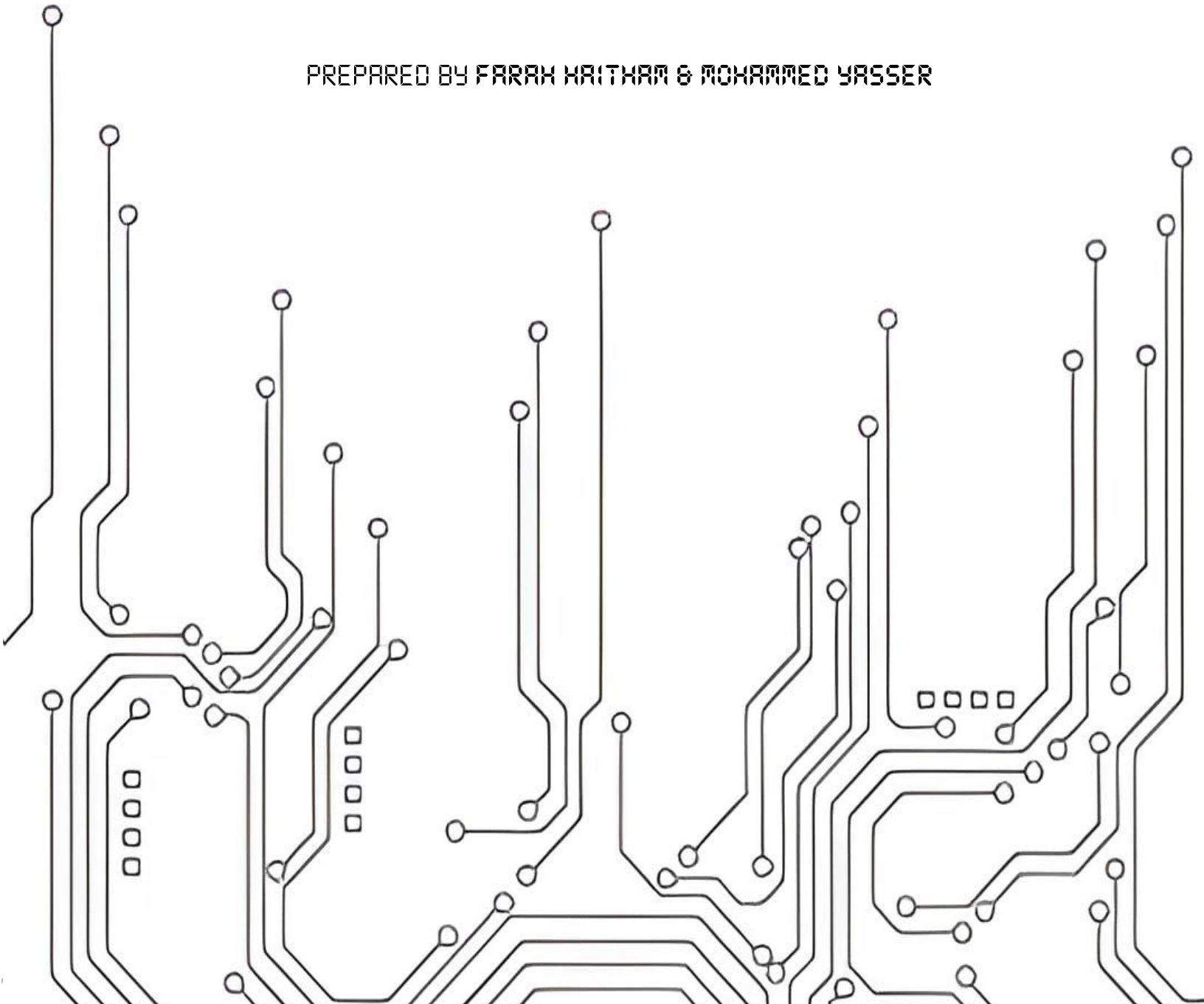
UVM-BASED VERIFICATION OF SPI SLAVE WITH SINGLE-PORT RAM

DIGITAL VERIFICATION DIPLOMA

Under the supervision of Eng. Kareem Wassem

PREPARED BY FARAH KAITHAM & MOHAMMED YASSER



PROJECT OVERVIEW

THIS PROJECT ENTAILS THE COMPREHENSIVE FUNCTIONAL VERIFICATION OF AN SPI SLAVE MODULE INTERFACED WITH A SINGLE PORT RAM MEMORY BLOCK USING THE UNIVERSAL VERIFICATION METHODOLOGY (UVM) FRAMEWORK. THE MAIN OBJECTIVE WAS TO ENSURE ROBUST AND RELIABLE OPERATION OF THE SPI SLAVE IN CONJUNCTION WITH MEMORY READ AND WRITE TRANSACTIONS, UNDER VARIED PROTOCOL SCENARIOS AND TIMING CONDITIONS.

THE VERIFICATION ENVIRONMENT WAS STRUCTURED MODULARLY AROUND UVM COMPONENTS INCLUDING AGENTS, DRIVERS, MONITORS, SEQUENCERS, AND SCOREBOARDS. SEPARATE AGENTS WERE DEVELOPED FOR THE SPI SLAVE AND THE RAM, FACILITATING INDEPENDENT AND COORDINATED STIMULUS GENERATION AND MONITORING. A GOLDEN REFERENCE MODEL WAS IMPLEMENTED TO VERIFY THE CORRECTNESS OF OUTPUT TRANSACTIONS BY COMPARING OUT BEHAVIOR AGAINST EXPECTED OUTPUTS.

RANDOMIZED TEST SEQUENCES WERE UTILIZED TO SIMULATE REALISTIC DATA TRANSFERS, INCLUDING WRITES AND READS VIA THE SPI INTERFACE TO RAM. FUNCTIONAL COVERAGE AND CODE COVERAGE METRICS WERE USED RIGOROUSLY TO IDENTIFY GAPS IN VERIFICATION AND ENSURE THOROUGH VALIDATION. KEY DESIGN BUGS DISCOVERED DURING SIMULATION WERE DEBUGGED AND RESOLVED, DEMONSTRATING THE EFFECTIVENESS OF THE ENVIRONMENT.

ULTIMATELY, SIMULATION RESULTS DEMONSTRATE CORRECT SPI SLAVE BEHAVIOR, ACCURATE SINGLE PORT RAM INTERFACING, AND RELIABLE DATA COMMUNICATION, CONFIRMING THE ROBUSTNESS OF THE DESIGN COMPONENTS UNDER DIVERSE OPERATIONAL CONDITIONS.

SECTION #1: VERIFICATION OF SINGLE PORT RAM

1. VERFICATION PLAN

	A	B	C	D	E
1	**Label**	**Design Requirement Description**	**Stimulus Generation**	**Functional Coverage**	**Functionality Check**
2	RAM_1	When the reset is asserted, the outputs (dout & tx_valid) should be zero	Directed at the start of the simulation then randomized with constrains to be off most of the time	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
3	RAM_2	In the write only sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b00, then write operation will happen and din[7:0] will be stored inside the wr_addr and tx_valid should be low	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to always be 2'b00 or 2'b01	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
4	RAM_3	In the write only sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b01, then write operation will happen and din[7:0] will be stored inside the memory using wr_addr as address and tx_valid should be low	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to always be 2'b00 or 2'b01	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
5	RAM_4	In the read only sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b10, then write operation will happen and din[7:0] will be stored inside rd_addr and tx_valid should be high	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to always be 2'b10 or 2'b11	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
6	RAM_5	In the read only sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b10, then read operation will happen as the dout should be the value of the memory at address equal to rd_addr and tx_valid should be low	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to always be 2'b10 or 2'b12	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
	In the write read sequence, when the	Randomization with constraints on rx_valid to be high most of	Coverage on the values of	The outputs are checked using both golden model and

Plan | Assertions | +

Ready | Accessibility: Good to go

	A	B	C	D	E
6				din[9:8] and tx_valid.	
7	RAM_6	In the write read sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b00, then write operation will happen and din[7:0] will be stored inside the wr_addr and tx_valid should be low	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to value based on the last operation, if the last operation was WRITE_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was WRITE DATA then din[9:8] can be 60% READ_ADDR or 40%WRITE_ADDR, if the last operation was READ_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was READ_DATA then din[9:8] can be 60% WRITE_ADDR or 40%READ_ADDR	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
8	RAM_7	In the write read sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b01, then write operation will happen and din[7:0] will be stored inside the memory using wr_addr as address and tx_valid should be low	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to value based on the last operation, if the last operation was WRITE_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was WRITE DATA then din[9:8] can be 60% READ_ADDR or 40%WRITE_ADDR, if the last operation was READ_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was READ_DATA then din[9:8] can be 60% WRITE_ADDR or 40%READ_ADDR	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
9	RAM_8	In the write read sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b10, then write operation will happen and din[7:0] will be stored inside rd_addr and tx_valid should be low	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to value based on the last operation, if the last operation was WRITE_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was WRITE DATA then din[9:8] can be 60% READ_ADDR or 40%WRITE_ADDR, if the last operation was READ_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was READ_DATA then din[9:8] can be 60% WRITE_ADDR or 40%READ_ADDR	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
10	RAM_9	In the write read sequence, when the rx_valid is asserted and the din[9:8] is equal to 2'b10, then read operation will happen as the dout should be the value of the memory at address equal to rd_addr and tx_valid should be high	Randomization with constraints on rx_valid to be high most of the time and din[9:8] to value based on the last operation, if the last operation was WRITE_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was WRITE DATA then din[9:8] can be 60% READ_ADDR or 40%WRITE_ADDR, if the last operation was READ_ADDR then din[9:8] can be 2'b00 or 2'b01, if the last op was READ_DATA then din[9:8] can be 60% WRITE_ADDR or 40%READ_ADDR	Coverage on the values of inputs rx_valid, din[9:8] and the value of the output tx_valid Cross coverage between din[9:8] and rx_valid. Cross coverage between din[9:8] and tx_valid.	The outputs are checked using both golden model and assertions
11					

Plan | Assertions | +

Ready | Accessibility: Good to go

2. CODE SNIPPETS

Interface

RAM_if.sv

```sv
interface RAM_IF(clk);
    input            clk;
    logic      [9:0] din;
    logic rst_n, rx_valid;

    logic [7:0] dout;
    logic       tx_valid;

    logic [7:0] dout_ref;
    logic       tx_valid_ref;
endinterface //RAM_IF
```

