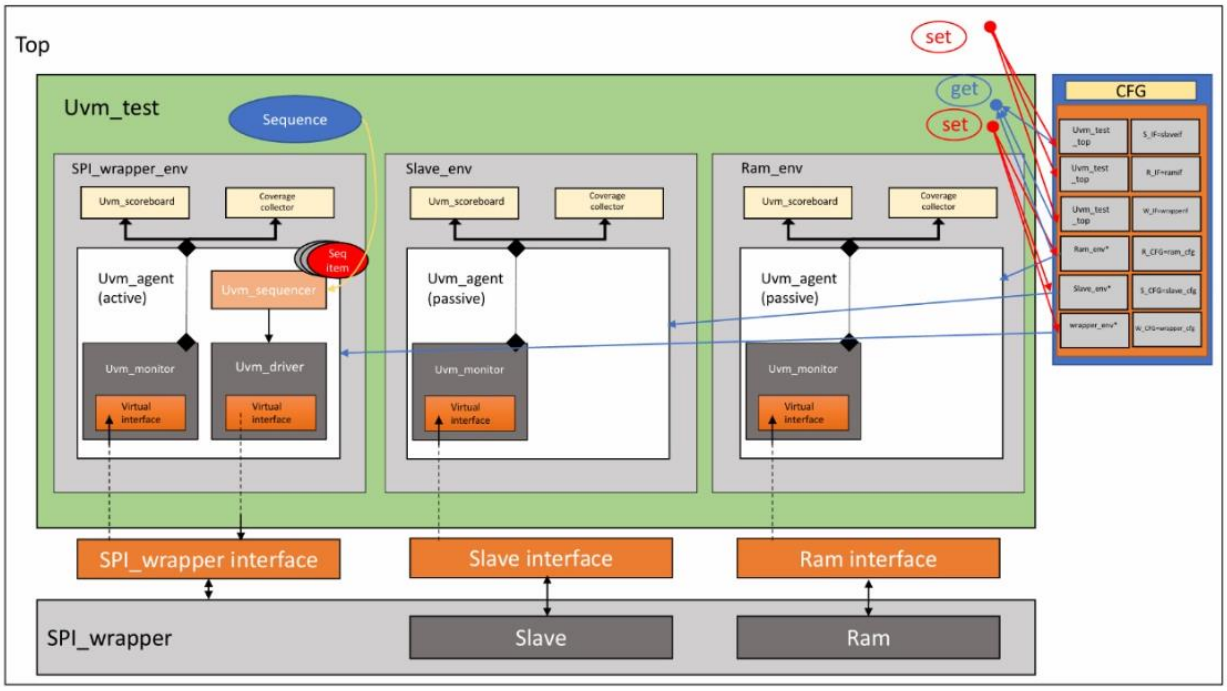
SPI


Top
Uvm_test
Sequence
SPI_wrapper_env
Uvm_scoreboard
Coverage collector
Seq item
Uvm_agent (active)
Uvm_sequencer
Uvm_monitor
Uvm_driver
Virtual interface
Virtual interface
Slave_env
Uvm_scoreboard
Coverage collector
Uvm_agent (passive)
Uvm_monitor
Virtual interface
Ram_env
Uvm_scoreboard
Coverage collector
Uvm_agent (passive)
Uvm_monitor
Virtual interface
set
get
set
CFG
SPI_wrapper interface
Slave interface
Ram interface
SPI_wrapper
Slave
Ram

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 begin        // begin & end has been added
        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]; // fixed to read from MEM[Rd_Addr] not MEM[Wr_Addr]
                default : dout <= 0;
            endcase
        end
        tx_valid <= (din[9] && din[8] && rx_valid)? 1'b1 : 1'b0;
    end
end

endmodule
```

Golden model:

```
module RAM_gold (din , rx_valid , tx_valid , clk , rst_n , dout);
    parameter MEM_WIDTH = 8 , MEM_DEPTH = 256 , ADDER_SIZE = 8;
    input [MEM_WIDTH+1:0] din;
    reg [ADDER_SIZE-1:0] addr_wr;
    reg [ADDER_SIZE-1:0] addr_rd;
    input rx_valid , clk , rst_n;
    output reg tx_valid;
    output reg [MEM_WIDTH-1:0] dout;

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

    always@(posedge clk) begin
        if(~rst_n) begin
            dout <= 0;
            tx_valid <= 0;
            addr_wr <= 0;
            addr_rd <= 0;
        end
        else begin
            if(rx_valid) begin
                case(din[9:8])
                    2'b00 : begin
                        addr_wr <= din[7:0];
                        tx_valid <= 0;
                    end
                    2'b01 : begin
                        mem [addr_wr] <= din[7:0];
                        tx_valid <= 0;
                    end
                    2'b10 : begin
                        addr_rd <= din[7:0];
                        tx_valid <= 0;
                    end
                    2'b11 : begin
                        dout <= mem [addr_rd];
                        tx_valid <= 1;
                    end

                endcase
            end
            else begin
                tx_valid <= 0;
            end

        end
    end
endmodule
```

Config db:

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

    class ram_config_obj extends uvm_object;
        `uvm_object_utils(ram_config_obj);

        virtual ram_if ram_config_vif;

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

    endclass


endpackage
```

SVA:

```
module ram_sva (
    input  clk,
    input  rx_valid,
    input  rst_n,
    input  [9:0] din,
    input  tx_valid,
    input  [7:0] dout
);

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


   // Assertions

    // rst Assertion
    rst_assert:assert        property  (@(posedge clk) !rst_n |=> (!tx_valid && !dout));
    rst_cover:cover          property  (@(posedge clk) !rst_n |=> (!tx_valid && !dout));

    // tx_valid 1 Assertion
    tx_valid_1_assert:assert property  (@(posedge clk) disable iff (!rst_n) (din[9:8] == 2'b00 || din[9:8] == 2'b01 || din[9:8] == 2'b10) |=> (!tx_valid));
    tx_valid_1_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (din[9:8] == 2'b00 || din[9:8] == 2'b01 || din[9:8] == 2'b10) |=> (!tx_valid));

    // tx_valid 2 Assertion
    tx_valid_2_assert:assert property  (@(posedge clk) disable iff (!rst_n) (din[9] && din[8] && rx_valid) |=> $rose(tx_valid) ##1 $fell(tx_valid) [->1]);
    tx_valid_2_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (din[9] && din[8] && rx_valid) |=> $rose(tx_valid) ##1 $fell(tx_valid) [->1]);

    // write Assertion
    write_assert:assert      property  (@(posedge clk) disable iff (!rst_n) (rx_valid && din[9:8] == 2'b00) |=> (din[9:8] == 2'b01) [->1]);
    write_cover:cover        property  (@(posedge clk) disable iff (!rst_n) (rx_valid && din[9:8] == 2'b00) |=> (din[9:8] == 2'b01) [->1]);

    // read Assertion
    read_assert:assert       property  (@(posedge clk) disable iff (!rst_n) (rx_valid && din[9:8] == 2'b10) |=> (din[9:8] == 2'b11) [->1]);
    read_cover:cover         property  (@(posedge clk) disable iff (!rst_n) (rx_valid && din[9:8] == 2'b10) |=> (din[9:8] == 2'b11) [->1]);


endmodule
```

Seq item:

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

   typedef enum bit [1:0] {
        WRITE_ADDR = 2'b00,
        WRITE_DATA = 2'b01,
        READ_ADDR  = 2'b10,
        READ_DATA  = 2'b11
    } operation_t;

    class ram_seq_item extends uvm_sequence_item;
        `uvm_object_utils(ram_seq_item);

        operation_t prev_op;
        rand logic  [9:0] din;
        rand bit  rst_n, rx_valid;

        logic [7:0] dout, dout_ref;
        bit   tx_valid, tx_valid_ref;


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

        constraint c_rst {rst_n dist {1 := 95 , 0 := 5};}

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


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

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

    endclass

endpackage
```

Rst seq:

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

Write only seq:

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

    class ram_write_seq extends uvm_sequence #(ram_seq_item);
        `uvm_object_utils(ram_write_seq);

        ram_seq_item seq_item;
        operation_t prev_op;

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

        task body;
            seq_item = ram_seq_item :: type_id :: create ("seq_item");
            prev_op = WRITE_ADDR; // Start with Write Address

            repeat (10000) begin
            start_item(seq_item);

                // Constraint 3: Write Address followed by Write Address or Write Data
                if (!seq_item.randomize() with {
                    seq_item.din[9:8] inside {2'b00, 2'b01};
                    if (prev_op == WRITE_ADDR) {
                        seq_item.din[9:8] inside {WRITE_ADDR, WRITE_DATA};
                    } else if (prev_op == WRITE_DATA) {
                        seq_item.din[9:8] == WRITE_ADDR; // After Write Data, go back to Write Address
                    }
                }) begin
                    `uvm_error("RAND_FAIL", "Randomization failed in write sequence")
                end

                prev_op = operation_t'(seq_item.din[9:8]);
                seq_item.prev_op = prev_op;

            finish_item(seq_item);
            end
        endtask

    endclass

endpackage
```

Read only seq:

```
1  package read_sequence;
6      class ram_read_seq extends uvm_sequence #(ram_seq_item);
7          `uvm_object_utils(ram_read_seq);
8
9          ram_seq_item seq_item;
10         operation_t prev_op;
11
12         function new(string name = "ram_read_seq");
13                 super.new(name);
14         endfunction
15
16         task body;
17             seq_item = ram_seq_item :: type_id :: create ("seq_item");
18             prev_op = READ_ADDR; // Start with Rrad Address
19
20             repeat (10000) begin
21             start_item(seq_item);
22             if (!seq_item.randomize() with {
23                     seq_item.din[9:8] inside {2'b10, 2'b11};
24                     if (prev_op == READ_ADDR) {
25                         seq_item.din[9:8] inside {READ_ADDR, READ_DATA};
26                     } else if (prev_op == READ_DATA) {
27                         seq_item.din[9:8] == READ_ADDR; // After Read Data, go back to Read Address
28                     }
29                 }) begin
30                     `uvm_error("RAND_FAIL", "Randomization failed in read sequence")
31                 end
32
33                 prev_op = operation_t'(seq_item.din[9:8]);
34                 seq_item.prev_op = prev_op;
35
36             finish_item(seq_item);
37             end
38         endtask
39
40     endclass
41
42 endpackage
```

Read write seq:

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

    class ram_Write_Read_seq extends uvm_sequence #(ram_seq_item);
        `uvm_object_utils(ram_Write_Read_seq);

        ram_seq_item seq_item;
        operation_t prev_op;

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

        task body;
            seq_item = ram_seq_item :: type_id :: create ("seq_item");
            prev_op = WRITE_ADDR; // Start with Write Address

            repeat (10000) begin
            start_item(seq_item);
            if (!seq_item.randomize() with {

                    // Write Address followed by Write Address or Write Data
                    if (prev_op == WRITE_ADDR) {
                        seq_item.din[9:8] inside {WRITE_ADDR, WRITE_DATA};
                    }
                    // After Write Data: 60% Read Address, 40% Write Address
                    else if (prev_op == WRITE_DATA) {
                        seq_item.din[9:8] dist { READ_ADDR := 60, WRITE_ADDR := 40 };
                    }
                    // Read Address followed by Read Address or Read Data
                    else if (prev_op == READ_ADDR) {
                        seq_item.din[9:8] inside {READ_ADDR, READ_DATA};
                    }
                    // After Read Data: 60% Write Address, 40% Read Address
                    else if (prev_op == READ_DATA) {
                        seq_item.din[9:8] dist { WRITE_ADDR := 60, READ_ADDR := 40 };
                    }
                }) begin
                    `uvm_error("RAND_FAIL", "Randomization failed in write_read sequence")
                end

                prev_op = operation_t'(seq_item.din[9:8]);
                seq_item.prev_op = prev_op;
```

```
                finish_item(seq_item);
                end
            endtask

        endclass

    endpackage
```

Sqr:

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

    class ram_sqr extends uvm_sequencer #(ram_seq_item);
        `uvm_component_utils(ram_sqr);
        function new(string name = "ram_sqr", uvm_component parent = null);
            super.new(name,parent);
        endfunction
    endclass
endpackage
```

Driver:

```
package ram_driver;
    import ram_config::*;
    import seq_item_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class ram_driver extends uvm_driver #(ram_seq_item);
        `uvm_component_utils(ram_driver);

        virtual ram_if ram_vif;
        ram_seq_item stim_seq_item;

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

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

            forever begin
                stim_seq_item = ram_seq_item :: type_id :: create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);
                ram_vif.din            = stim_seq_item.din;
                ram_vif.rst_n          = stim_seq_item.rst_n;
                ram_vif.rx_valid       = stim_seq_item.rx_valid;
                @(negedge ram_vif.clk);
                seq_item_port.item_done();
                `uvm_info("run_phase" , stim_seq_item.convert2string_stimulus(), UVM_HIGH)
            end

        endtask

    endclass

endpackage
```

Monitor:

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

    class ram_monitor extends uvm_monitor;
        `uvm_component_utils(ram_monitor);
        virtual ram_if ram_vif;
        ram_seq_item rsp_seq_item;
        uvm_analysis_port #(ram_seq_item) mon_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = ram_seq_item :: type_id :: create("rsp_seq_item");
                rsp_seq_item.din           = ram_vif.din;
                rsp_seq_item.rst_n         = ram_vif.rst_n;
                rsp_seq_item.rx_valid      = ram_vif.rx_valid;
                rsp_seq_item.dout          = ram_vif.dout;
                rsp_seq_item.tx_valid      = ram_vif.tx_valid;
                rsp_seq_item.dout_ref      = ram_vif.dout_ref;
                rsp_seq_item.tx_valid_ref  = ram_vif.tx_valid_ref;
                @(negedge ram_vif.clk);
                mon_ap.write(rsp_seq_item);
                `uvm_info("run_phase" , rsp_seq_item.convert2string(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

Agent:

```
package agent_pkg;
    import uvm_pkg::*;
   `include "uvm_macros.svh"
   import sequencer_pkg::*;
   import ram_driver::*;
   import monitor::*;
   import ram_config::*;
   import seq_item_pkg::*;

    class ram_agent extends uvm_agent;
        `uvm_component_utils(ram_agent);
        ram_sqr sqr;
        ram_driver drv;
        ram_monitor mon;
        ram_config_obj ram_cfg;
        uvm_analysis_port #(ram_seq_item) agt_ap;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            if(!uvm_config_db #(ram_config_obj) :: get(this, "", "CFG", ram_cfg)) begin
                `uvm_fatal("build_phase","Unable to get configration object");
            end

            sqr = ram_sqr     :: type_id :: create("sqr",this);
            drv = ram_driver  :: type_id :: create("drv",this);
            mon = ram_monitor :: type_id :: create("mon",this);

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

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            drv.ram_vif = ram_cfg.ram_config_vif;
            mon.ram_vif = ram_cfg.ram_config_vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
            mon.mon_ap.connect(agt_ap);
        endfunction

    endclass
endpackage
```

Coverage collector:

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

    class ram_coverage extends uvm_component;
    `uvm_component_utils(ram_coverage);

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


        covergroup CovCode;
            // 1. [9:8] coverpoint
            cp_din: coverpoint seq_item_cov.din[9:8] {
                bins IDLE             = {2'b00};
                bins WRITE            = {2'b01};
                bins READ_ADD         = {2'b10};
                bins READ_DATA        = {2'b11};
                bins transitions1     = (2'b00 => 2'b01);
                bins transitions2     = (2'b10 => 2'b11);
                bins total_transitions = (2'b00 => 2'b01 => 2'b10 => 2'b11);
            }

              // 2. rx_valid coverpoint
            cp_rx_valid: coverpoint seq_item_cov.rx_valid {
                bins rx_high = {1};
                bins rx_low  = {0};
            }

            cp_tx_valid: coverpoint seq_item_cov.tx_valid {
                bins tx_high = {1};
                bins tx_low  = {0};
            }

            //cross coverage  1
            cp_din__cp_rx_valid_1 : cross cp_din, cp_rx_valid {
                //option.cross_auto_bin_max = 0;
                bins cross_1 = binsof(cp_rx_valid.rx_high) && binsof(cp_din);
            }

            //cross coverage  2
            cp_din__cp_rx_valid_2 : cross cp_din, cp_tx_valid {
                option.cross_auto_bin_max = 0;
                bins cross_2 = binsof(cp_din.READ_DATA) && binsof(cp_tx_valid.tx_high);
            }



        endgroup

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

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

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

```

```
    task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(seq_item_cov);
                CovCode.sample();
            end
    endtask

    endclass
endpackage
```

Scoreboard:

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

    class ram_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(ram_scoreboard);
        uvm_analysis_export #(ram_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(ram_seq_item) sb_fifo;
        ram_seq_item seq_item_sb;


        int error_count   = 0;
        int correct_count = 0;

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

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                if(seq_item_sb.tx_valid != seq_item_sb.tx_valid_ref || seq_item_sb.dout != seq_item_sb.dout_ref) begin
                    $display("tx_valid = %h, tx_valid_ref = %h, dout = %h, dout_ref = %h, rst_n = %h, rx_valid = %h",
                            seq_item_sb.tx_valid, seq_item_sb.tx_valid_ref, seq_item_sb.dout, seq_item_sb.dout_ref, seq_item_sb.rst_n, seq_item_sb.rx_valid);
                    error_count++;
                end
                else begin
                    correct_count++;
                end
            end

        endtask

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

    endclass

endpackage
```

Env:

```
package ram_env;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ram_driver::*;
    import agent_pkg::*;
    import scoreboard_pkg::*;
    import cov_pkg::*;

    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.agt_ap.connect(sb.sb_export);
        agt.agt_ap.connect(cov.cov_export);
        endfunction

    endclass

endpackage
```

Test:

```
package ram_test;
    import ram_env::*;
    import ram_config::*;
    import write_sequence::*;
    import read_sequence::*;
    import Write_Read_sequence::*;
    import rst_sequence::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class ram_test extends uvm_test;
        `uvm_component_utils(ram_test);

        ram_config_obj ram_config_obj_test;
        ram_env env;
        ram_write_seq write_seq;
        ram_read_seq  read_seq;
        ram_Write_Read_seq write_read_seq;
        ram_rst_seq rst_seq;


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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = ram_env                          :: type_id :: create("env",this);
            ram_config_obj_test = ram_config_obj   :: type_id :: create("ram_config_obj_test");
            write_seq = ram_write_seq              :: type_id :: create ("write_seq");
            read_seq = ram_read_seq                :: type_id :: create ("read_seq");
            write_read_seq = ram_Write_Read_seq    :: type_id :: create ("write_read_seq");
            rst_seq  = ram_rst_seq                 :: type_id :: create ("rst_seq");

            if(!uvm_config_db #(virtual ram_if) :: get(this, "", "ram_IF", ram_config_obj_test.ram_config_vif))
                `uvm_fatal("build_phase", " test -  unable to get the virtual interface");

            uvm_config_db #(ram_config_obj) :: set(this, "*", "CFG", ram_config_obj_test);
```

```

        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("run_phase","assert rst ",UVM_LOW);
            rst_seq.start(env.agt.sqr);
            `uvm_info("run_phase","deassert rst ",UVM_LOW);

            `uvm_info("run_phase","write operation started",UVM_LOW);
            write_seq.start(env.agt.sqr);
            `uvm_info("run_phase","write operation ended",UVM_LOW);

            `uvm_info("run_phase","read operation started",UVM_LOW);
            read_seq.start(env.agt.sqr);
            `uvm_info("run_phase","read operation ended",UVM_LOW);

            `uvm_info("run_phase","write read  operation started",UVM_LOW);
            write_read_seq.start(env.agt.sqr);
            `uvm_info("run_phase","write read  operation ended",UVM_LOW);
            phase.drop_objection(this);

        endtask

    endclass

endpackage
```

Top:

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

module top ();
    bit clk;

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

    ram_if ramif (clk);
    RAM DUT (ramif.din,clk,ramif.rst_n,ramif.rx_valid,ramif.dout,ramif.tx_valid);
    RAM_gold gold (ramif.din , ramif.rx_valid , ramif.tx_valid_ref , clk , ramif.rst_n , ramif.dout_ref);

    bind RAM ram_sva checker_inst (.*);

    initial begin
        uvm_config_db #(virtual ram_if) :: set(null, "uvm_test_top", "ram_IF", ramif);
        run_test("ram_test");
    end

endmodule
```

Interface:

```
interface ram_if (clk);
  parameter MEM_WIDTH = 8 , MEM_DEPTH = 256 , ADDER_SIZE = 8;
  input clk;
  logic [MEM_WIDTH+1:0] din;
  bit rx_valid , clk , rst_n;
  bit tx_valid, tx_valid_ref;
  logic [MEM_WIDTH-1:0] dout, dout_ref;

endinterface : ram_if
```

Do file:

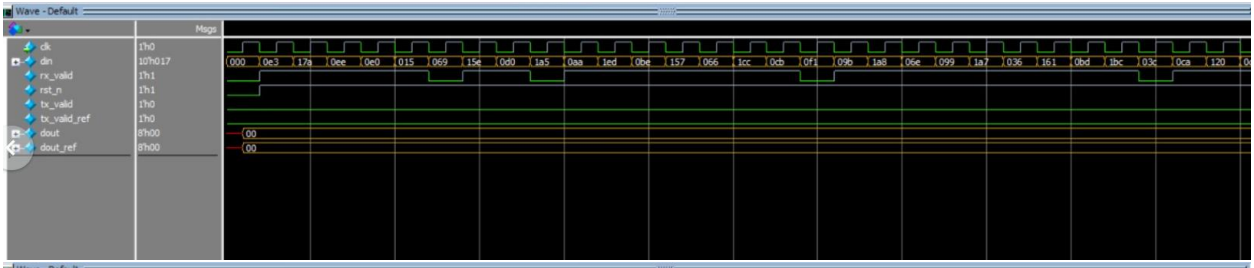
```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover
add wave /top/ramif/*
run -all
coverage save RAM.ucdb -onexit
coverage exclude -src RAM.v -line 27 -code s
coverage exclude -src RAM.v -line 27 -code b

```

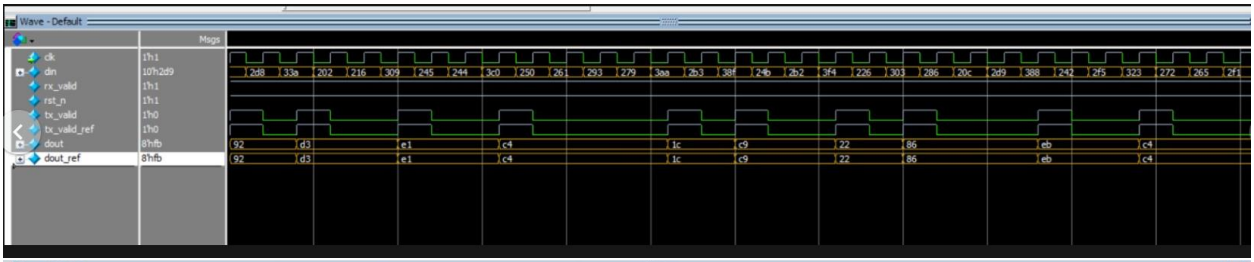
Src files:

```

RAM_if.sv
RAM.v
RAM_SVA.sv
RAM_config.sv
RAM_seq_item.sv
rst_seq_pkg.sv
write_seq_pkg.sv
Read_Seq.sv
Write_Read_Seq.sv
sequencer_pkg.sv
RAM_driver.sv
Monitor.sv
Agent.sv
coverage_collector_pkg.sv
scoreboard_pkg.sv
RAM_env.sv
RAM_test.sv
RAM_top.sv

```

Simulation:

```
Transcript
# * recording_detail has been set.                                    *
# *  To turn off, set 'recording_detail' to off:                      *
# * uvm_config_db#(int)            ::set(null, "", "recording_detail", 0); *
# * uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0); *
# ************************************************************************
# UVM_INFO RAM_test.sv(47) @ 2: uvm_test_top [run_phase] deassert rst
# UVM_INFO RAM_test.sv(49) @ 2: uvm_test_top [run_phase] write operation started
# UVM_INFO RAM_test.sv(51) @ 20002: uvm_test_top [run_phase] write operation ended
# UVM_INFO RAM_test.sv(53) @ 20002: uvm_test_top [run_phase] read operation started
# UVM_INFO RAM_test.sv(55) @ 40002: uvm_test_top [run_phase] read operation ended
# UVM_INFO RAM_test.sv(57) @ 40002: uvm_test_top [run_phase] write read  operation started
# UVM_INFO RAM_test.sv(59) @ 60002: uvm_test_top [run_phase] write read  operation ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 60002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO scoreboard_pkg.sv(51) @ 60002: uvm_test_top.env.sb [report_phase]  Successful transactions of RAM :       30001
# UVM_INFO scoreboard_pkg.sv(52) @ 60002: uvm_test_top.env.sb [report_phase]  failed transactions of RAM     :           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   14
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     2
# [run_phase]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 60002 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```





Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-	-	off	assert ($cast(seq,o))
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-	-	off	assert ($cast(seq,o))
/top/DUT/checker_inst/rst_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=>(~tx_valid8
/top/DUT/checker_inst/tx_valid_1_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((
/top/DUT/checker_inst/tx_valid_2_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (\`5
/top/DUT/checker_inst/write_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((r
/top/DUT/checker_inst/read_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((r

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/checker_inst/rst_cover	SVA	✓	Off	1562	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/checker_inst/tx_valid_1_cover	SVA	✓	Off	22441	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/checker_inst/tx_valid_2_cover	SVA	✓	Off	4066	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/checker_inst/write_cover	SVA	✓	Off	8150	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/checker_inst/read_cover	SVA	✓	Off	8056	1	Unlimited	1	100%		✓	0	0	0 ns	0

Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/cov_pkg/ram_coverage		100.00%							
TYPE CovCode		100.00%	100	100.00...		✓	auto(0)		
CVP CovCode::cp_din		100.00%	100	100.00...		✓			
CVP CovCode::cp_rx_valid		100.00%	100	100.00...		✓			
CVP CovCode::cp_tx_valid		100.00%	100	100.00...		✓			
CROSS CovCode::cp_din__cp...		100.00%	100	100.00...		✓			
CROSS CovCode::cp_din__cp...		100.00%	100	100.00...		✓			
INST \/cov_pkg::ram_covera...		100.00%	100	100.00...		✓			0
CVP cp_din		100.00%	100	100.00...		✓			
CVP cp_rx_valid		100.00%	100	100.00...		✓			
CVP cp_tx_valid		100.00%	100	100.00...		✓			
CROSS cp_din__cp_rx_va...		100.00%	100	100.00...		✓			
CROSS cp_din__cp_rx_va...		100.00%	100	100.00...		✓			

