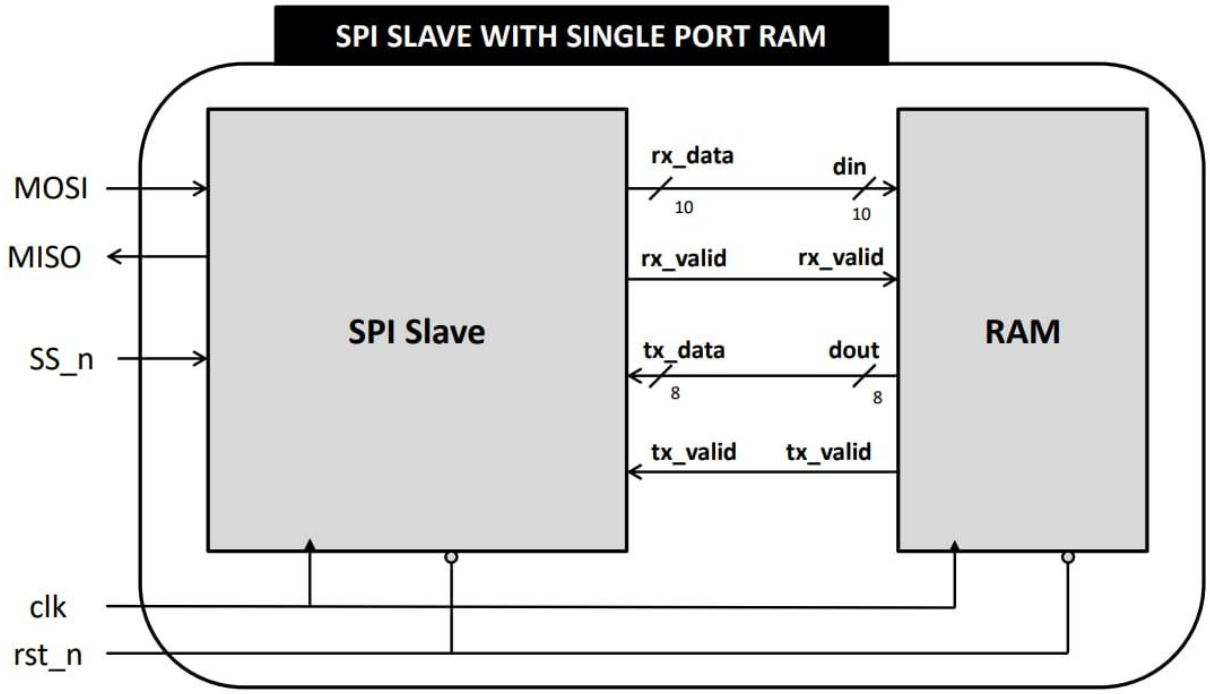
SPI Wrapper Project



By:

Rafat Elsheikh

Mohamed Mostafa

Youssef Tantawy

Team : SQUADRA

Contents

Part1 (SPI Slave) 6

Verification Plan 6

UVM Structure 7

Corrected Design Code 8

Golden Model Code 11

Interface Code 14

Shared Package Code 14

Sequence Item Code 14

Reset Sequence Code 16

Main Sequence Code 16

Sequencer Code 17

Configuration Object Code 17

Driver Code 17

Monitor Code 18

Agent Code 19

Scoreboard Code 20

Coverage Code 21

Environment Code 22

Test Code 23

Top Module Code 24

Source Files List 24

Do File 25

Waveform 25

Coverage Report 26

Bug Report 39

Sequences Explaining 40

Reset Sequence 40

Main Sequence 41

Assertion Table 41

Part 2 (RAM) 43

Design 43
Bug report 43
Verification Plan 44
Assertion Table 44
Do file 44
Source File 45
Interface 45
Golden model 45
Assertions 47
RAM Sequence item 49
Driver 51
Monitor 52
Scoreboard 53
Sequencer 54
Agent 55
Coverage 57
Sequence 59
    Reset Sequence 64
    Write Only Sequence 64
    Read Only Sequence 64
    Write-Read Sequence 65
    Main Sequence 65
Enviroment 65
Test 66
Config object 68
Top 69
UVM report 70
Wave Snippet 70
Covergroup 71
Assertion coverage 71
Code coverage 71

Coverage Report 73
Part 3 (Wrapper) 86
Verification Plan 86
UVM Structure 86
Corrected Design Code 87
Golden Model Code 88
Interface Code 88
Shared Package Code 88
Sequence Item Code 89
Reset Sequence Code 91
Write only Sequence Code 91
Read only Sequence Code 92
Write-Read Sequence Code 92
Configuration object Code 93
Sequencer Code 93
Driver Code 94
Monitor Code 94
Agent Code 95
Scoreboard Code 96
Coverage collector Code 97
Environment Code 98
Test Code 99
Top module Code 101
Source files list 102
DO file 102
Waveform 103
Wrapper_reset_seq 104
Wrapper_write_only_seq 104
Wrapper_read_only_seq 104
Wrapper_read_only_seq 104
Coverage Report 105

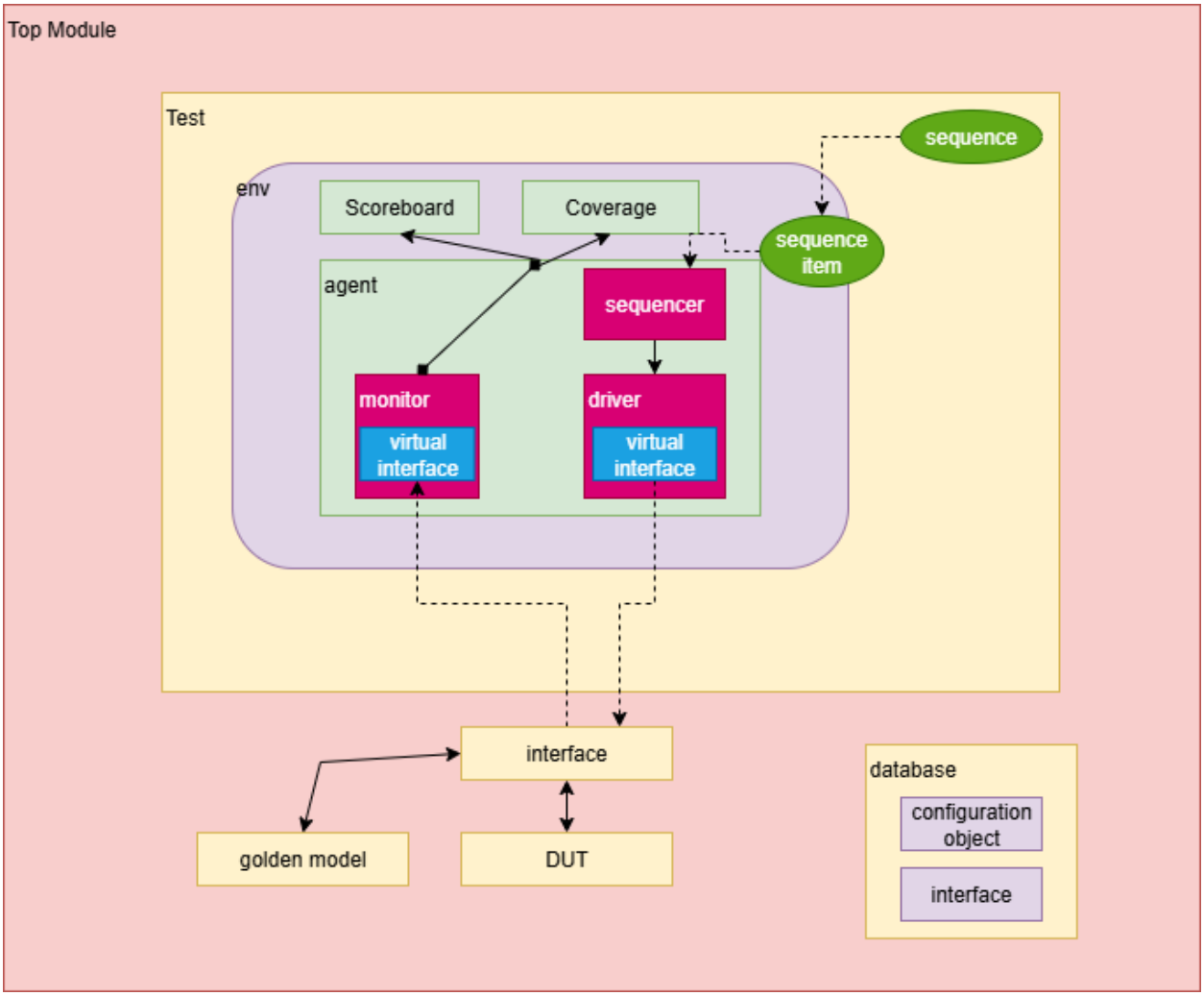
Assertion Table 117
Real time application for SPI Wrapper 117
- Consumer Electronics & IoT 117
- Automotive & Automotive-Grade Applications 118
- Microcontroller Platforms.....118

Part1 (SPI Slave)

Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
SPI_SLAVE_1	When the reset is asserted, the output value should be low	Directed at the start of the simulation then randomized to be inactive 95% of simulation time	-	A checker and a concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_2	When the reset is asserted, the current state should be IDLE	Directed at the start of the simulation then randomized to be inactive 95% of simulation time	-	A concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_3	When the current case is IDLE and SS_n is 0, the cs is CMD_CHK the next clock cycle	Randomization under contraints that SS_n should be high every 13 cycles for all cases except for read data it should be high every 23 cycles	SS_n cover the normal transition and the read data transition sequences	A concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_4	When the current case is any thing but CMD_CHK and SS_n is 1, the cs is IDLEthe next clock cycle	Randomization under contraints that SS_n should be high every 13 cycles for all cases except for read data it should be high every 23 cycles	SS_n cover the normal transition and the read data transition sequences	A concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_5	Current state in the next clock cycle depends on the current state and the inputs SS_n and MOSI	-	-	A concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_6	When MOSI equals 000 it's write address, 001 it's write data, 010 it's read address, 011 it's read data	Randomization under contraints that MOSI should be valid sequence	cover all valid sequences of MOSI and the cross between them and SS_n	A checker in the testbench to make sure the output is correct
SPI_SLAVE_7	When it's a valid MISO, after 10 cycles from the first MISO three bits, the rx_valid should be high	Randomization under contraints that MOSI should be valid sequence	cover all valid sequences of MOSI and the cross between them and SS_n	A checker and a concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_8	In the case of read data, tx_valid should be high	Randomization under constraints that in the case of read data, tx_valid should be high	-	A checker and a concurrent assertion in the testbench to make sure the output is correct
SPI_SLAVE_9	After receiving all the 10 bits from the MOSI, it should be ouput through rx_data	-	cover all values and transition in the 2 most bits in the rx_data	A checker and a concurrent assertion in the testbench to make sure the output is correct

UVM Structure



The interface has all the signals in the DUT and the golden model, the top module puts the interface in the database, the test takes the interface from the database and puts it in the configuration object and then puts the configuration object in the database again, after that the test runs the sequences using the sequencer.

The environment connects the scoreboard and coverage classes to the agent, and the driver with the sequencer and the monitor to the agent port and connect the interfaces of the driver and the monitor with the one in the configuration object.

The monitor assigns all the variables in the sequence item to the values in the interface and connect the sequence item to the port in the agent so the coverage and scoreboard classes be able to use it.

The coverage class contains all the coverage points and cross coverages needed and the scoreboard compares the output of the DUT and the output of the golden model using the variables in the sequence item.

Corrected Design Code

```
module SLAVE (MOSI, MISO, SS_n, clk, rst_n, rx_data, rx_valid, tx_data, tx_valid);
    localparam IDLE      = 3'b000;
    localparam WRITE     = 3'b001;
    localparam CHK_CMD   = 3'b010;
    localparam READ_ADD  = 3'b011;
    localparam READ_DATA = 3'b100;

    input            MOSI, clk, rst_n, SS_n, tx_valid;
    input      [7:0] tx_data;
    output reg [9:0] rx_data;
    output reg       rx_valid, MISO;

    reg [3:0] counter;
    reg       received_address;

    reg [2:0] cs, ns;

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

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

```
                    ns = IDLE;
                else
                    ns = READ_ADD;
            end
            READ_DATA : begin
                if (SS_n)
                    ns = IDLE;
                else
                    ns = READ_DATA;
            end
        endcase
    end

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

```
                            counter <= 8;
                        end
                    end
                end
            endcase
        end
    end

    `ifdef SIM
        property rst_MISO;
            @(posedge clk) rst_n == 0 |=> MISO == 0;
        endproperty

        property rst_rx_valid;
            @(posedge clk) rst_n == 0 |=> rx_valid == 0;
        endproperty

        property rst_rx_data;
            @(posedge clk) rst_n == 0 |=> rx_data == 0;
        endproperty

        property rx_valid_write_address;
            @(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 0) ##1 (MOSI == 0)
##1 (MOSI == 0) |-> ##10 rx_valid;
        endproperty

        property rx_valid_write_data;
            @(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 0) ##1 (MOSI == 0)
##1 (MOSI == 1) |-> ##10 rx_valid;
        endproperty

        property rx_valid_read_address;
            @(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 1) ##1 (MOSI == 1)
##1 (MOSI == 0) |-> ##10 rx_valid;
        endproperty

        property rx_valid_read_data;
            @(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 1) ##1 (MOSI == 1)
##1 (MOSI == 1) |-> ##10 rx_valid;
        endproperty

        property IDLE_to_CHK_CMD;
            @(posedge clk) disable iff (!rst_n) (cs == IDLE && SS_n == 0) |=> cs == CHK_CMD;
        endproperty

        property CHK_CMD_to_WRITE;
            @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && SS_n == 0 && MOSI == 0) |=>
cs == WRITE;
        endproperty

        property CHK_CMD_to_READ_ADD;
            @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && SS_n == 0 && MOSI == 1 &&
received_address == 0) |=> cs == READ_ADD;
        endproperty

        property CHK_CMD_to_READ_DATA;
            @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && SS_n == 0 && MOSI == 1 &&
received_address == 1) |=> cs == READ_DATA;
        endproperty

        property WRITE_to_IDLE;
            @(posedge clk) disable iff (!rst_n) (cs == WRITE && SS_n == 1) |=> cs == IDLE;
        endproperty

        property READ_ADD_to_IDLE;
```

```
180.             @(posedge clk) disable iff (!rst_n) (cs == READ_ADD && SS_n == 1) |=> cs == IDLE;
181.         endproperty
182.
183.         property READ_DATA_to_IDLE;
184.             @(posedge clk) disable iff (!rst_n) (cs == READ_DATA && SS_n == 1) |=> cs == IDLE;
185.         endproperty
186.
187.         a_rst_MISO: assert property (rst_MISO);
188.         c_rst_MISO: cover property (rst_MISO);
189.
190.         a_rst_rx_valid: assert property (rst_rx_valid);
191.         c_rst_rx_valid: cover property (rst_rx_valid);
192.
193.         a_rst_rx_data: assert property (rst_rx_data);
194.         c_rst_rx_data: cover property (rst_rx_data);
195.
196.         a_rx_valid_write_address: assert property (rx_valid_write_address);
197.         c_rx_valid_write_address: cover property (rx_valid_write_address);
198.
199.         a_rx_valid_write_data: assert property (rx_valid_write_data);
200.         c_rx_valid_write_data: cover property (rx_valid_write_data);
201.
202.         a_rx_valid_read_address: assert property (rx_valid_read_address);
203.         c_rx_valid_read_address: cover property (rx_valid_read_address);
204.
205.         a_rx_valid_read_data: assert property (rx_valid_read_data);
206.         c_rx_valid_read_data: cover property (rx_valid_read_data);
207.
208.         a_IDLE_to_CHK_CMD: assert property (IDLE_to_CHK_CMD);
209.         c_IDLE_to_CHK_CMD: cover property (IDLE_to_CHK_CMD);
210.
211.         a_CHK_CMD_to_WRITE: assert property (CHK_CMD_to_WRITE);
212.         c_CHK_CMD_to_WRITE: cover property (CHK_CMD_to_WRITE);
213.
214.         a_CHK_CMD_to_READ_ADD: assert property (CHK_CMD_to_READ_ADD);
215.         c_CHK_CMD_to_READ_ADD: cover property (CHK_CMD_to_READ_ADD);
216.
217.         a_CHK_CMD_to_READ_DATA: assert property (CHK_CMD_to_READ_DATA);
218.         c_CHK_CMD_to_READ_DATA: cover property (CHK_CMD_to_READ_DATA);
219.
220.         a_WRITE_to_IDLE: assert property (WRITE_to_IDLE);
221.         c_WRITE_to_IDLE: cover property (WRITE_to_IDLE);
222.
223.         a_READ_ADD_to_IDLE: assert property (READ_ADD_to_IDLE);
224.         c_READ_ADD_to_IDLE: cover property (READ_ADD_to_IDLE);
225.
226.         a_READ_DATA_to_IDLE: assert property (READ_DATA_to_IDLE);
227.         c_READ_DATA_to_IDLE: cover property (READ_DATA_to_IDLE);
228.     `endif
229. endmodule
230.
```

Golden Model Code

```
1. module SPI_slave_golden (clk, rst_n, SS_n, MOSI, tx_valid, tx_data, rx_valid, rx_data, MISO);
2.     parameter IDLE = 3'b000;
3.     parameter CHK_CMD = 3'b001;
4.     parameter WRITE = 3'b010;
5.     parameter READ_ADD = 3'b011;
6.     parameter READ_DATA = 3'b100;
7.
8.     input clk, rst_n, SS_n, MOSI, tx_valid;
```

```
    input [7:0] tx_data;

    output reg rx_valid, MISO;
    output reg [9:0] rx_data;

    (* fsm_encoding = "gray" *)
    reg [2:0] cs, ns;

    reg is_read_data;

    reg [3:0] count;

    // State Memory
    always @(posedge clk) begin
        if (~rst_n) begin
            cs <= IDLE;
        end else begin
            cs <= ns;
        end
    end

    // Next State Logic
    always @(*) begin
        case (cs)
            IDLE: begin
                if (~SS_n) begin
                    ns = CHK_CMD;
                end else begin
                    ns = IDLE;
                end
            end

            CHK_CMD: begin
                if (~SS_n) begin
                    if (~MOSI) begin
                        ns = WRITE;
                    end else if (~is_read_data) begin
                        ns = READ_ADD;
                    end else begin
                        ns = READ_DATA;
                    end
                end else begin
                    ns = IDLE;
                end
            end

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

            READ_ADD: begin
                if (~SS_n) begin
                    ns = READ_ADD;
                end else begin
                    ns = IDLE;
                end
            end

            READ_DATA: begin
                if (~SS_n) begin
                    ns = READ_DATA;
```

```
                end else begin
                    ns = IDLE;
                end
            end

            default: begin
                ns = IDLE;
            end
        endcase
    end

    // Output Logic
    always @(posedge clk) begin
        if (~rst_n) begin
            is_read_data <= 0;
            count <= 0;
            rx_valid <= 0;
            rx_data <= 0;
            MISO <= 0;
        end else begin
            case (cs)
                IDLE: rx_valid <= 0;
                CHK_CMD: count <= 0;

                WRITE: begin
                    if (count < 10) begin
                        rx_data[9 - count] <= MOSI;
                        count <= count + 1;
                    end else begin
                        rx_valid <= 1;
                    end
                end

                READ_ADD: begin
                    if (count < 10) begin
                        rx_data[9 - count] <= MOSI;
                        count <= count + 1;
                    end else begin
                        rx_valid <= 1;
                        is_read_data <= 1;
                    end
                end

                READ_DATA: begin
                    if (rx_valid && !tx_valid) begin
                        count <= 0;
                    end else if (count < 10 && !tx_valid) begin
                        rx_data[9 - count] <= MOSI;
                        count <= count + 1;
                    end else if (count == 10 && !tx_valid) begin
                        rx_valid <= 1;
                    end else if (tx_valid && count < 8) begin
                        is_read_data <= 0;
                        rx_valid <= 0;
                        MISO <= tx_data[7 - count];
                        count <= count + 1;
                    end
                end
            endcase
        end
    end
endmodule

```

Interface Code

```
interface SPI_slave_if (clk);
    input clk;
    logic rst_n;
    logic SS_n;
    logic MOSI;
    logic MISO;
    logic rx_valid;
    logic tx_valid;
    logic [9:0] rx_data;
    logic [7:0] tx_data;
    logic MISO_golden;
    logic rx_valid_golden;
    logic [9:0] rx_data_golden;
endinterface

```

Shared Package Code

```
package shared_pkg;
    int period;
    int count;
    bit have_address;
    logic [10:0] curr_op;
endpackage

```

Sequence Item Code

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

    class SPI_slave_seq_item extends uvm_sequence_item;
        `uvm_object_utils(SPI_slave_seq_item)

        rand logic rst_n;
        rand logic SS_n;
        logic MOSI;
        rand logic tx_valid;
        rand logic [7:0] tx_data;
        logic MISO;
        logic rx_valid;
        logic [9:0] rx_data;
        logic MISO_golden;
        logic rx_valid_golden;
        logic [9:0] rx_data_golden;
        rand logic [10:0] MOSI_arr;

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

        function string convert2string();
            return $sformatf("%s rst_n: %b, SS_n: %b, MOSI: %b, tx_valid: %b, tx_data: %h,
                              MISO: %b, rx_valid: %b, rx_data: %h,
                              MISO_golden: %b, rx_valid_golden: %b, rx_data_golden: %h",
```

```
30.                                 super.convert2string(), rst_n, SS_n, MOSI, tx_valid, tx_data,
31.                                 MISO, rx_valid, rx_data, MISO_golden, rx_valid_golden,
rx_data_golden);
32.         endfunction
33.
34.         function string convert2string_stimulus();
35.             return $sformatf("rst_n: %b, SS_n: %b, MOSI: %b, tx_valid: %b, tx_data: %h",
36.                                 rst_n, SS_n, MOSI, tx_valid, tx_data);
37.         endfunction
38.
39.         constraint SPI_SLAVE_1 {
40.             rst_n dist {1 :/ 95, 0 :/ 5};
41.         }
42.
43.         constraint SPI_SLAVE_6 {
44.             if (SS_n == 0) {
45.                 MOSI_arr[10:8] inside {3'b000, 3'b001, 3'b110, 3'b111};
46.
47.                 if (!have_address) {
48.                     MOSI_arr[10:8] != 3'b111;
49.                 }
50.             }
51.         }
52.
53.         constraint SPI_SLAVE_8 {
54.             if (count > 14) {
55.                 tx_valid == 1;
56.             } else {
57.                 tx_valid == 0;
58.             }
59.         }
60.
61.         function void post_randomize();
62.             if (count == 0) begin
63.                 curr_op = MOSI_arr;
64.             end
65.
66.             if (curr_op[10:8] == 3'b111) begin
67.                 period = 23;
68.             end else begin
69.                 period = 13;
70.             end
71.
72.             if (count == period) begin
73.                 SS_n = 1;
74.             end else begin
75.                 SS_n = 0;
76.             end
77.
78.             if (curr_op[10:8] == 3'b110) begin
79.                 have_address = 1;
80.             end
81.
82.             if (curr_op[10:8] == 3'b111 || !rst_n) begin
83.                 have_address = 0;
84.             end
85.
86.             if (count > 0 && count < 12) begin
87.                 MOSI = curr_op[11 - count];
88.             end else begin
89.                 MOSI = 0;
90.             end
91.
92.             if (!rst_n) begin
93.                 count = 0;
```

```
            end else begin
                if (count == period) begin
                    count = 0;
                end else begin
                    count++;
                end
            end
        endfunction
    endclass
endpackage

```

Reset Sequence Code

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

    class SPI_slave_rst_seq extends uvm_sequence #(SPI_slave_seq_item);
        `uvm_object_utils(SPI_slave_rst_seq);

        SPI_slave_seq_item seq_item;

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

        task body;
            seq_item = SPI_slave_seq_item::type_id::create("seq_item");

            start_item(seq_item);

            seq_item.rst_n = 0;

            finish_item(seq_item);
        endtask
    endclass
endpackage

```

Main Sequence Code

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

    class SPI_slave_main_seq extends uvm_sequence #(SPI_slave_seq_item);
        `uvm_object_utils(SPI_slave_main_seq);

        SPI_slave_seq_item seq_item;

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

        task body;
            repeat (10000) begin
                seq_item = SPI_slave_seq_item::type_id::create("seq_item");
```