Design

RAM.v

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

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

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

reg [7:0] MEM [255:0];

reg [7:0] Rd_Addr, Wr_Addr;

always @(posedge clk) begin
    if (~rst_n) begin
        dout <= 0;
        tx_valid <= 0;
        Rd_Addr <= 0;
        Wr_Addr <= 0;
    end
    else
        if (rx_valid) begin
            case (din[9:8])
                2'b00 : Wr_Addr <= din[7:0];
                2'b01 : MEM[Wr_Addr] <= din[7:0];
                2'b10 : Rd_Addr <= din[7:0];
                2'b11 : dout <= MEM[Rd_Addr]; // read addr instead of write addr //M2
                default : dout <= 0;
            endcase
            tx_valid <= (din[9] && din[8])? 1'b1 : 1'b0; //Begin - end //M1
        end

end

endmodule
```

Golden Model

RAM_ref.v

```v
module RAM_ref #(
    parameter ADDR_SIZE = 8,
    parameter MEM_DEPTH = 256
) (
    input clk, rst_n, rx_valid,
    input[9:0] rx_data,
    output reg tx_valid,
    output reg[7:0] tx_data
);

reg[ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];
reg[ADDR_SIZE-1:0] wr_address, rd_address;

always @(posedge clk) begin
    if(~rst_n) begin
        tx_valid <= 0;
        tx_data <= 0;
        rd_address <= 0;
        wr_address <= 0;
    end
    else begin
        if(rx_valid) begin
            if(rx_data[9:8] == 2'b00) begin
                wr_address <= rx_data[7:0];
                tx_valid <= 0;
            end
            else if(rx_data[9:8] == 2'b01) begin
                mem[wr_address] <= rx_data[7:0];
                tx_valid <= 0;
            end
            else if(rx_data[9:8] == 2'b10) begin
                rd_address <= rx_data[7:0];
                tx_valid <= 0;
            end
            else begin
                tx_data <= mem[rd_address];
                tx_valid <= 1;
            end
        end
    end
end

endmodule
```

SVA

RAM_sva.sv

```
import shared_pkg::*;
module RAM_sva(din,clk,rst_n,rx_valid,dout,tx_valid);
    input      [9:0] din;
    input            clk, rst_n, rx_valid;

    input  [7:0] dout;
    input       tx_valid;

    op_e op;

    always_comb begin
        case (din[9:8])
            2'b00 : op = WRITE_ADDR;
            2'b01 : op = WRITE_DATA;
            2'b10 : op = READ_ADDR;
            2'b11 : op = READ_DATA;
        endcase
    end

//RAM_1 // RAM_2, RAM_3 //RAM_4, RAM_5 //RAM_6, RAM_7, RAM_8, RAM_9

//Assertions
    a_reset_txValid: assert property (@(posedge clk) !rst_n |->##1  tx_valid == 1'b0);
    a_reset_dout: assert property (@(posedge clk) !rst_n |-> ##1 dout == 7'b0);

    a_wr_addr: assert property (@(posedge clk) disable iff (!rst_n) op == WRITE_ADDR && rx_valid |-> ##1 ~tx_valid );
    a_wr_data: assert property (@(posedge clk) disable iff (!rst_n) op == WRITE_DATA && rx_valid |-> ##1 ~tx_valid );
    a_rd_addr: assert property (@(posedge clk) disable iff (!rst_n) op == READ_ADDR && rx_valid |-> ##1 ~tx_valid );

    a_rd_data: assert property (@(posedge clk) disable iff (!rst_n) (op == READ_DATA && rx_valid && !is_readOnly) |->##1 $rose(tx_valid) ##[1:$] $fell(tx_valid));

    a_wr_addr_to_wr_data: assert property (@(posedge clk) disable iff (!rst_n) (op == WRITE_ADDR && rx_valid) |-> (##[1:$] op == WRITE_DATA && rx_valid));
    a_rd_addr_to_rd_data: assert property (@(posedge clk) disable iff (!rst_n) (op == READ_ADDR && rx_valid) |-> (##[1:$] op == READ_DATA && rx_valid));

//Coverage
    a_reset_txValid_cvr: cover property (@(posedge clk) !rst_n |->##1  tx_valid == 1'b0);
    a_reset_dout_cvr: cover property (@(posedge clk) !rst_n |-> ##1 dout == 7'b0);

    a_wr_addr_cvr: cover property (@(posedge clk) disable iff (!rst_n) op == WRITE_ADDR |-> ##1 ~tx_valid );
    a_wr_data_cvr: cover property (@(posedge clk) disable iff (!rst_n) op == WRITE_DATA |-> ##1 ~tx_valid );
    a_rd_addr_cvr: cover property (@(posedge clk) disable iff (!rst_n) op == READ_ADDR |-> ##1 ~tx_valid );

    a_rd_data_cvr: cover property (@(posedge clk) disable iff (!rst_n) (op == READ_DATA && rx_valid && !is_readOnly) |-> ##1 $rose(tx_valid) ##[1:$]  $fell(tx_valid));

    a_wr_addr_to_wr_data_cvr: cover property (@(posedge clk) disable iff (!rst_n) (op == WRITE_ADDR && rx_valid) |-> (##[1:$] op == WRITE_DATA && rx_valid));
    a_rd_addr_to_rd_data_cvr: cover property (@(posedge clk) disable iff (!rst_n) (op == READ_ADDR && rx_valid) |-> (##[1:$] op == READ_DATA && rx_valid));



endmodule
```

CONFIG

RAM_config.sv

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

        virtual RAM_IF ram_if;
        uvm_active_passive_enum is_active;

        function new(string name = "RAM_config");
            super.new(name);
        endfunction //new()
    endclass //RAM_config extends uvm_object
endpackage
```

Top

RAM_top.sv

```
import uvm_pkg::*;
`include"uvm_macros.svh"
import RAM_test_pkg::*;
module RAM_top;
    logic clk = 0;
    initial begin
        $readmemb("mem.dat", DUT.MEM);
        $readmemb("mem.dat", GM.mem);
        forever #1 clk = ~clk;
    end
    RAM_IF ram_if (clk);

    RAM DUT (
        .clk(ram_if.clk),
        .rst_n(ram_if.rst_n),
        .rx_valid(ram_if.rx_valid),
        .din(ram_if.din),
        .tx_valid(ram_if.tx_valid),
        .dout(ram_if.dout)
    );

    RAM_ref GM (
        .clk(ram_if.clk),
        .rst_n(ram_if.rst_n),
        .rx_valid(ram_if.rx_valid),
        .rx_data(ram_if.din),
        .tx_valid(ram_if.tx_valid_ref),
        .tx_data(ram_if.dout_ref)
    );

    bind DUT RAM_sva SVA (
        .clk(ram_if.clk),
        .rst_n(ram_if.rst_n),
        .rx_valid(ram_if.rx_valid),
        .din(ram_if.din),
        .tx_valid(ram_if.tx_valid),
        .dout(ram_if.dout)
    );

    initial begin
        uvm_config_db #(virtual RAM_IF)::set(null, "uvm_test_top", "RAM_IF", ram_if);
        run_test("RAM_test");
    end

endmodule
```

Test

RAM_test.sv

```
package RAM_test_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import shared_pkg::*;
    import RAM_config_pkg::*;
    import RAM_sequence_pkg::*;
    import RAM_seq_item_pkg::*;
    import RAM_env_pkg::*;
    class RAM_test extends uvm_test;
        `uvm_component_utils(RAM_test)

        RAM_env ram_env;

        RAM_config ram_cfg;

        RAM_reset_seq ram_reset;
        RAM_write_only_seq ram_write_only;
        RAM_read_only_seq ram_read_only;
        RAM_write_read_seq ram_write_read;

        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);
            ram_cfg = RAM_config::type_id::create("ram_cfg", this);
            ram_env = RAM_env::type_id::create("ram_env",this);

            ram_reset = RAM_reset_seq::type_id::create("ram_reset");
            ram_write_only = RAM_write_only_seq::type_id::create("ram_write_only");
            ram_read_only = RAM_read_only_seq::type_id::create("ram_read_only");
            ram_write_read = RAM_write_read_seq::type_id::create("ram_write_read");

            ram_cfg.is_active = UVM_ACTIVE;
            if (!uvm_config_db #(virtual RAM_IF)::get(this,"","RAM_IF",ram_cfg.ram_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the RAM interface");
            uvm_config_db #(RAM_config)::set(this, "*", "CFG_RAM", ram_cfg);

        endfunction

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

RAM_test.sv

```
package RAM_test_pkg;
    class RAM_test extends uvm_test;
        function void build_phase(uvm_phase phase);
            ram_cfg = RAM_config::type_id::create("ram_cfg", this);
            ram_env = RAM_env::type_id::create("ram_env",this);

            ram_reset = RAM_reset_seq::type_id::create("ram_reset");
            ram_write_only = RAM_write_only_seq::type_id::create("ram_write_only");
            ram_read_only = RAM_read_only_seq::type_id::create("ram_read_only");
            ram_write_read = RAM_write_read_seq::type_id::create("ram_write_read");

            ram_cfg.is_active = UVM_ACTIVE;
            if (!uvm_config_db #(virtual RAM_IF)::get(this,"","RAM_IF",ram_cfg.ram_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the RAM interface");
            uvm_config_db #(RAM_config)::set(this, "*", "CFG_RAM", ram_cfg);

        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);
            //RAM_1
            `uvm_info("run_phase", "Reset Asserted", UVM_LOW);
            ram_reset.start(ram_env.ram_agt.ram_sqr);
            `uvm_info("run_phase", "Reset Deasserted", UVM_LOW);
            // RAM_2, RAM_3
            `uvm_info("run_phase", "Write Only Seq Started", UVM_LOW);
            ram_write_only.start(ram_env.ram_agt.ram_sqr);
            `uvm_info("run_phase", "Write Only Seq Ended", UVM_LOW);
            //RAM_4, RAM_5
            `uvm_info("run_phase", "Read Only Seq Started", UVM_LOW);
            is_readOnly = 1;
            ram_read_only.start(ram_env.ram_agt.ram_sqr);
            is_readOnly = 0;
            `uvm_info("run_phase", "Read Only Seq Ended", UVM_LOW);
            //RAM_6, RAM_7, RAM_8, RAM_9
            `uvm_info("run_phase", "Write Read Seq Started", UVM_LOW);
            ram_write_read.start(ram_env.ram_agt.ram_sqr);
            `uvm_info("run_phase", "Write Read Seq Ended", UVM_LOW);

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

ENVIRONMENT

RAM_env.sv

```
package RAM_env_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import RAM_agent_pkg::*;
    import RAM_scoreboard_pkg::*;
    import RAM_coverage_pkg::*;
    class RAM_env extends uvm_env;
        `uvm_component_utils(RAM_env)

        RAM_agent ram_agt;
        RAM_scoreboard ram_sb;
        RAM_coverage ram_cvg;

        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);
            ram_agt = RAM_agent::type_id::create("ram_agt", this);
            ram_sb= RAM_scoreboard::type_id::create("ram_sb", this);
            ram_cvg = RAM_coverage::type_id::create("ram_cvg", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            ram_agt.agt_ap.connect(ram_sb.sb_axp);
            ram_agt.agt_ap.connect(ram_cvg.cvg_axp);
        endfunction


    endclass
endpackage
```