Functional coverage report:

fcover_report.txt

```
overage Report by instance with details

================================================================================
=== Instance: /top/DUT/checker_inst
=== Design Unit: work.ram_sva
================================================================================

irective Coverage:
    Directives                       5         5         0   100.00%

IRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
lame                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------------
top/DUT/checker_inst/rst_cover           ram_sva Verilog  SVA  RAM_SVA.sv(21)  1562 Covered
top/DUT/checker_inst/tx_valid_1_cover    ram_sva Verilog  SVA  RAM_SVA.sv(25)  22441 Covered
top/DUT/checker_inst/tx_valid_2_cover    ram_sva Verilog  SVA  RAM_SVA.sv(29)  4066 Covered
top/DUT/checker_inst/write_cover         ram_sva Verilog  SVA  RAM_SVA.sv(33)  8150 Covered
top/DUT/checker_inst/read_cover          ram_sva Verilog  SVA  RAM_SVA.sv(37)  8056 Covered

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

overgroup Coverage:
    Covergroups                      1        na        na   100.00%
        Coverpoints/Crosses          5        na        na        na
            Covergroup Bins         20        20         0   100.00%
--------------------------------------------------------------------------------------------------
overgroup                                        Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------
TYPE /cov_pkg/ram_coverage/CovCode                100.00%        100          -    Covered
    covered/total bins:                                20         20          -
    missing/total bins:                                 0         20          -
    % Hit:                                        100.00%        100          -
    Coverpoint cp_din                             100.00%        100          -    Covered
        covered/total bins:                             7          7          -
        missing/total bins:                             0          7          -
        % 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
```

fcover_report.txt

% Hit:	100.00%	100	-	
Cross cp_din__cp_rx_valid_1	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Cross cp_din__cp_rx_valid_2	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Covergroup instance \/cov_pkg::ram_coverage::CovCode				
	100.00%	100	-	Covered
covered/total bins:	20	20	-	
missing/total bins:	0	20	-	
% Hit:	100.00%	100	-	
Coverpoint cp_din	100.00%	100	-	Covered
covered/total bins:	7	7	-	
missing/total bins:	0	7	-	
% Hit:	100.00%	100	-	
bin IDLE	9992	1	-	Covered
bin WRITE	5045	1	-	Covered
bin READ_ADD	9952	1	-	Covered
bin READ_DATA	5011	1	-	Covered
bin transitions1	5045	1	-	Covered
bin transitions2	5010	1	-	Covered
bin total_transitions	501	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 rx_high	28467	1	-	Covered
bin rx_low	1534	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 tx_high	4531	1	-	Covered
bin tx_low	25470	1	-	Covered
Cross cp_din__cp_rx_valid_1	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin cross_1	28467	1	-	Covered
bin <total_transitions,rx_low>	26	1	-	Covered
bin <transitions2,rx_low>	254	1	-	Covered
bin <transitions1,rx_low>	277	1	-	Covered
bin <READ_DATA,rx_low>	254	1	-	Covered
bin <WRITE,rx_low>	277	1	-	Covered
bin <READ_ADD,rx_low>	508	1	-	Covered

fcover_report.txt

```
            bin <WRITE,rx_low>                          277          1          -    Covered
            bin <READ_ADD,rx_low>                       508          1          -    Covered
            bin <IDLE,rx_low>                           494          1          -    Covered
    Cross cp_din__cp_rx_valid_2                      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 cross_2                                 4531          1          -    Covered

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

----------------------------------------------------------------------------------------------------------------
 TYPE /cov_pkg/ram_coverage/CovCode                  100.00%        100          -    Covered
    covered/total bins:                                   20         20          -
    missing/total bins:                                    0         20          -
    % Hit:                                          100.00%        100          -
    Coverpoint cp_din                               100.00%        100          -    Covered
        covered/total bins:                                7          7          -
        missing/total bins:                                0          7          -
        % 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          -
    Cross cp_din__cp_rx_valid_1                     100.00%        100          -    Covered
        covered/total bins:                                8          8          -
        missing/total bins:                                0          8          -
        % Hit:                                      100.00%        100          -
    Cross cp_din__cp_rx_valid_2                     100.00%        100          -    Covered
        covered/total bins:                                1          1          -
        missing/total bins:                                0          1          -
        % Hit:                                      100.00%        100          -
 Covergroup instance \/cov_pkg::ram_coverage::CovCode
                                                    100.00%        100          -    Covered
    covered/total bins:                                   20         20          -
    missing/total bins:                                    0         20          -
    % Hit:                                          100.00%        100          -
    Coverpoint cp_din                               100.00%        100          -    Covered
        covered/total bins:                                7          7          -
        missing/total bins:                                0          7          -
        % Hit:                                      100.00%        100          -
```

fcover_report.txt

```
        covered/total bins:                          2         2         -
        missing/total bins:                          0         2         -
        % Hit:                                 100.00%       100         -
        bin rx_high                                28467         1         -    Covered
        bin rx_low                                  1534         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 tx_high                                 4531         1         -    Covered
        bin tx_low                                 25470         1         -    Covered
    Cross cp_din__cp_rx_valid_1                    100.00%       100         -    Covered
        covered/total bins:                          8         8         -
        missing/total bins:                          0         8         -
        % Hit:                                 100.00%       100         -
        Auto, Default and User Defined Bins:
            bin cross_1                            28467         1         -    Covered
            bin <total_transitions,rx_low>            26         1         -    Covered
            bin <transitions2,rx_low>                254         1         -    Covered
            bin <transitions1,rx_low>                277         1         -    Covered
            bin <READ_DATA,rx_low>                   254         1         -    Covered
            bin <WRITE,rx_low>                       277         1         -    Covered
            bin <READ_ADD,rx_low>                    508         1         -    Covered
            bin <IDLE,rx_low>                        494         1         -    Covered
    Cross cp_din__cp_rx_valid_2                    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 cross_2                             4531         1         -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                       Design Design   Lang File(Line)       Hits Status
                                           Unit   UnitType
--------------------------------------------------------------------------------------------------
/top/DUT/checker_inst/rst_cover            ram_sva Verilog  SVA  RAM_SVA.sv(21)  1562 Covered
/top/DUT/checker_inst/tx_valid_1_cover     ram_sva Verilog  SVA  RAM_SVA.sv(25)  22441 Covered
/top/DUT/checker_inst/tx_valid_2_cover     ram_sva Verilog  SVA  RAM_SVA.sv(29)  4066 Covered
/top/DUT/checker_inst/write_cover          ram_sva Verilog  SVA  RAM_SVA.sv(33)  8150 Covered
/top/DUT/checker_inst/read_cover           ram_sva Verilog  SVA  RAM_SVA.sv(37)  8056 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 5

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

Bug report:

```
20      else begin        // begin & end has been added
21          if (rx_valid) begin
22              case (din[9:8])
23                  2'b00 : Wr_Addr <= din[7:0];
24                  2'b01 : MEM[Wr_Addr] <= din[7:0];
25                  2'b10 : Rd_Addr <= din[7:0];
26                  2'b11 : dout <= MEM[Rd_Addr]; // fixed to read from MEM[Rd_Addr] not MEM[Wr_Addr]
27                  default : dout <= 0;
28              endcase
29          end
30          tx_valid <= (din[9] && din[8] && rx_valid)? 1'b1 : 1'b0;
31      end
32  end
33
34  endmodule
```

1.begin and end has been added

2.fixed to be mem[Rd_Addr]

Assertions table:

Feature	Assertion Name	Property Description
Reset Assertion	`rst_assert`	When reset is asserted, tx_valid should be low and dout should be low
TX Valid for Commands 00,01,10	`tx_valid_1_assert`	When din[9:8] is 00, 01, or 10, tx_valid should be low
TX Valid Rise for Command 11	`tx_valid_2_assert`	When din[9:8] is 11 and rx_valid is high, tx_valid should rise and eventually fall
Write Sequence	`write_assert`	After write address command (00) with rx_valid, eventually write data command (01) should occur
Read Sequence	`read_assert`	After read address command (10) with rx_valid, eventually read data command (11) should occur

SLAVE

Design:

```
module SLAVE (MOSI,MISO,SS_n,clk,rst_n,rx_data,rx_valid,tx_data,tx_valid);

localparam IDLE      = 3'b000;
localparam CHK_CMD   = 3'b001;//replaced
localparam WRITE     = 3'b010;// replaced
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 = 0; //should be intialized by zero

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) // '!' has been added.;
                        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;
        counter <= 10;

    end
    else begin
        case (cs)
            IDLE : begin
                rx_data  <= 0; //has been added
                rx_valid <= 0;
                //MISO <= 0; //has been added
            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; //removed
                    if (counter > 0) begin
                        MISO <= tx_data[counter-1];
                        counter <= counter - 1;
                    end
                    else begin
                        received_address <= 0;
                        rx_valid <= 0;//added
                    end
                end
                else begin
                    if (counter > 0) begin
                        rx_data[counter-1] <= MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        rx_valid <= 1;
                        counter <= 9; // 9 not 8
                    end
                end
            end
        endcase
    end
end

```

```
136
137 `ifdef SIM
138     // Assertions
139     trans1_assert:assert property  (@(posedge clk) disable iff (!rst_n) (cs == IDLE && !SS_n) |=> (cs == CHK_CMD));
140     trans1_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (cs == IDLE && !SS_n) |=> (cs == CHK_CMD));
141
142     trans2_assert:assert property  (@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n) |=> (cs == WRITE || cs == READ_ADD || cs == READ_DATA));
143     trans2_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n) |=> (cs == WRITE || cs == READ_ADD || cs == READ_DATA));
144
145     trans3_assert:assert property  (@(posedge clk) disable iff (!rst_n) (cs == WRITE && SS_n) |=> (cs == IDLE));
146     trans3_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (cs == WRITE && SS_n) |=> (cs == IDLE));
147
148     trans4_assert:assert property  (@(posedge clk) disable iff (!rst_n) (cs == READ_ADD && SS_n) |=> (cs == IDLE));
149     trans4_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (cs == READ_ADD && SS_n) |=> (cs == IDLE));
150
151     trans5_assert:assert property  (@(posedge clk) disable iff (!rst_n) (cs == READ_DATA && SS_n) |=> (cs == IDLE));
152     trans5_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (cs == READ_DATA && SS_n) |=> (cs == IDLE));
153
154
155 `endif
156
157 endmodule
```

Golden model:

```
1  // SPI Slave
2
3  module SPI_Slave_gold(MOSI ,MISO ,SS_n ,clk ,rst_n ,rx_data ,rx_valid ,tx_data ,tx_valid);
4      parameter IDLE = 3'b000 , CHK_CMD = 3'b001 , WRITE = 3'b010 , READ_ADD = 3'b011 , READ_DATA = 3'b100;
5      input MOSI ,SS_n ,clk ,rst_n ,tx_valid;
6      input [7:0] tx_data;
7      output reg MISO , rx_valid;
8      output reg [9:0] rx_data;
9
10     reg [2:0] cs , ns;
11
12     reg flag = 0;
13
14     reg [3:0] i;
15     //reg [2:0] i;
16
17     always @(posedge clk parameter IDLE = 3'b000 , CHK_CMD = 3'b001 , WRITE = 3'b010 , READ_ADD = 3'b011 , READ_DATA = 3'b100;
18         if(~rst_n) cs <= IDLE;
19         else cs <= ns;
20     end
21
22     always @(*) begin
23         case (cs)
24             IDLE : begin
25                 if(SS_n) ns = IDLE;
26                 else ns = CHK_CMD;
27             end
28
29             CHK_CMD : begin
30                 if(SS_n) ns = IDLE;
31                 else begin
32                     if(~MOSI) ns = WRITE;
33                     else begin
34                         if(flag) ns = READ_DATA;
35                         else ns = READ_ADD;
36                     end
37                 end
38             end
39
40             WRITE : begin
41                 if(SS_n) ns = IDLE;
42                 else ns = WRITE;
43             end
44
```

```
45              READ_ADD : begin
46                  if(SS_n) ns = IDLE;
47                  else ns = READ_ADD;
48              end
49
50              READ_DATA : begin
51                  if(SS_n) ns = IDLE;
52                  else ns = READ_DATA;
53              end
54
55             // default ns = IDLE;
56
57          endcase
58      end
59
60
61      always @(posedge clk ) begin
62          if(~rst_n) begin
63              rx_data  <= 0;
64              rx_valid <= 0;
65              MISO <= 0;
66              flag <= 0;
67           //   i <= 9;
68             // i <= 7;
69          end
70
71          else begin
72            case(cs)
73              IDLE: begin
74                  rx_data  <= 0;
75                  rx_valid <= 0;
76                //  MISO     <= 0;
77              end
78
79              CHK_CMD : begin
80                   i <= 10;
81                  // i <= 9;
82              end
83
```

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

            READ_ADD : begin
                if(i > 0) begin
                  // rx_valid <= 0;
                   rx_data[i-1] <= MOSI;
                   i <= i-1;
                end
                else begin
                    rx_valid <= 1;
                    flag <= 1;
                end
            end

            READ_DATA : begin
                if(tx_valid) begin
                  // rx_valid <= 0;
                   if (i > 0) begin
                       MISO <= tx_data[i-1];
                       i <= i - 1;
                   end
                   else begin
                       flag <= 0;
                       rx_valid <= 0;
                   end
                end

                else begin
                    if(i > 0) begin
                        //rx_valid <= 0;
                        rx_data[i-1] <= MOSI;
                        i <= i-1;
                    end
                                else begin
                                    rx_valid <= 1;
                                    i <= 9;
                                end
                            end
                        end

                    endcase

                end
            end

        endmodule
```

Debug report:

1.

```
READ_DATA : begin
    if (tx_valid) begin
       // rx_valid <= 0; //removed
        if (counter > 0) begin
            MISO <= tx_data[counter-1];
            counter <= counter - 1;
        end
        else begin
            received_address <= 0;
            rx_valid <= 0;//added
        end
    end
    else begin
        if (counter > 0) begin
            rx_data[counter-1] <= MOSI;
            counter <= counter - 1;
        end
        else begin
            rx_valid <= 1;
            counter <= 9; // 9 not 8
        end
    end
end
```

2.

```
4  localparam CHK_CMD    = 3'b001;//replaced
5  localparam WRITE      = 3'b010;// replaced
6  localparam READ_ADD   = 3'b011;
```

3.

```
16  reg      received_address = 0; //should be intialized by zero
17
```

4.

```
if (!received_address) // '!' has been added.;
    ns = READ_ADD;
else
    ns = READ_DATA;
```

5.

```
rx_data  <= 0; //has been added
```

The bugs were commented // here.