```
18.
19.                 start_item(seq_item);
20.
21.                 assert(seq_item.randomize());
22.
23.                 finish_item(seq_item);
24.             end
25.         endtask
26.     endclass
27. endpackage
28.
```

Sequencer Code

```
 1. package SPI_slave_sequencer_pkg;
 2.     import SPI_slave_seq_item_pkg::*;
 3.     import uvm_pkg::*;
 4.     `include "uvm_macros.svh"
 5.
 6.     class SPI_slave_sequencer extends uvm_sequencer #(SPI_slave_seq_item);
 7.         `uvm_component_utils(SPI_slave_sequencer);
 8.
 9.         function new(string name = "SPI_slave_sequencer", uvm_component parent = null);
10.             super.new(name, parent);
11.         endfunction
12.     endclass
13. endpackage
14.
```

Configuration Object Code

```
 1. package SPI_slave_config_pkg;
 2.     import uvm_pkg::*;
 3.     `include "uvm_macros.svh"
 4.
 5.     class SPI_slave_config extends uvm_object;
 6.         `uvm_object_utils(SPI_slave_config);
 7.
 8.         virtual SPI_slave_if SPI_slave_vif;
 9.         uvm_active_passive_enum is_active;
10.
11.         function new(string name = "SPI_slave_config");
12.             super.new(name);
13.         endfunction
14.     endclass
15. endpackage
16.
```

Driver Code

```
 1. package SPI_slave_driver_pkg;
 2.     import SPI_slave_seq_item_pkg::*;
 3.     import uvm_pkg::*;
 4.     `include "uvm_macros.svh"
 5.
 6.     class SPI_slave_driver extends uvm_driver #(SPI_slave_seq_item);
 7.         `uvm_component_utils(SPI_slave_driver);
```

```

        virtual SPI_slave_if SPI_slave_vif;
        SPI_slave_seq_item seq_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);

            forever begin
                seq_item = SPI_slave_seq_item::type_id::create("seq_item");

                seq_item_port.get_next_item(seq_item);

                SPI_slave_vif.rst_n = seq_item.rst_n;
                SPI_slave_vif.SS_n = seq_item.SS_n;
                SPI_slave_vif.MOSI = seq_item.MOSI;
                SPI_slave_vif.tx_valid = seq_item.tx_valid;
                SPI_slave_vif.tx_data = seq_item.tx_data;

                @(negedge SPI_slave_vif.clk);

                seq_item_port.item_done();

                `uvm_info("run_phase", seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask
    endclass
endpackage

```

Monitor Code

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

    class SPI_slave_monitor extends uvm_monitor;
        `uvm_component_utils(SPI_slave_monitor);

        virtual SPI_slave_if SPI_slave_vif;
        SPI_slave_seq_item seq_item;
        uvm_analysis_port #(SPI_slave_seq_item) monitor_ap;

        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);

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

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

            forever begin
                seq_item = SPI_slave_seq_item::type_id::create("seq_item");
```

```
28.
29.                 @(negedge SPI_slave_vif.clk);
30.
31.                 seq_item.rst_n = SPI_slave_vif.rst_n;
32.                 seq_item.SS_n = SPI_slave_vif.SS_n;
33.                 seq_item.MOSI = SPI_slave_vif.MOSI;
34.                 seq_item.tx_valid = SPI_slave_vif.tx_valid;
35.                 seq_item.tx_data = SPI_slave_vif.tx_data;
36.                 seq_item.MISO = SPI_slave_vif.MISO;
37.                 seq_item.rx_valid = SPI_slave_vif.rx_valid;
38.                 seq_item.rx_data = SPI_slave_vif.rx_data;
39.                 seq_item.MISO_golden = SPI_slave_vif.MISO_golden;
40.                 seq_item.rx_valid_golden = SPI_slave_vif.rx_valid_golden;
41.                 seq_item.rx_data_golden = SPI_slave_vif.rx_data_golden;
42.
43.                 monitor_ap.write(seq_item);
44.
45.                 `uvm_info("run_phase", seq_item.convert2string(), UVM_HIGH)
46.             end
47.         endtask
48.     endclass
49. endpackage
50.
```

Agent Code

```
 1. package SPI_slave_agent_pkg;
 2.     import SPI_slave_sequencer_pkg::*;
 3.     import SPI_slave_seq_item_pkg::*;
 4.     import SPI_slave_driver_pkg::*;
 5.     import SPI_slave_monitor_pkg::*;
 6.     import SPI_slave_config_pkg::*;
 7.     import uvm_pkg::*;
 8.     `include "uvm_macros.svh"
 9.
10.     class SPI_slave_agent extends uvm_agent;
11.         `uvm_component_utils(SPI_slave_agent);
12.
13.         SPI_slave_sequencer sequencer;
14.         SPI_slave_driver driver;
15.         SPI_slave_monitor monitor;
16.         SPI_slave_config cfg;
17.         uvm_analysis_port #(SPI_slave_seq_item) agent_ap;
18.
19.         function new(string name = "SPI_slave_agent", uvm_component parent = null);
20.             super.new(name, parent);
21.         endfunction
22.
23.         function void build_phase(uvm_phase phase);
24.             super.build_phase(phase);
25.
26.             if (!uvm_config_db #(SPI_slave_config)::get(this, "", "SPI_SLAVE_CFG", cfg))
27.                 `uvm_fatal("build_phase", "unable to get config object");
28.
29.             if (cfg.is_active == UVM_ACTIVE) begin
30.                 sequencer = SPI_slave_sequencer::type_id::create("sequencer", this);
31.                 driver = SPI_slave_driver::type_id::create("driver", this);
32.             end
33.
34.             monitor = SPI_slave_monitor::type_id::create("monitor", this);
35.             agent_ap = new("agent_ap", this);
36.         endfunction
```

```
37.
38.         function void connect_phase(uvm_phase phase);
39.             super.connect_phase(phase);
40.
41.             if (cfg.is_active == UVM_ACTIVE) begin
42.                 driver.SPI_slave_vif = cfg.SPI_slave_vif;
43.                 driver.seq_item_port.connect(sequencer.seq_item_export);
44.             end
45.
46.             monitor.SPI_slave_vif = cfg.SPI_slave_vif;
47.             monitor.monitor_ap.connect(agent_ap);
48.         endfunction
49.     endclass
50. endpackage
51.
```

Scoreboard Code

```
 1. package SPI_slave_scoreboard_pkg;
 2.     import SPI_slave_seq_item_pkg::*;
 3.     import uvm_pkg::*;
 4.     `include "uvm_macros.svh"
 5.
 6.     class SPI_slave_scoreboard extends uvm_scoreboard;
 7.         `uvm_component_utils(SPI_slave_scoreboard);
 8.
 9.         uvm_analysis_export #(SPI_slave_seq_item) sb_export;
10.         uvm_tlm_analysis_fifo #(SPI_slave_seq_item) sb_fifo;
11.         SPI_slave_seq_item seq_item;
12.
13.         int error_count, correct_count;
14.
15.         function new(string name = "SPI_slave_scoreboard", uvm_component parent = null);
16.             super.new(name, parent);
17.         endfunction
18.
19.         function void build_phase(uvm_phase phase);
20.             super.build_phase(phase);
21.
22.             sb_export = new("sb_export", this);
23.             sb_fifo = new("sb_fifo", this);
24.         endfunction
25.
26.         function void connect_phase(uvm_phase phase);
27.             super.connect_phase(phase);
28.
29.             sb_export.connect(sb_fifo.analysis_export);
30.         endfunction
31.
32.         task run_phase(uvm_phase phase);
33.             super.run_phase(phase);
34.
35.             forever begin
36.                 sb_fifo.get(seq_item);
37.
38.                 if (seq_item.MISO != seq_item.MISO_golden || seq_item.rx_valid !=
seq_item.rx_valid_golden ||
39.                     seq_item.rx_data != seq_item.rx_data_golden) begin
40.                     error_count++;
41.
42.                     `uvm_error("run_phase", $sformatf("Wrong Output: %s",
43.                                 seq_item.convert2string()));
```

```
                end else begin
                    correct_count++;

                    `uvm_info("run_phase", $sformatf("Correct Output: %s",
                                seq_item.convert2string()), UVM_HIGH);
                end
            end
        endtask

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

            `uvm_info("report_phase", $sformatf("total correct: %d, total error: %d",
correct_count, error_count), UVM_MEDIUM);
        endfunction
    endclass
endpackage

```

Coverage Code

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

    class SPI_slave_cov extends uvm_component;
        `uvm_component_utils(SPI_slave_cov);

        uvm_analysis_export #(SPI_slave_seq_item) cov_export;
        uvm_tlm_analysis_fifo #(SPI_slave_seq_item) cov_fifo;
        SPI_slave_seq_item seq_item;

        covergroup cg;
            cp_rx_data: coverpoint seq_item.rx_data[9:8] iff (seq_item.rst_n) {
                bins all_values[] = {2'b00, 2'b01, 2'b10, 2'b11};

                bins write_address_write_data = (2'b00 => 2'b01);
                bins write_address_read_address = (2'b00 => 2'b10);

                bins write_data_write_address = (2'b01 => 2'b00);
                bins write_data_read_data = (2'b01 => 2'b11);

                bins read_address_write_address = (2'b10 => 2'b00);
                bins read_address_read_data = (2'b10 => 2'b11);

                bins read_data_write_data = (2'b11 => 2'b01);
                bins read_data_read_address = (2'b11 => 2'b10);
            }

            cp_SS_n: coverpoint seq_item.SS_n iff (seq_item.rst_n) {
                bins normal_sequence = (1 => 0 [*13] => 1);
                bins read_data_sequence = (1 => 0 [*23] => 1);

                bins MOSI_related = (1 => 0 [*4]);
            }

            cp_MOSI: coverpoint seq_item.MOSI iff (seq_item.rst_n) {
                bins write_address = (0 => 0 => 0);
                bins write_data = (0 => 0 => 1);
                bins read_address = (1 => 1 => 0);
                bins read_data = (1 => 1 => 1);
```

```
            }

            cr_SS_n_MOSI: cross cp_SS_n, cp_MOSI iff (seq_item.rst_n) {
                illegal_bins SS_n_normal = binsof(cp_SS_n.normal_sequence);
                illegal_bins SS_n_read_data = binsof(cp_SS_n.read_data_sequence);
            }
        endgroup

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

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

            cov_export = new("cov_export", this);
            cov_fifo = new("cov_fifo", this);
        endfunction

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

            cov_export.connect(cov_fifo.analysis_export);
        endfunction

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

            forever begin
                cov_fifo.get(seq_item);
                cg.sample();
            end
        endtask
    endclass
endpackage

```

Environment Code

```
package SPI_slave_env_pkg;
    import SPI_slave_scoreboard_pkg::*;
    import SPI_slave_cov_pkg::*;
    import SPI_slave_agent_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class SPI_slave_env extends uvm_env;
        `uvm_component_utils(SPI_slave_env)

        SPI_slave_scoreboard sb;
        SPI_slave_agent agent;
        SPI_slave_cov cov;

        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);

            sb = SPI_slave_scoreboard::type_id::create("sb", this);
```

```
            agent = SPI_slave_agent::type_id::create("agent", this);
            cov = SPI_slave_cov::type_id::create("cov", 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);
        endfunction
    endclass
endpackage

```

Test Code

```
package SPI_slave_test_pkg;
    import SPI_slave_rst_seq_pkg::*;
    import SPI_slave_main_seq_pkg::*;
    import SPI_slave_env_pkg::*;
    import SPI_slave_config_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class SPI_slave_test extends uvm_test;
        `uvm_component_utils(SPI_slave_test)

        SPI_slave_env env;
        SPI_slave_config SPI_slave_cfg;
        virtual SPI_slave_if SPI_slave_vif;
        SPI_slave_rst_seq rst_seq;
        SPI_slave_main_seq main_seq;

        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);
            SPI_slave_cfg = SPI_slave_config::type_id::create("SPI_slave_cfg" ,this);
            rst_seq = SPI_slave_rst_seq::type_id::create("rst_seq", this);
            main_seq = SPI_slave_main_seq::type_id::create("main_seq", this);

            if (!uvm_config_db #(virtual SPI_slave_if)::get(this, "", "SPI_SLAVE_IF",
SPI_slave_cfg.SPI_slave_vif))
                `uvm_fatal("build_phase", "Test - unable to get the virtual interface");

            SPI_slave_cfg.is_active = UVM_ACTIVE;

            uvm_config_db #(SPI_slave_config)::set(this, "*", "SPI_SLAVE_CFG", SPI_slave_cfg);
        endfunction

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

            phase.raise_objection(this);

            `uvm_info("run_phase", "reset asserted", UVM_LOW);

            rst_seq.start(env.agent.sequencer);
```

```
46.
47.             `uvm_info("run_phase", "reset deasserted", UVM_LOW);
48.
49.             `uvm_info("run_phase", "stimulus generation started", UVM_LOW);
50.
51.             main_seq.start(env.agent.sequencer);
52.
53.             `uvm_info("run_phase", "stimulus generation ended", UVM_LOW);
54.
55.             phase.drop_objection(this);
56.         endtask
57.     endclass
58. endpackage
59.
```

Top Module Code

```
 1. import SPI_slave_test_pkg::*;
 2. import uvm_pkg::*;
 3. `include "uvm_macros.svh"
 4.
 5. module SPI_slave_top();
 6.     bit clk;
 7.
 8.     always #1 clk = ~clk;
 9.
10.     SPI_slave_if SPI_slave_ifi (clk);
11.     SLAVE dut (SPI_slave_ifi.MOSI, SPI_slave_ifi.MISO, SPI_slave_ifi.SS_n, SPI_slave_ifi.clk,
SPI_slave_ifi.rst_n,
12.                SPI_slave_ifi.rx_data, SPI_slave_ifi.rx_valid, SPI_slave_ifi.tx_data,
SPI_slave_ifi.tx_valid);
13.     SPI_slave_golden golden (SPI_slave_ifi.clk, SPI_slave_ifi.rst_n, SPI_slave_ifi.SS_n,
14.                              SPI_slave_ifi.MOSI, SPI_slave_ifi.tx_valid,
SPI_slave_ifi.tx_data,
15.                              SPI_slave_ifi.rx_valid_golden, SPI_slave_ifi.rx_data_golden,
SPI_slave_ifi.MISO_golden);
16.
17.     initial begin
18.         uvm_config_db #(virtual SPI_slave_if)::set(null, "uvm_test_top", "SPI_SLAVE_IF",
SPI_slave_ifi);
19.         run_test("SPI_slave_test");
20.     end
21. endmodule
22.
```

Source Files List

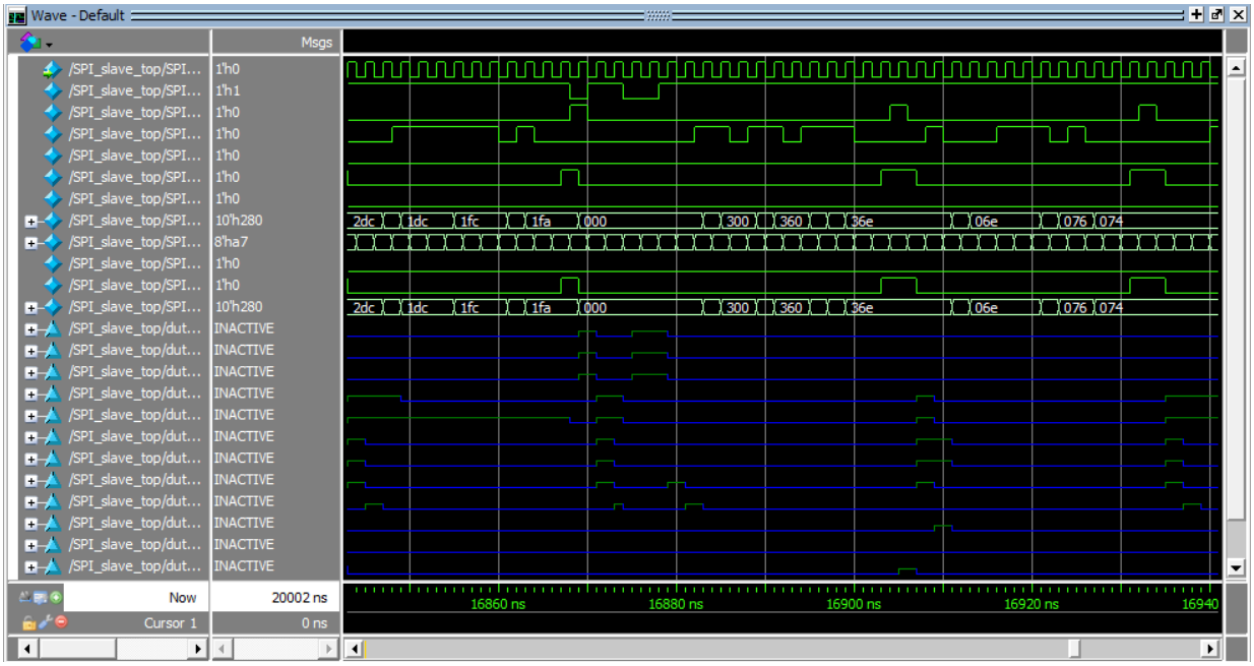
```
 1. SPI_slave_if.sv
 2. +define+SIM
 3. SPI_slave.sv
 4. SPI_slave_golden.v
 5. shared_pkg.sv
 6. SPI_slave_seq_item.sv
 7. SPI_slave_rst_seq.sv
 8. SPI_slave_main_seq.sv
 9. SPI_slave_sequencer.sv
10. SPI_slave_config.sv
11. SPI_slave_driver.sv
12. SPI_slave_monitor.sv
13. SPI_slave_agent.sv
```

```
14. SPI_slave_scoreboard.sv
15. SPI_slave_cov.sv
16. SPI_slave_env.sv
17. SPI_slave_test.sv
18. SPI_slave_top.sv
19.
```

Do File

```
1. vlib work
2. vlog -f src_files.list +cover -covercells
3. vsim -voptargs=+acc work.SPI_slave_top -classdebug -uvmcontrol=all -cover -l sim.log -sv_seed
1909088641
4. add wave /SPI_slave_top/SPI_slave_ifi/*
5. add wave /SPI_slave_top/dut/a_rst_MISO /SPI_slave_top/dut/a_rst_rx_valid
/SPI_slave_top/dut/a_rst_rx_data /SPI_slave_top/dut/a_rx_valid_write_address
/SPI_slave_top/dut/a_rx_valid_write_data /SPI_slave_top/dut/a_rx_valid_read_address
/SPI_slave_top/dut/a_rx_valid_read_data /SPI_slave_top/dut/a_IDLE_to_CHK_CMD
/SPI_slave_top/dut/a_CHK_CMD_to_WRITE /SPI_slave_top/dut/a_CHK_CMD_to_READ_ADD
/SPI_slave_top/dut/a_CHK_CMD_to_READ_DATA /SPI_slave_top/dut/a_WRITE_to_IDLE
/SPI_slave_top/dut/a_READ_ADD_to_IDLE /SPI_slave_top/dut/a_READ_DATA_to_IDLE
6. coverage save SPI_slave.ucdb -onexit
7. run -all
8.
```

Waveform



Coverage Report

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         74        74         0   100.00%

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

Toggle Coverage for instance /SPI_slave_top/SPI_slave_ifi --

                                           Node      1H->0L      0L->1H  "Coverage"
                                           ----------------------------------------
                                           MISO           1           1      100.00
                                     MISO_golden          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_data_golden[9-0]       1           1      100.00
                                       rx_valid           1           1      100.00
                                  rx_valid_golden         1           1      100.00
                                     tx_data[7-0]         1           1      100.00
                                       tx_valid           1           1      100.00

Total Node Count     =         37
Toggled Node Count   =         37
Untoggled Node Count =          0
```

```
Assertion Coverage:
    Assertions                      14        14         0   100.00%
-------------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
-------------------------------------------------------------------------
/SPI_slave_top/dut/a_rst_MISO
                     SPI_slave.sv(187)                  0          1
/SPI_slave_top/dut/a_rst_rx_valid
                     SPI_slave.sv(190)                  0          1
/SPI_slave_top/dut/a_rst_rx_data
                     SPI_slave.sv(193)                  0          1
/SPI_slave_top/dut/a_rx_valid_write_address
                     SPI_slave.sv(196)                  0          1
/SPI_slave_top/dut/a_rx_valid_write_data
                     SPI_slave.sv(199)                  0          1
/SPI_slave_top/dut/a_rx_valid_read_address
                     SPI_slave.sv(202)                  0          1
/SPI_slave_top/dut/a_rx_valid_read_data
                     SPI_slave.sv(205)                  0          1
/SPI_slave_top/dut/a_IDLE_to_CHK_CMD
                     SPI_slave.sv(208)                  0          1
/SPI_slave_top/dut/a_CHK_CMD_to_WRITE
                     SPI_slave.sv(211)                  0          1
/SPI_slave_top/dut/a_CHK_CMD_to_READ_ADD
                     SPI_slave.sv(214)                  0          1
/SPI_slave_top/dut/a_CHK_CMD_to_READ_DATA
                     SPI_slave.sv(217)                  0          1
/SPI_slave_top/dut/a_WRITE_to_IDLE
                     SPI_slave.sv(220)                  0          1
/SPI_slave_top/dut/a_READ_ADD_to_IDLE
                     SPI_slave.sv(223)                  0          1
/SPI_slave_top/dut/a_READ_DATA_to_IDLE
                     SPI_slave.sv(226)                  0          1
```

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        27        27         0   100.00%

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

Branch Coverage for instance /SPI_slave_top/dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File SPI_slave.sv
-----------------------------------IF Branch-----------------------------------
    19                                      3529     Count coming in to IF
    19              1                        510             if (~rst_n) begin
    22              1                       3019             else begin
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    30                                      1572     Count coming in to IF
    30              1                        460                     if (SS_n)
    32              1                       1112                     else
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-----------------------------------
    36                                      1322     Count coming in to IF
    38              1                       1322                     else begin
Branch totals: 1 hit of 1 branch = 100.00%

-----------------------------------IF Branch-----------------------------------
    39                                      1322     Count coming in to IF
    39              1                        911                         if (~MOSI)
    41              1                        411                         else begin
Branch totals: 2 hits of 2 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
    42                                    411     Count coming in to IF
    42              1                     304                             if (!received_address)
    44              1                     107                             else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    50                                   2573     Count coming in to IF
    50              1                     263                       if (SS_n)
    52              1                    2310                       else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    56                                   1874     Count coming in to IF
    56              1                     149                       if (SS_n)
    58              1                    1725                       else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    62                                    587     Count coming in to IF
    62              1                      48                       if (SS_n)
    64              1                     539                       else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    71                                  10001     Count coming in to IF
    71              1                     511              if (~rst_n) begin
    77              1                    9490              else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    86                                   3973     Count coming in to IF
    86              1                    3455                          if (counter > 0) begin
    90              1                     518                          else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    95                                   2734     Count coming in to IF
    95              1                    2238                          if (counter > 0) begin
    99              1                     496                          else begin
Branch totals: 2 hits of 2 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
   105                                  1013     Count coming in to IF
   105             1                     100                        if (tx_valid) begin
   115             1                     913                        else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   107                                   100     Count coming in to IF
   107             1                      92                             if (counter > 0) begin
   111             1                       8                             else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   116                                   913     Count coming in to IF
   116             1                     790                             if (counter > 0 && !rx_valid) begin
   120             1                     123                             else begin
Branch totals: 2 hits of 2 branches = 100.00%
```

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                       5         5         0   100.00%

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

Condition Coverage for instance /SPI_slave_top/dut --

  File SPI_slave.sv
----------------Focused Condition View-------------------
Line       86 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       95 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

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

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

```
----------------Focused Condition View-------------------
Line       107 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       116 Item    1  ((counter > 0) && ~rx_valid)
Condition totals: 2 of 2 input terms covered = 100.00%