COVERAGE

RAM_coverage.sv

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

    import RAM_seq_item_pkg::*;
    import shared_pkg::*;
    class RAM_coverage extends uvm_component;
        `uvm_component_utils(RAM_coverage)

        uvm_analysis_export #(RAM_seq_item) cvg_axp;
        uvm_tlm_analysis_fifo #(RAM_seq_item) cvg_fifo;

        RAM_seq_item cvg_seq_item;
//RAM_1 // RAM_2, RAM_3 //RAM_4, RAM_5 //RAM_6, RAM_7, RAM_8, RAM_9
        covergroup RAM_CVG;
            rx_valid_cp: coverpoint cvg_seq_item.rx_valid{
                bins Bin_rxValid_High = {1'b1};
                bins Bin_rxValid_Low = {1'b0};
            }
            tx_valid_cp: coverpoint cvg_seq_item.tx_valid{
                bins Bin_txValid_High = {1'b1};
                bins Bin_txValid_Low = {1'b0};
            }
           operation_cp: coverpoint cvg_seq_item.din[9:8]{
                bins Bin_Write_Addr = {WRITE_ADDR};
                bins Bin_Write_Data = {WRITE_DATA};
                bins Bin_Read_Addr = {READ_ADDR};
                bins Bin_Read_Data = {READ_DATA};
                bins Bin_Write_AddrToData = (WRITE_ADDR=>WRITE_DATA);
                bins Bin_Read_AddrToData = (READ_ADDR=>READ_DATA);
                bins Bin_Trans = (0 => 1), (1 => 3), (1 => 0), (0 => 2),
                                 (2 => 3), (2 => 0), (3 => 1), (3 => 2);
            }

            OP_rxValid_cross: cross operation_cp, rx_valid_cp {
                option.cross_auto_bin_max = 0;
                bins Bin_Write_Addr_rxValidHigh  = binsof(operation_cp.Bin_Write_Addr) && binsof(rx_valid_cp.Bin_rxValid_High);
                bins Bin_Write_Data_rxValidHigh  = binsof(operation_cp.Bin_Write_Data) && binsof(rx_valid_cp.Bin_rxValid_High);
                bins Bin_Read_Addr_rxValidHigh   = binsof(operation_cp.Bin_Read_Addr) && binsof(rx_valid_cp.Bin_rxValid_High);
                bins Bin_Read_Data_rxValidHigh   = binsof(operation_cp.Bin_Read_Data) && binsof(rx_valid_cp.Bin_rxValid_High);
            }
            Read_OP_cross: cross operation_cp,  tx_valid_cp {
                option.cross_auto_bin_max = 0;
                bins Bin_out = binsof(operation_cp) intersect {READ_DATA} && binsof(tx_valid_cp) intersect {1};
            }
```

RAM_coverage.sv

```
//RAM_1 // RAM_2, RAM_3 //RAM_4, RAM_5 //RAM_6, RAM_7, RAM_8, RAM_9
        covergroup RAM_CVG;
            operation_cp: coverpoint cvg_seq_item.din[9:8]{
            }

            OP_rxValid_cross: cross operation_cp, rx_valid_cp {
                option.cross_auto_bin_max = 0;
                bins Bin_Write_Addr_rxValidHigh  = binsof(operation_cp.Bin_Write_Addr) && binsof(rx_valid_cp.Bin_rxValid_High);
                bins Bin_Write_Data_rxValidHigh  = binsof(operation_cp.Bin_Write_Data) && binsof(rx_valid_cp.Bin_rxValid_High);
                bins Bin_Read_Addr_rxValidHigh   = binsof(operation_cp.Bin_Read_Addr) && binsof(rx_valid_cp.Bin_rxValid_High);
                bins Bin_Read_Data_rxValidHigh   = binsof(operation_cp.Bin_Read_Data) && binsof(rx_valid_cp.Bin_rxValid_High);
            }
            Read_OP_cross: cross operation_cp,  tx_valid_cp {
                option.cross_auto_bin_max = 0;
                bins Bin_out = binsof(operation_cp) intersect {READ_DATA} && binsof(tx_valid_cp) intersect {1};
            }
        endgroup
        function new (string name = "RAM_coverage", uvm_component parent = null);
            super.new(name, parent);
            RAM_CVG = new;
        endfunction

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

            cvg_axp = new ("cvg_axp", this);
            cvg_fifo = new ("cvg_fifo", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cvg_axp.connect(cvg_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cvg_fifo.get(cvg_seq_item);
                RAM_CVG.sample();
            end
        endtask
    endclass
endpackage
```


Scoreboard

RAM_scoreboard.sv

```
package RAM_scoreboard_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import RAM_seq_item_pkg::*;
    class RAM_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(RAM_scoreboard)

        logic [7:0] dout_ref;
        logic       tx_valid_ref;

        logic [7:0] Rd_Addr, Wr_Addr;

        logic [7:0] MEM_ref [logic[7:0]];

        uvm_analysis_export #(RAM_seq_item) sb_axp;
        uvm_tlm_analysis_fifo #(RAM_seq_item) sb_fifo;

        RAM_seq_item sb_seq_item;

        int error_cnt = 0, correct_cnt = 0;

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

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

        endfunction

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(sb_seq_item);
                // ref_model(sb_seq_item);
                if (
                    sb_seq_item.tx_valid_ref != sb_seq_item.tx_valid ||
                    sb_seq_item.dout_ref != sb_seq_item.dout
```

RAM_scoreboard.sv

```
package RAM_scoreboard_pkg;
    class RAM_scoreboard extends uvm_scoreboard;
        function new (string name = "RAM_scoreboard", uvm_component parent = null);
        endfunction

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

        endfunction

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(sb_seq_item);
                // ref_model(sb_seq_item);
                if (
                    sb_seq_item.tx_valid_ref != sb_seq_item.tx_valid ||
                    sb_seq_item.dout_ref != sb_seq_item.dout
                ) begin
                    error_cnt++;
                    `uvm_error("run_phase", $sformatf("Comparison Failed - Transaction Received DUT : %s, Ref Model : tx_valid = %b, dout = %h", sb_seq_item.convert2string,
                                                      sb_seq_item.tx_valid_ref, sb_seq_item.dout_ref));
                end
                else begin
                    correct_cnt++;
                    `uvm_info("run_phase", $sformatf("Correct RAM Comparison: %s", sb_seq_item.convert2string),UVM_HIGH);
                end
            end
        endtask

        function void report_phase (uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Correct Tests = %d, Error Tests = %d", correct_cnt, error_cnt),UVM_MEDIUM)

        endfunction

    endclass
endpackage
```


Agent

RAM_agent.sv

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

        RAM_config ram_cfg;

        RAM_sequencer ram_sqr;
        RAM_monitor ram_mon;
        RAM_driver ram_dvr;

        uvm_analysis_port #(RAM_seq_item) agt_ap;

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

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

            if (!uvm_config_db #(RAM_config)::get(this,"","CFG_RAM",ram_cfg))
                `uvm_fatal("build_phase", "Error - Agent cannot retrieve the config object");

            ram_mon = RAM_monitor::type_id::create("ram_mon", this);

            if (ram_cfg.is_active == UVM_ACTIVE) begin
                ram_dvr = RAM_driver::type_id::create("ram_dvr", this);
                ram_sqr = RAM_sequencer::type_id::create("ram_sqr", this);
            end

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

        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase);
            if(ram_cfg.is_active == UVM_ACTIVE) begin
                ram_dvr.ram_dvr_if = ram_cfg.ram_if;
                ram_dvr.seq_item_port.connect(ram_sqr.seq_item_export);
            end
```



```
            end
            ram_mon.ram_mon_if = ram_cfg.ram_if;
            ram_mon.mon_ap.connect(agt_ap);
        endfunction
    endclass
endpackage
```

Driver

RAM_driver.sv

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

        RAM_seq_item dvr_seq_item;
        virtual RAM_IF ram_dvr_if;

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

        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                dvr_seq_item = RAM_seq_item::type_id::create("dvr_seq_item");
                seq_item_port.get_next_item(dvr_seq_item);
                ram_dvr_if.rst_n = dvr_seq_item.rst_n;
                ram_dvr_if.rx_valid = dvr_seq_item.rx_valid;
                ram_dvr_if.din = dvr_seq_item.din;
                repeat(1)@(negedge ram_dvr_if.clk);
                seq_item_port.item_done();
                `uvm_info("run_phase", dvr_seq_item.convert2string_stimulus, UVM_HIGH);
            end
        endtask
    endclass
endpackage
```

Monitor

RAM_monitor.sv

```
package RAM_monitor_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import RAM_seq_item_pkg::*;
    class RAM_monitor extends uvm_monitor;
        `uvm_component_utils(RAM_monitor)

        virtual RAM_IF ram_mon_if;
        RAM_seq_item mon_seq_item;

        uvm_analysis_port #(RAM_seq_item) mon_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                mon_seq_item = RAM_seq_item::type_id::create("mon_seq_item");
                repeat(1)@(negedge ram_mon_if.clk);
                mon_seq_item.rst_n = ram_mon_if.rst_n;
                mon_seq_item.rx_valid = ram_mon_if.rx_valid;
                mon_seq_item.din = ram_mon_if.din;
                mon_seq_item.dout = ram_mon_if.dout;
                mon_seq_item.tx_valid = ram_mon_if.tx_valid;
                mon_seq_item.dout_ref = ram_mon_if.dout_ref;
                mon_seq_item.tx_valid_ref = ram_mon_if.tx_valid_ref;
                mon_ap.write(mon_seq_item);
                `uvm_info("run_phase", mon_seq_item.convert2string, UVM_HIGH);
            end
        endtask


    endclass
endpackage
```

Sequencer

RAM_sequencer.sv

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

Sequence

RAM_sequence.sv

```
package RAM_sequence_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import shared_pkg::*;
    import RAM_seq_item_pkg::*;
    class RAM_reset_seq extends uvm_sequence #(RAM_seq_item);
        `uvm_object_utils(RAM_reset_seq)

        RAM_seq_item ram_seq;

        function new(string name = "RAM_reset_seq");
            super.new(name);
        endfunction
    // RAM_1
        virtual task body();
            ram_seq = RAM_seq_item::type_id::create("ram_seq");
            start_item(ram_seq);
            ram_seq.rst_n = 0;
            ram_seq.rx_valid = 0;
            ram_seq.din = 0;
            finish_item(ram_seq);
        endtask
    endclass
    class RAM_write_only_seq extends uvm_sequence #(RAM_seq_item);
        `uvm_object_utils(RAM_write_only_seq)

