Benichips Team

By

Ahmed Farag Rabie

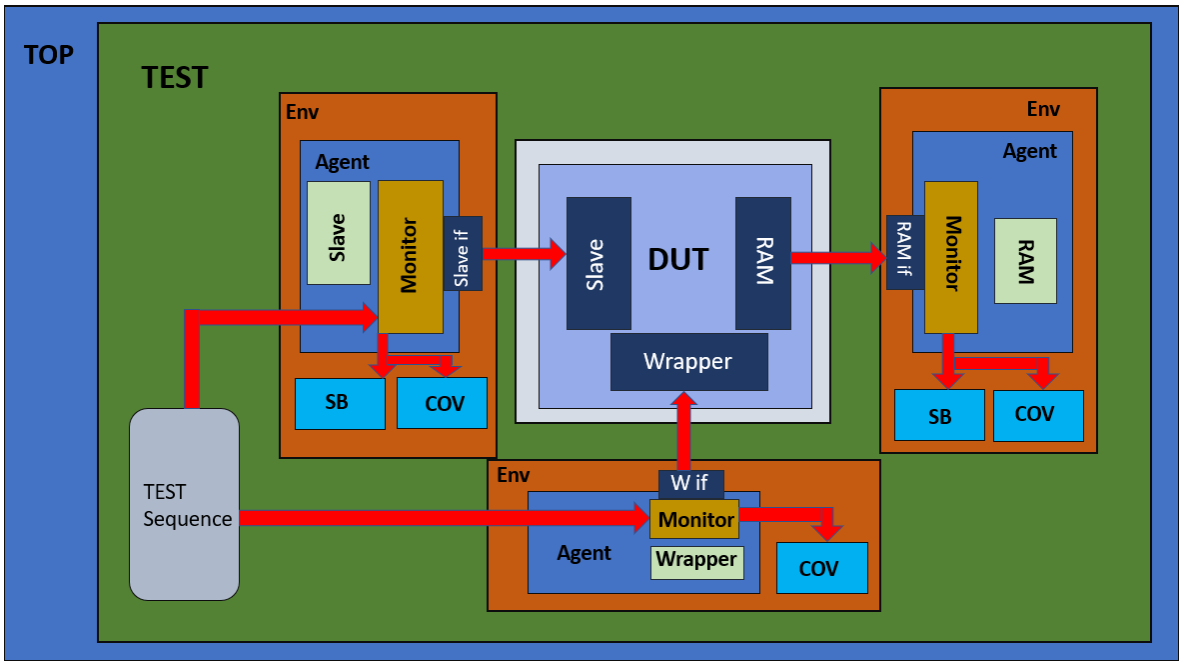
Mostafa Ayman Rabie

Youssef Ahmed Ebrahim

1-Verification Plan

Functionality Check	Functional Coverage	Stimulus Generation	Design Requirement Description	Label
SVA: Assertion check for all .relevant outputs at **rst_n = 0**	-	reset_sequence applied at test start; enforces rst_n is deasserted most of the time .	Verify that all DUT outputs (in SPI Slave, RAM, and Wrapper) assert to their low/inactive default state upon rst_n .assertion	**SEQ_RST**
SVA: Assertion check: Every Write Address must be *eventually* followed by a Write Data. Scoreboard confirms data written .correctly to RAM	Functional Coverage: Coverpoint on sequential ordering: Write Address Write Data (or Write .(Address	write_only_sequence enforces: Write Address shall always be followed by either Write Address or .Write Data operation	Verify Write Address Write Data sequential flow for RAM and .Wrapper	**SEQ_WR**
SVA: Assertion check: Every Read Address must be *eventually* followed by a .Read Data	Functional Coverage: Coverpoint on sequential ordering: Read Address Read Data (or Read .(Address	read_only_sequence enforces: Read Address shall always be followed by either Read Address or .Read Data operation	Verify Read Address Read Data sequential flow for RAM and .Wrapper	**SEQ_RD**
Scoreboard: Compares all Read Data against expected data stored in the model, verifying end-to-end data integrity.".	Cross Coverage: Between all operation types (din[9:8]) and rx_valid (when high). All sequential .coverages are exercised	write_read_sequence enforces the 60%/40% probability distributions required for the R/W flow, for write data, next operation is 60% read address and 40% write address, and for read data, next operation should be 60% write address and 40% rad address. for write address, next should be write address or write data, and for read address, next .should be read data	Verify mixed Read/Write transactions under weighted probability rules (60/40 splits after Write Data/Read Data .(operations	**SEQ_WR _RD**

2-UVM_Structure



2-Bugs in the slave

1-When received address is high the ns is READ_DATA

2-Mosi is not set to zero in IDLE

3-Bugs in the RAM

1-dout was take the write address not read

2-begin….end for the else

4-Code Snippets For Slave after modification:

1-interface

```
interface slave_if(input bit clk);
    logic       MOSI;
    logic       rst_n;
    logic       SS_n;
    logic       tx_valid;
    logic [7:0] tx_data;
    logic [9:0] rx_data;
    logic       rx_valid;
    logic       MISO;
    logic       rx_valid_ref;
    logic [9:0] rx_data_ref;
    logic       MISO_ref;

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

    modport REF (
        input  clk, MOSI, rst_n, SS_n, tx_valid, tx_data,
        output rx_valid_ref, MISO_ref, rx_data_ref
    );

endinterface
```

2-Design after edit bugs

```
module SLAVE (slave_if.DUT slaveif);

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


reg [3:0] counter;
reg       received_address;

reg [2:0] cs, ns;
reg [7:0] MISO_BUS;
reg [9:0] MOSI_BUS;

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

always @(*) begin
    case (cs)
        IDLE : begin
            if (slaveif.SS_n)
                ns = IDLE;
            else
                ns = CHK_CMD;
        end
        CHK_CMD : begin
            if (slaveif.SS_n)
                ns = IDLE;
            else begin
                if (~slaveif.MOSI)
                    ns = WRITE;
                else begin
                    if (received_address)
                        ns = READ_DATA;    //read data after recive address
                    else
                        ns = READ_ADD;
                end
            end
        end
        WRITE : begin
            if (slaveif.SS_n)
                ns = IDLE;
            else
                ns = WRITE;
        end
        READ_ADD : begin
            if (slaveif.SS_n)
                ns = IDLE;
            else
                ns = READ_ADD;
        end
        READ_DATA : begin
            if (slaveif.SS_n)
                ns = IDLE;
            else
                ns = READ_DATA;
        end
    endcase
end
```

```
always @(posedge  slaveif.clk) begin
    if (~slaveif.rst_n) begin
        slaveif.rx_valid <= 0;
        slaveif.rx_data <= 0;
        received_address <= 0;
        slaveif.MISO <= 0;
        MOSI_BUS<=0;
        MISO_BUS<=0;
    end
    else begin
        case (cs)
            IDLE : begin
                slaveif.rx_valid <= 0;
                slaveif.MISO <= 0;         // miso not set to zero in idle
                MOSI_BUS<=0;
                MISO_BUS<=0;
            end
            CHK_CMD : begin
                counter <= 10;
            end
            WRITE : begin
                if (counter > 0) begin
                    MOSI_BUS[counter-1] <= slaveif.MOSI;
                    counter <= counter - 1;
                end
                else begin
                    slaveif.rx_valid <= 1;
                    slaveif.rx_data<=MOSI_BUS;
                end
            end
            READ_ADD : begin
                if (counter > 0) begin
                    MOSI_BUS[counter-1] <= slaveif.MOSI;
                    counter <= counter - 1;
                end
                else begin
                    slaveif.rx_valid <= 1;
                    received_address <= 1;
                    slaveif.rx_data<=MOSI_BUS;
                end
            end
            READ_DATA : begin
                if (slaveif.tx_valid) begin
                    MISO_BUS<=slaveif.tx_data;
                    if (counter > 0) begin
                        slaveif.MISO <= MISO_BUS[counter-1];
                        counter <= counter - 1;
                    end
                    else begin
                        received_address <= 0;
                        slaveif.rx_valid <= 0; // for tx_valid to be asserted during read data
                    end
                end
                else begin
                    if (counter > 0) begin
                        MOSI_BUS[counter-1] <= slaveif.MOSI;
                        counter <= counter - 1;
                    end
                    else begin
                        slaveif.rx_valid <= 1;
                        slaveif.rx_data<=MOSI_BUS;
                        counter <= 8;
                    end
                end
            end
            default: begin
                slaveif.MISO <= 0;
            end
        endcase
    end
end
```