SVA:

```
module slave_sva (
   input clk,
   input MOSI,
   input rst_n,
   input SS_n,
   input tx_valid,
   input [7:0] tx_data,
   input [9:0] rx_data,
   input rx_valid,
   input MISO,
   input [2:0] cs,
   input [2:0] ns
);


    // Assertions

   // rst Assertion
   rst_assert:assert        property  (@(posedge clk) !rst_n |=> ((!MISO && !rx_valid && !rx_data)));
   rst_cover:cover          property  (@(posedge clk) !rst_n |=> ((!MISO && !rx_valid && !rx_data)));

   // Assertion 2
   tx_valid_1_assert:assert property  (@(posedge clk) disable iff (!rst_n) (rx_data[9:7] == 2'b000 || rx_data[9:7] == 2'b001 || rx_data[9:7] == 2'b110 || rx_data[9:7] == 2'b111) |->
   ##10 (rx_valid) && SS_n [->1]);
   tx_valid_1_cover:cover   property  (@(posedge clk) disable iff (!rst_n) (rx_data[9:7] == 2'b000 || rx_data[9:7] == 2'b001 || rx_data[9:7] == 2'b110 || rx_data[9:7] == 2'b111) |->
   ##10 (rx_valid) && SS_n [->1]);



endmodule
```

Config db:

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

Seq item:

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

    class slave_seq_item extends uvm_sequence_item;
        `uvm_object_utils(slave_seq_item);

        rand logic [7:0] tx_data;
        logic [9:0] rx_data ,rx_data_ref;
        logic rx_valid, MISO,MISO_ref ,rx_valid_ref;
        logic MOSI;
        rand logic SS_n;
        rand logic rst_n;
        rand bit tx_valid;
        rand bit[10:0] MOSI_bins;

         int counter,i;

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

     constraint reset_constraint {rst_n dist {0:=2, 1:=998};}

    constraint SS_N_constraint{
            if (~rst_n) SS_n == 1;

            else if (MOSI_bins[10:8] == 3'b111) {
                if(counter % 24 == 0)
                    SS_n == 1;
                else
                    SS_n == 0;
            }

            else {
                if (counter % 14 == 0)
                    SS_n == 1;
                else
                    SS_n == 0;
            }
    }

    constraint MOSI_bins_constraint{

      MOSI_bins[10:8] inside {3'b000,3'b001,3'b110,3'b111};

    }

```

```
49      constraint tx_valid_constraint{
50              if (MOSI_bins[10:8] == 3'b111)
51                {
52                  if(counter > 13)
53                    tx_valid == 1;
54                  else
55                    tx_valid == 0;
56                }
57              else
58                {
59                  tx_valid == 0;
60                }
61      }
62
63      function void post_randomize;
64                  counter ++;
65                  if(SS_n) begin
66                          counter = 1;
67                          i = 10;
68                  end
69                  else if (i >= 0 && counter > 2) begin
70                      MOSI = MOSI_bins[i];
71                      i--;
72                  end
73      endfunction
74
75
```

```
76       function string convert2string();
77         return $sformatf("%s reset = 0b%0b , mosi=%b , miso=%b , ss_n = %b ",super.convert2string(),rst_n, MOSI, MISO, SS_n);
78       endfunction
79
80       function string convert2string_stimulus();
81         return $sformatf("reset = 0b%0b , mosi=%b , ss_n = %b ",rst_n, MOSI, SS_n);
82       endfunction
83      endclass
84
85  endpackage
```

Rst seq:

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

    class slave_rst_seq extends uvm_sequence #(slave_seq_item);
        `uvm_object_utils(slave_rst_seq);

        slave_seq_item seq_item;

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

        task body;
            seq_item = slave_seq_item :: type_id :: create ("seq_item");
            start_item(seq_item);
            seq_item.rst_n      = 0;
            seq_item.SS_n       = 1;
            seq_item.tx_valid   = 0;
            seq_item.tx_data    = 0;
            seq_item.MOSI       = 0;
            finish_item(seq_item);
        endtask

    endclass

endpackage
```

Main seq:

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

    class slave_main_seq extends uvm_sequence #(slave_seq_item);
        `uvm_object_utils(slave_main_seq);

        slave_seq_item seq_item;

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

        task body ;
        seq_item= slave_seq_item :: type_id :: create("seq_item");
        repeat(10000) begin
            start_item(seq_item);
            if (seq_item.SS_n)
                seq_item.MOSI_bins.rand_mode(1);
            else
                seq_item.MOSI_bins.rand_mode(0);

            assert (seq_item.randomize());
            finish_item(seq_item);

        end
        endtask

 endclass

endpackage
```

Sqr:

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

    class slave_sqr extends uvm_sequencer #(slave_seq_item);
        `uvm_component_utils(slave_sqr);
        function new(string name = "slave_sqr", uvm_component parent = null);
            super.new(name,parent);
        endfunction
    endclass
endpackage
```

Driver:

```
1  package slave_driver;
2      import slave_config::*;
3      import seq_item_pkg::*;
4      import uvm_pkg::*;
5      `include "uvm_macros.svh"
6
7      class slave_driver extends uvm_driver #(slave_seq_item);
8          `uvm_component_utils(slave_driver);
9
10         virtual slave_if slave_vif;
11         slave_seq_item stim_seq_item;
12
13         function new(string name = "slave_driver", uvm_component parent = null);
14             super.new(name,parent);
15         endfunction
16
17         task run_phase(uvm_phase phase);
18             super.run_phase(phase);
19
20             forever begin
21                 stim_seq_item = slave_seq_item :: type_id :: create("stim_seq_item");
22                 seq_item_port.get_next_item(stim_seq_item);
23                 slave_vif.MOSI            = stim_seq_item.MOSI;
24                 slave_vif.rst_n           = stim_seq_item.rst_n;
25                 slave_vif.SS_n            = stim_seq_item.SS_n;
26                 slave_vif.tx_valid        = stim_seq_item.tx_valid;
27                 slave_vif.tx_data         = stim_seq_item.tx_data;
28                 @(negedge slave_vif.clk);
29                 seq_item_port.item_done();
30                 `uvm_info("run_phase" , stim_seq_item.convert2string_stimulus(), UVM_HIGH)
31
32             end
33
34         endtask
35
36     endclass
37
38 endpackage
```

Monitor:

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

    class slave_monitor extends uvm_monitor;
        `uvm_component_utils(slave_monitor);

        virtual slave_if slave_vif;
        slave_seq_item rsp_seq_item;
        uvm_analysis_port #(slave_seq_item) mon_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = slave_seq_item :: type_id :: create("rsp_seq_item");
                rsp_seq_item.MOSI          = slave_vif.MOSI;
                rsp_seq_item.rst_n         = slave_vif.rst_n;
                rsp_seq_item.SS_n          = slave_vif.SS_n;
                rsp_seq_item.tx_valid      = slave_vif.tx_valid;
                rsp_seq_item.tx_data       = slave_vif.tx_data;
                rsp_seq_item.rx_data       = slave_vif.rx_data;
                rsp_seq_item.rx_valid      = slave_vif.rx_valid;
                rsp_seq_item.MISO          = slave_vif.MISO;
                rsp_seq_item.rx_data_ref   = slave_vif.rx_data_ref;
                rsp_seq_item.rx_valid_ref  = slave_vif.rx_valid_ref;
                rsp_seq_item.MISO_ref      = slave_vif.MISO_ref;
                @(negedge slave_vif.clk);
                mon_ap.write(rsp_seq_item);
                `uvm_info("run_phase" , rsp_seq_item.convert2string(), UVM_HIGH)
            end

        endtask

    endclass
endpackage

```

Agent:

```
package agent_pkg;
   import uvm_pkg::*;
   `include "uvm_macros.svh"
   import sequencer_pkg::*;
   import slave_driver::*;
   import monitor::*;
   import slave_config::*;
   import seq_item_pkg::*;

    class slave_agent extends uvm_agent;
        `uvm_component_utils(slave_agent);
        slave_sqr sqr;
        slave_driver drv;
        slave_monitor mon;
        slave_config_obj slave_cfg;
        uvm_analysis_port #(slave_seq_item) agt_ap;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            if(!uvm_config_db #(slave_config_obj) :: get(this, "", "CFG", slave_cfg)) begin
                `uvm_fatal("build_phase","Unable to get configration object");
            end

            sqr = slave_sqr     :: type_id :: create("sqr",this);
            drv = slave_driver  :: type_id :: create("drv",this);
            mon = slave_monitor :: type_id :: create("mon",this);

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

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            drv.slave_vif = slave_cfg.slave_config_vif;
            mon.slave_vif = slave_cfg.slave_config_vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
            mon.mon_ap.connect(agt_ap);
        endfunction

    endclass
endpackage
```

Coverage collector:

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

    class slave_coverage extends uvm_component;
    `uvm_component_utils(slave_coverage);

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

        covergroup CovCode;
            // 1. rx_data[9:8] coverpoint
            cp_cmd: coverpoint seq_item_cov.rx_data[9:8] {
                bins IDLE      = {2'b00};
                bins WRITE     = {2'b01};
                bins READ_ADD  = {2'b10};
                bins READ_DATA = {2'b11};
            }

              // 2. SS_n coverpoint
            cp_ssn: coverpoint seq_item_cov.SS_n {
                bins normal_trans = (1 => 0 [*13] => 1);
                bins read_trans   = (1 => 0 [*23] => 1);
            }

            // 3. MOSI coverpoint
            cp_mosi: coverpoint seq_item_cov.MOSI {
                bins trans1 = (0 => 0 => 0);
                bins trans2 = (0 => 0 => 1);
                bins trans3 = (1 => 1 => 0);
                bins trans4 = (1 => 1 => 1);
            }


        endgroup

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

```

```
43
44      function void build_phase(uvm_phase phase);
45          super.build_phase(phase);
46          cov_export = new("cov_export",this);
47          cov_fifo   = new("cov_fifo",this);
48      endfunction
49
50      function void connect_phase(uvm_phase phase);
51          super.connect_phase(phase);
52          cov_export.connect(cov_fifo.analysis_export);
53      endfunction
54
55      task run_phase(uvm_phase phase);
56              super.run_phase(phase);
57              forever begin
58                  cov_fifo.get(seq_item_cov);
59                  CovCode.sample();
60              end
61      endtask
62
63
64      endclass
65
66  endpackage
```

Scoreboard:

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

    class slave_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(slave_scoreboard);
        uvm_analysis_export #(slave_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(slave_seq_item) sb_fifo;
        slave_seq_item seq_item_sb;


        int error_count   = 0;
        int correct_count = 0;

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

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                if(seq_item_sb.rx_data != seq_item_sb.rx_data_ref || seq_item_sb.rx_valid != seq_item_sb.rx_valid_ref || seq_item_sb.MISO != seq_item_sb.MISO_ref) begin
                  $display("rx_data = %h, rx_data_ref = %h, rx_valid = %h, rx_valid_ref = %h, MISO_ref = %h, MISO = %h, rst_n = %h, SS_n = %h",
                  seq_item_sb.rx_data, seq_item_sb.rx_data_ref, seq_item_sb.rx_valid, seq_item_sb.rx_valid_ref, seq_item_sb.MISO_ref, seq_item_sb.MISO, seq_item_sb.rst_n, seq_item_sb.SS_n);
                    error_count++;
                end
                else begin
                    `uvm_info("run_phase",$sformatf("correct out_ref outputs : %s ", seq_item_sb.convert2string()), UVM_HIGH);
                    correct_count++;
                end
            end

        endtask

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

    endclass

endpackage
```

Env:

```
package slave_env;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import slave_driver::*;
    import agent_pkg::*;
    import scoreboard_pkg::*;
    import cov_pkg::*;

    class slave_env extends uvm_env;
        `uvm_component_utils(slave_env);

        slave_agent agt;
        slave_scoreboard sb;
        slave_coverage cov;

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

        function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        agt = slave_agent      :: type_id :: create ("agt",this);
        sb  = slave_scoreboard :: type_id :: create ("sb",this);
        cov = slave_coverage   :: type_id :: create ("cov",this);
        endfunction

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

    endclass

endpackage
```

Test:

```
package slave_test;
    import slave_env::*;
    import slave_config::*;
    import main_sequence::*;
    import rst_sequence::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class slave_test extends uvm_test;
        `uvm_component_utils(slave_test);

        slave_config_obj slave_config_obj_test;
        slave_env env;
        slave_main_seq main_seq;
        slave_rst_seq rst_seq;


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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = slave_env                         :: type_id :: create("env",this);
            slave_config_obj_test = slave_config_obj :: type_id :: create("slave_config_obj_test");
            main_seq = slave_main_seq               :: type_id :: create ("main_seq");
            rst_seq  = slave_rst_seq                :: type_id :: create ("rst_seq");

            if(!uvm_config_db #(virtual slave_if) :: get(this, "", "slave_IF", slave_config_obj_test.slave_config_vif))
                `uvm_fatal("build_phase", " test -  unable to get the virtual interface");

            uvm_config_db #(slave_config_obj) :: set(this, "*", "CFG", slave_config_obj_test);

        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("run_phase","assert rst ",UVM_LOW);
            rst_seq.start(env.agt.sqr);
            `uvm_info("run_phase","deassert rst ",UVM_LOW);

            `uvm_info("run_phase","stimulus generation started",UVM_LOW);
            main_seq.start(env.agt.sqr);
            `uvm_info("run_phase","stimulus generation ended",UVM_LOW);
            phase.drop_objection(this);

        endtask

    endclass

endpackage
```

Top:

```
1  import slave_test::*;
2  import uvm_pkg::*;
3  `include "uvm_macros.svh"
4
5  module top ();
6      bit clk;
7
8      initial begin
9          forever begin
10             #1 clk = ~clk;
11         end
12     end
13
14     slave_if slaveif (clk);                                    input  [9:0] rx_data,
15     SLAVE DUT (slaveif.MOSI, slaveif.MISO, slaveif.SS_n, clk, slaveif.rst_n, slaveif.rx_data, slaveif.rx_valid, slaveif.tx_data, slaveif.tx_valid);
16     SPI_Slave_gold gold (slaveif.MOSI, slaveif.MISO_ref ,slaveif.SS_n ,clk ,slaveif.rst_n ,slaveif.rx_data_ref ,slaveif.rx_valid_ref ,slaveif.tx_data ,slaveif.tx_valid);
17
18     bind SLAVE slave_sva checker_inst (.*);
19
20     initial begin
21         uvm_config_db #(virtual slave_if) :: set(null, "uvm_test_top", "slave_IF", slaveif);
22         run_test("slave_test");
23     end
24
25 endmodule
```

Interface:

```
1  interface slave_if (clk);
2    input clk;
3    logic MOSI, rst_n, SS_n, tx_valid;
4    logic [7:0] tx_data;
5    logic [9:0] rx_data;
6    logic rx_valid, MISO;
7
8    logic [9:0] rx_data_ref;
9    logic rx_valid_ref, MISO_ref;
10
11
12 endinterface : slave_if
```

Do file:

```
vlib work
vlog -f src_files.list +define+SIM +cover -covercells
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover
add wave /top/slaveif/*
run -all
coverage save SLAVE_coverage.ucdb -du SLAVE

```

Src files:

```

slave_if.sv
SPI_slave.sv
Slave_gold.v
slave_SVA.sv
slave_config.sv
slave_seq_item.sv
rst_seq_pkg.sv
main_seq_pkg.sv
sequencer_pkg.sv
slave_driver.sv
Monitor.sv
Agent.sv
coverage_collector_pkg.sv
scoreboard_pkg.sv
slave_env.sv
slave_test.sv
slave_top.sv
```

Simulation:

```
# UVM_INFO @ 0: reporter [RNTST] Running test slave_test...
# UVM_INFO slave_test.sv(39) @ 0: uvm_test_top [run_phase] assert rst
# ****************************************************************************
# * Questa UVM Transaction Recording Turned ON.                              *
# * recording_detail has been set.                                           *
# *  To turn off, set 'recording_detail' to off:                             *
# * uvm_config_db#(int)            ::set(null, "", "recording_detail", 0); *
# * uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0); *
# ****************************************************************************
# UVM_INFO slave_test.sv(41) @ 2: uvm_test_top [run_phase] deassert rst
# UVM_INFO slave_test.sv(43) @ 2: uvm_test_top [run_phase] stimulus generation started
# UVM_INFO slave_test.sv(45) @ 20002: uvm_test_top [run_phase] stimulus generation ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO scoreboard_pkg.sv(51) @ 20002: uvm_test_top.env.sb [report_phase]  Successful transactions of Slave :       10001
# UVM_INFO scoreboard_pkg.sv(52) @ 20002: uvm_test_top.env.sb [report_phase]  failed transactions of Slave     :           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   10
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     2
# [run_phase]     4
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 20002 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-	-	off	assert ($cast(seq,o))
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-	-	off	assert ($cast(seq,o))
/main_sequence::slave_main_seq::body/#ublk#252445301#18/immed__35	Immediate	SVA	on	0	1	-	-	-	-	-	off	assert (randomize(...))
/top/DUT/trans1_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((
/top/DUT/trans2_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((
/top/DUT/trans3_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((
/top/DUT/trans4_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((
/top/DUT/trans5_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((
/top/DUT/checker_inst/rst_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=>((~MISO>.&
/top/DUT/checker_inst/tx_valid_1_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((((

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/trans1_cover	SVA	✓	Off	622	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/trans2_cover	SVA	✓	Off	619	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/trans3_cover	SVA	✓	Off	299	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/trans4_cover	SVA	✓	Off	123	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/trans5_cover	SVA	✓	Off	176	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/checker_inst/rst_cover	SVA	✓	Off	23	1	Unlimited	1	100%		✓	0	0	0 ns	0
/top/DUT/checker_inst/tx_valid_1_cover	SVA	✓	Off	4619	1	Unlimited	1	100%		✓	0	0	0 ns	0

Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/cov_pkg/slave_coverage		100.00%							
TYPE CovCode		100.00%	100	100.00...		✓	auto(0)		
CVP CovCode::cp_cmd		100.00%	100	100.00...		✓			
CVP CovCode::cp_ssn		100.00%	100	100.00...		✓			
CVP CovCode::cp_mosi		100.00%	100	100.00...		✓			
INST \/cov_pkg::slave_coverage::CovCode		100.00%	100	100.00...		✓		0	
CVP cp_cmd		100.00%	100	100.00...		✓			
bin WRITE_ADD		3143	1	100.00...		✓			
bin WRITE_DATA		1728	1	100.00...		✓			
bin READ_ADD		2713	1	100.00...		✓			
bin READ_DATA		2416	1	100.00...		✓			
CVP cp_ssn		100.00%	100	100.00...		✓			
bin normal_trans		449	1	100.00...		✓			
bin read_trans		154	1	100.00...		✓			
CVP cp_mosi		100.00%	100	100.00...		✓			
bin write_addr		1654	1	100.00...		✓			
bin write_data		1118	1	100.00...		✓			
bin read_addr		1220	1	100.00...		✓			
bin read_data		1920	1	100.00...		✓			

Functional coverage report:

```
    covered/total bins:                                10         10         -
    missing/total bins:                                 0         10         -
    % Hit:                                         100.00%        100         -
    Coverpoint cp_cmd                              100.00%        100         -    Covered
        covered/total bins:                             4          4         -
        missing/total bins:                             0          4         -
        % Hit:                                     100.00%        100         -
    Coverpoint cp_ssn                              100.00%        100         -    Covered
        covered/total bins:                             2          2         -
        missing/total bins:                             0          2         -
        % 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         -
 Covergroup instance \/cov_pkg::slave_coverage::CovCode
                                                   100.00%        100         -    Covered
    covered/total bins:                                10         10         -
    missing/total bins:                                 0         10         -
    % Hit:                                         100.00%        100         -
    Coverpoint cp_cmd                              100.00%        100         -    Covered
        covered/total bins:                             4          4         -
        missing/total bins:                             0          4         -
        % Hit:                                     100.00%        100         -
        bin WRITE_ADD                                3143          1         -    Covered
        bin WRITE_DATA                               1728          1         -    Covered
        bin READ_ADD                                 2713          1         -    Covered
        bin READ_DATA                                2416          1         -    Covered
    Coverpoint cp_ssn                              100.00%        100         -    Covered
        covered/total bins:                             2          2         -
        missing/total bins:                             0          2         -
        % Hit:                                     100.00%        100         -
        bin normal_trans                             449          1         -    Covered
        bin read_trans                               154          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_addr                               1654          1         -    Covered
        bin write_data                               1118          1         -    Covered
        bin read_addr                                1220          1         -    Covered
        bin read_data                                1920          1         -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

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

```
Coverage Report by instance with details

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

Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses         3        na        na        na
            Covergroup Bins        10        10         0   100.00%
--------------------------------------------------------------------------------------------------------------
Covergroup                                             Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /cov_pkg/slave_coverage/CovCode                 100.00%        100          -    Covered
    covered/total bins:                                    10         10          -
    missing/total bins:                                     0         10          -
    % Hit:                                             100.00%        100          -
    Coverpoint cp_cmd                                  100.00%        100          -    Covered
        covered/total bins:                                 4          4          -
        missing/total bins:                                 0          4          -
        % Hit:                                         100.00%        100          -
    Coverpoint cp_ssn                                  100.00%        100          -    Covered
        covered/total bins:                                 2          2          -
        missing/total bins:                                 0          2          -
        % 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          -
 Covergroup instance \/cov_pkg::slave_coverage::CovCode
                                                       100.00%        100          -    Covered
    covered/total bins:                                    10         10          -
    missing/total bins:                                     0         10          -
    % Hit:                                             100.00%        100          -
    Coverpoint cp_cmd                                  100.00%        100          -    Covered
        covered/total bins:                                 4          4          -
        missing/total bins:                                 0          4          -
        % Hit:                                         100.00%        100          -
        bin WRITE_ADD                                    3143          1          -    Covered
        bin WRITE_DATA                                   1728          1          -    Covered
        bin READ_ADD                                     2713          1          -    Covered
        bin READ_DATA                                    2416          1          -    Covered
    Coverpoint cp_ssn                                  100.00%        100          -    Covered
        covered/total bins:                                 2          2          -
        missing/total bins:                                 0          2          -
        % Hit:                                         100.00%        100          -
```

Assertions table:

Feature	Assertion Name	Property Description
Reset Assertion	`rst_assert`	When reset is asserted, MISO, rx_valid, and rx_data should be low
Command Sequence Validation	`tx_valid_1_assert`	After valid command sequence, rx_valid asserts after 10 cycles and SS_n eventually goes high
FSM: IDLE → CHK_CMD	`trans1_assert`	When in IDLE state and SS_n is low, next state should be CHK_CMD
FSM: CHK_CMD → WRITE/READ_ADD/READ_DATA	`trans2_assert`	When in CHK_CMD state and SS_n is low, next state should be WRITE, READ_ADD, or READ_DATA
FSM: WRITE → IDLE	`trans3_assert`	When in WRITE state and SS_n goes high, next state should be IDLE
FSM: READ_ADD → IDLE	`trans4_assert`	When in READ_ADD state and SS_n goes high, next state should be IDLE
FSM: READ_DATA → IDLE	`trans5_assert`	When in READ_DATA state and SS_n goes high, next state should be IDLE

Wrapper:

Design:

```
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);

endmodule
```

Golden model:

```
// SPI_Wrapper

module SPI_Wrapper(MOSI ,MISO ,SS_n ,clk ,rst_n);
    input MOSI ,SS_n ,clk ,rst_n;
    output MISO;

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

    SPI_Slave_gold DUT (MOSI ,MISO ,SS_n ,clk ,rst_n ,rx_data ,rx_valid ,tx_data ,tx_valid);
    RAM_gold dut (rx_data , rx_valid , tx_valid , clk , rst_n , tx_data);


endmodule
```

SVA:

```
module  wrapper_sva (
    input  clk,
    input  MOSI,
    input  SS_n,
    input  rst_n,
    input  MISO

);

   // rst Assertion
    rst_assert:assert        property  (@(posedge clk) !rst_n |=> (!MISO));
    rst_cover:cover          property  (@(posedge clk) !rst_n |=> (!MISO));

    sequence seq;
    $fell(SS_n) ##1 (MOSI [*3]) ##1 $rose(SS_n);
    endsequence

    property MISO_prop;
        @(posedge clk) disable iff (!rst_n)
            $fell(SS_n) |=>
                (not seq ##1 ($stable(MISO) throughout (!SS_n)) );
    endproperty

  assert property (MISO_prop);
  cover property (MISO_prop);



endmodule
```

Wrraper Config db:

```
1  package wrapper_config;
2      import uvm_pkg::*;
3      `include "uvm_macros.svh"
4
5      class wrapper_config_obj extends uvm_object;
6          `uvm_object_utils(wrapper_config_obj);
7
8          virtual wrapper_if wrapper_config_vif;
9          uvm_active_passive_enum is_active_wrapper;
10
11         function new(string name = "wrapper_config_obj");
12             super.new(name);
13         endfunction
14
15     endclass
16
17
18 endpackage
```

Ram config db:

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

    class ram_config_obj extends uvm_object;
        `uvm_object_utils(ram_config_obj);

        virtual ram_if ram_config_vif;
        uvm_active_passive_enum is_active_ram;

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

    endclass

endpackage
```

slave config db:

```
1  package slave_config;
2      import uvm_pkg::*;
3      `include "uvm_macros.svh"
4
5      class slave_config_obj extends uvm_object;
6          `uvm_object_utils(slave_config_obj);
7
8          virtual slave_if slave_config_vif;
9          uvm_active_passive_enum is_active_slave;
10
11         function new(string name = "slave_config_obj");
12             super.new(name);
13         endfunction
14
15     endclass
16
17
18 endpackage
```

Seq item:

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

    class wrapper_seq_item extends uvm_sequence_item;
      `uvm_object_utils(wrapper_seq_item);

      rand bit  rst_n, SS_n, MOSI;
      bit MISO, MISO_ref;
      rand bit[10:0] MOSI_bins;

      int counter,i;
      bit WRITE_ADD, READ_ADD, WRITE_DATA, READ_DATA;

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

    constraint rst_constraint {rst_n dist {1 := 999 , 0 := 1};}

        // ss_n constraint
        constraint SS_N_constraint {
            if (~rst_n)
                SS_n == 1;
            else if (MOSI_bins[10:8] == 3'b111) {
                if(counter % 24 == 0)
                    SS_n == 1;
                else
                    SS_n == 0;
            }
            else {
                if (counter % 14 == 0)
                    SS_n == 1;
                else
                    SS_n == 0;
            }
        }
```

```

      constraint write_constraint {
      if (WRITE_ADD)
          MOSI_bins [10:8] inside {3'b000 , 3'b001};
      }

      constraint read_constraint {
      MOSI_bins [10:8] inside {3'b111 , 3'b110};
      if (READ_ADD)
          MOSI_bins [10:8] == 3'b111;
      else if (READ_DATA)
          MOSI_bins [10:8] == 3'b110;
       }

      constraint write_read_constraint {
          if (WRITE_ADD)
              MOSI_bins [10:8] inside {3'b000 , 3'b001};

          else if (WRITE_DATA)
             MOSI_bins [10:8] dist {3'b000:=40 , 3'b110:=60};

          else if (READ_ADD)
             MOSI_bins [10:8] == {3'b111};

          else if (READ_DATA)
              MOSI_bins [10:8] dist {3'b110:=40 , 3'b000:=60};
      }

```

```
67          function void post_randomize;
68                  counter ++;
69                  if(SS_n) begin
70                          counter = 1;
71                          i = 10;
72                  end
73                  else if (i>=0 && counter >2) begin
74                      MOSI = MOSI_bins[i];
75                      i --;
76                  end
77                  foreach (MOSI_bins[i]) begin
78                      $display("mosi arr = %b", MOSI_bins);
79                  end
80
81                  if (MOSI_bins [10:8] == 3'b000)
82                      WRITE_ADD = 1'b1;
83                  else
84                      WRITE_ADD = 1'b0;
85
86                  if (MOSI_bins [10:8] == 3'b110)
87                      READ_ADD = 1'b1;
88                  else
89                      READ_ADD = 1'b0;
90
91                  if (MOSI_bins [10:8] == 3'b001)
92                      WRITE_DATA = 1'b1;
93                  else
94                      WRITE_DATA = 1'b0;
95
96                  if (MOSI_bins [10:8] == 3'b111)
97                      READ_DATA = 1'b1;
98                  else
99                      READ_DATA = 1'b0;
100     endfunction
```

```
101
102
103     function string convert2string();
104       return $sformatf("%s MOSI = %0d, SS_n = %0d, rst_n = %0d , MISO = %0d , MISO_ref = %0d , MOSI_bins = %b",
105       super.convert2string(), MOSI, SS_n, rst_n, MISO, MISO_ref, MOSI_bins);
106     endfunction
107
108   endclass
109
110 endpackage
```

Write only seq:

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

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

        wrapper_seq_item seq_item;

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

        task body();
            seq_item = wrapper_seq_item::type_id::create("seq_item");
            // write sequence
            seq_item.read_constraint.constraint_mode(0);
            seq_item.read_constraint.constraint_mode(0);
            repeat (1000) begin
                start_item(seq_item);
                if (seq_item.SS_n) begin
                    seq_item.MOSI_bins.rand_mode(1);
                    seq_item.write_constraint.constraint_mode(1);
                end
                else begin
                    seq_item.MOSI_bins.rand_mode(0);
                    seq_item.write_constraint.constraint_mode(0);
                    assert(seq_item.randomize() with {seq_item.MOSI_bins[10:8] inside {3'b000 , 3'b001};});
                end

                finish_item(seq_item);
            end
        endtask

  endclass

endpackage
```

Read only seq:

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

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

        wrapper_seq_item seq_item;

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

        task body();

       // read sequence
       seq_item = wrapper_seq_item :: type_id :: create ("seq_item");
            seq_item.write_constraint.constraint_mode(0);
            seq_item.write_read_constraint.constraint_mode(0);
            repeat (1000) begin
                start_item(seq_item);
                if (seq_item.SS_n) begin
                    seq_item.read_constraint.constraint_mode(1);
                    seq_item.MOSI_bins.rand_mode(1);
                end
                else begin
                    seq_item.read_constraint.constraint_mode(0);
                    seq_item.MOSI_bins.rand_mode(0);
                end
                assert(seq_item.randomize());
                finish_item(seq_item);
            end

    endtask

  endclass

endpackage
```

Write Read seq:

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

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

        wrapper_seq_item seq_item;

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

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

      // read and write
            seq_item.write_constraint.constraint_mode(0);
            seq_item.read_constraint.constraint_mode(0);
            repeat (1000) begin
                start_item(seq_item);
                if (seq_item.SS_n) begin
                    seq_item.write_read_constraint.constraint_mode(1);
                    seq_item.MOSI_bins.rand_mode(1);
                end
                else begin
                    seq_item.write_read_constraint.constraint_mode(0);
                    seq_item.MOSI_bins.rand_mode(0);
                end
                assert(seq_item.randomize());
                finish_item(seq_item);
      end
    endtask

  endclass

endpackage
```

Sqr:

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

```
package wrapper_driver;
    import wrapper_config::*;
    import 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 stim_seq_item;

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

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

            forever begin
                stim_seq_item = wrapper_seq_item :: type_id :: create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);
                wrapper_vif.SS_n        = stim_seq_item.SS_n;
                wrapper_vif.MOSI        = stim_seq_item.MOSI;
                wrapper_vif.rst_n       = stim_seq_item.rst_n;
                @(negedge wrapper_vif.clk);
                seq_item_port.item_done();
            end

        endtask

    endclass

endpackage
```

Monitor:

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

    class wrapper_monitor extends uvm_monitor;
        `uvm_component_utils(wrapper_monitor);

        virtual wrapper_if wrapper_vif;
        wrapper_seq_item rsp_seq_item;
        uvm_analysis_port #(wrapper_seq_item) mon_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = wrapper_seq_item :: type_id :: create("rsp_seq_item");
                rsp_seq_item.MOSI      = wrapper_vif.MOSI;
                rsp_seq_item.SS_n      = wrapper_vif.SS_n;
                rsp_seq_item.rst_n     = wrapper_vif.rst_n;
                rsp_seq_item.MISO      = wrapper_vif.MISO;
                rsp_seq_item.MISO_ref  = wrapper_vif.MISO_ref;
                @(negedge wrapper_vif.clk);
                mon_ap.write(rsp_seq_item);
            end
        endtask
    endclass
endpackage

```

Wrraper Agent:

```
package agent_pkg_wrapper;
   import uvm_pkg::*;
   `include "uvm_macros.svh"
   import sequencer_pkg::*;
   import wrapper_driver::*;
   import monitor_wrap::*;
   import wrapper_config::*;
   import seq_item_pkg::*;

    class wrapper_agent extends uvm_agent;
        `uvm_component_utils(wrapper_agent);
       wrapper_sqr sqr;
       wrapper_driver drv;
       wrapper_monitor mon;
       wrapper_config_obj wrapper_cfg;
       uvm_analysis_port #(wrapper_seq_item) agt_ap;


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

       function void build_phase(uvm_phase phase);
           super.build_phase(phase);
           if(!uvm_config_db #(wrapper_config_obj) :: get(this, "", "CFG_wrap", wrapper_cfg)) begin
               `uvm_fatal("build_phase","Unable to get configration object");
           end

           mon = wrapper_monitor :: type_id :: create("mon",this);

           if(wrapper_cfg.is_active_wrapper == UVM_ACTIVE) begin
               sqr = wrapper_sqr     :: type_id :: create("sqr",this);
               drv = wrapper_driver  :: type_id :: create("drv",this);
           end

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

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

             mon.wrapper_vif = wrapper_cfg.wrapper_config_vif;
             mon.mon_ap.connect(agt_ap);

             if(wrapper_cfg.is_active_wrapper == UVM_ACTIVE) begin
                 drv.wrapper_vif = wrapper_cfg.wrapper_config_vif;
                 drv.seq_item_port.connect(sqr.seq_item_export);
             end

         endfunction

     endclass
endpackage
```

Ram Agent:

```
package agent_pkg_ram;
    import uvm_pkg::*;
   `include "uvm_macros.svh"
   import sequencer_pkg_ram::*;
   import ram_driver::*;
   import monitor_ram::*;
   import ram_config::*;
   import seq_item_pkg_ram::*;

    class ram_agent extends uvm_agent;
        `uvm_component_utils(ram_agent);
        ram_sqr sqr;
        ram_driver drv;
        ram_monitor mon;
        ram_config_obj ram_cfg;
        uvm_analysis_port #(ram_seq_item) agt_ap;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            if(!uvm_config_db #(ram_config_obj) :: get(this, "", "CFG_ram", ram_cfg)) begin
                `uvm_fatal("build_phase","Unable to get configration object");
            end

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

            if(ram_cfg.is_active_ram == UVM_ACTIVE) begin
                sqr = ram_sqr     :: type_id :: create("sqr",this);
                drv = ram_driver  :: type_id :: create("drv",this);
            end

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

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

            mon.ram_vif = ram_cfg.ram_config_vif;
            mon.mon_ap.connect(agt_ap);

            if(ram_cfg.is_active_ram == UVM_ACTIVE) begin
                drv.ram_vif = ram_cfg.ram_config_vif;
                drv.seq_item_port.connect(sqr.seq_item_export);
            end

        endfunction

    endclass
endpackage
```

Slave Agent:

```
package agent_pkg_slave;
   import uvm_pkg::*;
   `include "uvm_macros.svh"
   import sequencer_pkg_slave::*;
   import slave_driver::*;
   import monitor_slave::*;
   import slave_config::*;
   import seq_item_pkg_slave::*;

    class slave_agent extends uvm_agent;
        `uvm_component_utils(slave_agent);
       slave_sqr sqr;
       slave_driver drv;
       slave_monitor mon;
       slave_config_obj slave_cfg;
       uvm_analysis_port #(slave_seq_item) agt_ap;

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

       function void build_phase(uvm_phase phase);
           super.build_phase(phase);
           if(!uvm_config_db #(slave_config_obj) :: get(this, "", "CFG_slave", slave_cfg)) begin
               `uvm_fatal("build_phase","Unable to get configration object");
           end

           mon = slave_monitor :: type_id :: create("mon",this);

           if(slave_cfg.is_active_slave == UVM_ACTIVE) begin
               sqr = slave_sqr     :: type_id :: create("sqr",this);
               drv = slave_driver  :: type_id :: create("drv",this);
           end

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

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

           mon.slave_vif = slave_cfg.slave_config_vif;
           mon.mon_ap.connect(agt_ap);

           if(slave_cfg.is_active_slave == UVM_ACTIVE) begin
               drv.slave_vif = slave_cfg.slave_config_vif;
               drv.seq_item_port.connect(sqr.seq_item_export);
           end

       endfunction

   endclass
endpackage
```

Coverage collector:

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

    class wrapper_config_obj extends uvm_object;
        `uvm_object_utils(wrapper_config_obj);

        virtual wrapper_if wrapper_config_vif;
        uvm_active_passive_enum is_active_wrapper;

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

    endclass


endpackage
```

Scoreboard:

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

    class wrapper_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(wrapper_scoreboard);
        uvm_analysis_export #(wrapper_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(wrapper_seq_item) sb_fifo;
        wrapper_seq_item seq_item_sb;


        int error_count   = 0;
        int correct_count = 0;