        RAM_seq_item ram_seq;

        function new(string name = "RAM_write_only_seq");
            super.new(name);
        endfunction
// RAM_2, RAM_3
        virtual task body();
            repeat(10000)begin
                ram_seq = RAM_seq_item::type_id::create("ram_seq");
                start_item(ram_seq);
                ram_seq.read_only_const.constraint_mode(0);
                ram_seq.write_read_const.constraint_mode(0);
                assert(ram_seq.randomize);
                finish_item(ram_seq);
            end
        endtask
    endclass
    class RAM_read_only_seq extends uvm_sequence #(RAM_seq_item);
        `uvm_object_utils(RAM_read_only_seq)
```

RAM_sequence.sv

```
// RAM_2, RAM_3
        endtask
    endclass
    class RAM_read_only_seq extends uvm_sequence #(RAM_seq_item);
        `uvm_object_utils(RAM_read_only_seq)

        RAM_seq_item ram_seq;

        function new(string name = "RAM_read_only_seq");
            super.new(name);
        endfunction
//RAM_4, RAM_5
        virtual task body();
            repeat(10000)begin
                ram_seq = RAM_seq_item::type_id::create("ram_seq");
                start_item(ram_seq);
                ram_seq.write_only_const.constraint_mode(0);
                ram_seq.write_read_const.constraint_mode(0);
                assert(ram_seq.randomize);
                finish_item(ram_seq);
            end
        endtask
    endclass
    class RAM_write_read_seq extends uvm_sequence #(RAM_seq_item);
        `uvm_object_utils(RAM_write_read_seq)

        RAM_seq_item ram_seq;

        function new(string name = "RAM_write_read_seq");
            super.new(name);
        endfunction
//RAM_6, RAM_7, RAM_8, RAM_9
        virtual task body();
            repeat(10000)begin
                ram_seq = RAM_seq_item::type_id::create("ram_seq");
                start_item(ram_seq);
                ram_seq.write_only_const.constraint_mode(0);
                ram_seq.read_only_const.constraint_mode(0);
                assert(ram_seq.randomize);
                finish_item(ram_seq);
            end
        endtask
    endclass
endpackage
```

Sequence Item

RAM_seq_item.sv

```
package RAM_seq_item_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import shared_pkg::*;
    class RAM_seq_item extends uvm_sequence_item;
        `uvm_object_utils(RAM_seq_item)
    // Signals Delarations
        rand logic      [9:0] din;
        rand logic rst_n, rx_valid;

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

        op_e wr_op[] = '{WRITE_ADDR, WRITE_DATA};
        op_e rd_op[] = '{READ_ADDR, READ_DATA};

        bit wr_addr, wr_data, rd_addr, rd_data;

        rand op_e op_rd_data, op_wr_data;

        static op_e last_op;
    // Functions
        function new(string name = "RAM_seq_item");
            super.new(name);
        endfunction

        function string convert2string();
            return $sformatf("%s - reset = %b, rx_valid = %b, din = %h ---- tx_valid = %b, dout = %h", super.convert2string, rst_n, rx_valid, din, tx_valid, dout);
        endfunction
        function string convert2string_stimulus();
            return $sformatf("%s - reset = %b, rx_valid = %b, din = %h", super.convert2string, rst_n, rx_valid, din);
        endfunction
        function void post_randomize();
            last_op = op_e'(din[9:8]);
        endfunction
    //Constraints
    //RAM_1
    constraint reset_const{
        rst_n dist {1:/90 , 0:/10};
    };
    // RAM_2, RAM_3
    constraint rx_valid_const{
        rx_valid dist {1:/90 , 0:/10};
```

RAM_seq_item.sv

```
package RAM_seq_item_pkg;
    // Functions
        function new(string name = "RAM_seq_item");
            super.new(name);
        endfunction

        function string convert2string();
            return $sformatf("%s - reset = %b, rx_valid = %b, din = %h ---- tx_valid = %b, dout = %h", super.convert2string, rst_n, rx_valid, din, tx_valid, dout);
        endfunction
        function string convert2string_stimulus();
            return $sformatf("%s - reset = %b, rx_valid = %b, din = %h", super.convert2string, rst_n, rx_valid, din);
        endfunction
        function void post_randomize();
            last_op = op_e'(din[9:8]);
        endfunction
    //Constraints
    //RAM_1
    constraint reset_const{
        rst_n dist {1:/90 , 0:/10};
    };
    // RAM_2, RAM_3
    constraint rx_valid_const{
        rx_valid dist {1:/90 , 0:/10};
    };
    // RAM_2, RAM_3 //
    constraint write_only_const{
        din[9:8] inside {wr_op};
    };
    //RAM_4, RAM_5
    constraint read_only_const{
        din[9:8] inside {rd_op};
    };
    //RAM_6, RAM_7, RAM_8, RAM_9
    constraint write_read_const{
        op_rd_data dist {WRITE_ADDR:=60 , READ_ADDR:=40};
        op_wr_data dist {READ_ADDR:=60 , WRITE_ADDR:=40};

        (last_op == READ_ADDR) -> din[9:8] inside {rd_op};
        (last_op == READ_DATA) -> din[9:8] == op_rd_data;
        (last_op == WRITE_ADDR) -> din[9:8] inside {wr_op};
        (last_op == WRITE_DATA) -> din[9:8] == op_wr_data;
    };
    endclass
endpackage
```


DO FILE

run_ram.do

```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.RAM_top -classdebug -uvmcontrol=all -cover
add wave /RAM_top/ram_if/*

run 0
add wave -position insertpoint  \
sim:/RAM_top/DUT/MEM \
sim:/RAM_top/DUT/Rd_Addr \
sim:/RAM_top/DUT/Wr_Addr
add wave -position insertpoint  \
sim:/shared_pkg::is_readOnly

coverage exclude -src RAM.v -line 27 -code s
coverage exclude -src RAM.v -line 27 -code b

coverage save RAM.ucdb -onexit

run -all


```

3. BUG REPORT

Bug #1: at the case din[9:8] = 2'b1, using wr_addr instead of rd_addr

before

```
    2'b10 : Rd_Addr <= din[7:0];
    2'b11 : dout <= MEM[Wr_Addr];
    default : dout <= 0;
```

after

```
2'b11 : dout <= MEM[Rd_Addr]; // read addr instead of write addr //M2
default : dout <= 0;
```

Bug #2: tx_valid should maintain its value when the rx_valid is not active

before

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

after

```
else
    if (rx_valid) begin
        case (din[9:8])
            2'b00 : Wr_Addr <= din[7:0];
            2'b01 : MEM[Wr_Addr] <= din[7:0];
            2'b10 : Rd_Addr <= din[7:0];
            2'b11 : dout <= MEM[Rd_Addr]; // read addr instead of write addr //M2
            default : dout <= 0;
        endcase
        tx_valid <= (din[9] && din[8])? 1'b1 : 1'b0; //Begin - end //M1
    end
```

4. COVERAGE

Code Coverage

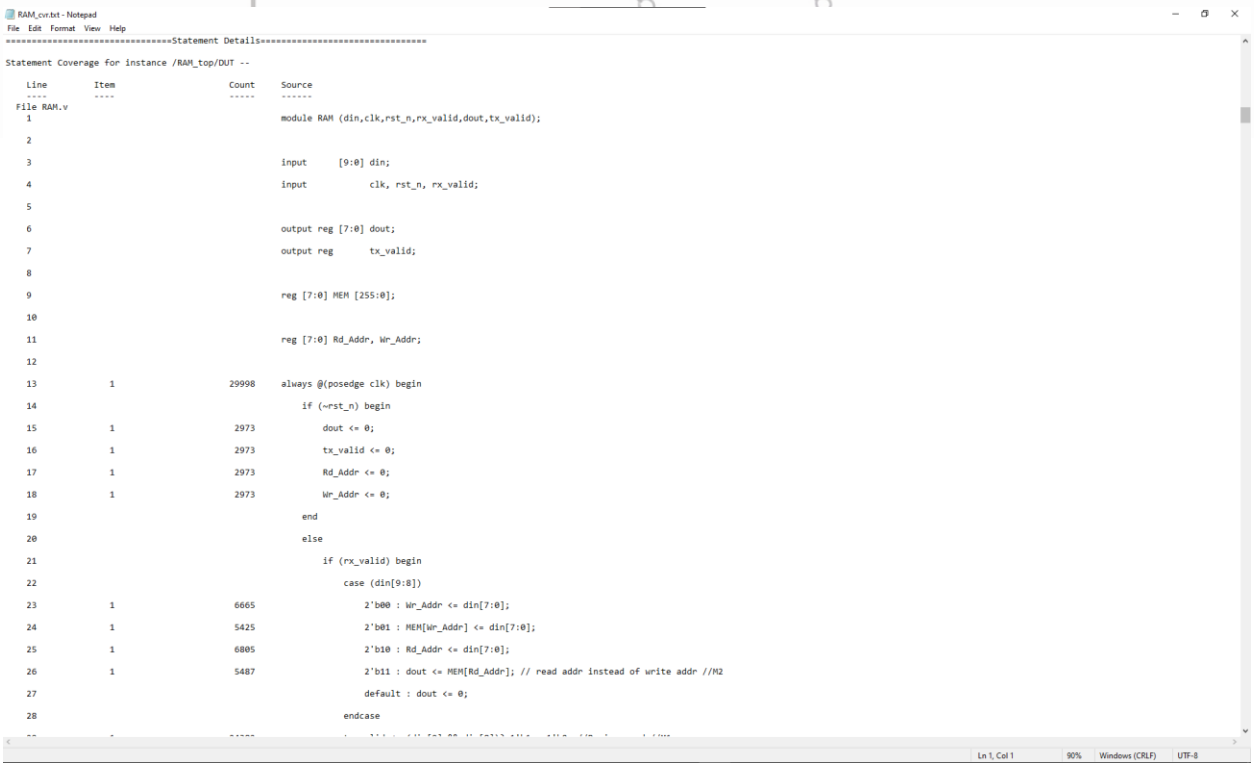
branch

RAM_cvr.txt - Notepad

File Edit Format View Help

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

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

Branch Coverage for instance /RAM_top/DUT

    Line         Item                    Count     Source
    ----         ----                    -----     ------
  File RAM.v
------------------------------------IF Branch------------------------------------
    14                                   29998     Count coming in to IF
    14              1                     2973         if (~rst_n) begin
    21              1                    24382             if (rx_valid) begin
                                          2643     All False Count
Branch totals: 3 hits of 3 branches = 100.00%

----------------------------------CASE Branch-----------------------------------
    22                                   24382     Count coming in to CASE
    23              1                     6665                 2'b00 : Wr_Addr <= din[7:0];
    24              1                     5425                 2'b01 : MEM[Wr_Addr] <= din[7:0];
    25              1                     6805                 2'b10 : Rd_Addr <= din[7:0];
    26              1                     5487                 2'b11 : dout <= MEM[Rd_Addr]; // read addr instead of write addr //M2
Branch totals: 4 hits of 4 branches = 100.00%
```