2-Assertion in the design

```
property asser1;
    @(posedge slaveif.clk) (~slaveif.rst_n) |=> (slaveif.rx_data == 0) && (slaveif.rx_valid == 0) && (slaveif.MISO == 0);
endproperty

property asser2;
    @(posedge slaveif.clk) disable iff(~slaveif.rst_n)
    (MOSI_BUS==3'b000 || MOSI_BUS==3'b110 || MOSI_BUS==3'b111||MOSI_BUS==3'b001) |-> ##10($rose(slaveif.rx_valid) && $rose(slaveif.SS_n)[->1]);
endproperty

property asser3;
    @(posedge slaveif.clk) disable iff(~slaveif.rst_n) (cs==IDLE && !slaveif.SS_n) |-> (ns==CHK_CMD);
endproperty

property asser4;
    @(posedge slaveif.clk) disable iff(~slaveif.rst_n) (cs==CHK_CMD && !slaveif.SS_n)|-> (ns==READ_DATA || ns==WRITE || ns==READ_ADD);
endproperty

property asser5;
    @(posedge slaveif.clk) disable iff(~slaveif.rst_n) (cs==WRITE && slaveif.SS_n) |-> (ns==IDLE);
endproperty
property asser6;
    @(posedge slaveif.clk) disable iff(~slaveif.rst_n) (cs==READ_DATA && slaveif.SS_n)|-> (ns==IDLE);
endproperty
property asser7;
    @(posedge slaveif.clk) disable iff(~slaveif.rst_n) (cs==READ_ADD && slaveif.SS_n)|-> (ns==IDLE);
endproperty

`ifdef SIM
as1:  assert property (asser1)  else $display("ERROR1");
as2:  assert property (asser2)  else $display("ERROR2");
as3:  assert property (asser3)  else $display("ERROR3");
as4:  assert property (asser4)  else $display("ERROR4");
as5:  assert property (asser5)  else $display("ERROR5");
as6:  assert property (asser6)  else $display("ERROR6");
as7:  assert property (asser7)  else $display("ERROR7");

cov1:  cover property (asser1);
cov2:  cover property (asser2);
cov3:  cover property (asser3);
cov4:  cover property (asser4);
cov5:  cover property (asser5);
cov6:  cover property (asser6);
cov7:  cover property (asser7);
`endif
endmodule
```

3-Sequence Item

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

    class slave_seq_item extends uvm_sequence_item;
        `uvm_object_utils(slave_seq_item)
             logic       MOSI;
        rand logic       rst_n;
        rand logic       SS_n;
        rand logic       tx_valid;
        rand logic [7:0] tx_data;
             logic [9:0] rx_data;
             logic       rx_valid;
             logic       MISO;

             logic [9:0] rx_data_ref;
             logic       rx_valid_ref;
             logic       MISO_ref;

        rand logic [10:0] mosi_data = 0;

        int counter;

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

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

        constraint SS_c {
            (mosi_data[9:8] == 2'b11) -> SS_n == (counter == 23);
            (mosi_data[9:8] != 2'b11) -> SS_n == (counter == 13);
        }

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

        constraint tx_valid_c {
            tx_valid == (mosi_data[9:8] == 2'b11);
        }

        function void pre_randomize();
            if((counter == 22)
                || counter == 12 && mosi_data[9:8] != 2'b11)
                counter = 0;
            else
                counter++;
        endfunction


        function void post_randomize();
            MOSI = mosi_data[10 - counter%11];
        endfunction


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

        function string convert2string_stimulus();
            return $sformatf("MOSI: %d, rst_n: %d, SS_n: %d, tx_valid: %d, tx_data: %d",
                             MOSI, rst_n, SS_n, tx_valid, tx_data);
        endfunction

    endclass
endpackage
```

4-Sequence

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

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

        slave_seq_item rst_item;

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

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


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

        slave_seq_item main_item;

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

        task body;
            main_item = slave_seq_item::type_id::create("main_item");
            repeat(10000)begin
                if((main_item.counter == 12 && main_item.mosi_data[9:8] != 2'b11)
                    || main_item.counter == 22)
                    main_item.mosi_data.rand_mode(1);
                else
                    main_item.mosi_data.rand_mode(0);
                start_item(main_item);
                assert(main_item.randomize());
                finish_item(main_item);

                `uvm_info("SEQUENCE MAIN", $sformatf("counter:
, mosi_data: %b, MOSI: %d, rst_n: %d, SS_n: %d, tx_valid: %d", main_item.counter, main_item.mosi_data, main_item.MOSI, main_item.rst_n, main_item.SS_n, main_item.tx_valid), UVM_LOW);
            end
        endtask
    endclass
endpackage
```

5-Sequencer

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

    class slave_sequencer extends uvm_sequencer #(slave_seq_item);
    `uvm_component_utils(slave_sequencer)

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

    endclass
endpackage
```

6-Configuration object

```
package slave_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class slave_config extends uvm_object;
        `uvm_object_utils(slave_config)
        uvm_active_passive_enum is_active;
        virtual slave_if vif;

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

7-Driver

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

    class slave_driver extends uvm_driver #(slave_seq_item);
        `uvm_component_utils(slave_driver)
        slave_seq_item drv_item;
        virtual slave_if vif;

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

        task run_phase(uvm_phase phase);
            forever begin
                drv_item = slave_seq_item::type_id::create("drv_item");
                seq_item_port.get_next_item(drv_item);
                vif.rst_n = drv_item.rst_n;
                vif.SS_n     = drv_item.SS_n;
                vif.tx_valid = drv_item.tx_valid;
                vif.tx_data  = drv_item.tx_data;
                vif.MOSI = drv_item.MOSI;
                @(negedge vif.clk);
                seq_item_port.item_done();
                `uvm_info("DRIVER RUN", drv_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

8-Monitor

```
package slave_monitor_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import slave_seq_item_pkg::*;
    class slave_monitor extends uvm_monitor;
        `uvm_component_utils(slave_monitor)
        slave_seq_item seq_item;
        virtual slave_if vif;

        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
                seq_item = slave_seq_item::type_id::create("seq_item");
                @(negedge vif.clk);
                seq_item.rst_n = vif.rst_n;
                seq_item.SS_n = vif.SS_n;
                seq_item.MOSI = vif.MOSI;
                seq_item.tx_data = vif.tx_data;
                seq_item.tx_valid = vif.tx_valid;
                seq_item.rx_valid = vif.rx_valid;
                seq_item.rx_data = vif.rx_data;
                seq_item.MISO = vif.MISO;
                seq_item.rx_valid_ref = vif.rx_valid_ref;
                seq_item.rx_data_ref = vif.rx_data_ref;
                seq_item.MISO_ref = vif.MISO_ref;
                mon_ap.write(seq_item);
                `uvm_info("run_phase", seq_item.convert2string(), UVM_HIGH);
            end
        endtask
    endclass
endpackage
```

9-Agent

```
package slave_agent_pkg;
    import slave_sequencer_pkg::*;
    import slave_driver_pkg::*;
    import slave_monitor_pkg::*;
    import slave_seq_item_pkg::*;
    import slave_config_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class slave_agent extends uvm_agent;
        `uvm_component_utils(slave_agent)

        slave_driver driver;
        slave_monitor monitor;
        slave_sequencer sqr;
        slave_config 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)::get(this, "", "SLAVE CFG", slave_cfg))
                `uvm_fatal("AGENT BUILD", "Failed to get CFG");

            if(slave_cfg.is_active == UVM_ACTIVE) begin
                sqr    = slave_sequencer::type_id::create("sqr", this);
                driver = slave_driver::type_id::create("driver", this);
            end

            agt_ap = new("agt_ap",this);
            monitor = slave_monitor::type_id::create("monitor", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            if(slave_cfg.is_active == UVM_ACTIVE) begin
                driver.seq_item_port.connect(sqr.seq_item_export);
                driver.vif = slave_cfg.vif;
            end

            monitor.mon_ap.connect(agt_ap);
            monitor.vif = slave_cfg.vif;
        endfunction
    endclass
endpackage
```

10-Scoreboard

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

    class slave_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(slave_scoreboard)
        slave_seq_item seq_item;
        uvm_analysis_export #(slave_seq_item) sb_exp;
        uvm_tlm_analysis_fifo #(slave_seq_item) fifo_export;

        int correct_count,error_count;

        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_exp = new("sb_exp",this);
            fifo_export = new("fifo_export",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            sb_exp.connect(fifo_export.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                fifo_export.get(seq_item);
                if(seq_item.rx_data != seq_item.rx_data_ref || seq_item.rx_valid != seq_item.rx_valid_ref || seq_item.MISO != seq_item.MISO_ref)begin
                    `uvm_error("SB RUN", $sformatf("Mismatch!\nMISO: %d Expected: %d, rx_data: %0d Expected: %0d\nrx_valid: %0d Expected: %0d",
                    |   |   |   seq_item.MISO, seq_item.MISO_ref, seq_item.rx_data, seq_item.rx_data_ref, seq_item.rx_valid, seq_item.rx_valid_ref));
                    error_count++;
                end
                else begin
                    `uvm_info("SB RUN", $sformatf("Match!\nMISO: %d Expected: %d, rx_data: %0d Expected: %0d\nrx_valid: %0d Expected: %0d",
                    |   |   |   seq_item.MISO, seq_item.MISO_ref, seq_item.rx_data, seq_item.rx_data_ref, seq_item.rx_valid, seq_item.rx_valid_ref), UVM_HIGH);
                    correct_count++;
                end
            end
        endtask

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("SB REPORT", $sformatf("errors: %0d", error_count), UVM_MEDIUM);
            `uvm_info("SB REPORT", $sformatf("correct: %0d", correct_count), UVM_MEDIUM);
        endfunction
    endclass
endpackage
```

11-Slave Ref

```
module SLAVE_REF (slave_if.REF slaveif);