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

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  --------------------------
  Row   1:          1  (counter > 0)_0       -
  Row   2:          1  (counter > 0)_1       ~rx_valid
  Row   3:          1  rx_valid_0            (counter > 0)
  Row   4:          1  rx_valid_1            (counter > 0)
```

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

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/SPI_slave_top/dut/c_rst_MISO           SLAVE  Verilog  SVA  SPI_slave.sv(188)
                                                                              511 Covered
/SPI_slave_top/dut/c_rst_rx_valid       SLAVE  Verilog  SVA  SPI_slave.sv(191)
                                                                              511 Covered
/SPI_slave_top/dut/c_rst_rx_data        SLAVE  Verilog  SVA  SPI_slave.sv(194)
                                                                              511 Covered
/SPI_slave_top/dut/c_rx_valid_write_address
                                        SLAVE  Verilog  SVA  SPI_slave.sv(197)
                                                                               45 Covered
/SPI_slave_top/dut/c_rx_valid_write_data SLAVE  Verilog  SVA  SPI_slave.sv(200)
                                                                               39 Covered
/SPI_slave_top/dut/c_rx_valid_read_address
                                        SLAVE  Verilog  SVA  SPI_slave.sv(203)
                                                                               36 Covered
/SPI_slave_top/dut/c_rx_valid_read_data SLAVE  Verilog  SVA  SPI_slave.sv(206)
                                                                               23 Covered
/SPI_slave_top/dut/c_IDLE_to_CHK_CMD    SLAVE  Verilog  SVA  SPI_slave.sv(209)
                                                                              859 Covered
/SPI_slave_top/dut/c_CHK_CMD_to_WRITE   SLAVE  Verilog  SVA  SPI_slave.sv(212)
                                                                              436 Covered
/SPI_slave_top/dut/c_CHK_CMD_to_READ_ADD SLAVE Verilog  SVA  SPI_slave.sv(215)
                                                                              276 Covered
/SPI_slave_top/dut/c_CHK_CMD_to_READ_DATA
                                        SLAVE  Verilog  SVA  SPI_slave.sv(218)
                                                                               98 Covered
/SPI_slave_top/dut/c_WRITE_to_IDLE      SLAVE  Verilog  SVA  SPI_slave.sv(221)
                                                                              240 Covered
/SPI_slave_top/dut/c_READ_ADD_to_IDLE   SLAVE  Verilog  SVA  SPI_slave.sv(224)
                                                                              134 Covered
/SPI_slave_top/dut/c_READ_DATA_to_IDLE  SLAVE  Verilog  SVA  SPI_slave.sv(227)
                                                                               45 Covered
```

```
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 /SPI_slave_top/dut --

FSM_ID: cs
    Current State Object : cs
    ----------------------
    State Value MapInfo :
    ---------------------
Line           State Name               Value
----           ----------               -----
  29                 IDLE                   0
  35              CHK_CMD                   2
  61            READ_DATA                   4
  55             READ_ADD                   3
  49                WRITE                   1
    Covered States :
    ----------------
                    State           Hit_count
                    -----           ---------
                     IDLE                 949
                  CHK_CMD                 911
                READ_DATA                 202
                 READ_ADD                 562
                    WRITE                 905
```

Covered Transitions :

Line	Trans_ID	Hit_count	Transition
33	0	911	IDLE -> CHK_CMD
45	1	104	CHK_CMD -> READ_DATA
43	2	286	CHK_CMD -> READ_ADD
40	3	469	CHK_CMD -> WRITE
37	4	52	CHK_CMD -> IDLE
63	5	104	READ_DATA -> IDLE
57	6	285	READ_ADD -> IDLE
51	7	469	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	37	37	0	100.00%

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

Statement Coverage for instance /SPI_slave_top/dut --

Line	Item	Count	Source
File SPI_slave.sv			
1			module SLAVE (MOSI, MISO, SS_n, clk, rst_n, rx_data, rx_valid, tx_data, tx_valid);
2			localparam IDLE = 3'b000;
3			localparam WRITE = 3'b001;
4			localparam CHK_CMD = 3'b010;
5			localparam READ_ADD = 3'b011;
6			localparam READ_DATA = 3'b100;
7			
8			input MOSI, clk, rst_n, SS_n, tx_valid;
9			input [7:0] tx_data;
10			output reg [9:0] rx_data;
11			output reg rx_valid, MISO;
12			
13			reg [3:0] counter;
14			reg received_address;
15			
16			reg [2:0] cs, ns;
17			
18	1	3529	always @(posedge clk) begin
19			if (~rst_n) begin
20	1	510	cs <= IDLE;
21			end
22			else begin
23	1	3019	cs <= ns;
24			end
25			end
26			
27	1	7928	always @(*) begin
28			case (cs)

```
29                                                      IDLE : begin
30                                                          if (SS_n)
31              1                    460                        ns = IDLE;
32                                                          else
33              1                   1112                        ns = CHK_CMD;
34                                                      end
35                                                      CHK_CMD : begin
36                                                          if (SS_n)
37                                                              ns = IDLE;
38                                                          else begin
39                                                              if (~MOSI)
40              1                    911                            ns = WRITE;
41                                                              else begin
42                                                                  if (!received_address)
43              1                    304                                ns = READ_ADD;
44                                                                  else
45              1                    107                                ns = READ_DATA;
46                                                              end
47                                                          end
48                                                      end
49                                                      WRITE : begin
50                                                          if (SS_n)
51              1                    263                        ns = IDLE;
52                                                          else
53              1                   2310                        ns = WRITE;
54                                                      end
55                                                      READ_ADD : begin
56                                                          if (SS_n)
57              1                    149                        ns = IDLE;
58                                                          else
59              1                   1725                        ns = READ_ADD;
60                                                      end
61                                                      READ_DATA : begin
62                                                          if (SS_n)
63              1                     48                        ns = IDLE;
64                                                          else
65              1                    539                        ns = READ_DATA;
66                                                      end
67                                                  endcase
68                                              end
69
```

```
70          1          10001      always @(posedge clk) begin
71                                    if (~rst_n) begin
72          1            511            rx_data <= 0;
73          1            511            rx_valid <= 0;
74          1            511            received_address <= 0;
75          1            511            MISO <= 0;
76                                    end
77                                    else begin
78                                        case (cs)
79                                            IDLE : begin
80          1            911                rx_valid <= 0;
81                                        end
82                                        CHK_CMD : begin
83          1            859                counter <= 10;
84                                        end
85                                        WRITE : begin
86                                            if (counter > 0) begin
87          1           3455                    rx_data[counter-1] <= MOSI;
88          1           3455                    counter <= counter - 1;
89                                            end
90                                            else begin
91          1            518                    rx_valid <= 1;
92                                            end
93                                        end
94                                        READ_ADD : begin
95                                            if (counter > 0) begin
96          1           2238                    rx_data[counter-1] <= MOSI;
97          1           2238                    counter <= counter - 1;
98                                            end
99                                            else begin
100         1            496                    rx_valid <= 1;
101         1            496                    received_address <= 1;
102                                           end
103                                       end
104                                       READ_DATA : begin
105                                           if (tx_valid) begin
106         1            100                    rx_valid <= 0;
107                                               if (counter > 0) begin
108         1             92                        MISO <= tx_data[counter-1];
109         1             92                        counter <= counter - 1;
```

```
107                                               if (counter > 0) begin
108         1             92                        MISO <= tx_data[counter-1];
109         1             92                        counter <= counter - 1;
110                                               end
111                                               else begin
112         1              8                        received_address <= 0;
113                                               end
114                                           end
115                                           else begin
116                                               if (counter > 0 && !rx_valid) begin
117         1            790                        rx_data[counter-1] <= MOSI;
118         1            790                        counter <= counter - 1;
119                                               end
120                                               else begin
121         1            123                        rx_valid <= 1;
122         1            123                        counter <= 8;
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         72        72         0   100.00%

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

Toggle Coverage for instance /SPI_slave_top/dut --

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

Total Node Count     =         36
Toggled Node Count   =         36
Untoggled Node Count =          0
```

```
Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses         4        na        na        na
            Covergroup Bins        23        23         0   100.00%
--------------------------------------------------------------------------------------------------------------
Covergroup                                           Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /SPI_slave_cov_pkg/SPI_slave_cov/cg            100.00%        100          -    Covered
    covered/total bins:                                   23         23          -
    missing/total bins:                                    0         23          -
    % Hit:                                            100.00%        100          -
    Coverpoint cp_rx_data                             100.00%        100          -    Covered
        covered/total bins:                               12         12          -
        missing/total bins:                                0         12          -
        % Hit:                                        100.00%        100          -
    Coverpoint cp_SS_n                                100.00%        100          -    Covered
        covered/total bins:                                3          3          -
        missing/total bins:                                0          3          -
        % Hit:                                        100.00%        100          -
    Coverpoint cp_MOSI                                100.00%        100          -    Covered
        covered/total bins:                                4          4          -
        missing/total bins:                                0          4          -
        % Hit:                                        100.00%        100          -
    Cross cr_SS_n_MOSI                                100.00%        100          -    Covered
        covered/total bins:                                4          4          -
        missing/total bins:                                0          4          -
        % Hit:                                        100.00%        100          -
 Covergroup instance \/SPI_slave_cov_pkg::SPI_slave_cov::cg
                                                      100.00%        100          -    Covered
    covered/total bins:                                   23         23          -
    missing/total bins:                                    0         23          -
    % Hit:                                            100.00%        100          -
```

```
    Coverpoint cp_rx_data                          100.00%        100          -    Covered
        covered/total bins:                             12         12          -
        missing/total bins:                              0         12          -
        % Hit:                                      100.00%        100          -
        bin all_values[0]                              3225          1          -    Covered
        bin all_values[1]                              1999          1          -    Covered
        bin all_values[2]                              2273          1          -    Covered
        bin all_values[3]                              1717          1          -    Covered
        bin write_address_write_data                    152          1          -    Covered
        bin write_address_read_address                  254          1          -    Covered
        bin write_data_write_address                    136          1          -    Covered
        bin write_data_read_data                         48          1          -    Covered
        bin read_address_write_address                  175          1          -    Covered
        bin read_address_read_data                      120          1          -    Covered
        bin read_data_write_data                         32          1          -    Covered
        bin read_data_read_address                       42          1          -    Covered
    Coverpoint cp_SS_n                             100.00%        100          -    Covered
        covered/total bins:                              3          3          -
        missing/total bins:                              0          3          -
        % Hit:                                      100.00%        100          -
        bin normal_sequence                            197          1          -    Covered
        bin read_data_sequence                          19          1          -    Covered
        bin MOSI_related                               439          1          -    Covered
    Coverpoint cp_MOSI                             100.00%        100          -    Covered
        covered/total bins:                              4          4          -
        missing/total bins:                              0          4          -
        % Hit:                                      100.00%        100          -
        bin write_address                              2552          1          -    Covered
        bin write_data                                 1197          1          -    Covered
        bin read_address                                916          1          -    Covered
        bin read_data                                   633          1          -    Covered
```

```
    Cross cr_SS_n_MOSI                             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 <MOSI_related,read_data>                46          1          -    Covered
            bin <MOSI_related,write_data>               81          1          -    Covered
            bin <MOSI_related,read_address>             75          1          -    Covered
            bin <MOSI_related,write_address>            77          1          -    Covered
        Illegal and Ignore Bins:
            illegal_bin SS_n_read_data                  17                     -    Occurred
            illegal_bin SS_n_normal                    104                     -    Occurred
```

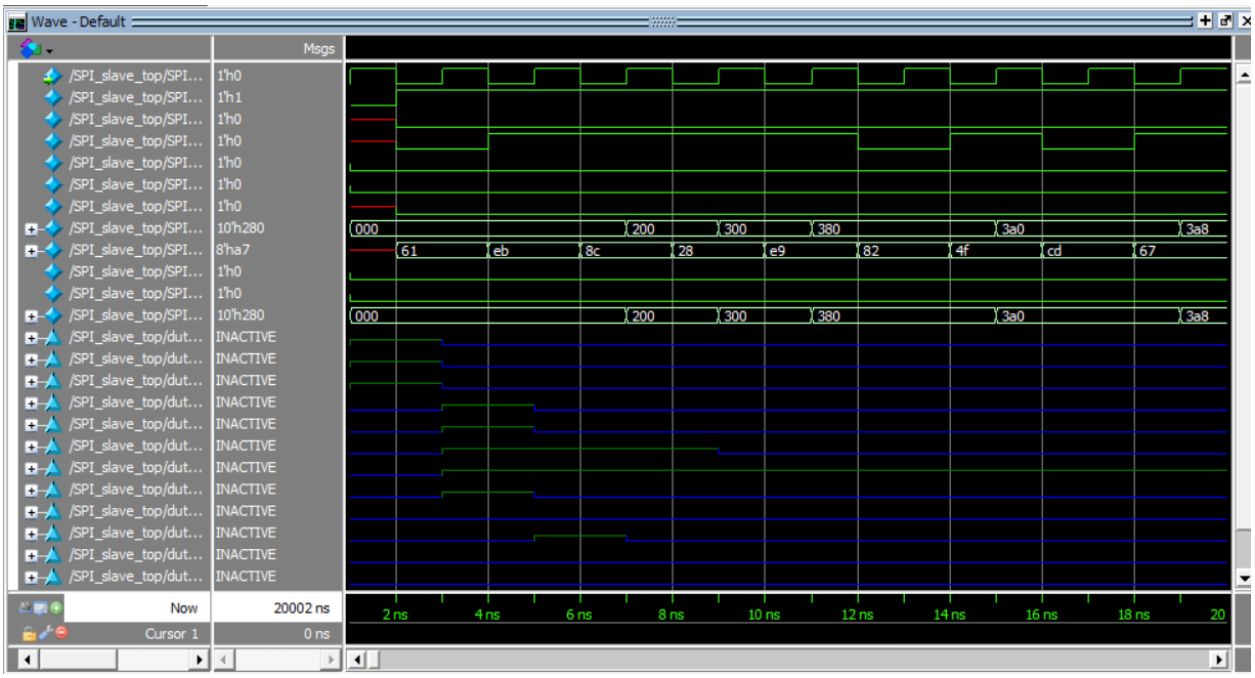
Bug Report

Bug	Fix
When in CHK_CMD, READ_ADD and READ_DATA cases were switched.	Edited the condition of the if condition to go to READ_ADD when it received_address is 0 not 1.
It never enters the phase of sending data to the Master in the case of READ_DATA.	Added that rx_valid should be low not only counter should be greater than 0 in the if condition.

Note: the WRITE encoding is 3’b001 while CHK_CMD is 3’b010 but we didn’t consider it as an error as it doesn’t affect the functionality of the design.

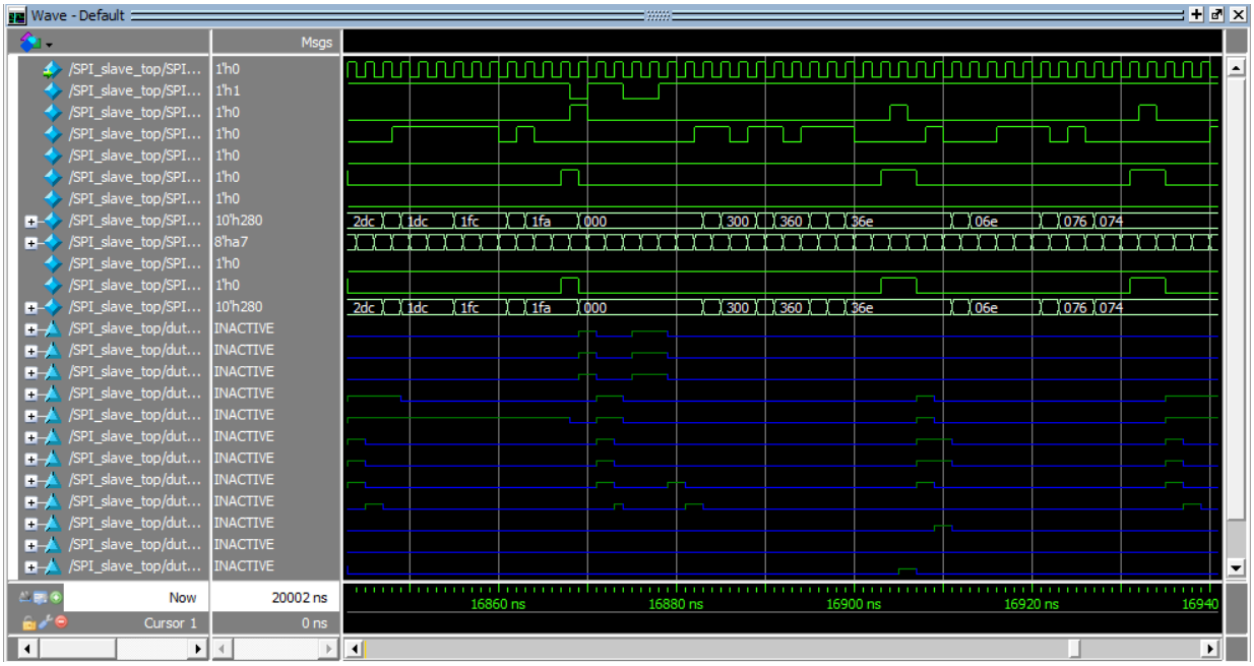
Sequences Explaining

Reset Sequence



The reset is asserted in the sequence item, then the driver assigns all the input signals in the interface to the values in the sequence item, so the output values are low.

Main Sequence



The values in the sequence item are randomized and there is a MOSI array that is randomized in the sequence item to make sure that the MOSI sequence along the whole operation is valid and there is a shared package that contains counter, period, flag indicates that there is an address in the read and an array that holds the MOSI array value along the whole operation as the MOSI array is randomized every clock cycle, and the sequence is continue randomizing along 10000 clock cycle, and every clock cycle the driver assigns all the input signals in the interface to the values in the sequence item.

Assertion Table

Feature	Assertion
When the reset is asserted, MISO should be low.	@(posedge clk) rst_n == 0 |=> MISO == 0;
When the reset is asserted, rx_valid should be low.	@(posedge clk) rst_n == 0 |=> rx_valid == 0;
When the reset is asserted, rx_data should be low.	@(posedge clk) rst_n == 0 |=> rx_data == 0;
When SS_n falls, in the next 3 clock cycles, if the pattern of 3'b000 MOSI happened, rx_valid should be high after another 10 clock cycles.	@(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 0) ##1 (MOSI == 0) ##1 (MOSI == 0) |-> ##10 rx_valid;
When SS_n falls, in the next 3 clock cycles, if the pattern of 3'b001 MOSI happened, rx_valid should be high after another 10 clock cycles.	@(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 0) ##1 (MOSI == 0) ##1 (MOSI == 1) |-> ##10 rx_valid;

When SS_n falls, in the next 3 clock cycles, if the pattern of 3'b110 MOSI happened, rx_valid should be high after another 10 clock cycles.	@(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 1) ##1 (MOSI == 1) ##1 (MOSI == 0) |-> ##10 rx_valid;
When SS_n falls, in the next 3 clock cycles, if the pattern of 3'b111 MOSI happened, rx_valid should be high after another 10 clock cycles.	@(posedge clk) disable iff (!rst_n) ($fell(SS_n)) ##1 (MOSI == 1) ##1 (MOSI == 1) ##1 (MOSI == 1) |-> ##10 rx_valid;
When the current case is IDLE and SS_n is low, after one clock cycle, the current case should be CHK_CMD.	@(posedge clk) disable iff (!rst_n) (cs == IDLE && SS_n == 0) |=> cs == CHK_CMD;
When the current case is CHK_CMD, SS_n is low and MOSI is low, after one clock cycle, the current case should be WRITE.	@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && SS_n == 0 && MOSI == 0) |=> cs == WRITE;
When the current case is CHK_CMD, SS_n is low, MOSI is high and received_address is low, after one clock cycle, the current case should be READ_ADD.	@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && SS_n == 0 && MOSI == 1 && received_address == 0) |=> cs == READ_ADD;
When the current case is CHK_CMD, SS_n is low, MOSI is high and received_address is high, after one clock cycle, the current case should be READ_DATA.	@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && SS_n == 0 && MOSI == 1 && received_address == 1) |=> cs == READ_DATA;
When the current case is WRITE and SS_n is high, after one clock cycle, the current case should be IDLE.	@(posedge clk) disable iff (!rst_n) (cs == WRITE && SS_n == 1) |=> cs == IDLE;
When the current case is READ_ADD and SS_n is high, after one clock cycle, the current case should be IDLE.	@(posedge clk) disable iff (!rst_n) (cs == READ_ADD && SS_n == 1) |=> cs == IDLE;
When the current case is READ_DATA and SS_n is high, after one clock cycle, the current case should be IDLE.	@(posedge clk) disable iff (!rst_n) (cs == READ_DATA && SS_n == 1) |=> cs == IDLE;

Part 2 (RAM)