STATEMENT



RAM_cvr.txt - Notepad

File Edit Format View Help

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

Statement Coverage for instance /RAM_top/DUT --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File RAM.v
    1                                               module RAM (din,clk,rst_n,rx_valid,dout,tx_valid);
    2
    3                                               input      [9:0] din;
    4                                               input            clk, rst_n, rx_valid;
    5
    6                                               output reg [7:0] dout;
    7                                               output reg       tx_valid;
    8
    9                                               reg [7:0] MEM [255:0];
    10
    11                                              reg [7:0] Rd_Addr, Wr_Addr;
    12
    13              1                   29998       always @(posedge clk) begin
    14                                                  if (~rst_n) begin
    15              1                    2973               dout <= 0;
    16              1                    2973               tx_valid <= 0;
    17              1                    2973               Rd_Addr <= 0;
    18              1                    2973               Wr_Addr <= 0;
    19                                                  end
    20                                                  else
    21                                                      if (rx_valid) begin
    22                                                          case (din[9:8])
    23              1                    6665                       2'b00 : Wr_Addr <= din[7:0];
    24              1                    5425                       2'b01 : MEM[Wr_Addr] <= din[7:0];
    25              1                    6805                       2'b10 : Rd_Addr <= din[7:0];
    26              1                    5487                       2'b11 : dout <= MEM[Rd_Addr]; // read addr instead of write addr //M2
    27                                                                  default : dout <= 0;
    28                                                          endcase
```

Ln 1, Col 1 | 90% | Windows (CRLF) | UTF-8

```
    29              1                   24382               tx_valid <= (din[9] && din[8])? 1'b1 : 1'b0; //Begin - end //M1
```

TOGGLE

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

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

Toggle Coverage for instance /RAM_top/DUT --

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

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

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

Functional Coverage

RAM_cvr.txt - Notepad

File Edit Format View Help

```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------------
Covergroup                                                 Metric       Goal       Bins    Status
----------------------------------------------------------------------------------------------------------------
 TYPE /RAM_coverage_pkg/RAM_coverage/RAM_CVG              100.00%        100          -    Covered
    covered/total bins:                                        16         16          -
    missing/total bins:                                         0         16          -
    % Hit:                                                100.00%        100          -
    Coverpoint rx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                     2          2          -
        missing/total bins:                                     0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint tx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                     2          2          -
        missing/total bins:                                     0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint operation_cp                               100.00%        100          -    Covered
        covered/total bins:                                     7          7          -
        missing/total bins:                                     0          7          -
        % Hit:                                            100.00%        100          -
    Cross OP_rxValid_cross                                100.00%        100          -    Covered
        covered/total bins:                                     4          4          -
        missing/total bins:                                     0          4          -
        % Hit:                                            100.00%        100          -
    Cross Read_OP_cross                                   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::RAM_CVG
                                                          100.00%        100          -    Covered
    covered/total bins:                                        16         16          -
    missing/total bins:                                         0         16          -
    % Hit:                                                100.00%        100          -
    Coverpoint rx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                     2          2          -
        missing/total bins:                                     0          2          -
        % Hit:                                            100.00%        100          -
        bin Bin_rxValid_High                                27030          1          -    Covered
        bin Bin_rxValid_Low                                  2971          1          -    Covered
    Coverpoint tx_valid_cp                                100.00%        100          -    Covered
        covered/total bins:                                     2          2          -
        missing/total bins:                                     0          2          -
        % Hit:                                            100.00%        100          -
        bin Bin_txValid_High                                 5979          1          -    Covered
        bin Bin_txValid_Low                                 24022          1          -    Covered
    Coverpoint operation_cp                               100.00%        100          -    Covered
        covered/total bins:                                     7          7          -
        missing/total bins:                                     0          7          -
        % Hit:                                            100.00%        100          -
        bin Bin_Write_Addr                                   8243          1          -    Covered
        bin Bin_Write_Data                                   6668          1          -    Covered
        bin Bin_Read_Addr                                    8336          1          -    Covered
        bin Bin_Read_Data                                    6754          1          -    Covered
        bin Bin_Write_AddrToData                             4166          1          -    Covered
        bin Bin_Read_AddrToData                              4243          1          -    Covered
        bin Bin_Trans                                       14776          1          -    Covered
    Cross OP_rxValid_cross                                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 Bin_Write_Addr_rxValidHigh                   7379          1          -    Covered
            bin Bin_Write_Data_rxValidHigh                   6043          1          -    Covered
            bin Bin_Read_Addr_rxValidHigh                    7523          1          -    Covered
            bin Bin_Read_Data_rxValidHigh                    6085          1          -    Covered
    Cross Read_OP_cross                                   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 Bin_out                                      5894          1          -    Covered

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

Assertion Coverage

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------------------------
/RAM_top/DUT/SVA/a_reset_txValid_cvr      RAM_sva Verilog  SVA  RAM_sva.sv(36)  2973 Covered
/RAM_top/DUT/SVA/a_reset_dout_cvr         RAM_sva Verilog  SVA  RAM_sva.sv(37)  2973 Covered
/RAM_top/DUT/SVA/a_wr_addr_cvr            RAM_sva Verilog  SVA  RAM_sva.sv(39)  6628 Covered
/RAM_top/DUT/SVA/a_wr_data_cvr            RAM_sva Verilog  SVA  RAM_sva.sv(40)  5410 Covered
/RAM_top/DUT/SVA/a_rd_addr_cvr            RAM_sva Verilog  SVA  RAM_sva.sv(41)  6555 Covered
/RAM_top/DUT/SVA/a_rd_data_cvr            RAM_sva Verilog  SVA  RAM_sva.sv(43)  1089 Covered
/RAM_top/DUT/SVA/a_wr_addr_to_wr_data_cvr
                                          RAM_sva Verilog  SVA  RAM_sva.sv(45)  5267 Covered
/RAM_top/DUT/SVA/a_rd_addr_to_rd_data_cvr
                                          RAM_sva Verilog  SVA  RAM_sva.sv(46)  5375 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 8
```

```
ASSERTION RESULTS:
--------------------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------------------
/RAM_top/DUT/SVA/a_reset_txValid
                     RAM_sva.sv(23)                 0          1
/RAM_top/DUT/SVA/a_reset_dout
                     RAM_sva.sv(24)                 0          1
/RAM_top/DUT/SVA/a_wr_addr
                     RAM_sva.sv(26)                 0          1
/RAM_top/DUT/SVA/a_wr_data
                     RAM_sva.sv(27)                 0          1
/RAM_top/DUT/SVA/a_rd_addr
                     RAM_sva.sv(28)                 0          1
/RAM_top/DUT/SVA/a_rd_data
                     RAM_sva.sv(30)                28          1
/RAM_top/DUT/SVA/a_wr_addr_to_wr_data
                     RAM_sva.sv(32)                 0          1
/RAM_top/DUT/SVA/a_rd_addr_to_rd_data
                     RAM_sva.sv(33)                 0          1
/RAM_sequence_pkg/RAM_write_only_seq/body/#ublk#33852551#34/immed__39
                     RAM_sequence.sv(39)            0          1
/RAM_sequence_pkg/RAM_read_only_seq/body/#ublk#33852551#54/immed__59
                     RAM_sequence.sv(59)            0          1
/RAM_sequence_pkg/RAM_write_read_seq/body/#ublk#33852551#74/immed__79
                     RAM_sequence.sv(79)            0          1