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


(* fsm_encoding = "sequential" *)
reg [2:0]cs,ns;
reg READ_FLAG;
reg [3:0] count;
reg [7:0] MISO_BUS;
reg [9:0] MOSI_BUS;
```

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

always@(*)begin
    case (cs)
    IDLE : begin
        if(slaveif.SS_n==0)
            ns=CHK_CMD;
        else
            ns=IDLE;
    end
    CHK_CMD : begin
        if(slaveif.SS_n)
        ns=IDLE;
        else if(slaveif.SS_n==0&&slaveif.MOSI==0)begin
            ns=WRITE;
        end
        else if (slaveif.SS_n==0&&slaveif.MOSI==1&&READ_FLAG==0)begin
            ns=READ_ADD;

        end

        else if(slaveif.SS_n==0&&slaveif.MOSI==1&&READ_FLAG==1)begin
            ns=READ_DATA;
        end
        else
            ns=CHK_CMD;
        end
    WRITE : begin
        if(slaveif.SS_n)
            ns=IDLE;
        else
            ns=WRITE;
        end
    READ_ADD : begin
        if(slaveif.SS_n)
            ns=IDLE;
        else
            ns=READ_ADD;
        end
    READ_DATA : begin
        if(slaveif.SS_n)
            ns=IDLE;
        else
            ns=READ_DATA;
    end
    endcase
end
```

```
always@(posedge slaveif.clk)begin
        if(~slaveif.rst_n)begin
            count<=10;
            slaveif.rx_valid_ref<=0;
            slaveif.rx_data_ref<=0;
            slaveif.MISO_ref<=0;
            READ_FLAG<=0;
            MISO_BUS<=0;
            MOSI_BUS<=0;
        end
        else begin
        case(cs)
            IDLE : begin
                count<=10;
                slaveif.rx_valid_ref<=0;
                slaveif.MISO_ref<=0;
                MOSI_BUS<=0;
                MISO_BUS<=0;
            end

            WRITE : begin
                if(count>0)begin
                    MOSI_BUS<={MOSI_BUS[9:0],slaveif.MOSI};
                    slaveif.rx_valid_ref<=0;
                    count<=count-1;
                end
                else  begin
                    slaveif.rx_data_ref<=MOSI_BUS;
                    slaveif.rx_valid_ref<=1;
                end
            end
            READ_ADD : begin
                if(count>0)begin
                    MOSI_BUS<={MOSI_BUS[9:0],slaveif.MOSI};
                    slaveif.rx_valid_ref<=0;
                    count<=count-1;
                end
                else begin
                    slaveif.rx_data_ref<=MOSI_BUS;
                    slaveif.rx_valid_ref<=1;
                    READ_FLAG<=1;
                end
            end

            READ_DATA : begin
                if (slaveif.tx_valid) begin
                    MISO_BUS<=slaveif.tx_data;
                    if (count > 0) begin
                        slaveif.MISO_ref <= MISO_BUS[count-1];
                        count <= count - 1;
                    end
                    else begin
                        READ_FLAG <= 0;
                        slaveif.rx_valid_ref <= 0;
                    end
                end
                else begin
                    if (count > 0) begin
                        MOSI_BUS[count-1] <= slaveif.MOSI;
                        count <= count - 1;
                    end
                    else begin
                        slaveif.rx_valid_ref <= 1;
                        slaveif.rx_data_ref <= MOSI_BUS;
                        count <= 8;
                    end
                end
            end
endcase
        end
end
endmodule
```

12-Coverage

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

    class slave_cov extends uvm_component;
        `uvm_component_utils(slave_cov)
        slave_seq_item seq_item;

        uvm_analysis_export #(slave_seq_item) cov_export;
        uvm_tlm_analysis_fifo #(slave_seq_item) fifo_export;


        covergroup g1;
        rx_data : coverpoint seq_item.rx_data[9:8];
        ss_n: coverpoint seq_item.SS_n {
            bins normal=(1 => 0[*13] => 1);
            bins Rd_dt=(1 => 0[*23] => 1);
        }
        mosi_cp: coverpoint seq_item.MOSI {
            bins write_add = (0 => 0 => 0);
            bins write_data = (0 => 0 => 1);
            bins READ_ADD = (1 => 1 => 0);
            bins READ_DATA = (1 => 1 => 1);
        }
        sceniario: cross ss_n,mosi_cp {
            ignore_bins illegal1 = binsof(ss_n.normal) && binsof(mosi_cp.READ_DATA);
            ignore_bins illegal2 = binsof(ss_n.Rd_dt) && binsof(mosi_cp.READ_ADD);
            ignore_bins illegal3 = binsof(ss_n.Rd_dt) && binsof(mosi_cp.write_add);
            ignore_bins illegal4 = binsof(ss_n.Rd_dt) && binsof(mosi_cp.write_data);
        }
        endgroup

        function new(string name ="slave_cov",uvm_component parent=null);
            super.new(name,parent);
            g1 = new;
        endfunction

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

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


        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                fifo_export.get(seq_item);
                g1.sample();
            end
        endtask
    endclass
endpackage
```

13-Environment

```
package slave_env_pkg;
    import slave_agent_pkg::*;
    import uvm_pkg::*;
    import slave_scoreboard_pkg::*;
    import slave_coverage_pkg::*;
    `include "uvm_macros.svh"

    class slave_env extends uvm_env;
        `uvm_component_utils(slave_env)

        slave_agent agent;
        slave_scoreboard sb;
        slave_cov 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);
            agent = slave_agent::type_id::create("agent", this);
            sb = slave_scoreboard::type_id::create("sb",this);
            cov = slave_cov::type_id::create("cov",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agent.agt_ap.connect(sb.sb_exp);
            agent.agt_ap.connect(cov.cov_export);
        endfunction
    endclass
endpackage
```

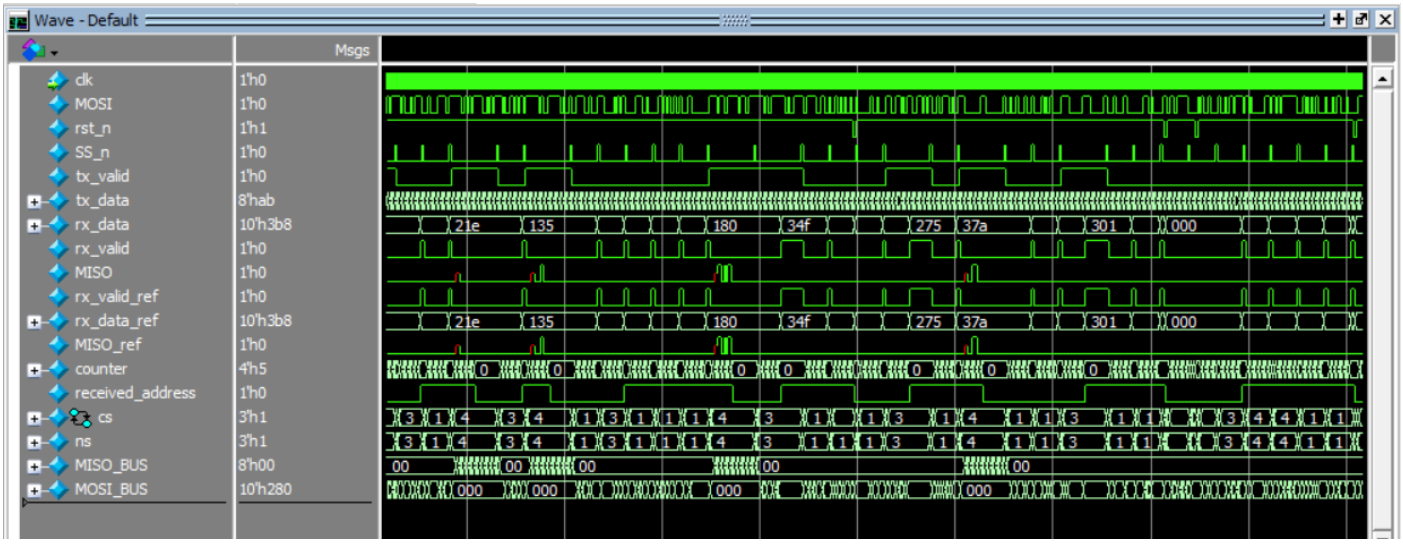
14-Test

```
package slave_test_pkg;
    import uvm_pkg::*;
    import slave_env_pkg::*;
    import slave_config_pkg::*;
    import slave_sequence_pkg::*;
    `include "uvm_macros.svh"

    class slave_test extends uvm_test;
        `uvm_component_utils(slave_test)

        slave_env env;
        slave_config slave_cfg;
        reset_seq rst_seq;
        main_seq mn_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);

            slave_cfg = slave_config::type_id::create("slave_cfg");
            slave_cfg.is_active = UVM_ACTIVE;

            if(!uvm_config_db#(virtual slave_if)::get(this, "", "SLAVE_VIF", slave_cfg.vif))
            `uvm_fatal("build phase","unable to get v_if");

            uvm_config_db#(slave_config)::set(this, "*", "CFG", slave_cfg);

            env = slave_env::type_id::create("env", this);
            rst_seq = reset_seq::type_id::create("rst_seq", this);
            mn_seq = main_seq::type_id::create("mn_seq", this);
        endfunction

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

            `uvm_info("TEST RUN", "Starting reset test.", UVM_LOW);
            rst_seq.start(env.agent.sqr);

            `uvm_info("TEST RUN", "Starting main test.", UVM_LOW);
            mn_seq.start(env.agent.sqr);

            `uvm_info("TEST RUN", "Test done.", UVM_LOW);
            phase.drop_objection(this);
        endtask
    endclass
endpackage
```

15-Top

```
import slave_test_pkg::*;
import uvm_pkg::*;

module slave_top();

    bit clk;
    always #1 clk = ~clk;

    slave_if slaveif(clk);
    SLAVE slave_dut(slaveif);
    SLAVE_REF slave_ref(slaveif);



    initial begin
        uvm_config_db #(virtual slave_if)::set(null, "uvm_test_top", "SLAVE_VIF", slaveif);
        run_test("slave_test");
    end
endmodule
```

16-Wave form



16-Do file