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

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                if(seq_item_sb.MISO != seq_item_sb.MISO_ref) begin
                    `uvm_error ("run_phase",$sformatf("comparsion failed, transaction received by the DUT:%s while the reference out_ref:0b%0b",seq_item_sb.convert2string(),
                                seq_item_sb.MISO_ref));
                    error_count++;
                end
                else begin
                    `uvm_info("run_phase",$sformatf("correct out_ref outputs : %s ", seq_item_sb.convert2string()), UVM_HIGH);
                    correct_count++;
                end

            end

        endtask

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase",$sformatf(" Successful transactions of wrapper : %d ", correct_count), UVM_MEDIUM);
            `uvm_info("report_phase",$sformatf(" failed transactions of wrapper     : %d ", error_count), UVM_MEDIUM);
        endfunction
    endclass

endpackage
```

Env:

```
package wrapper_env;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import agent_pkg_wrapper::*;
    import scoreboard_pkg_wrap::*;
    import cov_pkg_wrapper::*;

    class wrapper_env extends uvm_env;
        `uvm_component_utils(wrapper_env);

        wrapper_agent agt;
        wrapper_scoreboard sb;
        wrapper_coverage cov;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            agt = wrapper_agent      :: type_id :: create ("agt",this);
            sb  = wrapper_scoreboard :: type_id :: create ("sb",this);
            cov = wrapper_coverage   :: type_id :: create ("cov",this);
        endfunction

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

    endclass

endpackage
```

Test:

```
package wrapper_test;
    import wrapper_env::*;
    import wrapper_config::*;
    import write_sequence::*;
    import read_sequence::*;
    import Write_Read_sequence::*;
    import rst_sequence::*;
    import uvm_pkg::*;
    import slave_env::*;
    import ram_env::*;
    import ram_config::*;
    import slave_config::*;
    `include "uvm_macros.svh"

    class wrapper_test extends uvm_test;
        `uvm_component_utils(wrapper_test);

        wrapper_env env_act;
        slave_env env_pass_slave;
        ram_env env_pass_ram;

        wrapper_config_obj wrapper_config_obj_test;
        slave_config_obj slave_cfg;
        ram_config_obj ram_cfg;

        wrapper_write_seq write_only_seq;
        wrapper_read_seq read_only_seq;
        wrraper_Write_Read_seq read_write_seq;
        wrapper_rst_seq rst_seq;


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

```

```
36
37      function void build_phase(uvm_phase phase);
38          super.build_phase(phase);
39          env_act                 = wrapper_env        :: type_id :: create("env_act",this);
40          wrapper_config_obj_test = wrapper_config_obj :: type_id :: create("wrapper_config_obj_test");
41
42          env_pass_slave          = slave_env          :: type_id :: create ("env_pass_slave",this);
43          slave_cfg               = slave_config_obj   :: type_id :: create ("slave_cfg");
44
45          env_pass_ram            = ram_env            :: type_id :: create ("env_pass_ram",this);
46          ram_cfg                 = ram_config_obj     :: type_id :: create ("ram_cfg");
47
48          write_only_seq          = wrapper_write_seq        :: type_id :: create ("write_only_seq");
49          read_only_seq           = wrapper_read_seq         :: type_id :: create ("read_only_seq");
50          read_write_seq          = wrraper_Write_Read_seq   :: type_id :: create ("read_write_seq");
51          rst_seq                 = wrapper_rst_seq          :: type_id :: create ("rst_seq");
52
53          wrapper_config_obj_test.is_active_wrapper = UVM_ACTIVE;
54          slave_cfg.is_active_slave                 = UVM_PASSIVE;
55          ram_cfg.is_active_ram                     = UVM_PASSIVE;
56
57          if(!uvm_config_db #(virtual wrapper_if) :: get(this, "", "Wrapper_IF", wrapper_config_obj_test.wrapper_config_vif))
58              `uvm_fatal("build_phase", " test -  unable to get the first virtual interface");
59
60          if(!uvm_config_db #(virtual slave_if)   :: get(this,"","Slave_IF",slave_cfg.slave_config_vif))
61              `uvm_fatal("build_phase","test - unable to get the second virtul interface");
62
63          if(!uvm_config_db #(virtual ram_if)     :: get(this,"","Ram_IF",ram_cfg.ram_config_vif))
64              `uvm_fatal("build_phase","test - unable to get the third virtul interface");
65
66
67          uvm_config_db #(wrapper_config_obj) :: set(this,"env_act.*","CFG_wrap", wrapper_config_obj_test);
68          uvm_config_db #(slave_config_obj)   :: set(this,"env_pass_slave.*","CFG_slave",slave_cfg);
69          uvm_config_db #(ram_config_obj)     :: set(this,"env_pass_ram.*","CFG_ram",ram_cfg);
70
71      endfunction
```

```
72
73          task run_phase(uvm_phase phase);
74              super.run_phase(phase);
75
76              phase.raise_objection(this);
77
78              `uvm_info("run_phase","assert rst ",UVM_LOW);
79              rst_seq.start(env_act.agt.sqr);
80              `uvm_info("run_phase","deassert rst ",UVM_LOW);
81
82              `uvm_info("run_phase","write operation started",UVM_LOW);
83              write_only_seq.start(env_act.agt.sqr);
84              `uvm_info("run_phase","write operation ended",UVM_LOW);
85
86              `uvm_info("run_phase","read operation started",UVM_LOW);
87              read_only_seq.start(env_act.agt.sqr);
88              `uvm_info("run_phase","read operation ended",UVM_LOW);
89
90              `uvm_info("run_phase","write_read operation started",UVM_LOW);
91              read_write_seq.start(env_act.agt.sqr);
92              `uvm_info("run_phase","write_read operation ended",UVM_LOW);
93
94              phase.drop_objection(this);
95
96          endtask
97
98      endclass
99
100 endpackage
```

Top:

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

module top ();
    bit clk;

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

    wrapper_if wrapif (clk);
    slave_if slaveif (clk);
    ram_if ramif (clk);

    WRAPPER DUT_1 (wrapif.MOSI,wrapif.MISO,wrapif.SS_n,clk,wrapif.rst_n);
    SPI_Wrapper golden_1 (wrapif.MOSI ,wrapif.MISO_ref ,wrapif.SS_n ,clk ,wrapif.rst_n);
```

```
        assign ramif.rx_valid= DUT_1.rx_valid;
        assign ramif.din = DUT_1.rx_data_din;
        assign ramif.tx_valid= DUT_1.tx_valid;
        assign ramif.dout = DUT_1.tx_data_dout;
        assign ramif.rst_n    = DUT_1.rst_n;

        assign ramif.tx_valid_ref= golden_1.tx_valid;
        assign ramif.dout_ref = golden_1.tx_data;


        assign slaveif.MOSI= DUT_1.MOSI;
        assign slaveif.MISO = DUT_1.MISO;
        assign slaveif.tx_valid= DUT_1.tx_valid;
        assign slaveif.tx_data = DUT_1.tx_data_dout;
        assign slaveif.rx_data    = DUT_1.rx_data_din;
        assign slaveif.rx_valid    = DUT_1.rx_valid;
        assign slaveif.rst_n    = DUT_1.rst_n;
        assign slaveif.SS_n    = DUT_1.SS_n;

        assign slaveif.rx_data_ref= golden_1.rx_data;
        assign slaveif.rx_valid_ref = golden_1.rx_valid;
        assign slaveif.MISO_ref = golden_1.MISO;
```

```
    bind WRAPPER wrapper_sva checker1_inst (.*);
    bind SLAVE slave_sva checker2_inst (.*);
    bind RAM ram_sva checker3_inst (.*);

    initial begin
        $readmemh("mem.dat",DUT_1.RAM_instance.MEM);
        $readmemh("mem.dat",golden_1.dut.mem);
        uvm_config_db #(virtual wrapper_if) :: set(null,"uvm_test_top","Wrapper_IF",wrapif);
        uvm_config_db #(virtual slave_if)   :: set(null,"uvm_test_top","Slave_IF",slaveif);
        uvm_config_db #(virtual ram_if)     :: set(null,"uvm_test_top","Ram_IF",ramif);

        run_test("wrapper_test");
    end


endmodule
```

Interface:

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

endinterface : wrapper_if
```

Do file:

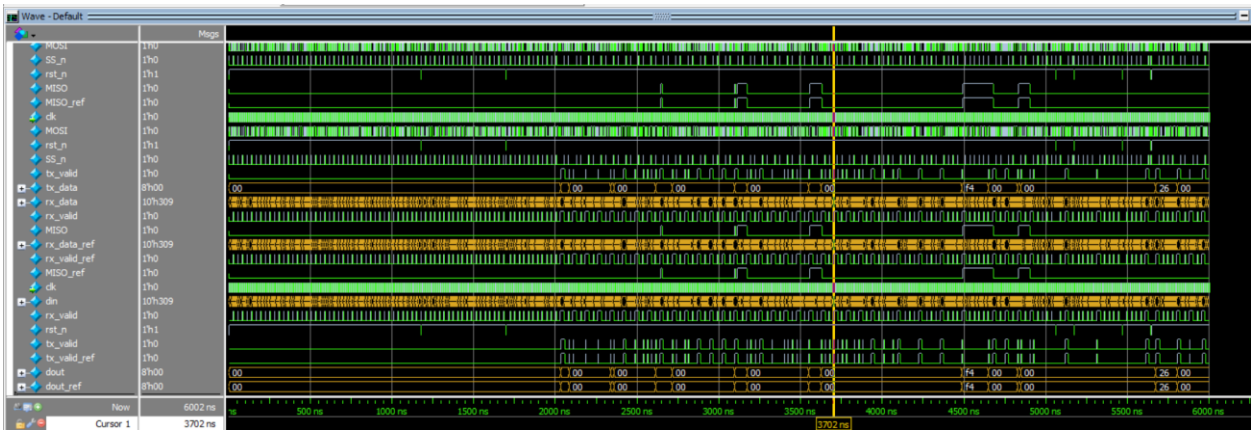
```
vlib work
vlog -f src_files.list +define+SIM +cover -covercells
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover
add wave /top/wrapif/*
add wave -position insertpoint sim:/top/slaveif/*
add wave -position insertpoint sim:/top/ramif/*
run -all
coverage save SPI_Wrapper_coverage.ucdb -du WRAPPER

```

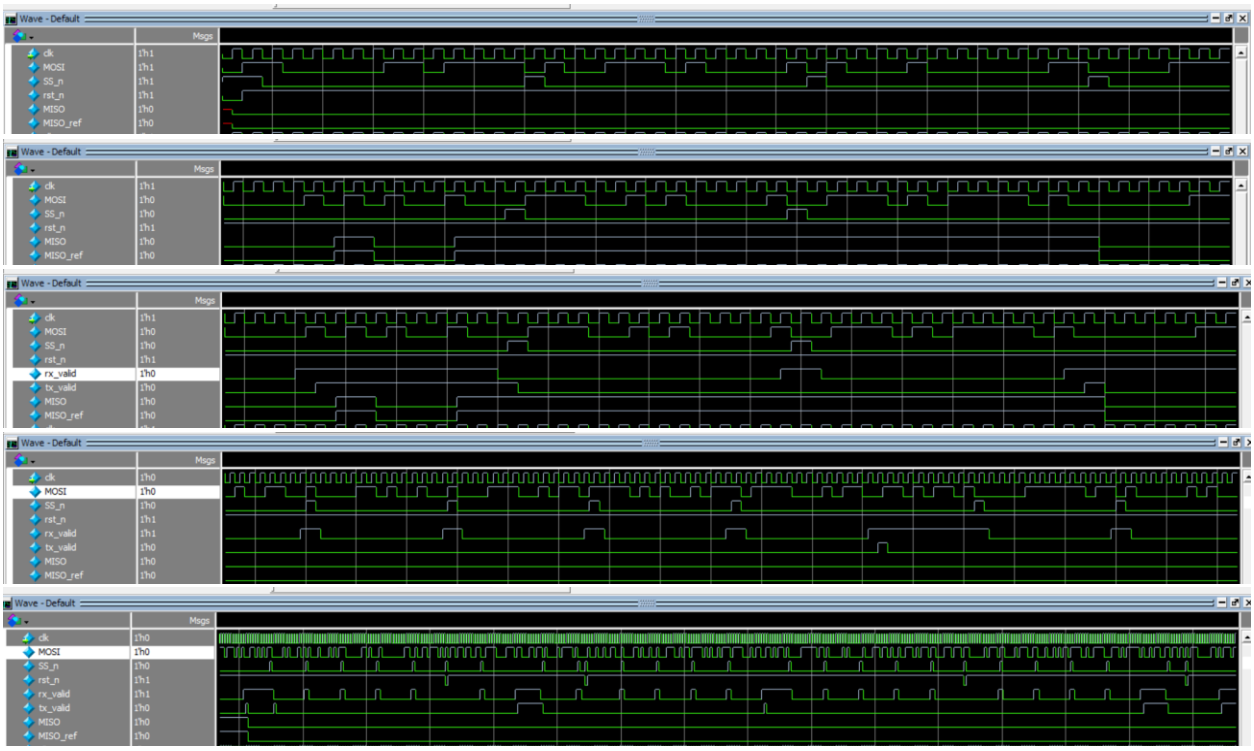
Src files:

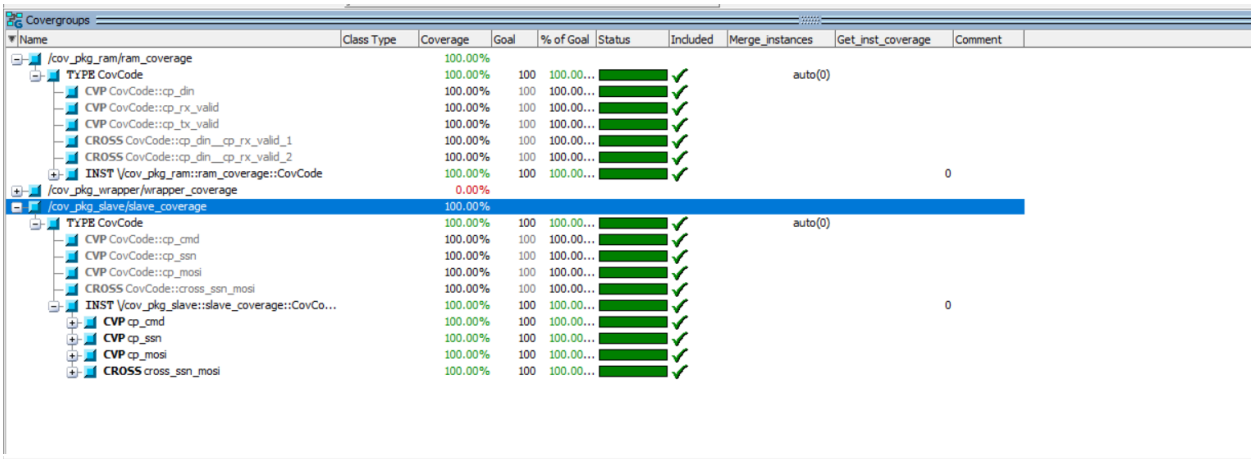
```
RAM_if.sv
RAM.v
RAM_SVA.sv
RAM_config.sv
RAM_seq_item.sv
Ram_rst_seq_pkg.sv
ram_write_seq_pkg.sv
Ram_Read_Seq.sv
ram_Write_Read_Seq.sv
Ram_sequencer_pkg.sv
RAM_driver.sv
Ram_Monitor.sv
Ram_Agent.sv
Ram_coverage_collector_pkg.sv
Ram_scoreboard_pkg.sv
RAM_env.sv