Total Coverage By Instance (filtered view): 77.20%
```

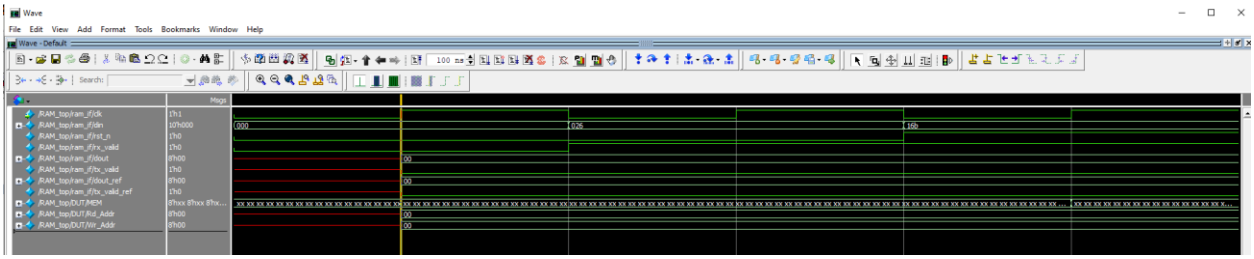
NOTE: A_RD_DATA ASSERTION WILL BE FIXED WHEN USING THE RAM INSIDE THE WRAPPER

5. ASSERTIONS TABLE

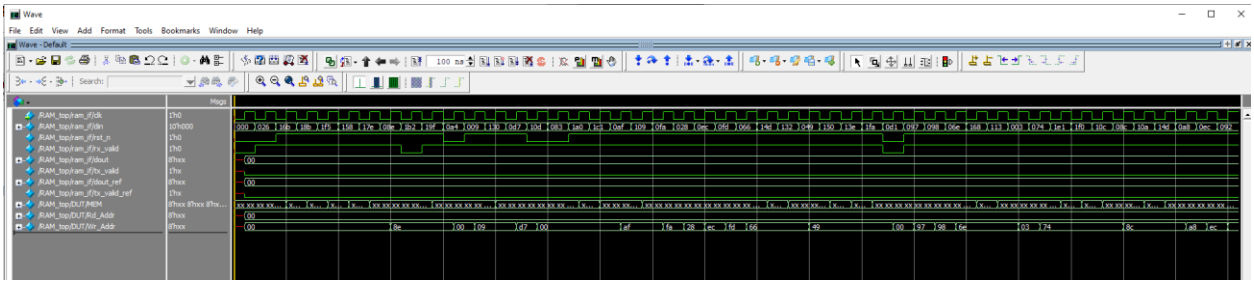
Feature	Assertion
When the reset is asserted, after 1 clk cycle tx_valid is low	(@(posedge clk) !rst_n |->##1 tx_valid == 1'b0)
When the reset is asserted, after 1 clk cycle dout is low	(@(posedge clk) !rst_n |-> ##1 dout == 7'b0)
when the reset is inactive and the din[9:8] = 2'b00 and the rx_valid is high, after 1 clk cycle tx_valid is low	(@(posedge clk) disable iff (!rst_n) op == WRITE_ADDR && rx_valid |-> ##1 ~tx_valid)
when the reset is inactive and the din[9:8] = 2'b01 and the rx_valid is high, after 1 clk cycle tx_valid is low	(@(posedge clk) disable iff (!rst_n) op == WRITE_DATA && rx_valid |-> ##1 ~tx_valid)
when the reset is inactive and the din[9:8] = 2'b10 and the rx_valid is high, after 1 clk cycle tx_valid is low	(@(posedge clk) disable iff (!rst_n) op == READ_ADDR && rx_valid |-> ##1 ~tx_valid)
when the reset is inactive and the din[9:8] = 2'b11 and the rx_valid is high, tx_valid is risen then after 1 or more clk cycles should fall	(@(posedge clk) disable iff (!rst_n) (op == READ_DATA && rx_valid && !is_readOnly) |->##1 $rose(tx_valid) ##[1:$] $fell(tx_valid))
when the reset is inactive and the din[9:8] = 2'b00 and the rx_valid is high, after 1 or more clk cycles din[9:8] should equal 2'b01 and rx_valid should be high	(@(posedge clk) disable iff (!rst_n) (op == WRITE_ADDR && rx_valid) |-> (##[1:$] op == WRITE_DATA && rx_valid))
when the reset is inactive and the din[9:8] = 2'b10 and the rx_valid is high, after 1 or more clk cycles din[9:8] should equal 2'b11 and rx_valid should be high	(@(posedge clk) disable iff (!rst_n) (op == READ_ADDR && rx_valid) |-> (##[1:$] op == READ_DATA && rx_valid))

6. SIMULATION SNIPPETS

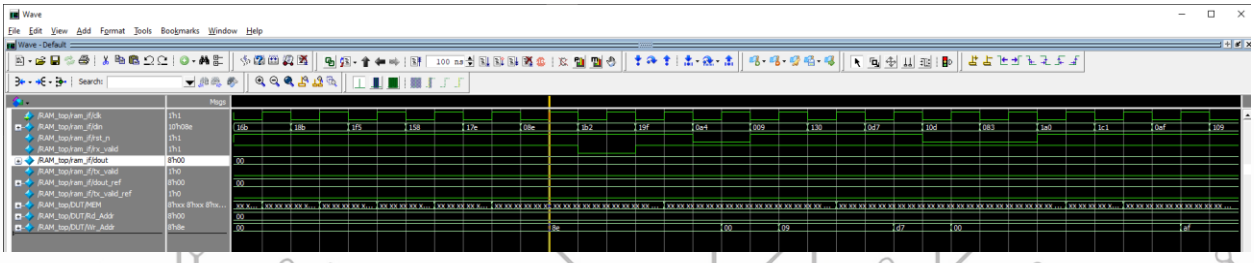
RESET SEQUENCE



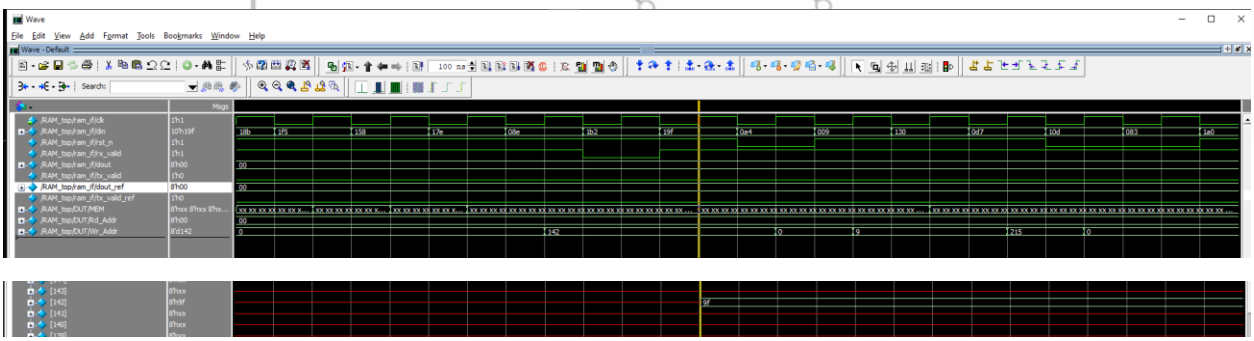
WRITE ONLY SEQUENCE



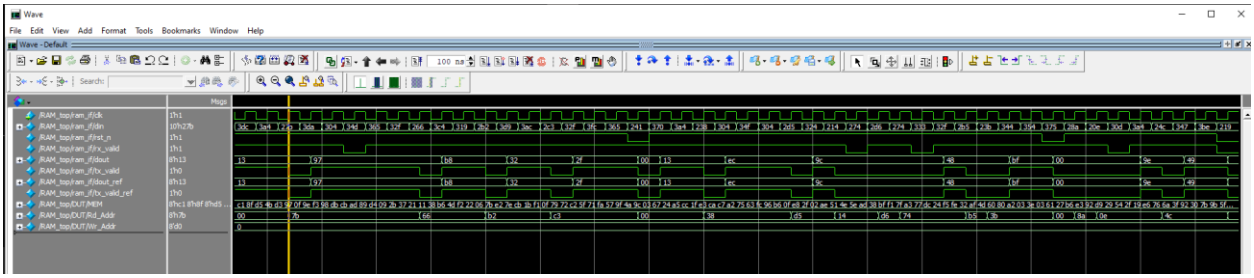
WRITE ADDRESS



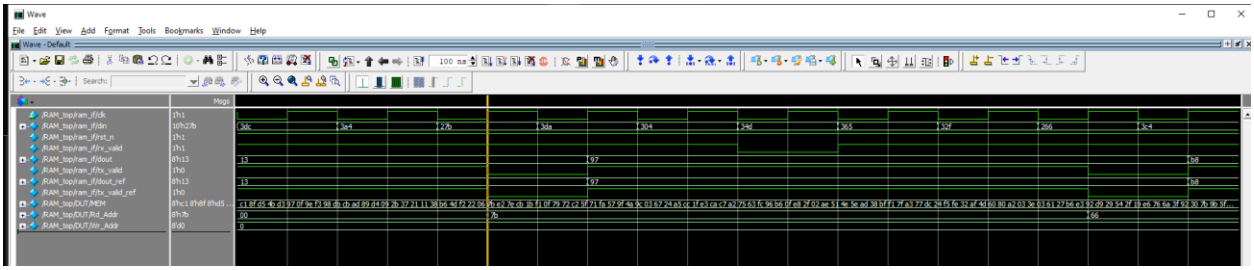
WRITE DATA



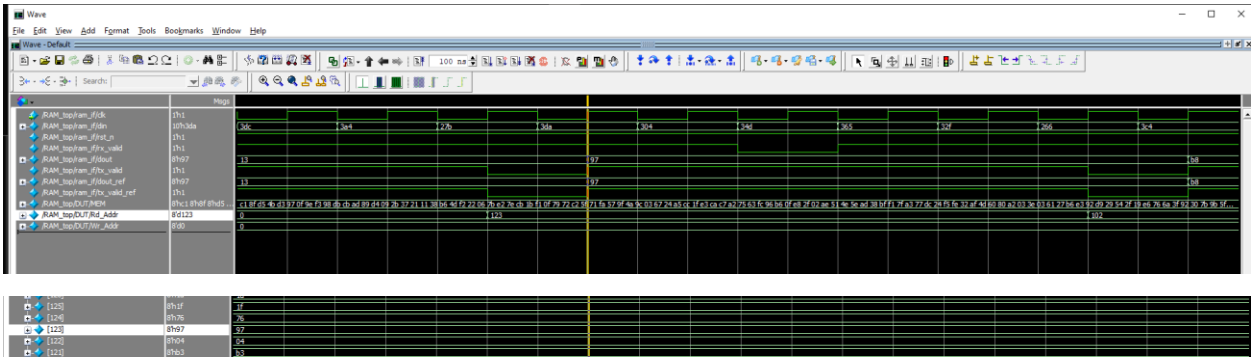
READ ONLY SEQUENCE



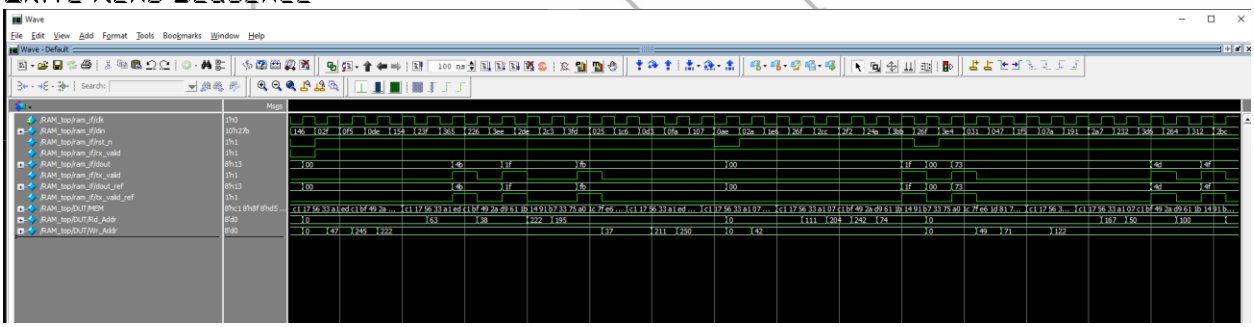
READ ADDRESS



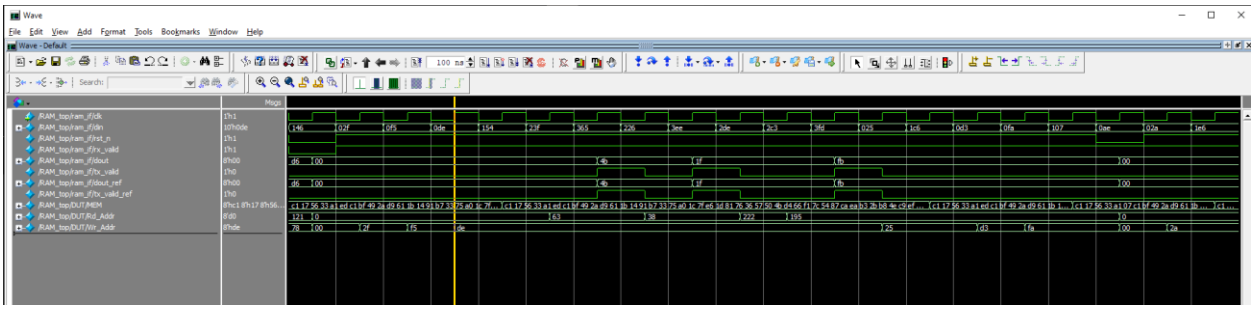
READ DATA



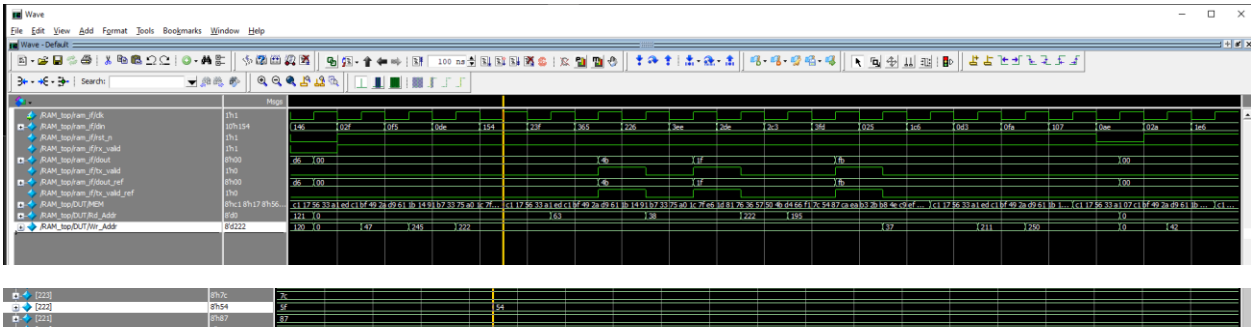
WRITE READ SEQUENCE



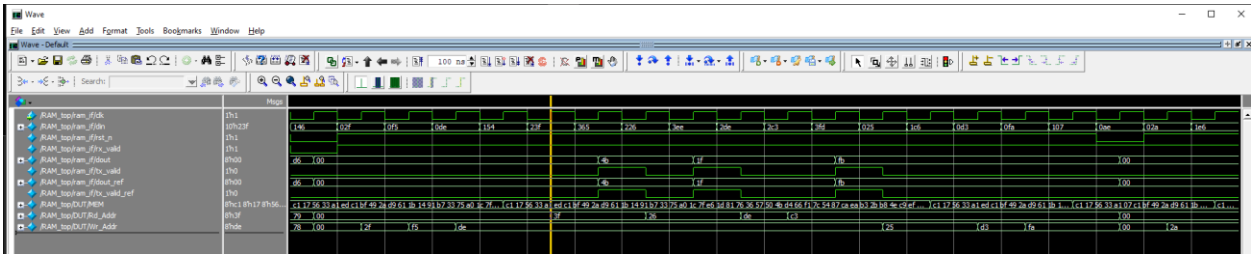
WRITE ADDR



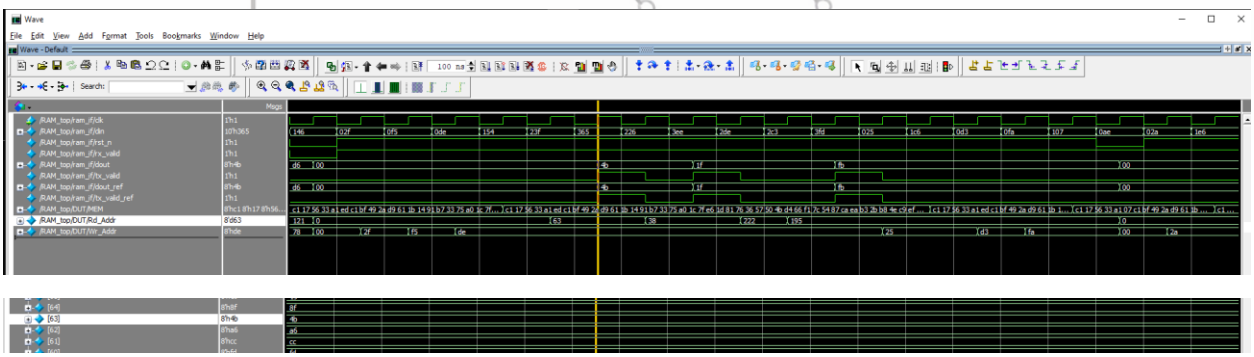
WRITE DATA



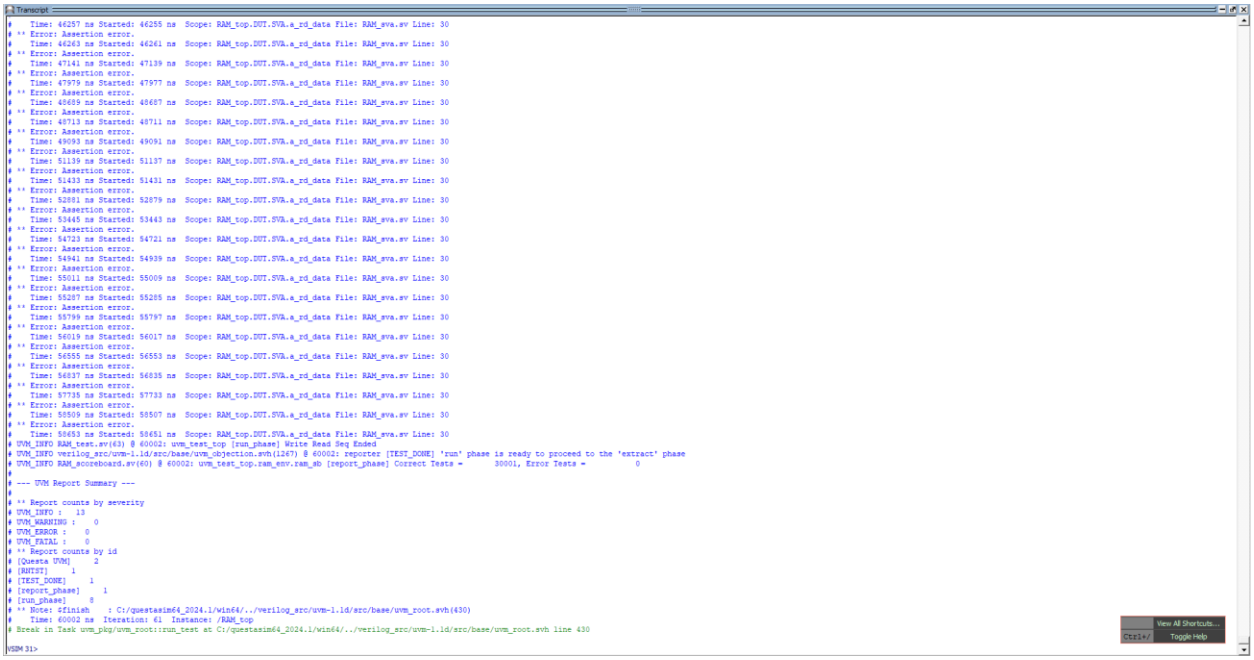
READ ADDR



READ DATA



TRANSCRIPT SNIPPETS



NOTE: A_RD_DATA ASSERTION WILL BE FIXED WHEN USING THE RAM INSIDE THE WRAPPER

SECTION #2: VERIFICATION OF SPI SLAVE

1. VERIFICATION PLAN

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
SLAVE_1	When the rst_n is asserted, the output MISO value should be low & the state cs turns back to IDLE	Directed at the start of the simulation, and then randomized with constraint that drives the reset to be off 99% of the simulation time	-	A checker in the reference model to check for the synchronous reset functionality
SLAVE_2	When the reset is deasserted, SS_n is asserted to tell the slave that the master starts communication & then deasserted to end communication	Randomized with constraint that drives the SS_n to be on every 13 clk cycles if the current state is normal operations or to be on every 23 clk cycles if the current state is read data	Covers transaction duration: 1 → 0 [*13] → 1 for normal operations & Check extended transaction: 1 → 0 [*23] → 1 for read data	A checker in the reference model to check for the functionality
SLAVE_4	When the reset is deasserted, the slave receives MOSI bit by bit & converts the data from serial "MOSI" to parallel	Randomized with a constraint that ensures that the most significant 3 bits follow the sequence of write address -> write data -> read address -> read data	Covers validate correct transition: 0->0->0 (for write address), 0->0->1 (for write data),1->1->0 (for read address), 1->1->1(for read data)	A checker in the reference model to check for the functionality
SLAVE_5	When the reset is deasserted, tx_valid is asserted when the slave is ready to send the parallel data to the RAM	Randomized with constraint that drives the tx_valid to be on when the current state is read data & the data is ready to be transmitted (when the counter indicates that)	-	A checker in the reference model to check for the functionality

2. CODE SNIPPETS

INTERFACE

SPI_slave_if.sv

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

DESIGN

SPI_slave.sv

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

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

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

reg [3:0] counter;
reg       received_address;

reg [2:0] cs, ns;

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

SPI_slave.sv

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

localparam IDLE      = 3'b000;
localparam CHK_CMD   = 3'b001; // (001) Check Command operation
localparam WRITE     = 3'b010; // (010) Write operation
localparam READ_ADD  = 3'b011;
localparam READ_DATA = 3'b100;

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

reg [3:0] counter;
reg       received_address;

reg [2:0] cs, ns;

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

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

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

SPI_slave.sv