Design

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


input      [9:0] din;
input            clk, rst_n, rx_valid;
output reg [7:0] dout;
output reg       tx_valid;
reg [7:0] MEM [255:0];
reg [7:0] Rd_Addr, Wr_Addr;
always @(posedge clk) begin
    if (~rst_n) begin
        dout <= 0;
        tx_valid <= 0;
        Rd_Addr <= 0;
        Wr_Addr <= 0;
    end
    else
        if (rx_valid) begin
            case (din[9:8])
                2'b00 : Wr_Addr <= din[7:0];
                2'b01 : MEM[Wr_Addr] <= din[7:0];
                2'b10 : Rd_Addr <= din[7:0];
                2'b11 : dout <= MEM[Rd_Addr]; //1
                default : dout <= 0;
            endcase
            tx_valid <= (din[9] && din[8])? 1'b1 : 1'b0; //2
        end
end
endmodule
```

Bug report

1- changed the case of read data to read from the Rd_Addr instead of Wr_Addr

2_ put the line that update the tx_valid into the if condition of rx_valid and removed the rx_valid from the condition of updating it to ensure that it's high till the end of connection

Verification Plan

Label	Design Requirement Description	Functional Coverage	Stimulus Generation	Functionality Check
Label_1	Reset Functionality	Cover reset assertion during different states (active reset and normal operation)	Directed: Assert reset at time 0 and randomly during operation (2% probability)	Check output dout=0, tx_valid=0, and address=0 after reset assertion
Label_2	Write Address Operation (din[9:8] = 2'b00)	Cover all address values (0-255) for write address operation	Directed + Randomized: opcode=2'b00, address values 0x00, 0xFF, 0xAA, 0x55, and 80% random addresses	Check tx_valid=0 when write address is performed
Label_3	Write Data Operation (din[9:8] = 2'b01)	Cover all data values (0-255) for write data operation	Directed + Randomized: opcode=2'b01, data values 0x00, 0xFF, 0xAA, 0x55, and 80% random data	Check tx_valid=0 when write data is performed
Label_4	Read Address Operation (din[9:8] = 2'b10)	Cover all address values (0-255) for read address operation	Directed + Randomized: opcode=2'b10, address values 0x00, 0xFF, 0xAA, 0x55, and 80% random addresses	Check tx_valid=0 when read address is performed
Label_5	Read Data Operation (din[9:8] = 2'b11)	Cover read_data bin	Directed + Randomized: opcode=2'b11	Check tx_valid=1 when read data is performed
Label_6	Full Write-Read Sequence	Cover full sequence: write_address → write_data → read_address → read_data	Write_read_sequence generates complete write-read cycles with 60% transition probability	Check data integrity: written data matches read data
Label_7	rx_valid Control	Cover rx_valid=1 and rx_valid=0	Randomized: rx_valid dist {1:=95, 0:=5}	Check that when rx_valid=0, no state changes occur
Label_8	Multiple Writes to Same Address	Cover multiple consecutive writes to the same address with different data	Directed: Write address once, then write multiple different data values	Check that last written data is what gets read
Label_9	Multiple Reads from Same Address	Cover multiple consecutive reads from the same address	Directed: Set read address once, then perform multiple read data operations	Check that all reads return the same data
Label_10	Memory Depth Coverage	Cover writes and reads to all 256 memory locations	Main_sequence with 100 random operations to achieve memory coverage	Check all memory locations can be written and read
Label_11	Back-to-Back Operations	Cover all combinations of consecutive operation transitions (16 combinations)	Main_sequence generates random back-to-back operations with rx_valid=1	Check correct behavior for all operation sequences
Label_12	Read from Unwritten Location	Cover scenario where read occurs before any write to that address	Read_only_sequence at the beginning (before writes)	Check that read returns initialized value (0x00)

Assertion Table

Assertion Name	Description	Property (SVA Code)
assert_reset_outputs	During active reset (rst_n=0), verify that all outputs (tx_valid and dout) are cleared to 0	`property p_reset_outputs; @(posedge clk) (!rst_n) |-> (tx_valid == 0 && dout == 0); endproperty assert_reset_outputs: assert property(p_reset_outputs) else $error("Outputs not cleared during reset");`
assert_tx_valid_input	When rx_valid is asserted and operation is NOT read_data (din[9:8] != 2'b11), tx_valid must remain low	`property p_tx_valid_during_input; @(posedge clk) (rx_valid && (din[9:8] != 2'b11)) |-> (tx_valid == 0); endproperty assert_tx_valid_input: assert property(p_tx_valid_during_input) else $error("tx_valid asserted during non-read operation");`
assert_tx_valid_read	When read_data operation occurs (rx_valid=1, din[9:8]=2'b11), tx_valid must pulse high for one cycle then return to zero	`property p_tx_valid_read_data; @(posedge clk) (rx_valid && din[9:8] == 2'b11) |=> (tx_valid == 1) ##1 (tx_valid == 0); endproperty assert_tx_valid_read: assert property(p_tx_valid_read_data) else $error("tx_valid pulse incorrect after read_data");`
assert_write_sequence	Write address operation (din[9:8]=2'b00) should eventually be followed by write data operation (din[9:8]=2'b01)	`sequence s_write_address; (rx_valid && din[9:8] == 2'b00); endsequence sequence s_write_data; (rx_valid && din[9:8] == 2'b01); endsequence property p_write_address_to_data; @(posedge clk) s_write_address |-> ##[1:$] s_write_data; endproperty assert write sequence: assert property(p write address to data) else $error("Write Address not followed by Write Data");`
assert_read_sequence	Read address operation (din[9:8]=2'b10) should eventually be followed by read data operation (din[9:8]=2'b11)	`sequence s_read_address; (rx_valid && din[9:8] == 2'b10); endsequence sequence s_read_data; (rx_valid && din[9:8] == 2'b11); endsequence property p_read_address_to_data; @(posedge clk) s_read_address |-> ##[1:$] s_read_data; endproperty assert_read_sequence: assert property(p_read_address_to_data) else $error("Read Address not followed by Read Data");`
assert_tx_valid_source	tx_valid can only be asserted if a read_data operation was performed in the previous cycle	`property p_tx_valid_only_on_read; @(posedge clk) disable iff(!rst_n) (tx_valid == 1) |-> $past(rx_valid && din[9:8] == 2'b11); endproperty assert_tx_valid_source: assert property(p_tx_valid_only_on_read) else $error("tx_valid asserted without read_data operation");`
assert_no_op	When rx_valid is de-asserted, no operations should occur and tx_valid must remain low	`property p_no_op_when_rx_invalid; @(posedge clk) disable iff(!rst_n) (!rx_valid) |-> (tx_valid == 0); endproperty assert_no_op: assert property(p_no_op_when_rx_invalid) else $error("Operation occurred when rx_valid=0");`
assert_dout_stability	dout signal should remain stable unless a read_data operation is being performed	`property p_dout_stable_when_no_read; @(posedge clk) disable iff(!rst_n) !(rx_valid && din[9:8] == 2'b11) |=> $stable(dout); endproperty assert_dout_stability: assert property(p_dout_stable_when_no_read) else $error("dout changed without read_data operation");`
assert_tx_valid_pulse_width	tx_valid should be a single-cycle pulse; if high in one cycle, must be low in the next	`property p_tx_valid_pulse; @(posedge clk) disable iff(!rst_n) (tx_valid == 1) |=> (tx_valid == 0); endproperty assert_tx_valid_pulse_width: assert property(p_tx_valid_pulse) else $error("tx_valid not single-cycle pulse");`
assert_rx_valid_stability	When rx_valid rises, it should remain stable for at least one clock cycle to prevent glitches	`property p_rx_valid_stable_during_operation; @(posedge clk) disable iff(!rst_n) $rose(rx_valid) |-> rx_valid; endproperty assert_rx_valid_stability: assert property(p_rx_valid_stable_during_operation) else $error("rx_valid glitch detected");`
assert_dout_valid_data	When tx_valid is asserted, dout must contain valid data (not X or Z)	`property p_dout_valid_with_tx; @(posedge clk) disable iff(!rst_n) (tx_valid == 1) |-> !$isunknown(dout); endproperty assert_dout_valid_data: assert property(p_dout_valid_with_tx) else $error("dout contains X/Z when tx_valid asserted");`
assert_address_range	Address value din[7:0] should be within valid range during address operations	`property p_address_range_check; @(posedge clk) disable iff(!rst_n) (rx_valid && (din[9:8] == 2'b00 || din[9:8] == 2'b10)) |-> (din[7:0] >= 0 && din[7:0] <= 255); endproperty assert_address_range: assert property(p_address_range_check) else $error("Address out of valid range");`
assert_operation_exclusivity	Only one operation type should be active at a time (no simultaneous write and read)	`property p_no_simultaneous_wr_rd; @(posedge clk) disable iff(!rst_n) (rx_valid && (din[9:8] == 2'b00 || din[9:8] == 2'b01)) |-> !(din[9:8] == 2'b10 || din[9:8] == 2'b11); endproperty assert_operation_exclusivity: assert property(p_no_simultaneous_wr_rd) else $error("Conflicting operations detected");`

Do file

vlib work

vlog -f src_files.list +cover -covercells

vsim -voptargs=+acc work.RAM_top -classdebug -uvmcontrol=all -cover -l sim.log -sv_seed 1463002509

add wave /RAM_top/vif/*

add wave /RAM_top/dut/ram_assertions_inst/assert_reset_outputs
/RAM_top/dut/ram_assertions_inst/assert_tx_valid_input
/RAM_top/dut/ram_assertions_inst/assert_tx_valid_read

/RAM_top/dut/ram_assertions_inst/assert_write_sequence
/RAM_top/dut/ram_assertions_inst/assert_read_sequence

coverage save RAM.ucdb -onexit

run -all

Source File

src_files.list

```
RAM_if.sv
RAM.v
RAM_golden.v
RAM_assertions.sv
RAM_config_obj_pkg.sv
RAM_seq_item.sv
RAM_sequence.sv
RAM_driver.sv
RAM_monitor.sv
RAM_scoreboard.sv
RAM_sequencer.sv
RAM_agent.sv
RAM_coverage.sv
RAM_env.sv
RAM_test.sv
RAM_top.sv
```

Interface

```
interface RAM_if(input bit clk);

    logic rst_n;

    logic rx_valid;

    logic [9:0] din;

    logic [7:0] dout;

    logic tx_valid;

    logic [7:0] dout_ref;

    logic tx_valid_ref;

endinterface
```

Golden model

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

    parameter MEM_DEPTH = 256;

    parameter ADDR_SIZE = 8;
```

```
    input clk, rst_n, rx_valid;
    input [ADDR_SIZE + 1 : 0] din;
    output reg tx_valid;
    output reg [ADDR_SIZE - 1 : 0] dout;
    reg [ADDR_SIZE - 1 : 0] wr_address;
    reg [ADDR_SIZE - 1 : 0] rd_address;
    reg [ADDR_SIZE - 1 : 0] mem [MEM_DEPTH - 1 : 0];


    always @(posedge clk) begin
        if (~rst_n) begin
            tx_valid <= 0;
            dout <= 0;
            wr_address <= 0;
            rd_address <= 0;
        end else if (rx_valid) begin
            case (din[ADDR_SIZE + 1 : ADDR_SIZE])
                2'b00: begin
                    tx_valid <= 0;
                    wr_address <= din[ADDR_SIZE - 1 : 0];
                end
                2'b01: begin
                    tx_valid <= 0;
                    mem[wr_address] <= din[ADDR_SIZE - 1 : 0];
                end
                2'b10: begin
                    tx_valid <= 0;
                    rd_address <= din[ADDR_SIZE - 1 : 0];
                end
                2'b11: begin
                    tx_valid <= 1;
                    dout <= mem[rd_address];
                end
                default: begin
```

```
                    tx_valid <= 0;
                    dout <= 0;
                end
            endcase
        end
    end
endmodule
```

Assertions

```
module RAM_assertions(
    input clk,
    input rst_n,
    input rx_valid,
    input [9:0] din,
    input [7:0] dout,
    input tx_valid
);
    property p_reset_outputs;
        @(posedge clk) (!rst_n) |=> (tx_valid == 0 && dout == 0);
    endproperty

    assert_reset_outputs: assert property(p_reset_outputs)
        else $error("Assertion Failed: Outputs not low during reset");
```

```
    cover_reset_outputs: cover property(p_reset_outputs);
```

```
    property p_tx_valid_during_input;
        @(posedge clk) (rst_n && rx_valid && (din[9:8] != 2'b11)) |=> (tx_valid == 0);
    endproperty
```

```
    assert_tx_valid_input: assert property(p_tx_valid_during_input)
        else $error("Assertion Failed: tx_valid asserted during address/data input");
```

```
    cover_tx_valid_input: cover property(p_tx_valid_during_input);


    property p_tx_valid_read_data;
        @(posedge clk) (rst_n && rx_valid && din[9:8] == 2'b11) |=> (tx_valid == 1);
    endproperty


    assert_tx_valid_read: assert property(p_tx_valid_read_data)
        else $error("Assertion Failed: tx_valid behavior incorrect after read_data");
```

```
    cover_tx_valid_read: cover property(p_tx_valid_read_data);
```

```
    sequence s_write_address;
        (rx_valid && din[9:8] == 2'b00);
    endsequence
```

```
    sequence s_write_data;
        (rx_valid && din[9:8] == 2'b01);
    endsequence
```

```
    property p_write_address_to_data;
        @(posedge clk) disable iff(!rst_n)
        s_write_address |-> ##[1:$] s_write_data;
    endproperty
```

```
    assert_write_sequence: assert property(p_write_address_to_data)
        else $error("Assertion Failed: Write Address not followed by Write Data");
```

```
    cover_write_sequence: cover property(p_write_address_to_data);
```

```
    sequence s_read_address;
```

```
        (rx_valid && din[9:8] == 2'b10);
    endsequence

    sequence s_read_data;
        (rx_valid && din[9:8] == 2'b11);
    endsequence

    property p_read_address_to_data;
        @(posedge clk) disable iff(!rst_n)
        s_read_address |-> ##[1:$] s_read_data;
    endproperty

    assert_read_sequence: assert property(p_read_address_to_data)
        else $error("Assertion Failed: Read Address not followed by Read Data");

    cover_read_sequence: cover property(p_read_address_to_data);
endmodule
```

RAM Sequence item

```
package RAM_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
class RAM_sequence_item extends uvm_sequence_item;
    `uvm_object_utils(RAM_sequence_item)
    rand bit rst_n;
    rand bit rx_valid;
    rand bit [9:0] din;
    bit [7:0] dout;
    bit tx_valid;
    bit [7:0] dout_ref;
    bit tx_valid_ref;
    constraint rst_c {
```

```
        rst_n dist {1 := 98, 0 := 2};
    }
    constraint rx_valid_c {
        rx_valid dist {1 := 95, 0 := 5};
    }

    constraint write_only_c {
        if (rx_valid && (din[9:8] == 2'b10))
            din[9:8] inside {2'b10, 2'b11};
    }
    constraint read_only_c {
        if (rx_valid && (din[9:8] == 2'b00))
            din[9:8] == 2'b01;
        if (rx_valid && (din[9:8] == 2'b01))
            din[9:8] == 2'b00;
    }
    constraint rw_c {
        if (rx_valid && (din[9:8] == 2'b10))
            din[9:8] inside {2'b10, 2'b11};
        if (rx_valid && (din[9:8] == 2'b11))
            din[9:8] dist {2'b00 := 60, 2'b10 := 40};
        if (rx_valid && (din[9:8] == 2'b00))
            din[9:8] == 2'b01;
        if (rx_valid && (din[9:8] == 2'b01))
            din[9:8] dist {2'b10 := 60, 2'b00 := 40};
    }
    function new(string name = "RAM_sequence_item");
        super.new(name);
    endfunction
    function string convert2string();
        return $sformatf("rst_n=%0b, rx_valid=%0b, din=%0h, dout=%0h, tx_valid=%0b | REF: dout_ref=%0h,
tx_valid_ref=%0b",
                         rst_n, rx_valid, din, dout, tx_valid, dout_ref, tx_valid_ref);
    endfunction
```

```
    function string convert2string_stimulus();
        return $sformatf("rst_n=%0b, rx_valid=%0b, din=%0h", rst_n, rx_valid, din);
    endfunction
endclass
endpackage
```

Driver

```
package RAM_driver_pkg ;
    import uvm_pkg::*;
    import RAM_pkg::*;
    `include "uvm_macros.svh"
class RAM_driver extends uvm_driver #(RAM_sequence_item);
    `uvm_component_utils(RAM_driver)

    virtual RAM_if vif;
    RAM_sequence_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_sequence_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();
            `uvm_info(get_type_name(), seq_item.convert2string_stimulus(), UVM_HIGH)
```

```
        end
    endtask
endclass
endpackage
```

Monitor

```
package RAM_monitor_pkg;
import uvm_pkg::*;
import RAM_pkg::*;
`include "uvm_macros.svh"
class RAM_monitor extends uvm_monitor;
    `uvm_component_utils(RAM_monitor)
    virtual RAM_if vif;
    RAM_sequence_item seq_item;
    uvm_analysis_port #(RAM_sequence_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_sequence_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;
            seq_item.dout_ref = vif.dout_ref;
```

```
            seq_item.tx_valid_ref = vif.tx_valid_ref;
            mon_ap.write(seq_item);
            `uvm_info(get_type_name(), seq_item.convert2string(), UVM_HIGH)
        end
    endtask
endclass
endpackage
```

Scoreboard

```
package RAM_scoreboard_pkg;
import uvm_pkg::*;
import RAM_pkg::*;
`include "uvm_macros.svh"
class RAM_scoreboard extends uvm_scoreboard;
    `uvm_component_utils(RAM_scoreboard)
    uvm_analysis_export #(RAM_sequence_item) sb_export;
    uvm_tlm_analysis_fifo #(RAM_sequence_item) sb_fifo;
    RAM_sequence_item seq_item;
    int correct_count = 0;
    int error_count = 0;
    function new(string name = "RAM_scoreboard", uvm_component parent = null);
        super.new(name, parent);
    endfunction
    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        sb_export = new("sb_export", this);
        sb_fifo = new("sb_fifo", this);
    endfunction
    function void connect_phase(uvm_phase phase);
        super.connect_phase(phase);
        sb_export.connect(sb_fifo.analysis_export);
    endfunction
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
```

```
        forever begin
            sb_fifo.get(seq_item);
            compare(seq_item);
        end
    endtask
    task compare(RAM_sequence_item seq_item);
        if (seq_item.tx_valid == seq_item.tx_valid_ref && seq_item.dout == seq_item.dout_ref) begin
            correct_count++;
            `uvm_info(get_type_name(), $sformatf("correct: DUT(dout=%0h, tx_valid=%0b) and REF(dout=%0h,
tx_valid=%0b)",
                    seq_item.dout, seq_item.tx_valid, seq_item.dout_ref, seq_item.tx_valid_ref),
UVM_HIGH)
        end else begin
            error_count++;
            `uvm_error(get_type_name(), $sformatf("eror: DUT(dout=%0h, tx_valid=%0b) and REF(dout=%0h,
tx_valid=%0b)",
                     seq_item.dout, seq_item.tx_valid, seq_item.dout_ref, seq_item.tx_valid_ref))
        end
    endtask
    function void report_phase(uvm_phase phase);
        super.report_phase(phase);
        `uvm_info(get_type_name(), $sformatf(" correct transactions: %0d", correct_count), UVM_LOW)
        `uvm_info(get_type_name(), $sformatf(" errors: %0d", error_count), UVM_LOW)
        if(error_count == 0)
            `uvm_info(get_type_name(), $sformatf(" No Errors Detected!"), UVM_LOW)
        else
            `uvm_error(get_type_name(), $sformatf(" %0d Errors Detected!", error_count))
    endfunction
endclass
endpackage
```

Sequencer

```
package RAM_sequencer_pkg;
    import uvm_pkg::*;
    import RAM_pkg::*;
```

```
    `include "uvm_macros.svh"
class RAM_sequencer extends uvm_sequencer #(RAM_sequence_item);
    `uvm_component_utils(RAM_sequencer)


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

Agent

```
package RAM_agent_pkg ;
    import uvm_pkg::*;
    import RAM_sequencer_pkg::*;
    import RAM_driver_pkg::*;
    import RAM_pkg::*;
    import RAM_monitor_pkg::*;
    import RAM_config_obj_pkg::*;
    `include "uvm_macros.svh"
class RAM_agent extends uvm_agent;
    `uvm_component_utils(RAM_agent)


    RAM_driver drv;
    RAM_sequencer sqr;
    RAM_monitor mon;
    RAM_config_obj cfg;
    uvm_analysis_port #(RAM_sequence_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_obj)::get(this, "", "RAM_config_obj", cfg))
                `uvm_fatal("build_phase", "unable to get config object");

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

        agent_ap = new("agent_ap", this);
        mon = RAM_monitor::type_id::create("mon", this);
    endfunction

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

        if(cfg.is_active == UVM_ACTIVE) begin
            drv.vif = cfg.vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
        end

        mon.vif = cfg.vif;
        mon.mon_ap.connect(agent_ap);
    endfunction
endclass
endpackage
```

Coverage

```
package RAM_coverage_pkg ;

import uvm_pkg::*;

import RAM_pkg::*;

`include "uvm_macros.svh"

class RAM_coverage extends uvm_component;

    `uvm_component_utils(RAM_coverage)


    uvm_analysis_export #(RAM_sequence_item) cov_export;

    uvm_tlm_analysis_fifo #(RAM_sequence_item) cov_fifo;

    RAM_sequence_item seq_item;
```

```
    covergroup cvr_grp;

        cp_din: coverpoint seq_item.din[9:8] {

            bins write_address = {2'b00};

            bins write_data = {2'b01};

            bins read_address = {2'b10};

            bins read_data = {2'b11};

        }

        cp_rx_valid: coverpoint seq_item.rx_valid {

            bins high = {1};

            bins low = {0};

        }

        cp_tx_valid: coverpoint seq_item.tx_valid {

            bins high = {1};

            bins low = {0};

        }
```

```
        cp_write_sequence: coverpoint seq_item.din[9:8] {

            bins write_add_to_write_data = (2'b00 => 2'b01);

        }

        cp_read_sequence: coverpoint seq_item.din[9:8] {

            bins read_add_to_read_data = (2'b10 => 2'b11);
```

```
        }

        cp_full_sequence: coverpoint seq_item.din[9:8] {
            bins full_wr_rd = (2'b00 => 2'b01 => 2'b10 => 2'b11);
        }

        cross_din_rx_valid: cross cp_din, cp_rx_valid {
            ignore_bins ignore_rx_low = binsof(cp_rx_valid.low);
        }
        cross_read_tx_valid: cross cp_din, cp_tx_valid {
            bins read_with_tx = binsof(cp_din.read_data) && binsof(cp_tx_valid.high);
            ignore_bins ignore_others = binsof(cp_din.write_address) ||  binsof(cp_din.write_data)
||  binsof(cp_din.read_address) ||binsof(cp_tx_valid.low);
        }
    endgroup

    function new(string name = "RAM_coverage", uvm_component parent = null);
        super.new(name, parent);
        cvr_grp = 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);
            cvr_grp.sample();
        end
    endtask
endclass
endpackage
```

Sequence

```
package RAM_sequence_pkg;
    import uvm_pkg::*;
    import RAM_pkg::*;
    `include "uvm_macros.svh"
class reset_sequence extends uvm_sequence #(RAM_sequence_item);
    `uvm_object_utils(reset_sequence)

    RAM_sequence_item seq_item;
```

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

```
    task body();
        seq_item = RAM_sequence_item::type_id::create("seq_item");
        start_item(seq_item);
        seq_item.rst_n = 0;
```

```
        seq_item.rx_valid = 0;
        seq_item.din = 0;
        finish_item(seq_item);

        // Release reset
        seq_item = RAM_sequence_item::type_id::create("seq_item");
        start_item(seq_item);
        seq_item.rst_n = 1;
        seq_item.rx_valid = 0;
        seq_item.din = 0;
        finish_item(seq_item);
    endtask
endclass
```

```
// Write Only Sequence
class write_only_sequence extends uvm_sequence #(RAM_sequence_item);
    `uvm_object_utils(write_only_sequence)

    RAM_sequence_item seq_item;
    bit [1:0] prev_op = 2'b00;

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

    task body();
        repeat(30) begin
            seq_item = RAM_sequence_item::type_id::create("seq_item");
            start_item(seq_item);
            assert(seq_item.randomize());

            seq_item.rx_valid = 1;
```

```
            case(prev_op)
                2'b00: begin
                    seq_item.din[9:8] = $urandom_range(0,1) ? 2'b00 : 2'b01;
                end
                2'b01: begin
                    seq_item.din[9:8] = $urandom_range(0,1) ? 2'b00 : 2'b01;
                end
                default: begin
                    seq_item.din[9:8] = 2'b00;
                end
            endcase

            prev_op = seq_item.din[9:8];
            finish_item(seq_item);
        end
    endtask
endclass
```

```
// Read Only Sequence
class read_only_sequence extends uvm_sequence #(RAM_sequence_item);
    `uvm_object_utils(read_only_sequence)

    RAM_sequence_item seq_item;
    bit [1:0] prev_op = 2'b10;

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

    task body();
        repeat(1000) begin
            seq_item = RAM_sequence_item::type_id::create("seq_item");
```

```
            start_item(seq_item);
            assert(seq_item.randomize());

            seq_item.rx_valid = 1;
            case(prev_op)
                2'b10: begin
                    seq_item.din[9:8] = $urandom_range(0,1) ? 2'b10 : 2'b11;
                end
                2'b11: begin
                    seq_item.din[9:8] = $urandom_range(0,1) ? 2'b10 : 2'b11;
                end
                default: begin
                    seq_item.din[9:8] = 2'b10;
                end
            endcase

            prev_op = seq_item.din[9:8];
            finish_item(seq_item);
        end
    endtask
endclass
```

```
// Write-Read Sequence
class write_read_sequence extends uvm_sequence #(RAM_sequence_item);
    `uvm_object_utils(write_read_sequence)

    RAM_sequence_item seq_item;
    bit [1:0] prev_op = 2'b00;

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

```
    task body();
        repeat(1000) begin
            seq_item = RAM_sequence_item::type_id::create("seq_item");
            start_item(seq_item);
            assert(seq_item.randomize());

            seq_item.rx_valid = 1;
            case(prev_op)
                2'b00: begin
                    seq_item.din[9:8] = $urandom_range(0,1) ? 2'b00 : 2'b01;
                end
                2'b01: begin
                    seq_item.din[9:8] = ($urandom_range(1,100) <= 60) ? 2'b10 : 2'b00;
                end
                2'b10: begin
                    seq_item.din[9:8] = $urandom_range(0,1) ? 2'b10 : 2'b11;
                end
                2'b11: begin
                    seq_item.din[9:8] = ($urandom_range(1,100) <= 60) ? 2'b00 : 2'b10;
                end
            endcase

            prev_op = seq_item.din[9:8];
            finish_item(seq_item);
        end
    endtask
endclass
```

```
// Main Sequence
class main_sequence extends uvm_sequence #(RAM_sequence_item);
    `uvm_object_utils(main_sequence)
```

```
    RAM_sequence_item seq_item;
```

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

```
    task body();
        repeat(1000) begin
            seq_item = RAM_sequence_item::type_id::create("seq_item");
            start_item(seq_item);
            assert(seq_item.randomize());
            seq_item.rx_valid = 1;
            finish_item(seq_item);
        end
    endtask
endclass
endpackage
```

Reset Sequence

This sequence initializes the RAM to a known state. It performs two operations: first, it asserts reset by setting `rst_n = 0` while clearing `rx_valid` and `din` to zero. Then it releases reset by setting `rst_n = 1`. This ensures the RAM module starts from a clean state before running other test sequences.

Write Only Sequence

This sequence performs only write operations to the RAM. It runs 30 iterations where each iteration randomly generates a sequence item and sets `rx_valid = 1` to enable the RAM. The operation code in `din[9:8]` is constrained to be either `2'b00` (set address) or `2'b01` (write data). The state machine tracks the previous operation, so after setting an address (`2'b00`), the next operation is likely to write data (`2'b01`), or it cycles back to setting a new address. This creates a realistic write pattern where addresses and data are paired together.

Read Only Sequence

This sequence performs only read operations on the RAM. It runs 1000 iterations with `rx_valid = 1`. The operation codes are constrained to `2'b10` (set address for reading) or `2'b11` (read data

from current address). Similar to the write sequence, it maintains state tracking so that after setting an address, the next operation is typically a read, creating a natural read pattern.

Write-Read Sequence

This is the most comprehensive sequence, running 1000 iterations that mix both write and read operations. Starting with write mode (`prev_op = 2'b00`), it transitions between write and read operations probabilistically. After a write operation (`2'b01`), there's a 60% chance to switch to reading (`2'b10`), otherwise it returns to address setup. Similarly, after a read operation (`2'b11`), there's a 60% chance to return to write mode (`2'b00`). This creates realistic scenarios where the RAM performs both operations intermixed.