```
vlib work
vlog +define+SIM -f src_files_slave.list +cover -covercells
vsim -voptargs=+acc work.slave_top -cover
add wave -position insertpoint sim:/slave_top/slaveif/*
add wave -position insertpoint sim:/slave_top/slave_dut/*

run -all
```

17- Slave Coverage

-Code coverage

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

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

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

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      47        47         0   100.00%
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         58        58         0   100.00%
```

-Functional Coverage

```
      % Hit:                                          100.00%        100          -
Covergroup instance \/slave_coverage_pkg::slave_cov::g1
                                                      100.00%        100          -    Covered
   covered/total bins:                                     14         14          -
   missing/total bins:                                      0         14          -
   % Hit:                                             100.00%        100          -
   Coverpoint rx_data                                 100.00%        100          -    Covered
       covered/total bins:                                  4          4          -
       missing/total bins:                                  0          4          -
       % Hit:                                         100.00%        100          -
       bin auto[0]                                      11447          1          -    Covered
       bin auto[1]                                       5615          1          -    Covered
       bin auto[2]                                       6939          1          -    Covered
       bin auto[3]                                       6001          1          -    Covered
   Coverpoint ss_n                                    100.00%        100          -    Covered
       covered/total bins:                                  2          2          -
       missing/total bins:                                  0          2          -
       % Hit:                                         100.00%        100          -
       bin normal                                        1484          1          -    Covered
       bin Rd_dt                                          382          1          -    Covered
   Coverpoint mosi_cp                                 100.00%        100          -    Covered
       covered/total bins:                                  4          4          -
       missing/total bins:                                  0          4          -
       % Hit:                                         100.00%        100          -
       bin write_add                                     5542          1          -    Covered
       bin write_data                                    3533          1          -    Covered
       bin READ_ADD                                      3448          1          -    Covered
       bin READ_DATA                                     4737          1          -    Covered
   Cross sceniario                                    100.00%        100          -    Covered
       covered/total bins:                                  4          4          -
       missing/total bins:                                  0          4          -
       % Hit:                                         100.00%        100          -
       Auto, Default and User Defined Bins:
           bin <Rd_dt,READ_DATA>                          195          1          -    Covered
           bin <normal,READ_ADD>                          382          1          -    Covered
           bin <normal,write_data>                        484          1          -    Covered
           bin <normal,write_add>                         618          1          -    Covered
       Illegal and Ignore Bins:
           ignore_bin illegal4                              0                     -    ZERO
           ignore_bin illegal3                              0                     -    ZERO
           ignore_bin illegal2                              0                     -    ZERO
           ignore_bin illegal1                              0                     -    ZERO
```

-Assertion Coverage

```
=== Instance: /wrapper_top/slave_dut
=== Design Unit: work.SLAVE
================================================================================

Assertion Coverage:
    Assertions                       7         7         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/wrapper_top/slave_dut/as1
                     slave.sv(171)                       0          1
/wrapper_top/slave_dut/as2
                     slave.sv(172)                       0          1
/wrapper_top/slave_dut/as3
                     slave.sv(173)                       0          1
/wrapper_top/slave_dut/as4
                     slave.sv(174)                       0          1
/wrapper_top/slave_dut/as5
                     slave.sv(175)                       0          1
/wrapper_top/slave_dut/as6
                     slave.sv(176)                       0          1
/wrapper_top/slave_dut/as7
                     slave.sv(177)                       0          1
```

5-Code Snippets For RAM after modification:

1-interface

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

2-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 //<<<<<<<<<<<<<<<
        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]; //<<<<<<<<<<<<<<<
                default : dout <= 0;
            endcase
        end

        tx_valid <= (din[9] && din[8] && rx_valid)? 1'b1 : 1'b0;
    end //<<<<<<<<<<<<<<<
end


    property rst_p;
        @(posedge clk)
        (!rst_n) |-> ##1  (!dout && !tx_valid);
    endproperty

    property tx_valid_p1;
        @(posedge clk) disable iff(!rst_n)
        (din[9:8] != 2'b11) |-> ##1 (tx_valid == 1'b0);
    endproperty

    property tx_valid_p2;
        @(posedge clk) disable iff(!rst_n)
        (din[9:8] == 2'b11 && rx_valid) |-> ##1 tx_valid ##[1:$] $fell(tx_valid);
    endproperty

    property wr_addr_p;
        @(negedge clk) disable iff(!rst_n)
        (din[9:8] == 2'b00) |-> ##[1:$] (din[9:8] == 2'b01);
    endproperty

    property rd_addr_p;
        @(negedge clk) disable iff(!rst_n)
        (din[9:8] == 2'b10) |-> ##[1:$] (din[9:8] == 2'b11);
    endproperty

    a1: assert property(rst_p);
    a2: assert property(tx_valid_p1);
    a3: assert property(tx_valid_p2);
    a4: assert property(wr_addr_p);
    a5: assert property(rd_addr_p);

    c1: cover property(rst_p);
    c2: cover property(tx_valid_p1);
    c3: cover property(tx_valid_p2);
    c4: cover property(wr_addr_p);
    c5: cover property(rd_addr_p);

endmodule
```

3-sequence item

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

    class ram_seq_item extends uvm_sequence_item;
        `uvm_object_utils(ram_seq_item);

        rand logic [9:0] din;
        rand logic       rst_n;
        rand logic       rx_valid;
             logic [7:0] dout;
             logic       tx_valid;
             logic [7:0] dout_ref;
             logic       tx_valid_ref;

             logic [9:0] prev_din = 0;

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

        function string convert2string();
            return $sformatf("%s din: %d rst_n: %d rx_valid: %d dout: %d tx_valid: %d",
                super.convert2string(), din, rst_n, rx_valid, dout, tx_valid);
        endfunction

        function string convert2string_stimulus();
            return $sformatf("din: %d rst_n: %d rx_valid: %d",
                din, rst_n, rx_valid);
        endfunction

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

        constraint wr_seq_c {
            din[9:8] inside {2'b00, 2'b01};
        }

        constraint rd_seq_c {
            din[9:8] inside {2'b10, 2'b11};
            if (prev_din[9:8] == 2'b10)
                din[9:8] == 2'b11;
            else if (prev_din[9:8] == 2'b11)
                din[9:8] == 2'b10;
        }

        constraint rd_wr_seq_c {
                if (prev_din[9:8] == 2'b00)
                    din[9:8] inside { 2'b00, 2'b01};

                else if(prev_din[9:8] == 2'b01)
                    din[9:8] dist {2'b10 := 6, 2'b00 := 4};

                else if (prev_din[9:8] == 2'b10)
                    din[9:8] == 2'b11;

                else // 11
                    din[9:8] dist {2'b10 := 4, 2'b00 := 6};
        }


        function void post_randomize();
            prev_din = din;
        endfunction

    endclass
endpackage
```

4-Sequence

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

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

        ram_seq_item rst_item;

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

        task body();
            rst_item = ram_seq_item::type_id::create("rst_item");
            rst_item.wr_seq_c.constraint_mode(0);
            rst_item.rd_seq_c.constraint_mode(0);
            rst_item.rd_wr_seq_c.constraint_mode(0);
            start_item(rst_item);
            rst_item.rst_n = 0;
            rst_item.din = 0;
            rst_item.rx_valid = 0;
            finish_item(rst_item);
        endtask
    endclass

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

        ram_seq_item wr_item;

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

        task body();
            wr_item = ram_seq_item::type_id::create("wr_item");
            wr_item.rd_seq_c.constraint_mode(0);
            wr_item.rd_wr_seq_c.constraint_mode(0);
            repeat(1000) begin
                start_item(wr_item);
                assert(wr_item.randomize());
                wr_item.din[9] = 1'b0;
                finish_item(wr_item);
            end
        endtask
    endclass
```

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

        ram_seq_item rd_item;

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

        task body();
            rd_item = ram_seq_item::type_id::create("rd_item");
            rd_item.wr_seq_c.constraint_mode(0);
            rd_item.rd_wr_seq_c.constraint_mode(0);
            repeat(1000) begin
                start_item(rd_item);
                assert(rd_item.randomize());
                rd_item.din[9] = 1'b1;
                finish_item(rd_item);
            end
        endtask
    endclass

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

        ram_seq_item rd_wr_item;

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

        task body();
            rd_wr_item = ram_seq_item::type_id::create("rd_wr_item");
            rd_wr_item.wr_seq_c.constraint_mode(0);
            rd_wr_item.rd_seq_c.constraint_mode(0);
            repeat(1000) begin
                start_item(rd_wr_item);
                assert(rd_wr_item.randomize());
                finish_item(rd_wr_item);
            end
        endtask
    endclass
endpackage
```

5-Sequencer

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

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

6-Driver

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

    class ram_driver extends uvm_driver #(ram_seq_item);
        `uvm_component_utils(ram_driver);
        virtual ram_if vif;
        ram_seq_item drv_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);
            drv_item = ram_seq_item::type_id::create("drv_item");
            forever begin
                seq_item_port.get_next_item(drv_item);
                vif.din = drv_item.din;
                vif.rx_valid = drv_item.rx_valid;
                vif.rst_n = drv_item.rst_n;
                seq_item_port.item_done();
                @(negedge vif.clk);
                `uvm_info("DRIVER", drv_item.convert2string_stimulus(), UVM_HIGH);
            end
        endtask
    endclass
endpackage
```

7-Configuration object

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

    class ram_config extends uvm_object;
        `uvm_object_utils(ram_config);
        virtual ram_if vif;
        uvm_active_passive_enum is_active;

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

8-Monitor

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

    class ram_monitor extends uvm_monitor;
        `uvm_component_utils(ram_monitor);
        ram_seq_item mon_item;
        virtual ram_if vif;
        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);
            forever begin
                mon_item = ram_seq_item::type_id::create("mon_item");
                @(negedge vif.clk);
                mon_item.rx_valid = vif.rx_valid;
                mon_item.din = vif.din;
                mon_item.rst_n = vif.rst_n;
                mon_item.dout = vif.dout;
                mon_item.tx_valid = vif.tx_valid;
                mon_item.dout_ref = vif.dout_ref;
                mon_item.tx_valid_ref = vif.tx_valid_ref;
                mon_ap.write(mon_item);
                `uvm_info("MONITOR RUN", mon_item.convert2string(), UVM_HIGH);
            end
        endtask
    endclass

endpackage
```