```
    always @(posedge clk) begin
        else begin
            case (cs)
                READ_ADD : begin
                end
                else begin
                    rx_valid <= 1;
                    received_address <= 1;
                end
            end
            READ_DATA : begin
                if (tx_valid) begin
                    rx_valid <= 0;
                    if (counter > 0) begin
                        MISO <= tx_data[counter-1];
                        counter <= counter - 1;
                    end
                    else begin
                        received_address <= 0;
                    end
                end
                else begin
                    if (counter > 0) begin
                        rx_data[counter-1] <= MOSI;
                        counter <= counter - 1;
                        rx_valid <= 0;
                    end
                    else begin
                        rx_valid <= 1;
                        counter <= 9;
                    end
                end
            end
        endcase
    end
end

`ifdef SIM

property rst_p;
    @(posedge clk) (!rst_n) |=> (!MISO && !rx_valid && (rx_data == 10'b00_0000_0000) && (cs == IDLE) && !received_address);
endproperty

property write_seq_p;
    @(posedge clk) disable iff (!rst_n) (SS_n == 1 ##2 MOSI == 0) |=> MOSI == 0;
endproperty

property read_seq_p;
    @(posedge clk) disable iff (!rst_n) (SS_n == 1 ##2 MOSI == 1) |=> MOSI == 1;
endproperty

sequence write1_seq_s;
    SS_n == 1 ##2 MOSI == 0 ##1 MOSI == 0 ##1 MOSI == 0;
endsequence
```

SPI_slave.sv