Slave_if.sv
SPI_slave.sv
Slave_SVA.sv
Slave_shared_pkg.sv
Slave_config.sv
Slave_seq_item.sv
Slave_rst_seq_pkg.sv
Slave_main_seq_pkg.sv
Slave_sequencer_pkg.sv
Slave_driver.sv
Slave_Monitor.sv
Slave_Agent.sv
Slave_coverage_collector_pkg.sv
Slave_scoreboard_pkg.sv
Slave_env.sv

Wrapper_if.sv
SPI_wrapper.v
SPI_Wrapper_golden.v
Wrapper_SVA.sv
Wrapper_shared_pkg.sv
Wrapper_config.sv
Wrapper_seq_item.sv
Wrapper_rst_seq_pkg.sv
Wrapper_write_seq_pkg.sv
Wrapper_Read_Seq.sv
Wrapper_Write_Read_Seq.sv
Wrapper_sequencer_pkg.sv
Wrapper_driver.sv
Wrapper_Monitor.sv
Wrapper_Agent.sv
Wrapper_coverage_collector_pkg.sv
Wrapper_scoreboard_pkg.sv
Wrapper_env.sv
Wrapper_test.sv
Wrapper_top.sv
```

Simulation:

```
Transcript
# mosi arr = 00010000011
# mosi arr = 00010000011
# mosi arr = 00010000011
# mosi arr = 00010000011
# mosi arr = 00010000011
# mosi arr = 00010000011
# mosi arr = 00010000011
# UVM_INFO Wrapper_test.sv(92) @ 6002: uvm_test_top [run_phase] write_read operation ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 6002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO Wrapper_scoreboard_pkg.sv(52) @ 6002: uvm_test_top.env_act.sb [report_phase]  Successful transactions of wrapper :        3001
# UVM_INFO Wrapper_scoreboard_pkg.sv(53) @ 6002: uvm_test_top.env_act.sb [report_phase]  failed transactions of wrapper     :           0
# UVM_INFO Ram_scoreboard_pkg.sv(51) @ 6002: uvm_test_top.env_pass_ram.sb [report_phase]  Successful transactions of RAM :        3001
# UVM_INFO Ram_scoreboard_pkg.sv(52) @ 6002: uvm_test_top.env_pass_ram.sb [report_phase]  failed transactions of RAM     :           0
# UVM_INFO Slave_scoreboard_pkg.sv(51) @ 6002: uvm_test_top.env_pass_slave.sb [report_phase]  Successful transactions of Slave :        3001
# UVM_INFO Slave_scoreboard_pkg.sv(52) @ 6002: uvm_test_top.env_pass_slave.sb [report_phase]  failed transactions of Slave     :           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   18
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     6
# [run_phase]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 6002 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```




Wave - Default
Msgs
clk
MOSI
SS_n
rst_n
MISO
MISO_ref
rx_valid
tx_valid



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/cov_pkg_ram/ram_coverage		100.00%							
TYPE CovCode		100.00%	100	100.00...			auto(0)		
CVP CovCode::cp_din		100.00%	100	100.00...					
CVP CovCode::cp_rx_valid		100.00%	100	100.00...					
CVP CovCode::cp_tx_valid		100.00%	100	100.00...					
CROSS CovCode::cp_din__cp_rx_valid_1		100.00%	100	100.00...					
CROSS CovCode::cp_din__cp_rx_valid_2		100.00%	100	100.00...					
INST \/cov_pkg_ram::ram_coverage::CovCode		100.00%	100	100.00...					0
/cov_pkg_wrapper/wrapper_coverage		0.00%							
/cov_pkg_slave/slave_coverage		100.00%							
TYPE CovCode		100.00%	100	100.00...			auto(0)		
CVP CovCode::cp_cmd		100.00%	100	100.00...					
CVP CovCode::cp_ssn		100.00%	100	100.00...					
CVP CovCode::cp_mosi		100.00%	100	100.00...					
CROSS CovCode::cross_ssn_mosi		100.00%	100	100.00...					
INST \/cov_pkg_slave::slave_coverage::CovCo...		100.00%	100	100.00...					0
CVP cp_cmd		100.00%	100	100.00...					
CVP cp_ssn		100.00%	100	100.00...					
CVP cp_mosi		100.00%	100	100.00...					
CROSS cross_ssn_mosi		100.00%	100	100.00...					

fcover_report.txt

```
Coverage Report by instance with details

================================================================================
=== Instance: /top/DUT_1/RAM_instance/checker3_inst
=== Design Unit: work.ram_sva
================================================================================

Directive Coverage:
    Directives                       5         5         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                 Design Design   Lang File(Line)      Hits Status
                                     Unit   UnitType
--------------------------------------------------------------------------------------------
/top/DUT_1/RAM_instance/checker3_inst/rst_cover
                                     ram_sva Verilog  SVA  RAM_SVA.sv(21)      7 Covered
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_1_cover
                                     ram_sva Verilog  SVA  RAM_SVA.sv(25)   2335 Covered
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_2_cover
                                     ram_sva Verilog  SVA  RAM_SVA.sv(31)     64 Covered
/top/DUT_1/RAM_instance/checker3_inst/write_cover
                                     ram_sva Verilog  SVA  RAM_SVA.sv(35)    106 Covered
/top/DUT_1/RAM_instance/checker3_inst/read_cover
                                     ram_sva Verilog  SVA  RAM_SVA.sv(39)    289 Covered

================================================================================
=== Instance: /top/DUT_1/SLAVE_instance/checker2_inst
=== Design Unit: work.slave_sva
================================================================================

Directive Coverage:
    Directives                       2         2         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                 Design Design   Lang File(Line)      Hits Status
                                     Unit   UnitType
--------------------------------------------------------------------------------------------
/top/DUT_1/SLAVE_instance/checker2_inst/rst_cover
                                     slave_sva Verilog  SVA  Slave_SVA.sv(20)    7 Covered
/top/DUT_1/SLAVE_instance/checker2_inst/tx_valid_1_cover
                                     slave_sva Verilog  SVA  Slave_SVA.sv(24)1485 Covered

================================================================================
=== Instance: /top/DUT_1/SLAVE_instance
=== Design Unit: work.SLAVE
================================================================================
```

fcover_report.txt

```
TOTAL COVERGROUP COVERAGE: 66.66%  COVERGROUP TYPES: 3

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/top/DUT_1/RAM_instance/checker3_inst/rst_cover
                                        ram_sva Verilog  SVA  RAM_SVA.sv(21)     7 Covered
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_1_cover
                                        ram_sva Verilog  SVA  RAM_SVA.sv(25)  2335 Covered
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_2_cover
                                        ram_sva Verilog  SVA  RAM_SVA.sv(31)    64 Covered
/top/DUT_1/RAM_instance/checker3_inst/write_cover
                                        ram_sva Verilog  SVA  RAM_SVA.sv(35)   106 Covered
/top/DUT_1/RAM_instance/checker3_inst/read_cover
                                        ram_sva Verilog  SVA  RAM_SVA.sv(39)   289 Covered
/top/DUT_1/SLAVE_instance/trans1_cover  SLAVE  Verilog  SVA  SPI_slave.sv(141)
                                                                            187 Covered
/top/DUT_1/SLAVE_instance/trans2_cover  SLAVE  Verilog  SVA  SPI_slave.sv(144)
                                                                            186 Covered
/top/DUT_1/SLAVE_instance/trans3_cover  SLAVE  Verilog  SVA  SPI_slave.sv(147)
                                                                             98 Covered
/top/DUT_1/SLAVE_instance/trans4_cover  SLAVE  Verilog  SVA  SPI_slave.sv(150)
                                                                             19 Covered
/top/DUT_1/SLAVE_instance/trans5_cover  SLAVE  Verilog  SVA  SPI_slave.sv(153)
                                                                             63 Covered
/top/DUT_1/SLAVE_instance/checker2_inst/rst_cover
                                        slave_sva Verilog  SVA  Slave_SVA.sv(20)   7 Covered
/top/DUT_1/SLAVE_instance/checker2_inst/tx_valid_1_cover
                                        slave_sva Verilog  SVA  Slave_SVA.sv(24)1485 Covered
/top/DUT_1/checker1_inst/cover__MISO_prop
                                        wrapper_sva Verilog  SVA  Wrapper_SVA.sv(25)
                                                                            187 Covered
/top/DUT_1/checker1_inst/rst_cover      wrapper_sva Verilog  SVA  Wrapper_SVA.sv(12)
                                                                              7 Covered

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

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))
/Write_Read_sequence::wrraper_Write_Read_seq::body/#ublk#179112213#25/immed__35	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))
/read_sequence::wrapper_read_seq::body/#ublk#71049845#25/immed__35	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))
/write_sequence::wrapper_write_seq::body/#ublk#92345829#24/immed__30	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))
/top/DUT_1/RAM_instance/checker3_inst/rst_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=>(~tx_valid&
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_1_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((d
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_2_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((dr
/top/DUT_1/RAM_instance/checker3_inst/write_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((rx
/top/DUT_1/RAM_instance/checker3_inst/read_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((rx
/top/DUT_1/SLAVE_instance/trans1_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((cs
/top/DUT_1/SLAVE_instance/trans2_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((cs
/top/DUT_1/SLAVE_instance/trans3_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((cs
/top/DUT_1/SLAVE_instance/trans4_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((cs
/top/DUT_1/SLAVE_instance/trans5_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((((cs
/top/DUT_1/SLAVE_instance/checker2_inst/rst_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=>((~MISO&&
/top/DUT_1/SLAVE_instance/checker2_inst/tx_valid_1_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) (((((
/top/DUT_1/checker1_inst/rst_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|=>~MISO)
/top/DUT_1/checker1_inst/assert__MISO_prop	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (~rst_n) ((($fe

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT_1/RAM_instance/checker3_inst/rst_cover	SVA	✓	Off	7	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_1_cover	SVA	✓	Off	2335	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/RAM_instance/checker3_inst/tx_valid_2_cover	SVA	✓	Off	64	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/RAM_instance/checker3_inst/write_cover	SVA	✓	Off	106	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/RAM_instance/checker3_inst/read_cover	SVA	✓	Off	289	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/trans1_cover	SVA	✓	Off	187	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/trans2_cover	SVA	✓	Off	186	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/trans3_cover	SVA	✓	Off	98	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/trans4_cover	SVA	✓	Off	19	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/trans5_cover	SVA	✓	Off	63	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/checker2_inst/rst_cover	SVA	✓	Off	7	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/SLAVE_instance/checker2_inst/tx_valid_1_cover	SVA	✓	Off	1485	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/checker1_inst/cover__MISO_prop	SVA	✓	Off	187	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0
/top/DUT_1/checker1_inst/rst_cover	SVA	✓	Off	7	1	Unlimited	1	100%	✓	✓	0	0	0 ns	0

Assertions table:

Feature	Assertion Name	Property Description
Reset Assertion	`rst_assert`	When reset is asserted, MISO should be low in the next cycle
MISO Stability	`MISO_prop`	After SS_n falls, MISO should remain stable throughout the SPI transaction until SS_n rises

---

BY : MOHAMED NASSER