Main Sequence

This is the simplest and most general sequence. It runs 1000 iterations where each sequence item is fully randomized without any constraints on the operation codes. All fields (`din`, `rx_valid`, `rst_n`) are randomized by the sequence item's randomize() method, providing unconstrained stimulus to test the RAM across all possible input combinations.

Enviroment

```
package RAM_env_pkg;

import uvm_pkg::*;

import RAM_agent_pkg::*;

import RAM_scoreboard_pkg::*;

import RAM_coverage_pkg::*;

`include "uvm_macros.svh"

class RAM_env extends uvm_env;

    `uvm_component_utils(RAM_env)

    RAM_agent agt;

    RAM_scoreboard sb;

    RAM_coverage cov;

    function new(string name = "RAM_env", uvm_component parent = null);

        super.new(name, parent);

    endfunction


    function void build_phase(uvm_phase phase);

        super.build_phase(phase);

        agt = RAM_agent::type_id::create("agt", this);
```

```
        sb = RAM_scoreboard::type_id::create("sb", this);
        cov = RAM_coverage::type_id::create("cov", this);
    endfunction
```

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

Test

```
package RAM_test_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import RAM_pkg::*;
    import RAM_sequence_pkg::*;
    import RAM_env_pkg::*;
    import RAM_config_obj_pkg::*;
class RAM_test extends uvm_test;
    `uvm_component_utils(RAM_test)
    RAM_env env;
    RAM_config_obj cfg;
    virtual RAM_if vif;
    reset_sequence rst_seq;
    write_only_sequence wr_seq;
    read_only_sequence rd_seq;
    write_read_sequence wr_rd_seq;
    main_sequence main_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);

        cfg = RAM_config_obj::type_id::create("cfg", this);

        env = RAM_env::type_id::create("env", this);

        rst_seq = reset_sequence::type_id::create("rst_seq", this);

        wr_seq = write_only_sequence::type_id::create("wr_seq", this);

        rd_seq = read_only_sequence::type_id::create("rd_seq", this);

        wr_rd_seq = write_read_sequence::type_id::create("wr_rd_seq", this);

        main_seq = main_sequence::type_id::create("main_seq", this);


        if (!uvm_config_db #(virtual RAM_if)::get(this, "", "RAM_IF", cfg.vif))

            `uvm_fatal("build_phase", "Test - unable to get the wrapper virtual interface");


        cfg.is_active = UVM_ACTIVE;


        uvm_config_db#(RAM_config_obj)::set(this, "*", "RAM_config_obj", cfg);

    endfunction


    task run_phase(uvm_phase phase);

        super.run_phase(phase);

        phase.raise_objection(this);


        `uvm_info(get_type_name(), "Starting Reset Sequence", UVM_MEDIUM)


        rst_seq.start(env.agt.sqr);


        #10;


        `uvm_info(get_type_name(), "Starting Write Only Sequence", UVM_MEDIUM)
```

```
        wr_seq.start(env.agt.sqr);

        #10;

        `uvm_info(get_type_name(), "Starting Read Only Sequence", UVM_MEDIUM)

        rd_seq.start(env.agt.sqr);

        #10;

        `uvm_info(get_type_name(), "Starting Write-Read Sequence", UVM_MEDIUM)

        wr_rd_seq.start(env.agt.sqr);

        #10;


        `uvm_info(get_type_name(), "Starting Main Sequence", UVM_MEDIUM)

        main_seq.start(env.agt.sqr);

        #100;

        phase.drop_objection(this);
    endtask
endclass

endpackage
```

Config object

```
package RAM_config_obj_pkg;

import uvm_pkg::*;
```

```
`include "uvm_macros.svh"

class RAM_config_obj extends uvm_object;

  `uvm_object_utils(RAM_config_obj)

  virtual RAM_if vif;

  uvm_active_passive_enum is_active ;


  function new(string name="RAM_config_obj");

    super.new(name);

  endfunction

endclass

endpackage
```

Top

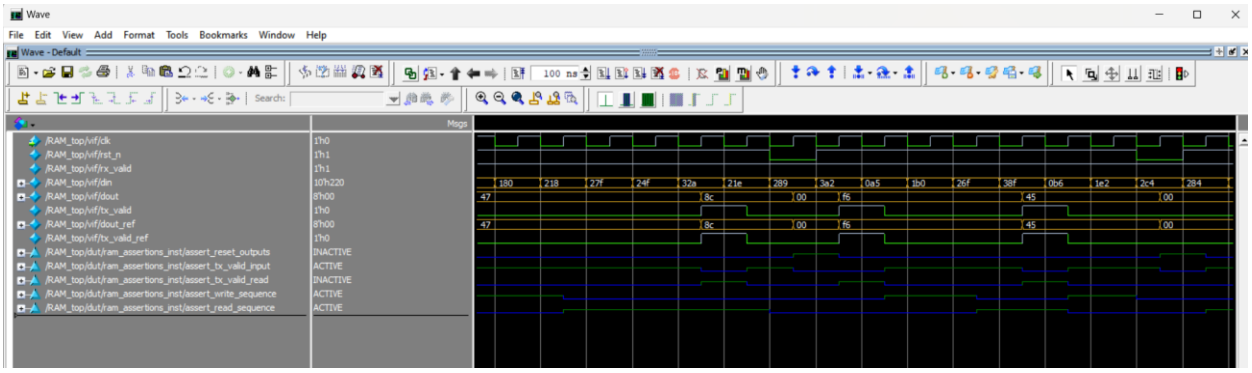
```
import uvm_pkg::*;

import RAM_pkg::*;

import RAM_test_pkg::*;

`include "uvm_macros.svh"

module RAM_top;

    bit clk;

    initial begin

        clk = 0;

        forever #5 clk = ~clk;

    end

    RAM_if vif(clk);

    RAM dut
(.din(vif.din),.clk(vif.clk),.rst_n(vif.rst_n),.rx_valid(vif.rx_valid),.dout(vif.dout),.tx_valid(vif.tx_va
lid));

    RAM_golden golden
(.clk(vif.clk), .rst_n(vif.rst_n),.din(vif.din),.rx_valid(vif.rx_valid),.dout(vif.dout_ref),.tx_valid(vif.
tx_valid_ref) );

    bind RAM RAM_assertions ram_assertions_inst
(.clk(clk),.rst_n(rst_n),.rx_valid(rx_valid),.din(din),.dout(dout), .tx_valid(tx_valid));


    initial begin

        uvm_config_db#(virtual RAM_if)::set(null, "uvm_test_top", "RAM_IF", vif);
```

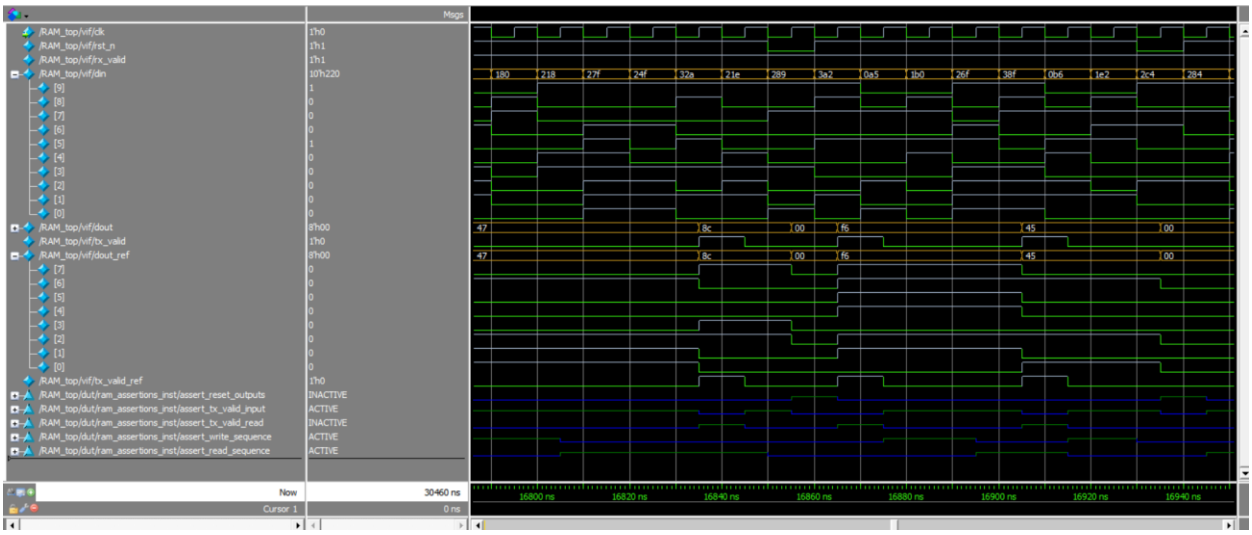
```
        run_test("RAM_test");

    end


endmodule
```

UVM report

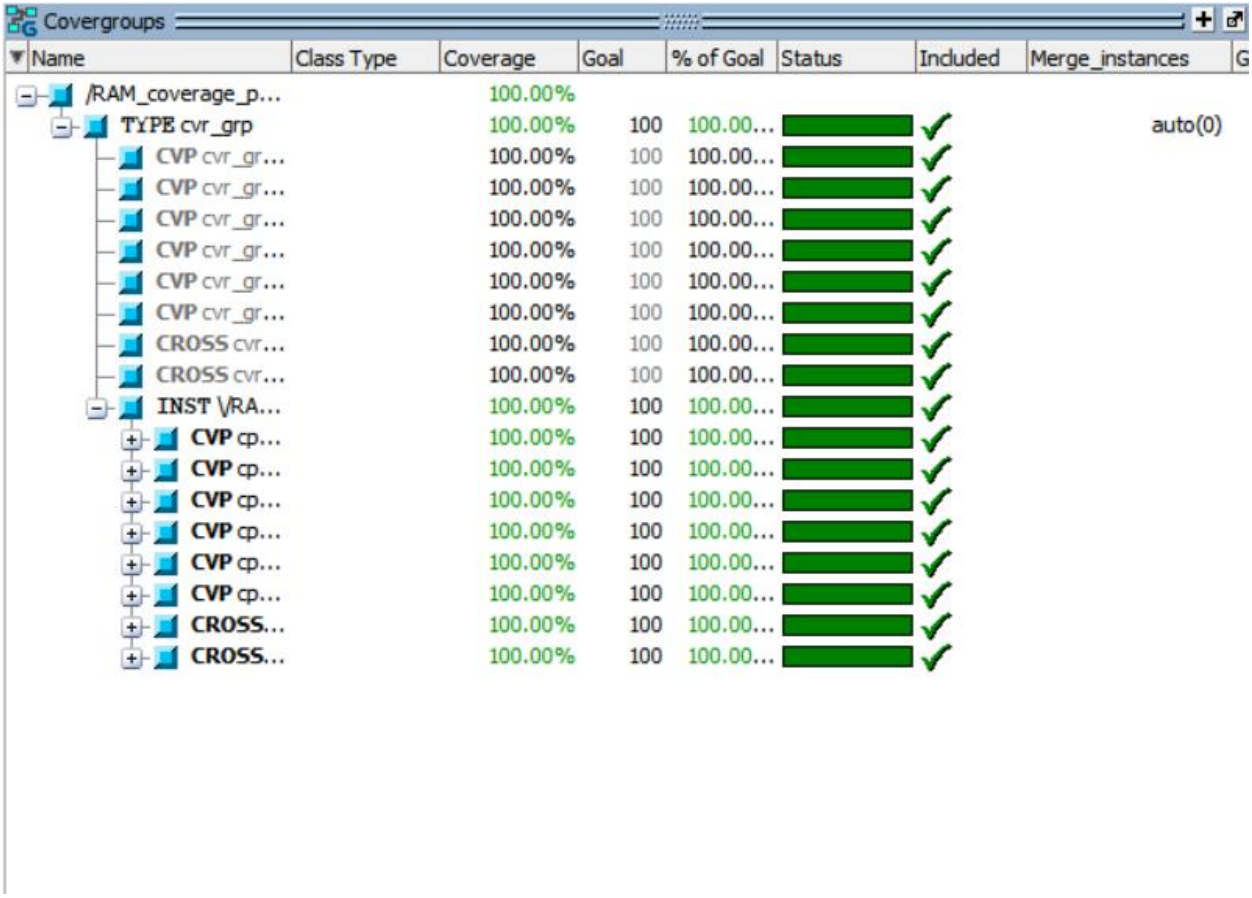
```
# UVM_INFO RAM_test.sv(51) @ 30: uvm_test_top [RAM_test] Starting Write Only Sequence
# UVM_INFO RAM_test.sv(57) @ 340: uvm_test_top [RAM_test] Starting Read Only Sequence
# UVM_INFO RAM_test.sv(63) @ 10350: uvm_test_top [RAM_test] Starting Write-Read Sequence
# UVM_INFO RAM_test.sv(70) @ 20360: uvm_test_top [RAM_test] Starting Main Sequence
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 30460: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO RAM_scoreboard.sv(44) @ 30460: uvm_test_top.env.sb [RAM_scoreboard]  correct transactions: 3046
# UVM_INFO RAM_scoreboard.sv(45) @ 30460: uvm_test_top.env.sb [RAM_scoreboard]  errors: 0
# UVM_INFO RAM_scoreboard.sv(47) @ 30460: uvm_test_top.env.sb [RAM_scoreboard]  No Errors Detected!
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   12
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RAM_scoreboard]     3
# [RAM_test]     5
# [RNTST]     1
# [TEST_DONE]     1
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
```

Wave Snippet

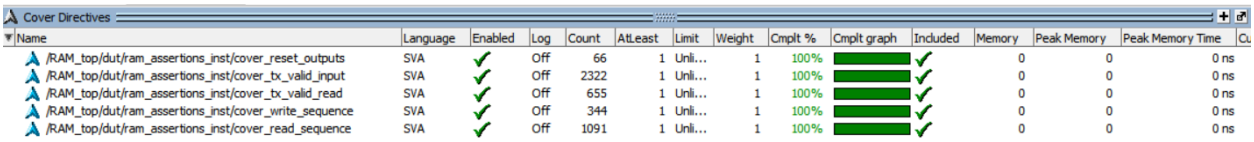




Covergroup

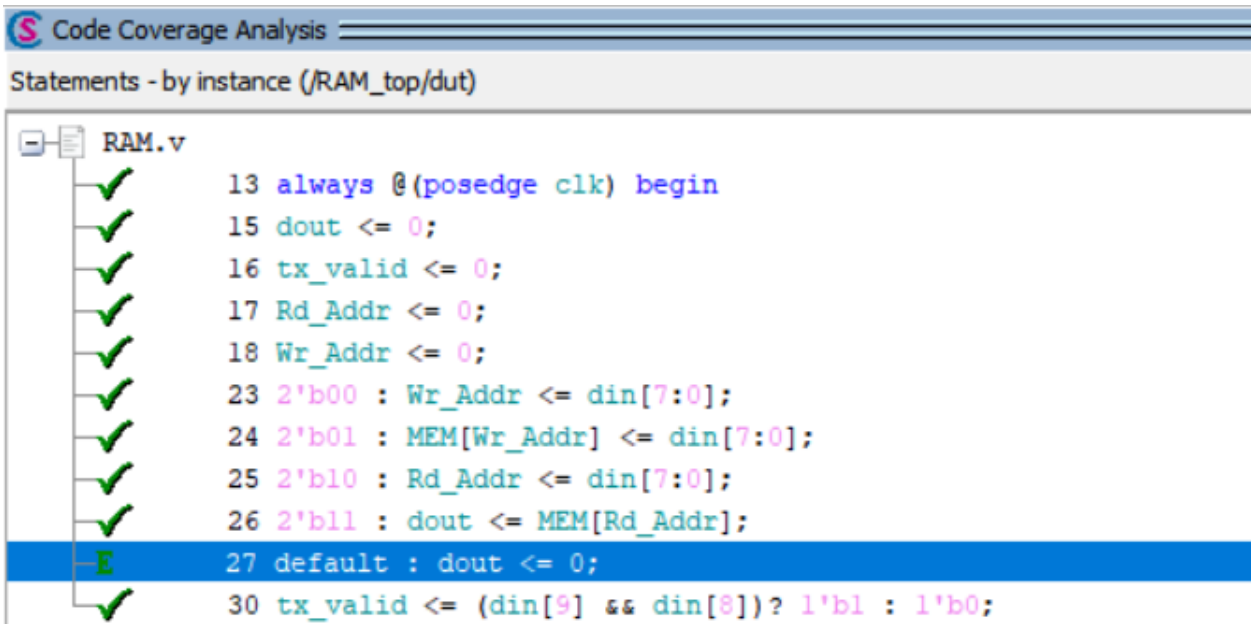


Assertion coverage

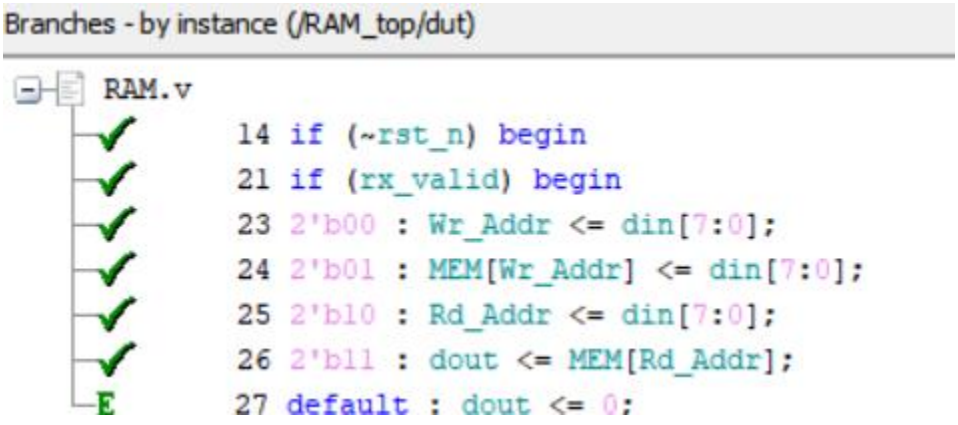


Code coverage

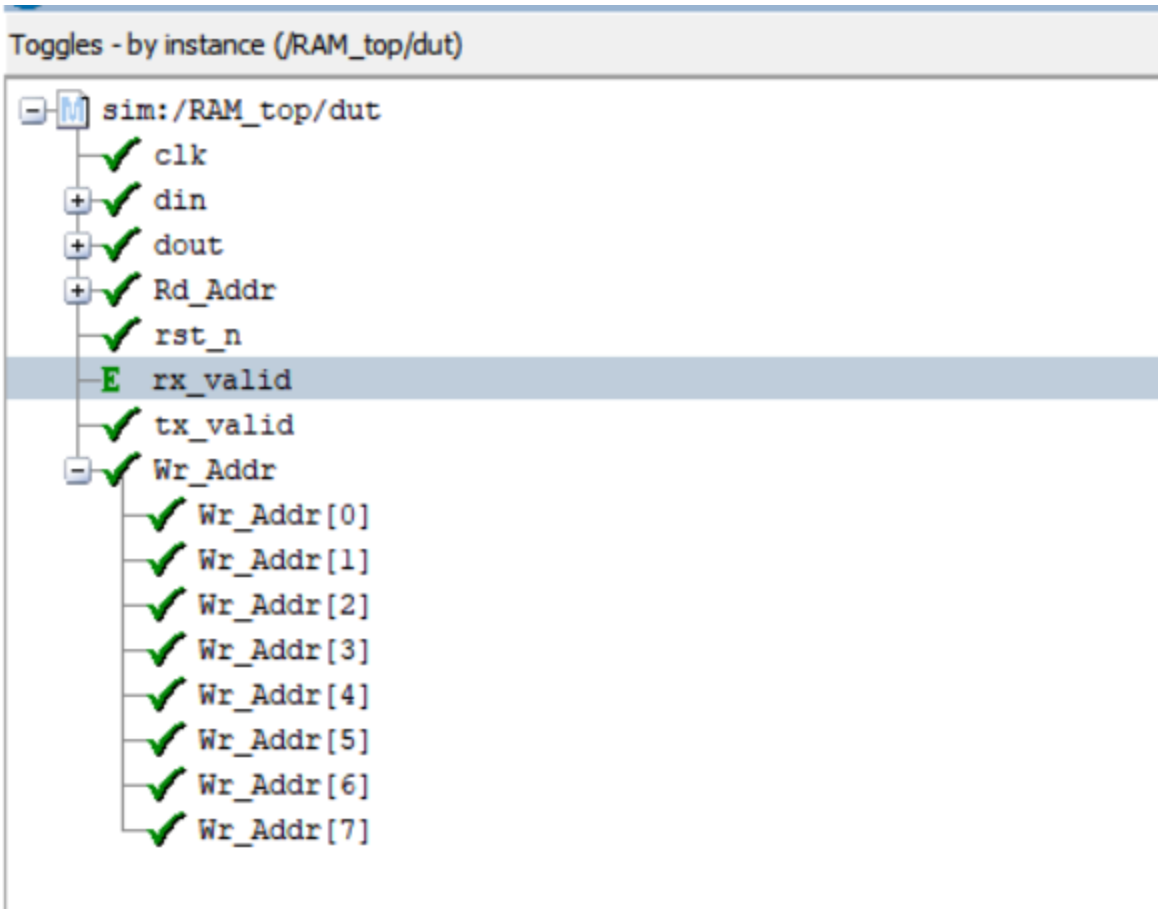
Statement with excluded defualt statement



Branch



Toggle



Coverage Report

Coverage Report by instance with details

==

=== Instance: /RAM_top/dut/ram_assertions_inst

=== Design Unit: work.RAM_assertions

==

Assertion Coverage:

Assertions	5	5	0	100.00%

Name	File(Line)	Failure Count	Pass Count
/RAM_top/dut/ram_assertions_inst/assert_reset_outputs	RAM_assertions.sv(13)	0	1
/RAM_top/dut/ram_assertions_inst/assert_tx_valid_input	RAM_assertions.sv(22)	0	1
/RAM_top/dut/ram_assertions_inst/assert_tx_valid_read	RAM_assertions.sv(31)	0	1
/RAM_top/dut/ram_assertions_inst/assert_write_sequence	RAM_assertions.sv(49)	0	1
/RAM_top/dut/ram_assertions_inst/assert_read_sequence	RAM_assertions.sv(67)	0	1

Directive Coverage:

Directives	5	5	0	100.00%

DIRECTIVE COVERAGE:

Name	Design Unit	Design UnitType	Lang	File(Line)	Hits	Status
/RAM_top/dut/ram_assertions_inst/cover_reset_outputs						

RAM_assertions Verilog SVA RAM_assertions.sv(16)

66 Covered

/RAM_top/dut/ram_assertions_inst/cover_tx_valid_input

RAM_assertions Verilog SVA RAM_assertions.sv(25)

2322 Covered

/RAM_top/dut/ram_assertions_inst/cover_tx_valid_read

RAM_assertions Verilog SVA RAM_assertions.sv(34)

655 Covered

/RAM_top/dut/ram_assertions_inst/cover_write_sequence

RAM_assertions Verilog SVA RAM_assertions.sv(52)

344 Covered

/RAM_top/dut/ram_assertions_inst/cover_read_sequence

RAM_assertions Verilog SVA RAM_assertions.sv(70)

1091 Covered

==

=== Instance: /RAM_coverage_pkg

=== Design Unit: work.RAM_coverage_pkg

==

Covergroup Coverage:

Covergroups	1	na	na	100.00%
Coverpoints/Crosses	8	na	na	na
Covergroup Bins	16	16	0	100.00%