9-Agent

```
package ram_agent_pkg;
    import uvm_pkg::*;
    import ram_sequencer_pkg::*;
    import ram_driver_pkg::*;
    import ram_config_pkg::*;
    import ram_seq_item_pkg::*;
    import ram_monitor_pkg::*;
    `include "uvm_macros.svh"

    class ram_agent extends uvm_agent;
        `uvm_component_utils(ram_agent);
        ram_sequencer sqr;
        ram_driver drv;
        ram_config cfg;
        ram_monitor mon;
        uvm_analysis_port #(ram_seq_item) agent_ap;

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

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

            if(!uvm_config_db #(ram_config)::get(this, "", "CFG", cfg))
                `uvm_fatal("AGENT BUILD", "get failure!");

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

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

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

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

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

10-Scoreboard

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

    class ram_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(ram_scoreboard);
        uvm_analysis_export #(ram_seq_item) sb_exp;
        uvm_tlm_analysis_fifo #(ram_seq_item) sb_fifo;
        ram_seq_item sb_item;

        int error_count, correct_count;

        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_exp = new("sb_exp", this);
            sb_fifo = new("sb_fifo", this);
        endfunction

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(sb_item);
                if(sb_item.dout !== sb_item.dout_ref || sb_item.tx_valid !== sb_item.tx_valid_ref) begin
                    `uvm_error("SR_SCB", $sformatf("Mismatch!\ndout: %0d Expected: %0d\ntx_valid: %0d Expected: %0d",
                        sb_item.dout, sb_item.dout_ref, sb_item.tx_valid, sb_item.tx_valid_ref));
                    error_count++;
                end
                else begin
                    `uvm_info("SR_SCB", $sformatf("Mmatch!\ndout: %0d Expected: %0d\ntx_valid: %0d Expected: %0d",
                        sb_item.dout, sb_item.dout_ref, sb_item.tx_valid, sb_item.tx_valid_ref), UVM_HIGH);
                    correct_count++;
                end
            end


        endtask


        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("SCB_REPORT", $sformatf("\ntotal correct: %0d\ntotal errors: %0d", correct_count, error_count), UVM_MEDIUM);
        endfunction
    endclass
endpackage
```

11-RAM_ref

```
module RAM_REF #(
    parameter MEM_DEPTH = 256,
    parameter ADDR_SIZE = 8
)(
    input       [9:0]   din,
    input               clk,
    input               rst_n,
    input               rx_valid,
    output reg  [7:0]   dout,
    output reg          tx_valid
);
    reg [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];

    reg [ADDR_SIZE-1:0] write_address;
    reg [ADDR_SIZE-1:0] read_address;

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

12-Coverage

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

    class ram_coverage extends uvm_component;
        `uvm_component_utils(ram_coverage);
        uvm_analysis_export #(ram_seq_item) cov_exp;
        uvm_tlm_analysis_fifo #(ram_seq_item) cov_fifo;
        ram_seq_item cov_item;

        covergroup cov_gp;
            rx_valid_p: coverpoint cov_item.rx_valid{
                option.weight = 0;
            }

            din_p_all: coverpoint cov_item.din[9:8];

            all_transition: coverpoint cov_item.din[9:8] {
                bins wr_data_after_wr_address = (2'b00 => 2'b01);
                bins rd_data_after_rd_address = (2'b10 => 2'b11);
                bins all_transition = (2'b00 => 2'b01 => 2'b10 => 2'b11);
            }
            c1: cross din_p_all, rx_valid_p {
                ignore_bins b1 = binsof(rx_valid_p) intersect {0};
            }

            c2: cross din_p_all , rx_valid_p {
                option.cross_auto_bin_max = 0;
                bins b2 = binsof(rx_valid_p) intersect {1}
                    && binsof(din_p_all) intersect {2'b11};
           }

        endgroup

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

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

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

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            cov_fifo.get(cov_item);
            cov_gp.sample();
        end
    endtask
    endclass
endpackage
```

13-Environment

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

    class ram_env extends uvm_env;
        `uvm_component_utils(ram_env);
        ram_agent agt;
        ram_scoreboard sb;
        ram_coverage cov;

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

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

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

14-Test

```
package ram_test_pkg;
    import ram_env_pkg::*;
    import ram_config_pkg::*;
    import ram_sequence_pkg::*;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class ram_test extends uvm_test;
        `uvm_component_utils(ram_test);
        ram_env env;
        ram_config cfg;
        ram_reset_seq rst_seq;
        ram_wr_seq wr_seq;
        ram_rd_seq rd_seq;
        ram_rd_wr_seq rd_wr_seq;
        function new(string name = "ram_test", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            cfg = ram_config::type_id::create("cfg");
            if(!uvm_config_db #(virtual ram_if)::get(this, "", "RAM_VIF", cfg.vif))
                `uvm_fatal("TEST BUilD", "Failure in getting the interface");
            cfg.is_active = UVM_ACTIVE;
            uvm_config_db #(ram_config)::set(this, "*", "CFG", cfg);

            rst_seq = ram_reset_seq::type_id::create("rst_seq");
            wr_seq = ram_wr_seq::type_id::create("wr_seq");
            rd_seq = ram_rd_seq::type_id::create("rd_seq");
            rd_wr_seq = ram_rd_wr_seq::type_id::create("rd_wr_seq");
            env = ram_env::type_id::create("env", this);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("TEST_RUN", "starting reset test", UVM_LOW);
            rst_seq.start(env.agt.sqr);
            `uvm_info("TEST_RUN", "starting WRITE test", UVM_LOW);
            wr_seq.start(env.agt.sqr);
            `uvm_info("TEST_RUN", "starting READ test", UVM_LOW);
            rd_seq.start(env.agt.sqr);
            `uvm_info("TEST_RUN", "starting READ WRITE test", UVM_LOW);
            rd_wr_seq.start(env.agt.sqr);
            `uvm_info("TEST_RUN", "test done", UVM_LOW);
            #5;
            phase.drop_objection(this);
        endtask

    endclass

endpackage
```

15-Top

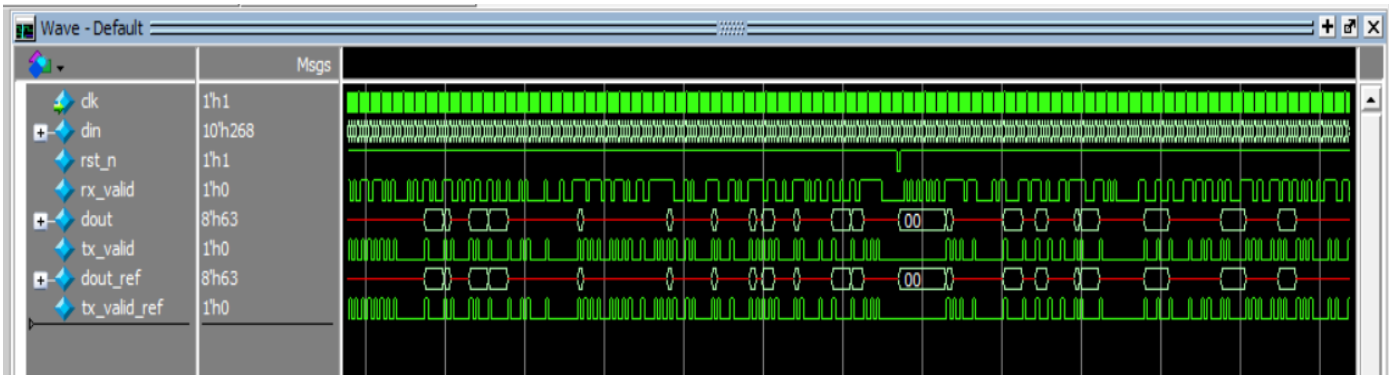
```
import ram_test_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

module ram_top();

    logic clk;
    initial clk = 0;
    always #1 clk = ~clk;

    ram_if ramif(clk);
    RAM ram_dut(ramif.din, clk, ramif.rst_n, ramif.rx_valid, ramif.dout, ramif.tx_valid);
    RAM_REF ram_ref(ramif.din, clk, ramif.rst_n, ramif.rx_valid, ramif.dout_ref, ramif.tx_valid_ref);


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

16-Do file

```
vlib work
vlog +define+SIM -f src_files.list +cover -covercells
vsim -voptargs=+acc work.ram_top -cover

coverage save RAM_COV.ucdb -onexit
coverage exclude -src RAM.sv -line 27 -code s
add wave -position insertpoint sim:/ram_top/ramif/*
run -all
```

17-wave form

There is unknown values because not the whole ram is written and we don't initialize it to zero



18-Coverage report

-Code coverage

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         8         0   100.00%
```

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

I exclude the default statement and branch

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

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         8         0   100.00%
```

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

-Assertion coverage

