --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File Wrapper.sv
    1                                               module Wrapper (Wrapper_inf.dut if_wrapper , SPI_slave_inf.dut if_c , RAM_if.dut if_ram);

    2

    3                                                   RAM   RAM_instance   (if_ram);

    4                                                   SPI_slave SLAVE_instance (if_c);

    5

    6               1                       10626       always @(*) begin

    7               1                       10626           if_c.MOSI = if_wrapper.MOSI;

    8               1                       10626           if_c.SS_n = if_wrapper.SS_n;

    9               1                       10626           if_c.rst_n = if_wrapper.rst_n;

    10              1                       10626           if_wrapper.MISO = if_c.MISO;

    11              1                       10626           if_c.rst_n = if_wrapper.rst_n;

    12

    13                                                      if(if_c.rx_valid) begin

    14              1                        2481               if_ram.din = if_c.rx_data;

    15                                                      end

    16              1                       10626           if_ram.rx_valid = if_c.rx_valid;

    17              1                       10626           if_c.tx_data = if_ram.dout;

    18              1                       10626           if_c.tx_valid = if_ram.tx_valid;

    19              1                       10626           if_ram.rst_n = if_wrapper.rst_n;
```



```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      10        10         0   100%

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

Statement Coverage for instance /top/Wrapper_inst/RAM_instance --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File RAM.sv
    1                                               module RAM (RAM_if.dut DUT_if);

    2

    3                                               logic[7:0] MEM [255:0];

    4                                               logic [7:0] Rd_Addr, Wr_Addr;

    5

    6               1                    11000      always @(posedge DUT_if.clk) begin

    7                                                   if (~DUT_if.rst_n) begin

    8               1                       90              DUT_if.dout <= 0;

    9               1                       90              DUT_if.tx_valid <= 0;

    10              1                       90              Rd_Addr <= 0;

    11              1                       90              Wr_Addr <= 0;

    12                                                  end

    13                                                  else  begin

    14                                                      if (DUT_if.rx_valid) begin

    15                                                          case (DUT_if.din[9:8])

    16              1                      623                      2'b00 : Wr_Addr <= DUT_if.din[7:0];

    17              1                      261                      2'b01 : MEM[Wr_Addr] <= DUT_if.din[7:0];

    18              1                      214                      2'b10 : Rd_Addr <= DUT_if.din[7:0];

    19              1                      687                      2'b11 : DUT_if.dout <= MEM[Rd_Addr];

    20                                                          endcase

    21                                                      end

    22              1                    10910              DUT_if.tx_valid <= (DUT_if.din[9] && DUT_if.din[8])? 1'b1 : 1'b0;
```



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

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

Branch Coverage for instance /top/Wrapper_inst/SLAVE_instance

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File SPI_slave.sv
------------------------------------IF Branch------------------------------------
    16                                      3651     Count coming in to IF
    16              1                         90             if (~if_c.rst_n) begin

    19              1                       3561             else begin

Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------CASE Branch----------------------------------
    27                                      7596     Count coming in to CASE
    28              1                       1475                 IDLE : begin

    34              1                       1001                 CHK_CMD : begin

    48              1                       3180                 WRITE : begin

    54              1                       1066                 READ_ADD : begin

    60              1                        873                 READ_DATA : begin

                                                1     All False Count
Branch totals: 6 hits of 6 branches = 100.00%

------------------------------------IF Branch------------------------------------
    29                                      1475     Count coming in to IF
    29              1                        738                     if (if_c.SS_n)

    31              1                        737                     else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    35                                      1001     Count coming in to IF
    35              1                          9                     if (if_c.SS_n)

    37              1                        992                     else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    38                                       992     Count coming in to IF
    38              1                        723                         if (~if_c.MOSI)

    40              1                        269                         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    41                                       269     Count coming in to IF
    41              1                        150                             if (received_address) //fix
```



```
================================================================================
=== Instance: /top/Wrapper_inst
=== Design Unit: work.Wrapper
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         2         2         0   100.00%

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

Branch Coverage for instance /top/Wrapper_inst

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File Wrapper.sv
-----------------------------------IF Branch-----------------------------------
    13                                     10626     Count coming in to IF
    13              1                       2481             if(if_c.rx_valid) begin

                                            8145     All False Count
Branch totals: 2 hits of 2 branches = 100.00%
```