Covergroup	Metric	Goal	Bins	Status

```
----------------------------------------------------------------------------------------------------
TYPE /RAM_coverage_pkg/RAM_coverage/cvr_grp        100.00%       100        -    Covered
  covered/total bins:                    16        16        -
  missing/total bins:                     0        16        -
  % Hit:                             100.00%       100        -
  Coverpoint cp_din                  100.00%       100        -    Covered
    covered/total bins:                   4         4        -
    missing/total bins:                   0         4        -
    % Hit:                           100.00%       100        -
  Coverpoint cp_rx_valid             100.00%       100        -    Covered
    covered/total bins:                   2         2        -
    missing/total bins:                   0         2        -
    % Hit:                           100.00%       100        -
  Coverpoint cp_tx_valid             100.00%       100        -    Covered
    covered/total bins:                   2         2        -
    missing/total bins:                   0         2        -
    % Hit:                           100.00%       100        -
  Coverpoint cp_write_sequence       100.00%       100        -    Covered
    covered/total bins:                   1         1        -
    missing/total bins:                   0         1        -
    % Hit:                           100.00%       100        -
  Coverpoint cp_read_sequence        100.00%       100        -    Covered
    covered/total bins:                   1         1        -
    missing/total bins:                   0         1        -
    % Hit:                           100.00%       100        -
  Coverpoint cp_full_sequence        100.00%       100        -    Covered
```

covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Cross cross_din_rx_valid	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Cross cross_read_tx_valid	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Covergroup instance \/RAM_coverage_pkg::RAM_coverage::cvr_grp				
	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Coverpoint cp_din	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
bin write_address	380	1	-	Covered
bin write_data	190	1	-	Covered
bin read_address	1810	1	-	Covered
bin read_data	666	1	-	Covered
Coverpoint cp_rx_valid	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	

% Hit:	100.00%	100	-	
bin high	3043	1	-	Covered
bin low	3	1	-	Covered
Coverpoint cp_tx_valid	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin high	655	1	-	Covered
bin low	2391	1	-	Covered
Coverpoint cp_write_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin write_add_to_write_data	173	1	-	Covered
Coverpoint cp_read_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin read_add_to_read_data	424	1	-	Covered
Coverpoint cp_full_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin full_wr_rd	43	1	-	Covered
Cross cross_din_rx_valid	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 <read_data,high>                    666         1         -    Covered
      bin <write_data,high>                   190         1         -    Covered
      bin <read_address,high>                 1810        1         -    Covered
      bin <write_address,high>                377         1         -    Covered
    Illegal and Ignore Bins:
      ignore_bin ignore_rx_low                3                     -    Occurred
  Cross cross_read_tx_valid                   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_with_tx                        655         1         -    Covered
    Illegal and Ignore Bins:
      ignore_bin ignore_others                2391                  -    Occurred

================================================================================
============
=== Instance: /RAM_sequence_pkg
=== Design Unit: work.RAM_sequence_pkg
================================================================================
============

Assertion Coverage:
  Assertions                      4         4         0   100.00%
--------------------------------------------------------------------
Name                File(Line)                Failure      Pass
```

	Count	Count
/RAM_sequence_pkg/write_only_sequence/body/#ublk#33852551#44/immed__47		
RAM_sequence.sv(47)	0	1
/RAM_sequence_pkg/read_only_sequence/body/#ublk#33852551#80/immed__83		
RAM_sequence.sv(83)	0	1
/RAM_sequence_pkg/write_read_sequence/body/#ublk#33852551#116/immed__119		
RAM_sequence.sv(119)	0	1
/RAM_sequence_pkg/main_sequence/body/#ublk#33852551#154/immed__157		
RAM_sequence.sv(157)	0	1

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /RAM_coverage_pkg/RAM_coverage/cvr_grp	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Coverpoint cp_din	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Coverpoint cp_rx_valid	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	

% Hit:	100.00%	100	-	
Coverpoint cp_tx_valid	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
Coverpoint cp_write_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Coverpoint cp_read_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Coverpoint cp_full_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Cross cross_din_rx_valid	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Cross cross_read_tx_valid	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	

Covergroup instance \/RAM_coverage_pkg::RAM_coverage::cvr_grp

100.00%	100	-	Covered

covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Coverpoint cp_din	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
bin write_address	380	1	-	Covered
bin write_data	190	1	-	Covered
bin read_address	1810	1	-	Covered
bin read_data	666	1	-	Covered
Coverpoint cp_rx_valid	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin high	3043	1	-	Covered
bin low	3	1	-	Covered
Coverpoint cp_tx_valid	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin high	655	1	-	Covered
bin low	2391	1	-	Covered
Coverpoint cp_write_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	

bin write_add_to_write_data	173	1	-	Covered
Coverpoint cp_read_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin read_add_to_read_data	424	1	-	Covered
Coverpoint cp_full_sequence	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin full_wr_rd	43	1	-	Covered
Cross cross_din_rx_valid	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 <read_data,high>	666	1	-	Covered
bin <write_data,high>	190	1	-	Covered
bin <read_address,high>	1810	1	-	Covered
bin <write_address,high>	377	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ignore_rx_low	3		-	Occurred
Cross cross_read_tx_valid	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_with_tx 655 1 - Covered

Illegal and Ignore Bins:

ignore_bin ignore_others 2391 - Occurred

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:

Name Design Design Lang File(Line) Hits Status

Unit UnitType

/RAM_top/dut/ram_assertions_inst/cover_reset_outputs

RAM_assertions Verilog SVA RAM_assertions.sv(16)

66 Covered

/RAM_top/dut/ram_assertions_inst/cover_tx_valid_input

RAM_assertions Verilog SVA RAM_assertions.sv(25)

2322 Covered

/RAM_top/dut/ram_assertions_inst/cover_tx_valid_read

RAM_assertions Verilog SVA RAM_assertions.sv(34)

655 Covered

/RAM_top/dut/ram_assertions_inst/cover_write_sequence

RAM_assertions Verilog SVA RAM_assertions.sv(52)

344 Covered

/RAM_top/dut/ram_assertions_inst/cover_read_sequence

RAM_assertions Verilog SVA RAM_assertions.sv(70)

1091 Covered

TOTAL DIRECTIVE COVERAGE: 100.00% COVERS: 5

ASSERTION RESULTS:

Name File(Line) Failure Pass

Count Count

/RAM_top/dut/ram_assertions_inst/assert_reset_outputs

RAM_assertions.sv(13) 0 1

/RAM_top/dut/ram_assertions_inst/assert_tx_valid_input

RAM_assertions.sv(22) 0 1

/RAM_top/dut/ram_assertions_inst/assert_tx_valid_read

RAM_assertions.sv(31) 0 1

/RAM_top/dut/ram_assertions_inst/assert_write_sequence

RAM_assertions.sv(49) 0 1

/RAM_top/dut/ram_assertions_inst/assert_read_sequence

RAM_assertions.sv(67) 0 1

/RAM_sequence_pkg/write_only_sequence/body/#ublk#33852551#44/immed__47

RAM_sequence.sv(47) 0 1

/RAM_sequence_pkg/read_only_sequence/body/#ublk#33852551#80/immed__83

RAM_sequence.sv(83) 0 1

/RAM_sequence_pkg/write_read_sequence/body/#ublk#33852551#116/immed__119

RAM_sequence.sv(119) 0 1

/RAM_sequence_pkg/main_sequence/body/#ublk#33852551#154/immed__157

RAM_sequence.sv(157) 0 1

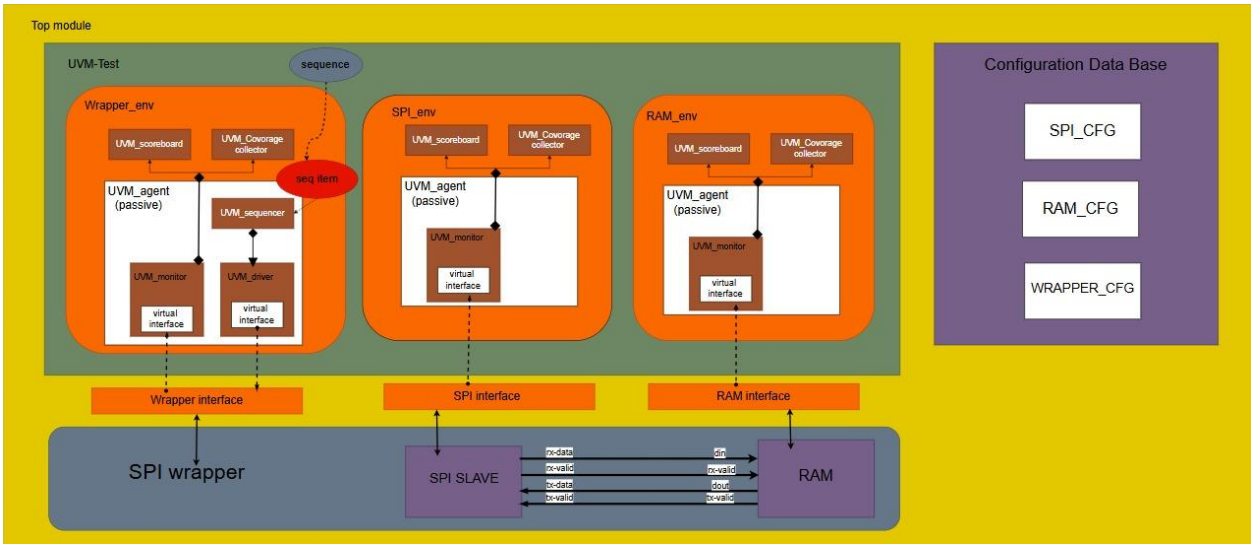
Total Coverage By Instance (filtered view): 100.00%

Part 3 (Wrapper)

Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
WRAPPER_1	Reset asserted sets MISO, rx_valid, rx_data_din to 0.	Directed reset sequence (rst_n = 0) at test start.	-	Assertion (reset_check) and checker with golden model comparison.
WRAPPER_2	Reset sets state to IDLE.	Randomized rst_n (95% high, 5% low) across sequences.	-	Assertion and scoreboard check vs. golden model.
WRAPPER_3	IDLE with SS_n = 0 sets cs high (13 cycles, 23 for read data).	Randomized with period constraints in sequences.	SS_n coverpoint.	Assertion and scoreboard timing check.
WRAPPER_4	Non-CMD/CHK with SS_n = 1 sets IDLE next cycle.	Randomized SS_n and MOSI transitions.	SS_n, cp_MOSI coverpoints.	Assertion and RAM_assertions check.
WRAPPER_5	MOSI 00D/001/110/111 for write addr/data, read addr/data.	Directed and randomized in operation sequences.	cp_MOSI bins for commands.	Checker and scoreboard vs. golden model.
WRAPPER_6	Valid MISO after 10 cycles: 010 read addr, 011 read data.	Randomized in ro_seq and wr_seq with constraints.	MOSI and SS_n cross coverage.	Assertion (miso_stable) and checker.
WRAPPER_7	Valid MISO after 10 cycles with valid MOSI sequence.	Randomized in wo_seq with write constraints.	MOSI and SS_n cross coverage.	Assertion and RAM_assertions check.
WRAPPER_8	Read data sets tx_valid high.	Randomized in ro_seq and wr_seq.	Data transition sequences.	Assertion and scoreboard check.
WRAPPER_9	10 MOSI bits output via rx_data.	Randomized across sequences with serialized MOSI.	rx_data 2 MSBs transitions.	Checker vs. golden rx_data_golden.

UVM Structure



The interface has all the signals in the DUT and the golden model, the top module puts the interface in the database, the test takes the interface from the database and puts it in the configuration object and then puts the configuration object in the database again, after that the test runs the sequences using the sequencer.

The environment connects the scoreboard and coverage classes to the agent, and the driver with the sequencer and the monitor to the agent port and connect the interfaces of the driver and the monitor with the one in the configuration object.

The monitor assigns all the variables in the sequence item to the values in the interface and connect the sequence item to the port in the agent so the coverage and scoreboard classes be able to use it.

The coverage class contains all the coverage points and cross coverages needed and the scoreboard compares the output of the DUT and the output of the golden model using the variables in the sequence item.

The test creates 3 environments and 3 cfgs for the RAM and WRAPPER and SPI then assign the RAM and SPI environments as passive and the wrapper env as active .

Corrected Design Code

wrapper.sv > ...

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

input  MOSI, SS_n, clk, rst_n;
output MISO;

wire [9:0] rx_data_din;
wire       rx_valid;
wire       tx_valid;
wire [7:0] tx_data_dout;

RAM   RAM_instance   (rx_data_din,clk,rst_n,rx_valid,tx_data_dout,tx_valid);
SLAVE SLAVE_instance (MOSI,MISO,SS_n,clk,rst_n,rx_data_din,rx_valid,tx_data_dout,tx_valid);

`ifdef SIM

  property reset_check;
    @(posedge clk) rst_n == 0 |=> (MISO == 0 && rx_valid == 0 && rx_data_din == 0);
  endproperty

  property miso_stable_when_not_read;
    @(posedge clk) disable iff (!rst_n)
      ((!tx_valid) && $fell(SS_n)) |=> (SS_n == 0 && $stable(MISO)) [*11];
  endproperty

  a_reset_check: assert property (reset_check);
  c_reset_check: cover property (reset_check);
  a_miso_stable_when_not_read: assert property (miso_stable_when_not_read);
  c_miso_stable_when_not_read: cover property (miso_stable_when_not_read);

`endif

endmodule
```

Golden Model Code

wrapper_golden.v > ...

```v
module SPI_wrapper_golden (clk, rst_n, SS_n, MOSI, MISO_golden);
    input  clk, rst_n, MOSI, SS_n;
    output MISO_golden;

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

    SPI_slave_golden inst1 (clk, rst_n, SS_n, MOSI, tx_valid, tx_data, rx_valid, rx_data, MISO_golden);
    RAM_golden inst2 (clk, rst_n, rx_data, rx_valid, tx_data, tx_valid);
endmodule
```

Interface Code

Wrapper_if.sv > ...

```sv
interface wrapper_if(clk);

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

endinterface : wrapper_if
```

Shared Package Code

shared_pkg.sv > ...

```sv
package shared_pkg;
    int period;
    int count;
    bit have_address;
    logic [10:0] curr_op;

    bit was_write_address, was_write_data, was_read_address, was_read_data;
endpackage
```

Sequence Item Code

wrapper_seq_item.sv > ...

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

    class wrapper_seq_item extends uvm_sequence_item;
        `uvm_object_utils(wrapper_seq_item)

        rand logic rst_n;
        rand logic SS_n;
        logic MOSI;
        logic MISO;
        logic MISO_golden;
        rand logic [10:0] MOSI_arr;

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

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

        function string convert2string_stimulus();
            return $sformatf("rst_n: %b, SS_n: %b, MOSI: %b, MISO: %b, MISO_golden: %b",
                             rst_n, SS_n, MOSI, MISO, MISO_golden);
        endfunction

        constraint reset_c {
            rst_n dist {1 :/ 95, 0 :/ 5};
        }

        constraint random_arr_c {
            MOSI_arr[10:8] inside {3'b000, 3'b001, 3'b110, 3'b111};
            if (!have_address) {
                MOSI_arr[10:8] != 3'b111;
            }
        }

        constraint c_next_operation {
            if (was_write_address) {
                soft MOSI_arr[9] == 1'b0;
            } else if (was_read_address) {
                soft MOSI_arr[9:8] == 2'b11;
            }
        }

        constraint c_after_read_data {
            if (was_read_data) {
                soft MOSI_arr[9:8] == 2'b10;
            }
        }

        constraint c_after_read_address {
            if (!was_read_address) {
                MOSI_arr[9:8] != 2'b11;
            }
        }

        constraint c_after_write_and_read_data {
            if (was_write_data) {
                soft MOSI_arr[9:8] dist {2'b10 :/ 60, 2'b00 :/ 40};
            } else if (was_read_data) {
                soft MOSI_arr[9:8] dist {2'b00 :/ 60, 2'b10 :/ 40};
            }
        }

```

```
69          function void post_randomize();
70              if (count == 0) begin
71                  curr_op = MOSI_arr;
72              end
73
74              if (curr_op[10:8] == 3'b111) begin
75                  period = 23;
76              end else begin
77                  period = 13;
78              end
79
80              if (count == period) begin
81                  SS_n = 1;
82              end else begin
83                  SS_n = 0;
84              end
85
86
87              if (curr_op[10:8] == 3'b110) begin
88                  have_address = 1;
89              end
90
91              if (curr_op[10:8] == 3'b111 || !rst_n) begin
92                  have_address = 0;
93              end
94
95              if (count > 0 && count < 12) begin
96                  MOSI = curr_op[11 - count];
97              end else begin
98                  MOSI = 0;
99              end
100
101             if (curr_op[9:8] == 2'b00) begin
102                 was_write_address = 1;
103             end else begin
104                 was_write_address = 0;
```

```
104                 was_write_address = 0;
105             end
106
107             if (curr_op[9:8] == 2'b01) begin
108                 was_write_data = 1;
109             end else begin
110                 was_write_data = 0;
111             end
112
113             if (curr_op[9:8] == 2'b10) begin
114                 was_read_address = 1;
115             end else begin
116                 was_read_address = 0;
117             end
118
119             if (curr_op[9:8] == 2'b11) begin
120                 was_read_data = 1;
121             end else begin
122                 was_read_data = 0;
123             end
124
125             if (!rst_n) begin
126                 count = 0;
127             end else begin
128                 if (count == period) begin
129                     count = 0;
130                 end else begin
131                     count++;
132                 end
133             end
134         endfunction
135     endclass
136 endpackage
```

Reset Sequence Code

wrapper_reset_seq.sv > ...

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

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

        wrapper_seq_item seq_item;

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

        task body();
            seq_item = wrapper_seq_item::type_id::create("seq_item");

            start_item(seq_item);

            seq_item.rst_n = 0;

            finish_item(seq_item);
        endtask
    endclass
endpackage
```

Write only Sequence Code

wrapper_wo_seq.sv > ...

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

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

        wrapper_seq_item seq_item;

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

        task body;
            repeat(1000) begin
                seq_item = wrapper_seq_item::type_id::create("seq_item");

                start_item(seq_item);

                assert(seq_item.randomize() with {
                    MOSI_arr[9] == 1'b0;
                });

                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

Read only Sequence Code

wrapper_ro_seq.sv > ...

```
1  package wrapper_ro_seq_pkg;
2      import wrapper_seq_item_pkg::*;
3      import uvm_pkg::*;
4      `include "uvm_macros.svh"
5
6      class wrapper_ro_seq extends uvm_sequence #(wrapper_seq_item);
7          `uvm_object_utils(wrapper_ro_seq)
8
9          wrapper_seq_item seq_item;
10
11         function new(string name = "wrapper_ro_seq");
12             super.new(name);
13         endfunction
14
15         task body;
16             repeat(1000) begin
17                 seq_item = wrapper_seq_item::type_id::create("seq_item");
18
19                 start_item(seq_item);
20
21                 assert(seq_item.randomize() with {
22                     MOSI_arr[9] == 1'b1;
23                 });
24
25                 finish_item(seq_item);
26             end
27         endtask
28     endclass
29 endpackage
```

Write-Read Sequence Code

wrapper_wr_seq.sv > ...

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

Configuration object Code

wrapper_cfg.sv > ...

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

    class wrapper_config extends uvm_object;
        `uvm_object_utils(wrapper_config);

        virtual wrapper_if wrapper_vif;
        uvm_active_passive_enum is_active;

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

Sequencer Code

wrapper_sqr.sv > {} wrapper_sqr_pkg > wrapper_sqr

```
package wrapper_sqr_pkg;

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

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

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

Driver Code

wrapper_drv.sv > ...

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

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

        virtual wrapper_if wrapper_vif;
        wrapper_seq_item seq_item;

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

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

            forever begin
                seq_item = wrapper_seq_item::type_id::create("seq_item");

                seq_item_port.get_next_item(seq_item);

                wrapper_vif.rst_n = seq_item.rst_n;
                wrapper_vif.SS_n = seq_item.SS_n;
                wrapper_vif.MOSI = seq_item.MOSI;

                @(negedge wrapper_vif.clk);

                seq_item_port.item_done();

                `uvm_info("run_phase", seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

Monitor Code

wrapper_monitor.sv > {} wrapper_monitor_pkg > wrapper_monitor

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

    class wrapper_monitor extends uvm_monitor;
        `uvm_component_utils(wrapper_monitor);

        virtual wrapper_if wrapper_vif;
        wrapper_seq_item seq_item;
        uvm_analysis_port #(wrapper_seq_item) monitor_ap;

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

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

            monitor_ap = new("monitor_ap", this);
        endfunction
```

```
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);

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

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

            forever begin
                seq_item = wrapper_seq_item::type_id::create("seq_item");

                @(negedge wrapper_vif.clk);

                seq_item.rst_n = wrapper_vif.rst_n;
                seq_item.SS_n = wrapper_vif.SS_n;
                seq_item.MOSI = wrapper_vif.MOSI;
                seq_item.MISO = wrapper_vif.MISO;
                seq_item.MISO_golden = wrapper_vif.MISO_golden;

                monitor_ap.write(seq_item);

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

Agent Code

wrapper_agent.sv > {} wrapper_agent_pkg > wrapper_agent

```
package wrapper_agent_pkg;
    import wrapper_sqr_pkg::*;
    import wrapper_seq_item_pkg::*;
    import wrapper_driver_pkg::*;
    import wrapper_monitor_pkg::*;
    import wrapper_config_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class wrapper_agent extends uvm_agent;
        `uvm_component_utils(wrapper_agent);

        wrapper_sqr sequencer;
        wrapper_driver driver;
        wrapper_monitor monitor;
        wrapper_config cfg;
        uvm_analysis_port #(wrapper_seq_item) agent_ap;

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

```
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);

            if (!uvm_config_db #(wrapper_config)::get(this, "", "wrapper_CFG", cfg))
                `uvm_fatal("build_phase", "unable to get config object");

            if (cfg.is_active == UVM_ACTIVE) begin
                sequencer = wrapper_sqr::type_id::create("sequencer", this);
                driver = wrapper_driver::type_id::create("driver", this);
            end

            monitor = wrapper_monitor::type_id::create("monitor", this);
            agent_ap = new("agent_ap", this);
        endfunction

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

            if (cfg.is_active == UVM_ACTIVE) begin
                driver.wrapper_vif = cfg.wrapper_vif;
                driver.seq_item_port.connect(sequencer.seq_item_export);
            end

            monitor.wrapper_vif = cfg.wrapper_vif;
            monitor.monitor_ap.connect(agent_ap);
        endfunction
    endclass
endpackage
```

Scoreboard Code

wrapper_sb.sv > ...

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

    class wrapper_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(wrapper_scoreboard);

        uvm_analysis_export #(wrapper_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(wrapper_seq_item) sb_fifo;
        wrapper_seq_item seq_item;

        int error_count, correct_count;

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

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

            sb_export = new("sb_export", this);
            sb_fifo = new("sb_fifo", this);
        endfunction

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

            sb_export.connect(sb_fifo.analysis_export);
        endfunction

```

```
    task run_phase(uvm_phase phase);
        super.run_phase(phase);

        forever begin
            sb_fifo.get(seq_item);

            if (seq_item.MISO != seq_item.MISO_golden) begin
                error_count++;

                `uvm_error("run_phase", $sformatf("Wrong Output: %s",
                           seq_item.convert2string()));
            end else begin
                correct_count++;

                `uvm_info("run_phase", $sformatf("Correct Output: %s",
                           seq_item.convert2string()), UVM_HIGH);
            end
        end
    endtask

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

        `uvm_info("report_phase", $sformatf("total correct: %d, total error: %d", correct_count, error_count), UVM_MEDIUM)
    endfunction
endclass
endpackage
```

Coverage collector Code

wrapper_cov.sv > ...

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

    class wrapper_cov extends uvm_component;
        `uvm_component_utils(wrapper_cov);

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

        covergroup cg;
           S_n: coverpoint seq_item.SS_n ;

            cp_MOSI: coverpoint seq_item.MOSI iff (seq_item.rst_n) {
                bins write_address = (0 => 0 => 0);
                bins write_data = (0 => 0 => 1);
                bins read_address = (1 => 1 => 0);
                bins read_data = (1 => 1 => 1);
            }
        endgroup

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

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

            cov_export = new("cov_export", this);
            cov_fifo = new("cov_fifo", this);
        endfunction

```

```
        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);

            cov_export.connect(cov_fifo.analysis_export);
        endfunction

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

            forever begin
                cov_fifo.get(seq_item);
                cg.sample();
            end
        endtask
    endclass
endpackage
```

Note : we removed the coverpoint of (full transfer) bc it will never happens in the wrapper due to serial input data .