```
Assertion Coverage:
    Assertions                       5         5         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/ram_top/ram_dut/a1  RAM.sv(61)                             0          1
/ram_top/ram_dut/a2  RAM.sv(62)                             0          1
/ram_top/ram_dut/a3  RAM.sv(63)                             0          1
/ram_top/ram_dut/a4  RAM.sv(64)                             0          1
/ram_top/ram_dut/a5  RAM.sv(65)                             0          1
Branch Coverage:
```

-Functional Coverage

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

----------------------------------------------------------------------------------------------------------------
 TYPE /ram_coverage_pkg/ram_coverage/cov_gp               100.00%        100          -    Covered
    covered/total bins:                                        14         14          -
    missing/total bins:                                         0         14          -
    % Hit:                                                100.00%        100          -
    Coverpoint rx_valid_p                                   0.00%        100          -    ZERO
        covered/total bins:                                     2          2          -
        missing/total bins:                                     0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint din_p_all                                  100.00%        100          -    Covered
        covered/total bins:                                     4          4          -
        missing/total bins:                                     0          4          -
        % Hit:                                            100.00%        100          -
    Coverpoint all_transition                             100.00%        100          -    Covered
        covered/total bins:                                     3          3          -
        missing/total bins:                                     0          3          -
        % Hit:                                            100.00%        100          -
    Cross c1                                              100.00%        100          -    Covered
        covered/total bins:                                     4          4          -
        missing/total bins:                                     0          4          -
        % Hit:                                            100.00%        100          -
    Cross c2                                              100.00%        100          -    Covered
        covered/total bins:                                     1          1          -
        missing/total bins:                                     0          1          -
        % Hit:                                            100.00%        100          -
 Covergroup instance \/ram_coverage_pkg::ram_coverage::cov_gp
                                                          100.00%        100          -    Covered
    covered/total bins:                                        14         14          -
    missing/total bins:                                         0         14          -
    % Hit:                                                100.00%        100          -
    Coverpoint rx_valid_p [1]                             100.00%        100          -    Covered
        covered/total bins:                                     2          2          -
        missing/total bins:                                     0          2          -
        % Hit:                                            100.00%        100          -
        bin auto[0]                                          1451          1          -    Covered
        bin auto[1]                                          1551          1          -    Covered
    Coverpoint din_p_all                                  100.00%        100          -    Covered
        covered/total bins:                                     4          4          -
        missing/total bins:                                     0          4          -
        % Hit:                                            100.00%        100          -
        bin auto[0]                                           914          1          -    Covered
        bin auto[1]                                           692          1          -    Covered
        bin auto[2]                                           699          1          -    Covered
        bin auto[3]                                           697          1          -    Covered
    Coverpoint all_transition                             100.00%        100          -    Covered
        covered/total bins:                                     3          3          -
        missing/total bins:                                     0          3          -
        % Hit:                                            100.00%        100          -
        bin wr_data_after_wr_address                          455          1          -    Covered
        bin rd_data_after_rd_address                          697          1          -    Covered
        bin all_transition                                    131          1          -    Covered
    Cross c1                                              100.00%        100          -    Covered
        covered/total bins:                                     4          4          -
        missing/total bins:                                     0          4          -
        % Hit:                                            100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <auto[3],auto[1]>                             369          1          -    Covered
            bin <auto[1],auto[1]>                             353          1          -    Covered
            bin <auto[2],auto[1]>                             376          1          -    Covered
            bin <auto[0],auto[1]>                             453          1          -    Covered
        Illegal and Ignore Bins:
            ignore_bin b1                                    1451                     -    Occurred
    Cross c2                                              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 b2                                            369          1          -    Covered
```

5-Code Snippets For Wrapper after modification:

1-interface

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

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

2-Design

```
module WRAPPER (wrapper_if.DUT wrapperif);

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

RAM     ram_dut   (rx_data_din, wrapperif.clk, wrapperif.rst_n, rx_valid, tx_data_dout, tx_valid);
RAM_REF ram_ref(rx_data_din, wrapperif.clk, wrapperif.rst_n, rx_valid, tx_data_dout_ref, tx_valid_ref);

`ifdef SIM
rst_as  : assert property (@(posedge wrapperif.clk) (~wrapperif.rst_n) |=> (~wrapperif.MISO));
rst_cov : cover  property (@(posedge wrapperif.clk) (~wrapperif.rst_n) |=> (~wrapperif.MISO));

MISO_as  : assert property (@(posedge wrapperif.clk) disable iff(~wrapperif.rst_n) (rx_data_din[9:8] != 2'b11) |=> $stable(wrapperif.MISO));
MISO_cov : cover  property (@(posedge wrapperif.clk) disable iff(~wrapperif.rst_n) (rx_data_din[9:8] != 2'b11) |=> $stable(wrapperif.MISO));
`endif
endmodule
```

3-Sequencer

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

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

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

    endclass
endpackage
```

4-Sequence item

```
package wrapper_seq_item_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;

        rand logic SS_n;
        rand logic MOSI;
        logic MISO;

        rand logic [10:0] mosi_data = 0;
        logic [10:0] prev_mosi_data = 0;

        int counter;

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

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

        constraint SS_c {
            (mosi_data[9:8] == 2'b11) -> SS_n == (counter == 23);
            (mosi_data[9:8] != 2'b11) -> SS_n == (counter == 13);
        }

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

        constraint wr_seq_c {
            mosi_data[10:8] inside {3'b000, 3'b001};
        }

        constraint rd_seq_c {
            mosi_data[10:8] inside {3'b110, 3'b111};
            if (prev_mosi_data[9:8] == 2'b10)
                mosi_data[9:8] == 2'b11;
            else if (prev_mosi_data[9:8] == 2'b11)
                mosi_data[9:8] == 2'b10;
        }

        constraint rd_wr_seq_c {
                if (prev_mosi_data[9:8] == 2'b00)
                    mosi_data[9:8] inside { 2'b00, 2'b01};

                else if(prev_mosi_data[9:8] == 2'b01)
                    mosi_data[9:8] dist {2'b10 := 6, 2'b00 := 4};

                else if (prev_mosi_data[9:8] == 2'b10)
                    mosi_data[9:8] == 2'b11;

                else // 11
                    mosi_data[9:8] dist {2'b10 := 4, 2'b00 := 6};
        }

        function void pre_randomize();
            if((counter == 23)
                || counter == 13 && mosi_data[9:8] != 2'b11)
                counter = 0;
            else
                counter++;
        endfunction
```

```
    function void post_randomize();
        prev_mosi_data = mosi_data;
        MOSI = mosi_data[10 - counter%11];
    endfunction


    function string convert2string();
        return $sformatf("%s rst_n = %0b, SS_n = %0b, MOSI = %0b, MISO = %0b",
            super.convert2string(), rst_n, SS_n, MOSI, MISO);
    endfunction

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

5-Sequence

```
package wrapper_sequence_pkg;
    import uvm_pkg::*;
    import wrapper_seq_item_pkg::*;
    import ram_seq_item_pkg::*;
    import slave_seq_item_pkg::*;
    `include"uvm_macros.svh"

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

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

        task body;
            rst_item = wrapper_seq_item::type_id::create("rst_item");
            start_item(rst_item);
            rst_item.rst_n = 0;
            rst_item.SS_n = 1;
            rst_item.MOSI = 0;
            finish_item(rst_item);
        endtask
    endclass

    class wrapper_wr_seq extends uvm_sequence #(wrapper_seq_item);
        `uvm_object_utils(wrapper_wr_seq)
        wrapper_seq_item wr_item;

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

        task body();
            wr_item = wrapper_seq_item::type_id::create("wr_item");
            wr_item.rd_seq_c.constraint_mode(0);
            wr_item.rd_wr_seq_c.constraint_mode(0);


            repeat(10000) begin
                if((wr_item.counter == 13 && wr_item.mosi_data[9:8] != 2'b11)
                    || wr_item.counter == 23) begin
                    wr_item.mosi_data.rand_mode(1);
                    wr_item.wr_seq_c.constraint_mode(1);
                    end
                else begin
                    wr_item.mosi_data.rand_mode(0);
                    wr_item.wr_seq_c.constraint_mode(0);
                end
                start_item(wr_item);
                assert(wr_item.randomize());

                `uvm_info("SEQUENCE MAIN", $sformatf("counter:
, mosi_data: %b, MOSI: %b, SS_n: %b",wr_item.counter,wr_item.mosi_data, wr_item.MOSI, wr_item.SS_n), UVM_HIGH)
                finish_item(wr_item);
            end
        endtask
    endclass
```

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

        wrapper_seq_item rd_item;

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

        task body();
            rd_item = wrapper_seq_item::type_id::create("rd_item");
            rd_item.wr_seq_c.constraint_mode(0);
            rd_item.rd_wr_seq_c.constraint_mode(0);

            repeat(10000) begin
                start_item(rd_item);
                if((rd_item.counter == 13 && rd_item.mosi_data[9:8] != 2'b11)
                    || rd_item.counter == 23) begin
                    rd_item.mosi_data.rand_mode(1);
                    rd_item.rd_seq_c.constraint_mode(1);
                    end
                else begin
                    rd_item.mosi_data.rand_mode(0);
                    rd_item.rd_seq_c.constraint_mode(0);
                end
                assert(rd_item.randomize());
                finish_item(rd_item);

            end
        endtask

    endclass

    class wrapper_rd_wr_seq extends uvm_sequence #(wrapper_seq_item);
        `uvm_object_utils(wrapper_rd_wr_seq)
        wrapper_seq_item rd_wr_item;

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

        task body();
            rd_wr_item = wrapper_seq_item::type_id::create("rd_wr_item");
            rd_wr_item.wr_seq_c.constraint_mode(0);
            rd_wr_item.rd_seq_c.constraint_mode(0);


            repeat(10000) begin
                start_item(rd_wr_item);
                if((rd_wr_item.counter == 13 && rd_wr_item.mosi_data[9:8] != 2'b11)
                    || rd_wr_item.counter == 23) begin
                    rd_wr_item.mosi_data.rand_mode(1);
                    rd_wr_item.rd_wr_seq_c.constraint_mode(1);
                    end
                else begin
                    rd_wr_item.mosi_data.rand_mode(0);
                    rd_wr_item.rd_wr_seq_c.constraint_mode(0);
                end
                assert(rd_wr_item.randomize());
                finish_item(rd_wr_item);
            end

            start_item(rd_wr_item);

            finish_item(rd_wr_item);
        endtask
    endclass
endpackage
```

6-Driver

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

    class wrapper_driver extends uvm_driver #(wrapper_seq_item);
        `uvm_component_utils(wrapper_driver)
        wrapper_seq_item drv_item;
        virtual wrapper_if vif;

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

        task run_phase(uvm_phase phase);
            forever begin
                drv_item = wrapper_seq_item::type_id::create("drv_item");
                seq_item_port.get_next_item(drv_item);
                vif.rst_n = drv_item.rst_n;
                vif.SS_n     = drv_item.SS_n;
                vif.MOSI = drv_item.MOSI;
                @(negedge vif.clk);
                seq_item_port.item_done();
                `uvm_info("DRIVER RUN", drv_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask
    endclass
endpackage
```

7-Configuration object

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

class wrapper_config extends uvm_object;
    `uvm_object_utils(wrapper_config)

    virtual wrapper_if vif;

    uvm_active_passive_enum is_active;


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

8-Monitor

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

    class wrapper_monitor extends uvm_monitor;
        `uvm_component_utils(wrapper_monitor)
        wrapper_seq_item seq_item;
        virtual wrapper_if vif;

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                seq_item = wrapper_seq_item::type_id::create("seq_item");
                @(negedge vif.clk);
                seq_item.rst_n = vif.rst_n;
                seq_item.SS_n = vif.SS_n;
                seq_item.MOSI = vif.MOSI;
                seq_item.MISO = vif.MISO;
                `uvm_info("run_phase", seq_item.convert2string(), UVM_HIGH);
            end
        endtask
    endclass
endpackage
```

9-Agent

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

    class wrapper_agent extends uvm_agent;
        `uvm_component_utils(wrapper_agent)

        wrapper_driver driver;
        wrapper_monitor monitor;
        wrapper_sequencer sqr;
        wrapper_config cfg;

        uvm_analysis_port #(wrapper_seq_item) agt_ap;

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

```
    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        if(!uvm_config_db#(wrapper_config)::get(this, "", "WRAPPER CFG", cfg))
            `uvm_fatal("AGENT BUILD", "Failed to get CFG");

            if(cfg.is_active == UVM_ACTIVE) begin
                sqr    = wrapper_sequencer::type_id::create("sqr", this);
                driver = wrapper_driver::type_id::create("driver", this);
            end
            monitor = wrapper_monitor::type_id::create("monitor", this);
    endfunction

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

            if(cfg.is_active == UVM_ACTIVE) begin
                driver.seq_item_port.connect(sqr.seq_item_export);
                driver.vif = cfg.vif;
            end
            monitor.vif = cfg.vif;
    endfunction
  endclass
endpackage
```

10-Environment

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

    class wrapper_env extends uvm_env;
        `uvm_component_utils(wrapper_env)

        wrapper_agent agent;


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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            agent = wrapper_agent::type_id::create("agent", this);
        endfunction
    endclass
endpackage
```

11-Test

```
package wrapper_test_pkg;

    import uvm_pkg::*;
    import wrapper_seq_item_pkg::*;
    import wrapper_sequence_pkg::*;
    import wrapper_config_pkg::*;
    import slave_config_pkg::*;
    import ram_config_pkg::*;
    import wrapper_env_pkg::*;
    import slave_env_pkg::*;
    import ram_env_pkg::*;
    `include "uvm_macros.svh"

    class wrapper_test extends uvm_test;
        `uvm_component_utils(wrapper_test)

        wrapper_env wrapperenv;
        slave_env slaveenv;
        ram_env ramenv;

        wrapper_config wrapper_cfg;
        slave_config slave_cfg;
        ram_config ram_cfg;

        wrapper_rst_seq rst_seq;
        wrapper_wr_seq wr_seq;
        wrapper_rd_seq rd_seq;
        wrapper_rd_wr_seq wr_rd_seq;

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

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

            wrapperenv = wrapper_env::type_id::create("wrapperenv", this);
            slaveenv = slave_env::type_id::create("slaveenv", this);
            ramenv = ram_env::type_id::create("ramenv", this);

            wrapper_cfg = wrapper_config::type_id::create("wrapper_cfg");
            slave_cfg = slave_config::type_id::create("slave_cfg");
            ram_cfg = ram_config::type_id::create("ram_cfg");

            rst_seq = wrapper_rst_seq::type_id::create("rst_seq");
            wr_seq = wrapper_wr_seq::type_id::create("wr_seq");
            rd_seq = wrapper_rd_seq::type_id::create("rd_seq");
            wr_rd_seq = wrapper_rd_wr_seq::type_id::create("wr_rd_seq");

            if(!uvm_config_db #(virtual wrapper_if)::get(this, "", "WRAPPER VIF", wrapper_cfg.vif))
                `uvm_fatal("build_phase", "Test - Unable to get the virtual interface fo the SPI_wrapper form the uvm_config_db");

            if(!uvm_config_db #(virtual slave_if)::get(this, "", "SLAVE VIF", slave_cfg.vif))
                `uvm_fatal("build_phase", "Test - Unable to get the virtual interface fo the SPI_wrapper form the uvm_config_db");

            if(!uvm_config_db #(virtual ram_if)::get(this, "", "RAM VIF", ram_cfg.vif))
                `uvm_fatal("build_phase", "Test - Unable to get the virtual interface fo the SPI_wrapper form the uvm_config_db");

            uvm_config_db #(wrapper_config)::set(this, "*", "WRAPPER CFG", wrapper_cfg);
            uvm_config_db #(slave_config)::set(this, "*", "SLAVE CFG", slave_cfg);
            uvm_config_db #(ram_config)::set(this, "*", "RAM CFG", ram_cfg);

            wrapper_cfg.is_active = UVM_ACTIVE;
            slave_cfg.is_active = UVM_PASSIVE;
            ram_cfg.is_active = UVM_PASSIVE;
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            `uvm_info("run_phase_test", "Rest Asserted", UVM_LOW);
            rst_seq.start(wrapperenv.agent.sqr);
            `uvm_info("run_phase_test", "Rest Deasserted", UVM_LOW);

            `uvm_info("run_phase_test", "Write only operation is started", UVM_LOW);
            wr_seq.start(wrapperenv.agent.sqr);
            `uvm_info("run_phase_test", "Write only operation is ended", UVM_LOW);

            `uvm_info("run_phase_test", "read only operation is started", UVM_LOW);
            rd_seq.start(wrapperenv.agent.sqr);
            `uvm_info("run_phase_test", "read only operation is ended", UVM_LOW);

            `uvm_info("run_phase_test", "write and read operation is started", UVM_LOW);
            wr_rd_seq.start(wrapperenv.agent.sqr);
            `uvm_info("run_phase_test", "write and read operation is ended", UVM_LOW);
            phase.drop_objection(this);
        endtask
    endclass
endpackage
```