Environment Code

wrapper_env.sv > {} wrapper_env_pkg > wrapper_env

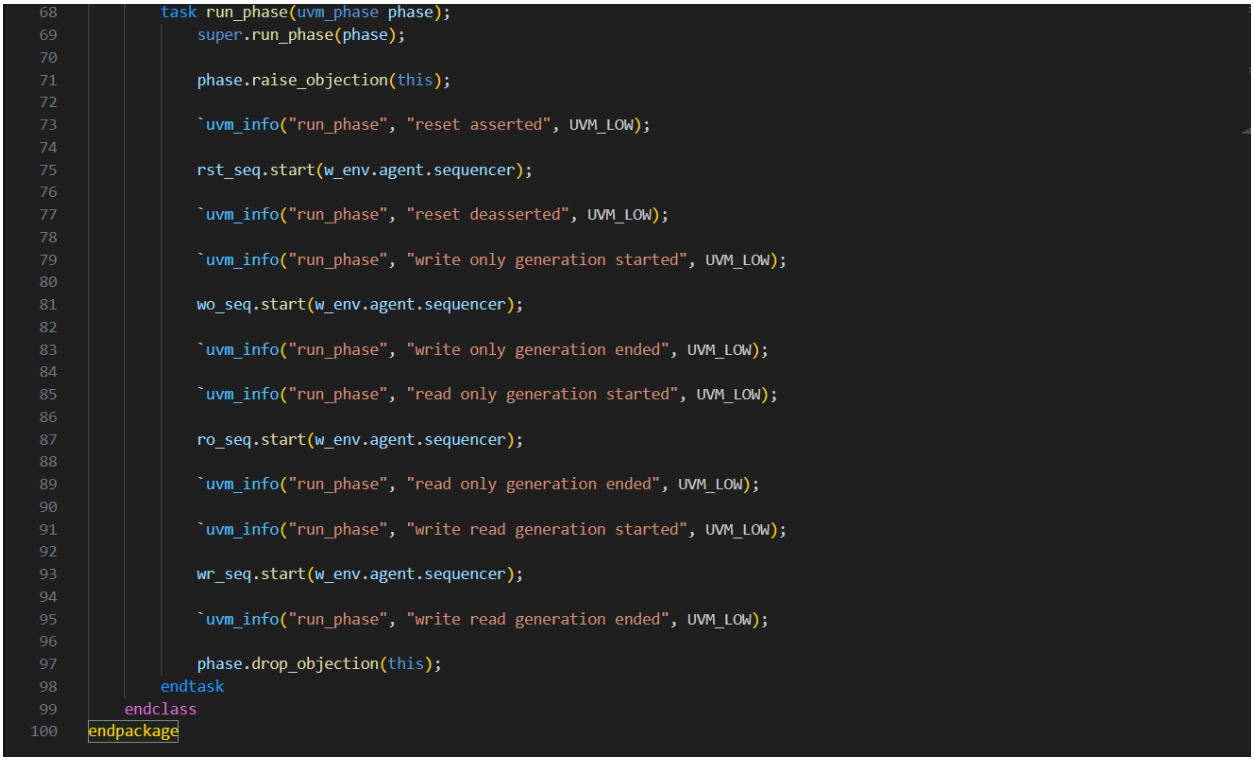
```
package wrapper_env_pkg;
    import wrapper_scoreboard_pkg::*;
    import wrapper_config_pkg::*;
    import wrapper_agent_pkg::*;
    import wrapper_cov_pkg::*;

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

    class wrapper_env extends uvm_env;
        `uvm_component_utils(wrapper_env)

        wrapper_scoreboard sb;
        wrapper_agent agent;
        wrapper_cov cov;

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

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

            sb = wrapper_scoreboard::type_id::create("sb", this);
            agent = wrapper_agent::type_id::create("agent", this);
            cov = wrapper_cov::type_id::create("cov", 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);
        endfunction
    endclass
endpackage
```

Test Code

wrapper_test.sv > {} wrapper_test_pkg > wrapper_test

```
package wrapper_test_pkg;
    import wrapper_rst_seq_pkg::*;
    import wrapper_wr_seq_pkg::*;
    import wrapper_ro_seq_pkg::*;
    import wrapper_wo_seq_pkg::*;
    import wrapper_env_pkg::*;
    import RAM_env_pkg::*;
    import SPI_slave_env_pkg::*;
    import wrapper_config_pkg::*;
    import RAM_config_obj_pkg::*;
    import SPI_slave_config_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class wrapper_test extends uvm_test;
        `uvm_component_utils(wrapper_test)

        wrapper_env w_env;
        RAM_env R_env;
        SPI_slave_env SPI_env;
        wrapper_config wrapper_cfg;
        RAM_config_obj RAM_cfg;
        SPI_slave_config SPI_slave_cfg;
        virtual wrapper_if wrapper_vif;
        wrapper_rst_seq rst_seq;
        wrapper_wo_seq wo_seq;
        wrapper_ro_seq ro_seq;
        wrapper_wr_seq wr_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);
```

```
        function void build_phase(uvm_phase phase);
            super.build_phase(phase);

            w_env = wrapper_env::type_id::create("w_env", this);
            wrapper_cfg = wrapper_config::type_id::create("wrapper_cfg", this);
            R_env = RAM_env::type_id::create("R_env", this);
            RAM_cfg = RAM_config_obj::type_id::create("RAM_cfg", this);
            SPI_env = SPI_slave_env::type_id::create("SPI_env", this);
            SPI_slave_cfg = SPI_slave_config::type_id::create("SPI_slave_cfg", this);
            rst_seq = wrapper_rst_seq::type_id::create("rst_seq", this);
            wo_seq = wrapper_wo_seq::type_id::create("wo_seq", this);
            ro_seq = wrapper_ro_seq::type_id::create("ro_seq", this);
            wr_seq = wrapper_wr_seq::type_id::create("wr_seq", this);

            if (!uvm_config_db #(virtual wrapper_if)::get(this, "", "wrapper_IF", wrapper_cfg.wrapper_vif))
                `uvm_fatal("build_phase", "Test - unable to get the wrapper virtual interface");

            if (!uvm_config_db #(virtual RAM_if)::get(this, "", "RAM_IF", RAM_cfg.vif))
                `uvm_fatal("build_phase", "Test - unable to get the RAM virtual interface");

            if (!uvm_config_db #(virtual SPI_slave_if)::get(this, "", "SPI_slave_IF", SPI_slave_cfg.SPI_slave_vif))
                `uvm_fatal("build_phase", "Test - unable to get the SPI_slave virtual interface");

            wrapper_cfg.is_active = UVM_ACTIVE;
            RAM_cfg.is_active = UVM_PASSIVE;
            SPI_slave_cfg.is_active = UVM_PASSIVE;

            uvm_config_db #(wrapper_config)::set(this, "*", "wrapper_CFG", wrapper_cfg);
            uvm_config_db #(RAM_config_obj)::set(this, "*", "RAM_config_obj", RAM_cfg);
            uvm_config_db #(SPI_slave_config)::set(this, "*", "SPI_slave_config", SPI_slave_cfg);
        endfunction
```



```
68          task run_phase(uvm_phase phase);
69              super.run_phase(phase);
70
71              phase.raise_objection(this);
72
73              `uvm_info("run_phase", "reset asserted", UVM_LOW);
74
75              rst_seq.start(w_env.agent.sequencer);
76
77              `uvm_info("run_phase", "reset deasserted", UVM_LOW);
78
79              `uvm_info("run_phase", "write only generation started", UVM_LOW);
80
81              wo_seq.start(w_env.agent.sequencer);
82
83              `uvm_info("run_phase", "write only generation ended", UVM_LOW);
84
85              `uvm_info("run_phase", "read only generation started", UVM_LOW);
86
87              ro_seq.start(w_env.agent.sequencer);
88
89              `uvm_info("run_phase", "read only generation ended", UVM_LOW);
90
91              `uvm_info("run_phase", "write read generation started", UVM_LOW);
92
93              wr_seq.start(w_env.agent.sequencer);
94
95              `uvm_info("run_phase", "write read generation ended", UVM_LOW);
96
97              phase.drop_objection(this);
98          endtask
99      endclass
100 endpackage
```

Top module Code

wrapper_top.sv > ...

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

module top();

bit clk ;

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

SPI_slave_if SPI_slave_ifi (clk);
RAM_if vif(clk);
wrapper_if w_if(clk);

WRAPPER DUT (w_if.MOSI, w_if.MISO, w_if.SS_n, w_if.clk, w_if.rst_n);
SPI_wrapper_golden GOLD(w_if.clk, w_if.rst_n, w_if.SS_n, w_if.MOSI, w_if.MISO_golden);

bind WRAPPER RAM_assertions wrapper_sva (.clk(DUT.clk), .rst_n(DUT.rst_n),
                                         .rx_valid(DUT.rx_valid), .din(DUT.rx_data_din),
                                         .dout(DUT.tx_data_dout), .tx_valid(DUT.tx_valid));

assign SPI_slave_ifi.rst_n = DUT.rst_n;
assign SPI_slave_ifi.SS_n = DUT.SS_n;
assign SPI_slave_ifi.MOSI = DUT.MOSI;
assign SPI_slave_ifi.tx_data = DUT.tx_data_dout;
assign SPI_slave_ifi.tx_valid = DUT.tx_valid;
assign SPI_slave_ifi.rx_data = DUT.rx_data_din;
assign SPI_slave_ifi.rx_valid = DUT.rx_valid;
assign SPI_slave_ifi.MISO = DUT.MISO;
assign SPI_slave_ifi.rx_data_golden = GOLD.rx_data;
assign SPI_slave_ifi.rx_valid_golden = GOLD.rx_valid;
assign SPI_slave_ifi.MISO_golden = GOLD.MISO_golden;

```

```
assign vif.rst_n = DUT.rst_n;
assign vif.din = DUT.rx_data_din;
assign vif.rx_valid = DUT.rx_valid;
assign vif.dout = DUT.tx_data_dout;
assign vif.tx_valid = DUT.tx_valid;
assign vif.dout_ref = GOLD.tx_data;
assign vif.tx_valid_ref = GOLD.tx_valid;



initial begin
    uvm_config_db#(virtual wrapper_if)::set(null, "uvm_test_top", "wrapper_IF", w_if);
    uvm_config_db#(virtual SPI_slave_if)::set(null, "uvm_test_top", "SPI_slave_IF", SPI_slave_ifi);
    uvm_config_db#(virtual RAM_if)::set(null, "uvm_test_top", "RAM_IF", vif);
    run_test("wrapper_test");
end


endmodule
```

Source files list

src_files.list

```
SPI_slave_if.sv
RAM_if.sv
Wrapper_if.sv
+define+SIM
SPI_slave.sv
RAM.v
RAM_assertions.sv
wrapper.sv
SPI_slave_golden.v
RAM_golden.v
wrapper_golden.v
shared_pkg.sv
SPI_slave_seq_item.sv
RAM_seq_item.sv
wrapper_seq_item.sv
SPI_slave_rst_seq.sv
SPI_slave_main_seq.sv
RAM_sequence.sv
wrapper_reset_seq.sv
wrapper_wo_seq.sv
wrapper_ro_seq.sv
wrapper_wr_seq.sv
SPI_slave_sequencer.sv
RAM_sequencer.sv
wrapper_sqr.sv
SPI_slave_config.sv
RAM_config_obj_pkg.sv
wrapper_cfg.sv
SPI_slave_driver.sv
RAM_driver.sv
wrapper_drv.sv
SPI_slave_monitor.sv
RAM_monitor.sv
wrapper_monitor.sv
SPI_slave_agent.sv
RAM_agent.sv
wrapper_agent.sv
SPI_slave_scoreboard.sv
```

```
SPI_slave_scoreboard.sv
RAM_scoreboard.sv
wrapper_sb.sv
SPI_slave_cov.sv
RAM_coverage.sv
wrapper_cov.sv
SPI_slave_env.sv
RAM_env.sv
wrapper_env.sv
wrapper_test.sv
wrapper_top.sv
```

DO file

vlib work

vlog -f src_files.list +cover -covercells

vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover -l sim.log -sv_seed 1374903841

add wave /top/w_if/*

add wave /top/SPI_slave_ifi/*

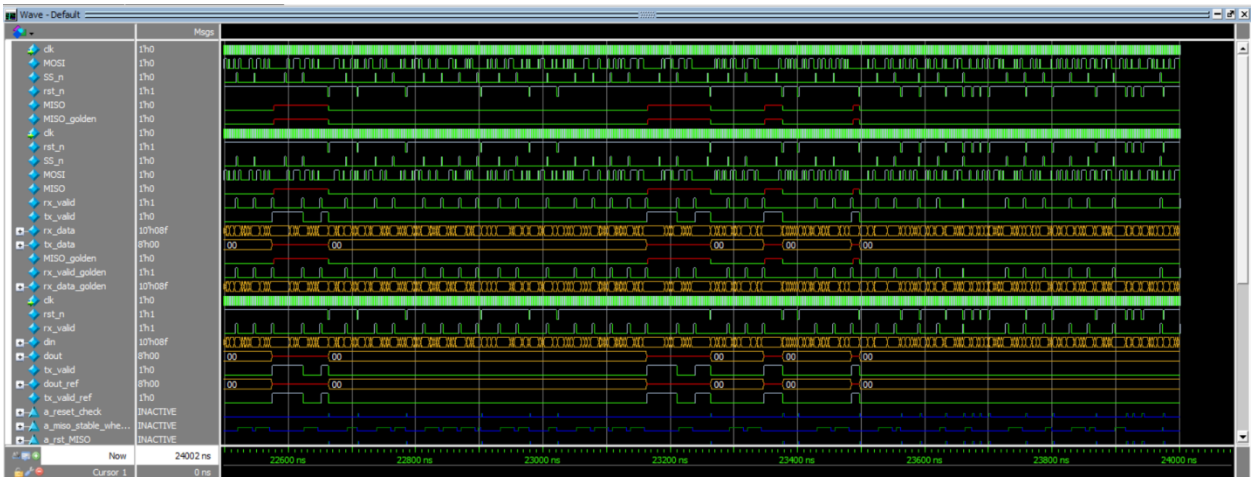
add wave /top/vif/*

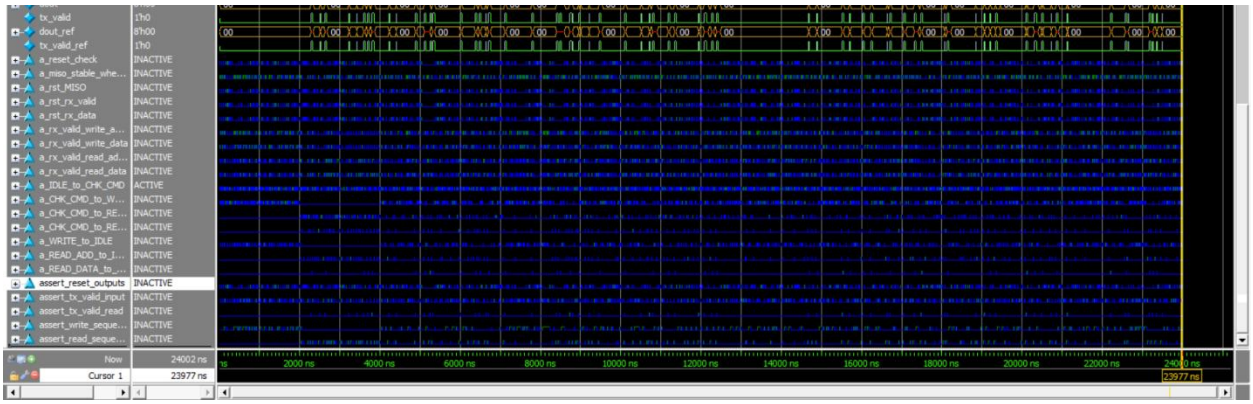
add wave /top/DUT/a_reset_check /top/DUT/a_miso_stable_when_not_read
/top/DUT/SLAVE_instance/a_rst_MISO /top/DUT/SLAVE_instance/a_rst_rx_valid
/top/DUT/SLAVE_instance/a_rst_rx_data /top/DUT/SLAVE_instance/a_rx_valid_write_address
/top/DUT/SLAVE_instance/a_rx_valid_write_data
/top/DUT/SLAVE_instance/a_rx_valid_read_address
/top/DUT/SLAVE_instance/a_rx_valid_read_data
/top/DUT/SLAVE_instance/a_IDLE_to_CHK_CMD
/top/DUT/SLAVE_instance/a_CHK_CMD_to_WRITE
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_ADD
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_DATA
/top/DUT/SLAVE_instance/a_WRITE_to_IDLE
/top/DUT/SLAVE_instance/a_READ_ADD_to_IDLE
/top/DUT/SLAVE_instance/a_READ_DATA_to_IDLE
/top/DUT/wrapper_sva/assert_reset_outputs /top/DUT/wrapper_sva/assert_tx_valid_input
/top/DUT/wrapper_sva/assert_tx_valid_read /top/DUT/wrapper_sva/assert_write_sequence
/top/DUT/wrapper_sva/assert_read_sequence

coverage save WRAPPER.ucdb -onexit

run -all

Waveform





Wrapper_reset_seq

Purpose: Generates a reset sequence for the design under test.

Description: This sequence creates a single wrapper_seq_item and sets the rst_n signal to 0 (active-low reset). It uses the UVM start_item and finish_item methods to send the reset stimulus to the driver. The sequence is typically used to initialize or reset the design to a known state.

Wrapper_write_only_seq

Purpose: Generates a sequence for write-only operations.

Description: This sequence creates 1000 wrapper_seq_item transactions, each randomized with a constraint that the 10th bit (MOSI_arr[9]) of the MOSI_arr array is set to 0. This likely indicates a write operation in a serial protocol context. Each transaction is sent to the driver using start_item and finish_item.

Wrapper_read_only_seq

Purpose: Generates a sequence for read-only operations.

Description: This sequence creates 1000 wrapper_seq_item transactions, each randomized with a constraint that the 10th bit (MOSI_arr[9]) of the MOSI_arr array is set to 1. This likely indicates a read operation in the context of a serial protocol (e.g., SPI). Each transaction is sent to the driver using start_item and finish_item.

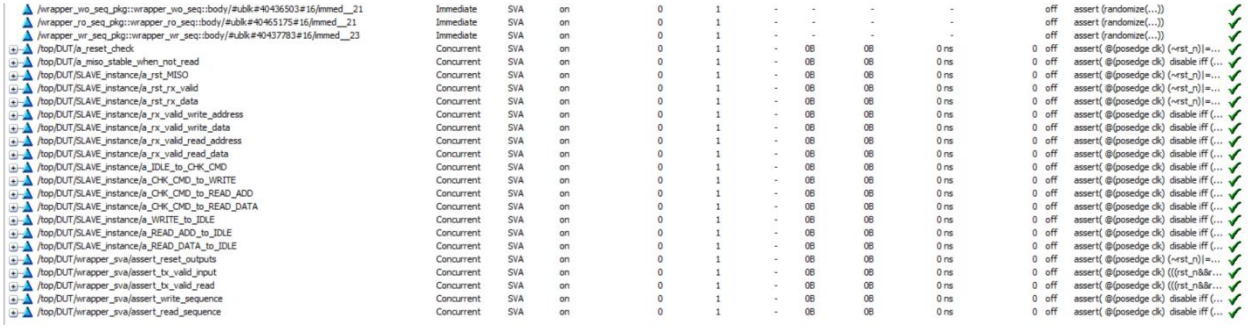
Wrapper_read_only_seq

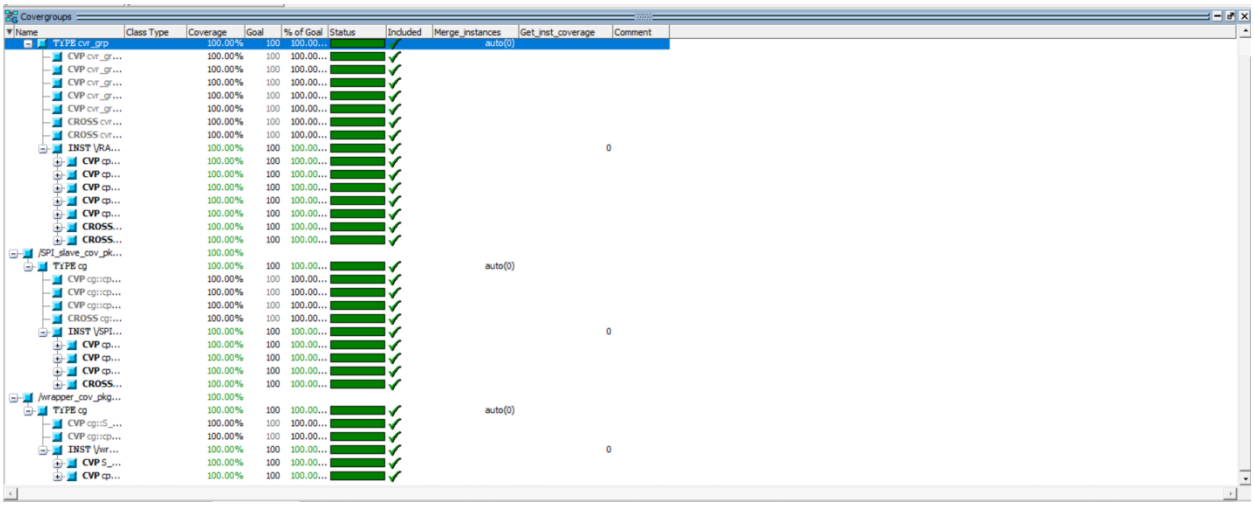
Purpose: Generates a sequence for write operations.

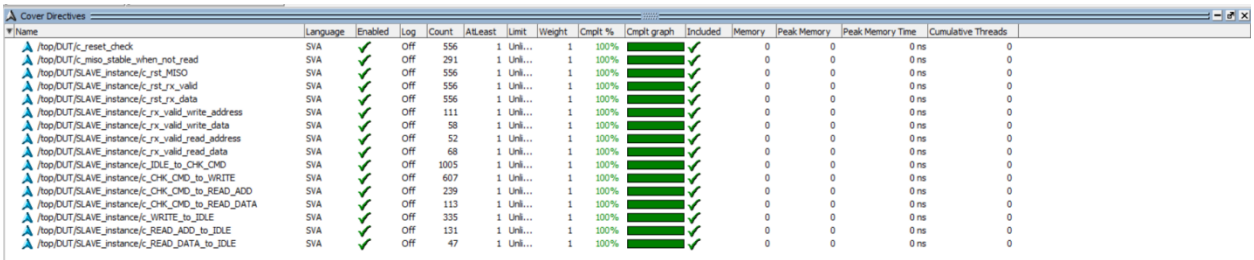
Description: This sequence generates 10,000 wrapper_seq_item transactions, each fully randomized without specific constraints except that the c_after_read_data constraint is disabled (constraint_mode(0)). This suggests the sequence is used for write operations where randomization covers various data patterns, and the disabled constraint avoids post-read data restrictions.

These sequences are designed to test different aspects of a design, likely a serial interface wrapper, by generating controlled (reset) and randomized (read/write) stimuli.

Coverage Report







```
Coverage Report by Instance with details

================================================================================
=== Instance: /top/SPI_slave_if1
=== Design Unit: work.SPI_slave_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         74        74         0   100.00%

----------------------------------Toggle Details----------------------------------

Toggle Coverage for Instance /top/SPI_slave_if1 --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                          MISO           1           1      100.00
                                    MISO_golden          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_data_golden[9-0]      1           1      100.00
                                      rx_valid           1           1      100.00
                                 rx_valid_golden         1           1      100.00
                                    tx_data[7-0]         1           1      100.00
                                      tx_valid           1           1      100.00

Total Node Count     =         37
Toggled Node Count   =         37
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (74 of 74 bins)

================================================================================
=== Instance: /top/vif
=== Design Unit: work.RAM_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         62        62         0   100.00%

----------------------------------Toggle Details----------------------------------

Toggle Coverage for Instance /top/vif --

                                          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
                                   dout_ref[7-0]         1           1      100.00
                                         rst_n           1           1      100.00
                                      rx_valid           1           1      100.00
                                      tx_valid           1           1      100.00
                                  tx_valid_ref           1           1      100.00

Total Node Count     =         31
Toggled Node Count   =         31
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (62 of 62 bins)

================================================================================
=== Instance: /top/w_if
=== Design Unit: work.wrapper_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         12        12         0   100.00%

----------------------------------Toggle Details----------------------------------

Toggle Coverage for Instance /top/w_if --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                          MISO           1           1      100.00
                                    MISO_golden          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     =          6
Toggled Node Count   =          6
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (12 of 12 bins)

================================================================================
=== Instance: /top/DUT/RAM_instance
=== Design Unit: work.RAM
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         7         0   100.00%

----------------------------------Branch Details----------------------------------

Branch Coverage for Instance /top/DUT/RAM_instance

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File RAM.v
------------------------------------IF Branch------------------------------------
    14                                      6139     Count coming in to IF
    14              1                        554         if (~rst_n) begin

    21              1                       1091             if (rx_valid) begin

                                            4494     All False Count
```

```
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         7         0   100.00%