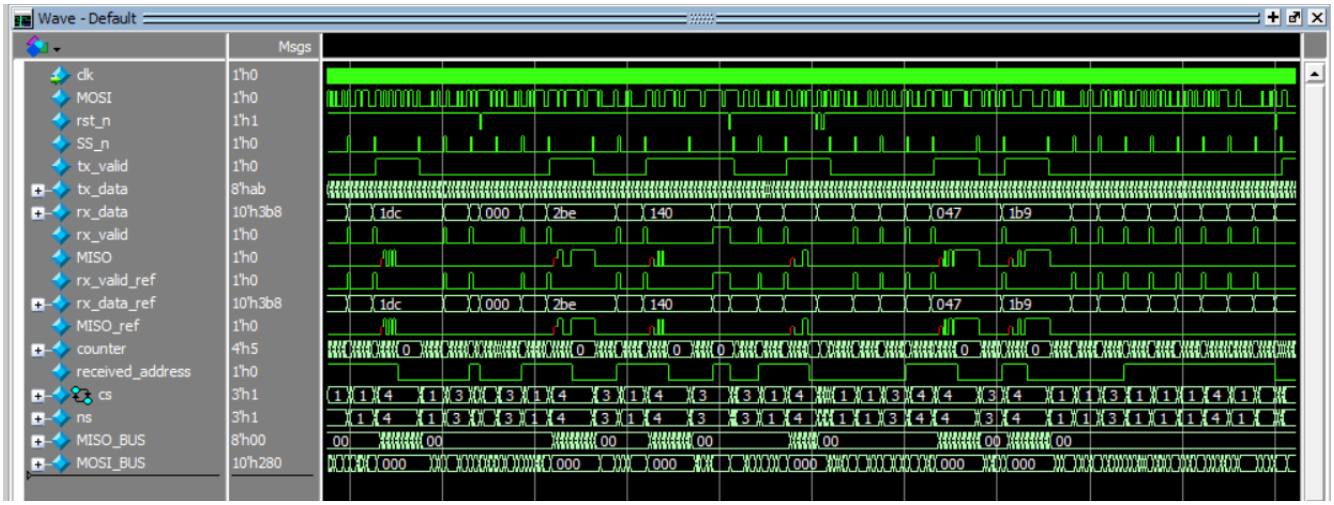
12-Top

```
import uvm_pkg::*;
import wrapper_test_pkg::*;
module wrapper_top();
    bit clk;

    always #1 clk = ~clk;

    wrapper_if wrapperif(clk);
    slave_if slaveif(clk);
    ram_if ramif(clk);
    WRAPPER dut(wrapperif);
    SLAVE slave_dut (slaveif);
    SLAVE_REF slave_ref (slaveif);

    assign wrapperif.MISO = slaveif.MISO;

    assign slaveif.MOSI = wrapperif.MOSI;
    assign slaveif.SS_n = wrapperif.SS_n;
    assign slaveif.rst_n = wrapperif.rst_n;
    assign slaveif.tx_valid = dut.tx_valid;
    assign slaveif.tx_data = dut.tx_data_dout;

    assign ramif.din = dut.ram_dut.din;
    assign ramif.rst_n = dut.ram_dut.rst_n;
    assign ramif.rx_valid = dut.ram_dut.rx_valid;
    assign ramif.clk = dut.ram_dut.clk;
    assign ramif.dout = dut.ram_dut.dout;
    assign ramif.tx_valid = dut.ram_dut.tx_valid;
    assign ramif.dout_ref = dut.ram_ref.dout;
    assign ramif.tx_valid_ref = dut.ram_ref.tx_valid;

    assign dut.rx_data_din = slaveif.rx_data;
    assign dut.rx_valid = slaveif.rx_valid;


    initial begin
        uvm_config_db #(virtual wrapper_if)::set(null, "uvm_test_top", "WRAPPER VIF", wrapperif);
        uvm_config_db #(virtual slave_if)::set(null, "uvm_test_top", "SLAVE VIF", slaveif);
        uvm_config_db #(virtual ram_if)::set(null, "uvm_test_top", "RAM VIF", ramif);
        run_test ("wrapper_test");
    end
endmodule
```

13-Do file

```
vlib work
vlog +define+SIM -f src_files_wrapper.list +cover -covercells
vsim -voptargs=+acc work.wrapper_top -cover
add wave -position insertpoint sim:/wrapper_top/slaveif/*
add wave -position insertpoint sim:/wrapper_top/ramif/*
add wave -position insertpoint sim:/wrapper_top/wrapperif/*
add wave -position insertpoint sim:/wrapper_top/slave_dut/*

add wave -position insertpoint sim:/wrapper_top/dut/ram_dut/*
add wave -position insertpoint sim:/wrapper_top/slave_dut.received_address
add wave -position insertpoint sim:/wrapper_top/slaveif/*
add wave -position insertpoint sim:/wrapper_top/ramif/*
add wave -position insertpoint sim:/wrapper_top/wrapperif/*
coverage save WRAPPER_COV.ucdb -onexit

coverage exclude -src RAM.sv -line 27 -code b
coverage exclude -src slave.sv -line 135 -code s
coverage exclude -src slave.sv -line 134 -code b

run -all
```

14-Wave form



15-Transcript

```
# UVM_INFO slave_scoreboard.sv(48) @ 20002: uvm_test_top.env.sb [SB REPORT] errors: 0
# UVM_INFO slave_scoreboard.sv(49) @ 20002: uvm_test_top.env.sb [SB REPORT] correct: 10001
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :10009
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SB REPORT]     2
# [SEQUENCE MAIN] 10000
# [TEST RUN]     3
# [TEST_DONE]     1
# ** Note: $finish    : F:/Program Files/Mentor_Graphics_QuestaSim_2021.1x64/win64/../verilog_src/uvm-1.
#    Time: 20002 ns  Iteration: 61  Instance: /slave_top
```

16-Coverage Report

-Code coverage

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         58        58         0   100.00%
```

-slave_dut

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

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

```
Statement Coverage:
    Enabled Coverage              Bins       Hits     Misses  Coverage
    ----------------              ----       ----     ------  --------
    Statements                      47         47          0   100.00%
```

```
Toggle Coverage:
    Enabled Coverage              Bins       Hits     Misses  Coverage
    ----------------              ----       ----     ------  --------
    Toggles                         58         58          0   100.00%
```

-RAM_dut

```
Branch Coverage:
    Enabled Coverage              Bins       Hits     Misses  Coverage
    ----------------              ----       ----     ------  --------
    Branches                         8          8          0   100.00%
```

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

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

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

-Assertion Coverage

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

Assertion Coverage:
    Assertions                       2         2         0   100.00%
--------------------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------------------
/wrapper_top/dut/rst_as
                     wrapper.sv(13)                      0          1
/wrapper_top/dut/MISO_as
                     wrapper.sv(16)                      0          1
```

-Functional Coverage

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

--------------------------------------------------------------------------------------------------------------
 TYPE /ram_coverage_pkg/ram_coverage/cov_gp            100.00%        100          -    Covered
    covered/total bins:                                    14         14          -
    missing/total bins:                                     0         14          -
    % Hit:                                            100.00%        100          -
    Coverpoint rx_valid_p                               0.00%        100          -    ZERO
        covered/total bins:                                 2          2          -
        missing/total bins:                                 0          2          -
        % Hit:                                        100.00%        100          -
    Coverpoint din_p_all                              100.00%        100          -    Covered
        covered/total bins:                                 4          4          -
        missing/total bins:                                 0          4          -
        % Hit:                                        100.00%        100          -
    Coverpoint all_transition                         100.00%        100          -    Covered
        covered/total bins:                                 3          3          -
        missing/total bins:                                 0          3          -
        % Hit:                                        100.00%        100          -
    Cross c1                                          100.00%        100          -    Covered
        covered/total bins:                                 4          4          -
        missing/total bins:                                 0          4          -
        % Hit:                                        100.00%        100          -
    Cross c2                                          100.00%        100          -    Covered
        covered/total bins:                                 1          1          -
        missing/total bins:                                 0          1          -
        % Hit:                                        100.00%        100          -
 Covergroup instance \/ram_coverage_pkg::ram_coverage::cov_gp
                                                      100.00%        100          -    Covered
    covered/total bins:                                    14         14          -
    missing/total bins:                                     0         14          -
    % Hit:                                            100.00%        100          -
    Coverpoint rx_valid_p [1]                         100.00%        100          -    Covered
        covered/total bins:                                 2          2          -
        missing/total bins:                                 0          2          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                     23724          1          -    Covered
        bin auto[1]                                      6278          1          -    Covered
    Coverpoint din_p_all                              100.00%        100          -    Covered
        covered/total bins:                                 4          4          -
        missing/total bins:                                 0          4          -
        % Hit:                                        100.00%        100          -
        bin auto[0]                                     11447          1          -    Covered
        bin auto[1]                                      5615          1          -    Covered
        bin auto[2]                                      6939          1          -    Covered
        bin auto[3]                                      6001          1          -    Covered
    Coverpoint all_transition                         100.00%        100          -    Covered
        covered/total bins:                                 3          3          -
        missing/total bins:                                 0          3          -
        % Hit:                                        100.00%        100          -
        bin wr_data_after_wr_address                      145          1          -    Covered
        bin rd_data_after_rd_address                       29          1          -    Covered
        bin all_transition                                  3          1          -    Covered
```

Cross c1	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[3],auto[1]>	1947	1	-	Covered
bin <auto[1],auto[1]>	895	1	-	Covered
bin <auto[2],auto[1]>	2528	1	-	Covered
bin <auto[0],auto[1]>	908	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin b1	23724		-	Occurred
Cross c2	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 b2	1947	1	-	Covered
TYPE /slave_coverage_pkg/slave_cov/g1	100.00%	100	-	Covered
covered/total bins:	14	14	-	
missing/total bins:	0	14	-	
% Hit:	100.00%	100	-	
Coverpoint rx_data	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Coverpoint ss_n	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
Coverpoint mosi_cp	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Cross sceniario	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Covergroup instance \/slave_coverage_pkg::slave_cov::g1				
	100.00%	100	-	Covered
covered/total bins:	14	14	-	
missing/total bins:	0	14	-	
% Hit:	100.00%	100	-	
Coverpoint rx_data	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
bin auto[0]	11447	1	-	Covered
bin auto[1]	5615	1	-	Covered
bin auto[2]	6939	1	-	Covered
bin auto[3]	6001	1	-	Covered
Coverpoint ss_n	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin normal	1484	1	-	Covered
bin Rd_dt	382	1	-	Covered
Coverpoint mosi_cp	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
bin write_add	5542	1	-	Covered
bin write_data	3533	1	-	Covered
bin READ_ADD	3448	1	-	Covered
bin READ_DATA	4737	1	-	Covered

Coverpoint mosi_cp	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
bin write_add	5542	1	-	Covered
bin write_data	3533	1	-	Covered
bin READ_ADD	3448	1	-	Covered
bin READ_DATA	4737	1	-	Covered
Cross sceniario	100.00%	100	-	Covered
covered/total bins:	4	4	-	
missing/total bins:	0	4	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <Rd_dt,READ_DATA>	195	1	-	Covered
bin <normal,READ_ADD>	382	1	-	Covered
bin <normal,write_data>	484	1	-	Covered
bin <normal,write_add>	618	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin illegal4	0		-	ZERO
ignore_bin illegal3	0		-	ZERO
ignore_bin illegal2	0		-	ZERO
ignore_bin illegal1	0		-	ZERO