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

Branch Coverage for instance /top/DUT/RAM_instance

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File RAM.v
------------------------------------IF Branch------------------------------------
    14                                     6139     Count coming in to IF
    14              1                       554         if (~rst_n) begin
    21              1                      1091             if (rx_valid) begin
                                           4494     All False Count
Branch totals: 3 hits of 3 branches = 100.00%

---------------------------------CASE Branch---------------------------------
    22                                     1091     Count coming in to CASE
    23              1                       436                  2'b00 : Wr_Addr <= din[7:0];
    24              1                       254                  2'b01 : MEM[Wr_Addr] <= din[7:0];
    25              1                       268                  2'b10 : Rd_Addr <= din[7:0];
    26              1                       133                  2'b11 : dout <= MEM[Rd_Addr];
Branch totals: 4 hits of 4 branches = 100.00%


Expression Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----   -------    ------  --------
    Expressions                      2         2         0   100.00%

================================Expression Details================================

Expression Coverage for instance /top/DUT/RAM_instance --

  File RAM.v
----------------Focused Expression View-----------------
Line       30 Item    1  (din[9] && din[8])
Expression totals: 2 of 2 input terms covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
      din[9]         Y
      din[8]         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  din[9]_0              -
  Row   2:          1  din[9]_1              din[8]
  Row   3:          1  din[8]_0              din[9]
  Row   4:          1  din[8]_1              din[9]


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

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

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

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File RAM.v
    1                                                module RAM (din,clk,rst_n,rx_valid,dout,tx_valid);
    2
    3                                                input      [9:0] din;
    4                                                input            clk, rst_n, rx_valid;
    5
    6                                                output reg [7:0] dout;
    7                                                output reg       tx_valid;
    8
    9                                                reg [7:0] MEM [255:0];
    10
    11                                               reg [7:0] Rd_Addr, Wr_Addr;
    12
    13              1                       6139     always @(posedge clk) begin
    14                                                   if (~rst_n) begin
    15              1                        554             dout <= 0;
    16              1                        554             tx_valid <= 0;
    17              1                        554             Rd_Addr <= 0;
    18              1                        554             Wr_Addr <= 0;
    19                                                   end
    20                                                   else
    21                                                       if (rx_valid) begin
    22                                                           case (din[9:8])
    23              1                        436                 2'b00 : Wr_Addr <= din[7:0];
```

```
    ----------------            ----      ----    ------  --------
    Toggles                       76        76         0   100.00%

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

Toggle Coverage for instance /top/DUT/RAM_instance --

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

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

Toggle Coverage      =     100.00% (76 of 76 bins)

================================================================================
=== Instance: /top/DUT/SLAVE_instance
=== Design Unit: work.SLAVE
================================================================================

Assertion Coverage:
    Assertions                       14        14         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/top/DUT/SLAVE_instance/a_rst_MISO
                     SPI_slave.sv(187)                   0          1
/top/DUT/SLAVE_instance/a_rst_rx_valid
                     SPI_slave.sv(190)                   0          1
/top/DUT/SLAVE_instance/a_rst_rx_data
                     SPI_slave.sv(193)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_write_address
                     SPI_slave.sv(196)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_write_data
                     SPI_slave.sv(199)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_read_address
                     SPI_slave.sv(202)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_read_data
                     SPI_slave.sv(205)                   0          1
/top/DUT/SLAVE_instance/a_IDLE_to_CHK_CMD
                     SPI_slave.sv(208)                   0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_WRITE
                     SPI_slave.sv(211)                   0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_ADD
                     SPI_slave.sv(214)                   0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_DATA
                     SPI_slave.sv(217)                   0          1
/top/DUT/SLAVE_instance/a_WRITE_to_IDLE
                     SPI_slave.sv(220)                   0          1
/top/DUT/SLAVE_instance/a_READ_ADD_to_IDLE
                     SPI_slave.sv(223)                   0          1
/top/DUT/SLAVE_instance/a_READ_DATA_to_IDLE
                     SPI_slave.sv(226)                   0          1
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        27        27         0   100.00%

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

Branch Coverage for instance /top/DUT/SLAVE_instance

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File SPI_slave.sv
------------------------------------IF Branch------------------------------------
    19                                     4112     Count coming in to IF
    19             1                        555             if (~rst_n) begin

    22             1                       3557             else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    30                                     1820     Count coming in to IF
    30             1                        569                     if (SS_n)

    32             1                       1251                     else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    36                                     1439     Count coming in to IF
    38             1                       1439                     else begin

Branch totals: 1 hit of 1 branch = 100.00%

------------------------------------IF Branch------------------------------------
    39                                     1439     Count coming in to IF
    39             1                       1051                         if (~MOSI)

    41             1                        388                         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    42                                      388     Count coming in to IF
    42             1                        257                             if (!received_address)

    44             1                        131                             else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    50                                     3405     Count coming in to IF
    50             1                        376                     if (SS_n)
    --             -                        ----                     -
```

```
------------------------------------IF Branch------------------------------------
    56                                  1607     Count coming in to IF
    56              1                    143                     if (SS_n)

    58              1                   1464                     else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    62                                   635     Count coming in to IF
    62              1                     50                     if (SS_n)

    64              1                    585                     else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    71                                 12000     Count coming in to IF
    71              1                    555             if (~rst_n) begin

    77              1                  11445             else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    86                                  5545     Count coming in to IF
    86              1                   4813                         if (counter > 0) begin

    90              1                    732                         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    95                                  2205     Count coming in to IF
    95              1                   1924                         if (counter > 0) begin

    99              1                    281                         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   105                                  1639     Count coming in to IF
   105              1                    576                         if (tx_valid) begin

   115              1                   1063                         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   107                                   576     Count coming in to IF
   107              1                    477                             if (counter > 0) begin

   111              1                     99                             else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   116                                  1063     Count coming in to IF
   116              1                    922                             if (counter > 0 && !rx_valid) begin

   120              1                    141                             else begin

Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----   -------    ------  --------
    Conditions                       5         5         0   100.00%

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

Condition Coverage for Instance /top/DUT/SLAVE_instance --

  File SPI_slave.sv
----------------Focused Condition View-------------------
Line       86 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line       95 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line      107 Item    1  (counter > 0)
Condition totals: 1 of 1 input term covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line      116 Item    1  ((counter > 0) && ~rx_valid)
```

```
     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  ------------------------
  Row   1:          1  (counter > 0)_0       -
  Row   2:          1  (counter > 0)_1       ~rx_valid
  Row   3:          1  rx_valid_0            (counter > 0)
  Row   4:          1  rx_valid_1            (counter > 0)


Directive Coverage:
    Directives                      14       14         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------
/top/DUT/SLAVE_instance/c_rst_MISO       SLAVE  Verilog  SVA  SPI_slave.sv(188)
                                                                              556 Covered
/top/DUT/SLAVE_instance/c_rst_rx_valid   SLAVE  Verilog  SVA  SPI_slave.sv(191)
                                                                              556 Covered
/top/DUT/SLAVE_instance/c_rst_rx_data    SLAVE  Verilog  SVA  SPI_slave.sv(194)
                                                                              556 Covered
/top/DUT/SLAVE_instance/c_rx_valid_write_address
                                         SLAVE  Verilog  SVA  SPI_slave.sv(197)
                                                                              111 Covered
/top/DUT/SLAVE_instance/c_rx_valid_write_data
                                         SLAVE  Verilog  SVA  SPI_slave.sv(200)
                                                                               58 Covered
/top/DUT/SLAVE_instance/c_rx_valid_read_address
                                         SLAVE  Verilog  SVA  SPI_slave.sv(203)
                                                                               52 Covered
/top/DUT/SLAVE_instance/c_rx_valid_read_data
                                         SLAVE  Verilog  SVA  SPI_slave.sv(206)
                                                                               68 Covered
/top/DUT/SLAVE_instance/c_IDLE_to_CHK_CMD
                                         SLAVE  Verilog  SVA  SPI_slave.sv(209)
                                                                             1005 Covered
/top/DUT/SLAVE_instance/c_CHK_CMD_to_WRITE
                                         SLAVE  Verilog  SVA  SPI_slave.sv(212)
                                                                              607 Covered
/top/DUT/SLAVE_instance/c_CHK_CMD_to_READ_ADD
                                         SLAVE  Verilog  SVA  SPI_slave.sv(215)
                                                                              239 Covered
/top/DUT/SLAVE_instance/c_CHK_CMD_to_READ_DATA
                                         SLAVE  Verilog  SVA  SPI_slave.sv(218)
                                                                              113 Covered
/top/DUT/SLAVE_instance/c_WRITE_to_IDLE  SLAVE  Verilog  SVA  SPI_slave.sv(221)
                                                                              335 Covered
/top/DUT/SLAVE_instance/c_READ_ADD_to_IDLE
                                         SLAVE  Verilog  SVA  SPI_slave.sv(224)
                                                                              131 Covered
/top/DUT/SLAVE_instance/c_READ_DATA_to_IDLE
                                         SLAVE  Verilog  SVA  SPI_slave.sv(227)
                                                                               47 Covered
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/DUT/SLAVE_instance --

FSM_ID: cs
    Current State Object : cs
    ---------------------
    State Value MapInfo :
    ---------------------
Line           State Name              Value
----           ----------              -----
  29                 IDLE                  0
  35              CHK_CMD                  2
  61            READ_DATA                  4
  55             READ_ADD                  3
  49                WRITE                  1
    Covered States :
    ---------------
                    State           Hit_count
                    -----           ---------
                     IDLE                1097
                  CHK_CMD                1051
                READ_DATA                 231
                 READ_ADD                 487
                    WRITE                1246
    Covered Transitions :
    ---------------------
Line           Trans_ID          Hit_count          Transition
----           --------          ---------          ----------
  33                  0                1051          IDLE -> CHK_CMD
  45                  1                 118          CHK_CMD -> READ_DATA
  43                  2                 248          CHK_CMD -> READ_ADD
  40                  3                 639          CHK_CMD -> WRITE
  37                  4                  46          CHK_CMD -> IDLE
  63                  5                 118          READ_DATA -> IDLE
  57                  6                 248          READ_ADD -> IDLE
  51                  7                 638          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                      37        37         0   100.00%

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

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

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File SPI_slave.sv
```

File Edit Format View Help

```
1                                       module SLAVE (MOSI, MISO, SS_n, clk, rst_n, rx_data, rx_valid, tx_data, tx_valid);
2                                           localparam IDLE      = 3'b000;
3                                           localparam WRITE     = 3'b001;
4                                           localparam CHK_CMD   = 3'b010;
5                                           localparam READ_ADD  = 3'b011;
6                                           localparam READ_DATA = 3'b100;
7
8                                           input            MOSI, clk, rst_n, SS_n, tx_valid;
9                                           input      [7:0] tx_data;
10                                          output reg [9:0] rx_data;
11                                          output reg       rx_valid, MISO;
12
13                                          reg [3:0] counter;
14                                          reg       received_address;
15
16                                          reg [2:0] cs, ns;
17
18          1                   4112        always @(posedge clk) begin
19                                              if (~rst_n) begin
20          1                    555                cs <= IDLE;
21                                              end
22                                              else begin
23          1                   3557                cs <= ns;
24                                              end
25                                          end
26
27          1                   8906        always @(*) begin
28                                              case (cs)
29                                                  IDLE : begin
30                                                      if (SS_n)
31          1                    569                    ns = IDLE;
32                                                      else
33          1                   1251                    ns = CHK_CMD;
34                                                  end
35                                                  CHK_CMD : begin
36                                                      if (SS_n)
37                                                          ns = IDLE;
38                                                      else begin
39                                                          if (~MOSI)
40          1                   1051                        ns = WRITE;
41                                                          else begin
42                                                              if (!received_address)
43          1                    257                            ns = READ_ADD;
44                                                              else
45          1                    131                            ns = READ_DATA;
46                                                          end
47                                                      end
48                                                  end
49                                                  WRITE : begin
50                                                      if (SS_n)
51          1                    376                    ns = IDLE;
52                                                      else
53          1                   3029                    ns = WRITE;
54                                                  end
55                                                  READ_ADD : begin
56                                                      if (SS_n)
57          1                    143                    ns = IDLE;
58                                                      else
59          1                   1464                    ns = READ_ADD;
```

```
  94                                                    READ_ADD : begin
  95                                                        if (counter > 0) begin
  96              1                      1924                   rx_data[counter-1] <= MOSI;
  97              1                      1924                   counter <= counter - 1;
  98                                                        end
  99                                                        else begin
 100              1                       281                   rx_valid <= 1;
 101              1                       281                   received_address <= 1;
 102                                                        end
 103                                                    end
 104                                                    READ_DATA : begin
 105                                                        if (tx_valid) begin
 106              1                       576                   rx_valid <= 0;
 107                                                            if (counter > 0) begin
 108              1                       477                       MISO <= tx_data[counter-1];
 109              1                       477                       counter <= counter - 1;
 110                                                            end
 111                                                            else begin
 112              1                        99                       received_address <= 0;
 113                                                            end
 114                                                        end
 115                                                        else begin
 116                                                            if (counter > 0 && !rx_valid) begin
 117              1                       922                       rx_data[counter-1] <= MOSI;
 118              1                       922                       counter <= counter - 1;
 119                                                            end
 120                                                            else begin
 121              1                       141                       rx_valid <= 1;
 122              1                       141                       counter <= 8;

ggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         72        72         0   100.00%

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

ggle Coverage for Instance /top/DUT/SLAVE_instance --

                                      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
                                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
                                     rst_n           1           1      100.00
                                rx_data[9-0]         1           1      100.00
                                    rx_valid         1           1      100.00
                                tx_data[0-7]         1           1      100.00
                                    tx_valid         1           1      100.00

tal Node Count     =         36
ggled Node Count   =         36
toggled Node Count =          0

ggle Coverage      =     100.00% (72 of 72 bins)

================================================================================
= Instance: /top/DUT/wrapper_sva
= Design Unit: work.RAM_assertions
================================================================================

sertion Coverage:
    Assertions                       5         5         0   100.00%
    --------------------------------------------------------------------
me                  File(Line)                    Failure      Pass
                                                  Count        Count
    --------------------------------------------------------------------
op/DUT/wrapper_sva/assert_reset_outputs
                    RAM_assertions.sv(12)                0          1
op/DUT/wrapper_sva/assert_tx_valid_input
                    RAM_assertions.sv(17)                0          1
op/DUT/wrapper_sva/assert_tx_valid_read
                    RAM_assertions.sv(23)                0          1
op/DUT/wrapper_sva/assert_write_sequence
                    RAM_assertions.sv(37)                0          1
op/DUT/wrapper_sva/assert_read_sequence
                    RAM_assertions.sv(53)                0          1
ggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         44        44         0   100.00%

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

```
Coverage Report by instance with details

================================================================================
=== Instance: /top/DUT/SLAVE_instance
=== Design Unit: work.SLAVE
================================================================================

Assertion Coverage:
    Assertions                      14        14         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/top/DUT/SLAVE_instance/a_rst_MISO
                     SPI_slave.sv(187)                   0          1
/top/DUT/SLAVE_instance/a_rst_rx_valid
                     SPI_slave.sv(190)                   0          1
/top/DUT/SLAVE_instance/a_rst_rx_data
                     SPI_slave.sv(193)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_write_address
                     SPI_slave.sv(196)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_write_data
                     SPI_slave.sv(199)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_read_address
                     SPI_slave.sv(202)                   0          1
/top/DUT/SLAVE_instance/a_rx_valid_read_data
                     SPI_slave.sv(205)                   0          1
/top/DUT/SLAVE_instance/a_IDLE_to_CHK_CMD
                     SPI_slave.sv(208)                   0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_WRITE
                     SPI_slave.sv(211)                   0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_ADD
                     SPI_slave.sv(214)                   0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_DATA
                     SPI_slave.sv(217)                   0          1
/top/DUT/SLAVE_instance/a_WRITE_to_IDLE
                     SPI_slave.sv(220)                   0          1
/top/DUT/SLAVE_instance/a_READ_ADD_to_IDLE
                     SPI_slave.sv(223)                   0          1
/top/DUT/SLAVE_instance/a_READ_DATA_to_IDLE
                     SPI_slave.sv(226)                   0          1

================================================================================
=== Instance: /top/DUT/wrapper_sva
=== Design Unit: work.RAM_assertions
================================================================================

Assertion Coverage:
    Assertions                       5         5         0   100.00%
```

```
================================================================================
=== Instance: /top/DUT/wrapper_sva
=== Design Unit: work.RAM_assertions
================================================================================

Assertion Coverage:
    Assertions                       5         5         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/top/DUT/wrapper_sva/assert_reset_outputs
                     RAM_assertions.sv(12)                0          1
/top/DUT/wrapper_sva/assert_tx_valid_input
                     RAM_assertions.sv(17)                0          1
/top/DUT/wrapper_sva/assert_tx_valid_read
                     RAM_assertions.sv(23)                0          1
/top/DUT/wrapper_sva/assert_write_sequence
                     RAM_assertions.sv(37)                0          1
/top/DUT/wrapper_sva/assert_read_sequence
                     RAM_assertions.sv(53)                0          1

================================================================================
=== Instance: /top/DUT
=== Design Unit: work.WRAPPER
================================================================================

Assertion Coverage:
    Assertions                       2         2         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/top/DUT/a_reset_check
                     wrapper.sv(25)                       0          1
/top/DUT/a_miso_stable_when_not_read
                     wrapper.sv(27)                       0          1

================================================================================
=== Instance: /wrapper_wo_seq_pkg
=== Design Unit: work.wrapper_wo_seq_pkg
================================================================================

Assertion Coverage:
    Assertions                       1         1         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
```

assertion_report.txt

```
Name                    File(Line)                    Failure      Pass
                                                      Count        Count
--------------------------------------------------------------------------
/wrapper_wo_seq_pkg/wrapper_wo_seq/body/#ublk#40436503#16/immed__21
                        wrapper_wo_seq.sv(21)               0          1

============================================================================
=== Instance: /wrapper_ro_seq_pkg
=== Design Unit: work.wrapper_ro_seq_pkg
============================================================================

Assertion Coverage:
    Assertions                        1         1         0   100.00%
--------------------------------------------------------------------------
Name                    File(Line)                    Failure      Pass
                                                      Count        Count
--------------------------------------------------------------------------
/wrapper_ro_seq_pkg/wrapper_ro_seq/body/#ublk#40465175#16/immed__21
                        wrapper_ro_seq.sv(21)               0          1

============================================================================
=== Instance: /wrapper_wr_seq_pkg
=== Design Unit: work.wrapper_wr_seq_pkg
============================================================================

Assertion Coverage:
    Assertions                        1         1         0   100.00%
--------------------------------------------------------------------------
Name                    File(Line)                    Failure      Pass
                                                      Count        Count
--------------------------------------------------------------------------
/wrapper_wr_seq_pkg/wrapper_wr_seq/body/#ublk#40437783#16/immed__23
                        wrapper_wr_seq.sv(23)               0          1


ASSERTION RESULTS:
--------------------------------------------------------------------------
Name                    File(Line)                    Failure      Pass
                                                      Count        Count
--------------------------------------------------------------------------
/top/DUT/a_reset_check
                        wrapper.sv(25)                      0          1
/top/DUT/a_miso_stable_when_not_read
                        wrapper.sv(27)                      0          1
/top/DUT/SLAVE_instance/a_rst_MISO
                        SPI_slave.sv(187)                   0          1
/top/DUT/SLAVE_instance/a_rst_rx_valid
                        SPI_slave.sv(190)                   0          1
/top/DUT/SLAVE_instance/a_rst_rx_data
```

assertion_report.txt

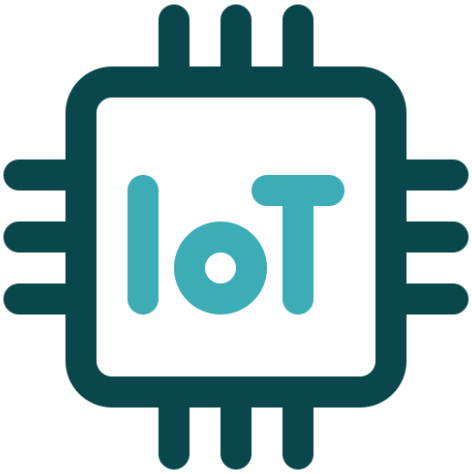
```
                    wrapper.sv(27)                    0          1
/top/DUT/SLAVE_instance/a_rst_MISO
                    SPI_slave.sv(187)                 0          1
/top/DUT/SLAVE_instance/a_rst_rx_valid
                    SPI_slave.sv(190)                 0          1
/top/DUT/SLAVE_instance/a_rst_rx_data
                    SPI_slave.sv(193)                 0          1
/top/DUT/SLAVE_instance/a_rx_valid_write_address
                    SPI_slave.sv(196)                 0          1
/top/DUT/SLAVE_instance/a_rx_valid_write_data
                    SPI_slave.sv(199)                 0          1
/top/DUT/SLAVE_instance/a_rx_valid_read_address
                    SPI_slave.sv(202)                 0          1
/top/DUT/SLAVE_instance/a_rx_valid_read_data
                    SPI_slave.sv(205)                 0          1
/top/DUT/SLAVE_instance/a_IDLE_to_CHK_CMD
                    SPI_slave.sv(208)                 0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_WRITE
                    SPI_slave.sv(211)                 0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_ADD
                    SPI_slave.sv(214)                 0          1
/top/DUT/SLAVE_instance/a_CHK_CMD_to_READ_DATA
                    SPI_slave.sv(217)                 0          1
/top/DUT/SLAVE_instance/a_WRITE_to_IDLE
                    SPI_slave.sv(220)                 0          1
/top/DUT/SLAVE_instance/a_READ_ADD_to_IDLE
                    SPI_slave.sv(223)                 0          1
/top/DUT/SLAVE_instance/a_READ_DATA_to_IDLE
                    SPI_slave.sv(226)                 0          1
/top/DUT/wrapper_sva/assert_reset_outputs
                    RAM_assertions.sv(12)             0          1
/top/DUT/wrapper_sva/assert_tx_valid_input
                    RAM_assertions.sv(17)             0          1
/top/DUT/wrapper_sva/assert_tx_valid_read
                    RAM_assertions.sv(23)             0          1
/top/DUT/wrapper_sva/assert_write_sequence
                    RAM_assertions.sv(37)             0          1
/top/DUT/wrapper_sva/assert_read_sequence
                    RAM_assertions.sv(53)             0          1
/wrapper_wo_seq_pkg/wrapper_wo_seq/body/#ublk#40436503#16/immed__21
                    wrapper_wo_seq.sv(21)             0          1
/wrapper_ro_seq_pkg/wrapper_ro_seq/body/#ublk#40465175#16/immed__21
                    wrapper_ro_seq.sv(21)             0          1
/wrapper_wr_seq_pkg/wrapper_wr_seq/body/#ublk#40437783#16/immed__23
                    wrapper_wr_seq.sv(23)             0          1

Total Coverage By Instance (filtered view): 100.00%
```

Assertion Table

Feature	Assertion
Whenever the reset (rst_n) is asserted low, MISO, rx_valid, and rx_data_din are low	`@(posedge clk) rst_n == 0
Whenever no transmission is valid and SS_n falls, MISO remains stable for 11 cycles	`@(posedge clk) disable iff (!rst_n) ((!tx_valid) && $fell(SS_n))

Real time application for SPI Wrapper

➢ Consumer Electronics & IoT



- **Environmental Monitoring Systems**

Systems deploy numerous sensors to measure temperature, humidity, and air quality, using SPI to communicate with a central microcontroller for rapid data collection and transmission

- **Audio Processing Digital**

audio systems use SPI where the microcontroller sends audio data to a digital-to-analog converter (DAC) while receiving status information concurrently, minimizing delay and maximizing throughput crucial for real-time applications

➢ Automotive & Automotive-Grade Applications

- **Real-Time Clock (RTC) Integration**

Real-time clock chips communicate via SPI to provide alarms and timing information.

- **Sensor Data Transmission**

Touchscreen sensors send pen-down interrupts, temperature sensors send thermal alerts, and vehicle tire pressure sensors transmit data through SPI interfaces.

➢ Microcontroller Platforms

➢ 

- **IoT Development Boards**

Arduino and ESP8266/ESP32 microcontrollers use SPI to connect to external sensors and actuators, enabling real-time analytics and decision-making for IoT applications. An SPI wrapper abstracts these complexities, handling clock polarity, chip select management, and multiple device arbitration—making it essential for any system requiring reliable, fast inter-chip communication.