```
`ifdef SIM

property rst_p;
    @(posedge clk) (!rst_n) |=> (!MISO && !rx_valid && (rx_data == 10'b00_0000_0000) && (cs == IDLE) && !received_address);
endproperty

property write_seq_p;
    @(posedge clk) disable iff (!rst_n) (SS_n == 1 ##2 MOSI == 0) |=> MOSI == 0;
endproperty

property read_seq_p;
    @(posedge clk) disable iff (!rst_n) (SS_n == 1 ##2 MOSI == 1) |=> MOSI == 1;
endproperty

sequence write1_seq_s;
    SS_n == 1 ##2 MOSI == 0 ##1 MOSI == 0 ##1 MOSI == 0;
endsequence

sequence read1_seq_s;
    SS_n == 1 ##2 MOSI == 1 ##1 MOSI == 1 ##1 MOSI == 0;
endsequence

sequence write2_seq_s;
    SS_n == 1 ##2 MOSI == 0 ##1 MOSI == 0 ##1 MOSI == 1;
endsequence

sequence read2_seq_s;
    SS_n == 1 ##2 MOSI == 1 ##1 MOSI == 1 ##1 MOSI == 1;
endsequence

sequence rx_ss_n_s;
    ##10 rx_valid ##[1:$] SS_n;
endsequence

property rx_valid_p;
    @(posedge clk) disable iff (!rst_n)
    (write1_seq_s or read1_seq_s or write2_seq_s or read2_seq_s) |-> rx_ss_n_s;
endproperty

property cs_p1;
    @(posedge clk) disable iff (!rst_n) (SS_n) |=> (cs == IDLE);
endproperty

property cs_p2;
    @(posedge clk) disable iff (!rst_n) (cs == IDLE && !SS_n) |=> (cs == CHK_CMD);
endproperty

property cs_p3;
    @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && !MOSI) |=> (cs == WRITE);
endproperty

property cs_p4;
    @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && MOSI && received_address) |=> (cs == READ_DATA);
endproperty
```

Ln 125, Col 39 Spaces: 4 UTF-8 CRLF {} System Verilog

SPI_slave.sv

```
property cs_p4;
    @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && MOSI && received_address) |=> (cs == READ_DATA);
endproperty

property cs_p6;
    @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && MOSI && !received_address) |=> (cs == READ_ADD);
endproperty

property cs_p7;
    @(posedge clk) disable iff (!rst_n) (cs == WRITE && !SS_n) |=> (cs == WRITE);
endproperty

property cs_p8;
    @(posedge clk) disable iff (!rst_n) (cs == READ_ADD && !SS_n) |=> (cs == READ_ADD);
endproperty

property cs_p9;
    @(posedge clk) disable iff (!rst_n) (cs == READ_DATA && !SS_n) |=> (cs == READ_DATA);
endproperty

property counter_rst_p;
    @(posedge clk) disable iff (!rst_n) (cs == CHK_CMD) |=> (counter == 4'b1010);
endproperty

property counter_write_p;
    @(posedge clk) disable iff (!rst_n)
    ((cs == WRITE || cs == READ_ADD || cs == READ_DATA) && counter > 0) |=> (counter == $past(counter) - 1'b1);
endproperty

property received_address_p1;
    @(posedge clk) disable iff (!rst_n) (cs == READ_ADD && counter == 0) |=> (received_address);
endproperty

property received_address_p2;
    @(posedge clk) disable iff (!rst_n) (cs == READ_DATA && tx_valid && counter == 0) |=> (!received_address);
endproperty

assert property (rst_p);
assert property (write_seq_p);
assert property (read_seq_p);
assert property (rx_valid_p);
assert property (cs_p1);
assert property (cs_p2);
assert property (cs_p3);
assert property (cs_p4);
assert property (cs_p6);
assert property (cs_p7);
assert property (cs_p8);
assert property (cs_p9);
assert property (counter_rst_p);
assert property (counter_write_p);
assert property (received_address_p1);
assert property (received_address_p2);
```

Ln 125, Col 39 Spaces: 4 UTF-8 CRLF {} System Verilog

```
cover property (rst_p);
cover property (write_seq_p);
cover property (read_seq_p);
cover property (rx_valid_p);
cover property (cs_p1);
cover property (cs_p2);
cover property (cs_p3);
cover property (cs_p4);
cover property (cs_p6);
cover property (cs_p7);
cover property (cs_p8);
cover property (cs_p9);
cover property (counter_rst_p);
cover property (counter_write_p);
cover property (received_address_p1);
cover property (received_address_p2);

`endif

endmodule
```

Golden Model

SPI_slave_golden_model.v

```
module SLAVE_golden_model (
    input clk, rst_n, SS_n, MOSI, tx_valid,
    input[7:0] tx_data,
    output reg MISO, rx_valid,
    output reg[9:0] rx_data
);
parameter IDLE = 3'b000;
parameter CHK_CMD = 3'b001;
parameter WRITE = 3'b010;
parameter READ_ADD = 3'b011;
parameter READ_DATA = 3'b100;

// (* fsm_encoding = "gray" *)
reg[2:0] cs, ns;
reg rx_type;
reg[4:0] cnt;

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

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

always @(posedge clk) begin
```

SPI_slave_golden_model.v

```

always @(posedge clk) begin
    if(~rst_n) begin
        MISO <= 0;
        rx_type <= 0;
        rx_valid <= 0;
        rx_data <= 0;
        cnt <= 0;
    end
    else begin
        case(cs)
        IDLE: rx_valid <= 0;
        CHK_CMD: cnt <= 0;
        WRITE: begin
            cnt <= cnt + 1;
                if (cnt < 10) rx_data[9-cnt] <= MOSI;
            if (cnt == 10) rx_valid <= 1;
        end
        READ_ADD: begin
            cnt <= cnt + 1;
                if (cnt < 10) rx_data[9-cnt] <= MOSI;
            if (cnt == 10) begin
                rx_valid <= 1;
                rx_type <= 1;
            end
        end
        READ_DATA: begin
            if(!tx_valid) begin
                cnt <= cnt + 1;
                if (cnt < 10) rx_data[9-cnt] <= MOSI;
                else begin
                    rx_type <= 0;
                end
                if (cnt == 10) cnt <= 1;
                rx_valid <= (cnt == 10);
            end
            else begin
                rx_valid <= 0;
                cnt <= cnt + 1;
                if (cnt >= 2 && cnt <= 9) MISO <= tx_data[9 - cnt];
                else begin
                    rx_type <= 0;
                end
            end
        end
        endcase
    end
end

endmodule

```

Ln 3, Col 23 (7 selected) Spaces: 4 UTF-8 CRLF {} Verilog

Config

SPI_slave_config.sv

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

class SPI_slave_config extends uvm_object;
    `uvm_object_utils(SPI_slave_config);

    virtual SPI_slave_if slave_if;
    uvm_active_passive_enum is_active;

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

endpackage
```

Top

SPI_slave_top.sv

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

module SPI_slave_top ();
bit clk;
initial begin
    clk=0;
    forever begin
        #1 clk=~clk;
    end
end

SPI_slave_if slave_if(clk);

SLAVE DUT(
    .clk(clk), .rst_n(slave_if.rst_n), .SS_n(slave_if.SS_n),
    .tx_valid(slave_if.tx_valid),.tx_data(slave_if.tx_data), .MOSI(slave_if.MOSI),
    .rx_data(slave_if.rx_data), .rx_valid(slave_if.rx_valid), .MISO(slave_if.MISO)
);

SLAVE_golden_model GOLDEN(
    .clk(clk), .rst_n(slave_if.rst_n), .SS_n(slave_if.SS_n),
    .tx_valid(slave_if.tx_valid),.tx_data(slave_if.tx_data), .MOSI(slave_if.MOSI),
    .rx_data(slave_if.rx_data_ref), .rx_valid(slave_if.rx_valid_ref), .MISO(slave_if.MISO_ref)
);

initial begin
    uvm_config_db#(virtual SPI_slave_if)::set(null,"uvm_test_top","SLAVE_IF",slave_if);
    run_test("SPI_slave_test");
end

endmodule
```

Test

SPI_slave_test.sv

```
package SPI_slave_test_pkg;
import SPI_slave_env_pkg::*;
import SPI_slave_config_pkg::*;
import SPI_slave_reset_sequence_pkg::*;
import SPI_slave_main_sequence_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class SPI_slave_test extends uvm_test;
    `uvm_component_utils(SPI_slave_test)
    SPI_slave_env env;
    SPI_slave_config slave_cfg;
    SPI_slave_reset_sequence rst_seq;
    SPI_slave_main_sequence main_seq;


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

    function void build_phase (uvm_phase phase);
        super.build_phase(phase);
        env = SPI_slave_env::type_id::create("env", this);
        slave_cfg = SPI_slave_config::type_id::create("slave_cfg");
        rst_seq = SPI_slave_reset_sequence::type_id::create("rst_seq");
        main_seq = SPI_slave_main_sequence::type_id::create("main_seq");

        if (!uvm_config_db #(virtual SPI_slave_if)::get(this, "", "SLAVE_IF", slave_cfg.slave_if))
            `uvm_fatal("build_phase", "Test-unable to get the virtual interface of the slave from the uvm_config_db")

        slave_cfg.is_active = UVM_ACTIVE;

        uvm_config_db #(SPI_slave_config)::set(this, "*", "CFG_SLAVE", slave_cfg);
    endfunction : build_phase

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

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

        `uvm_info("run_phase", "Main Sequence Started", UVM_LOW);
        main_seq.start(env.agt.sqr);
        `uvm_info("run_phase", "Main Sequence Ended", UVM_LOW);

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

Environment

SPI_slave_env.sv

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

class SPI_slave_env extends uvm_env;
    `uvm_component_utils(SPI_slave_env)

    SPI_slave_agent agt;
    SPI_slave_scoreboard sb;
    SPI_slave_coverage cov;

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

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        agt = SPI_slave_agent::type_id::create("agt",this);
        sb = SPI_slave_scoreboard::type_id::create("sb",this);
        cov = SPI_slave_coverage::type_id::create("cov",this);
    endfunction

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

Coverage

SPI_slave_coverage.sv

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

class SPI_slave_coverage extends  uvm_scoreboard;
    `uvm_component_utils(SPI_slave_coverage);

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

    covergroup covgrp;

        rx_data_cp: coverpoint seq_item_cov.rx_data[9:8] {
            bins rx_data_all_values[] = {[0:3]};
            bins rx_data_trans[] = (0 => 1), (1 => 3), (1 => 0), (0 => 2),
                                   (2 => 3), (2 => 0), (3 => 1), (3 => 2);
        }

        SS_n_cp: coverpoint seq_item_cov.SS_n {
            bins normal_trans = (1 => 0[*13] => 1);
            bins extended_trans = (1 => 0[*23] => 1);
        }

        MOSI_cp: coverpoint seq_item_cov.MOSI{
            bins write_address = (0 => 0 => 0);
            bins write_data    = (0 => 0 => 1);
            bins read_address  = (1 => 1 => 0);
            bins read_data     = (1 => 1 => 1);
        }

        SS_n_MOSI_cr: cross SS_n_cp, MOSI_cp{
            ignore_bins not_r_data_ext1 = binsof(MOSI_cp.write_address) && binsof(SS_n_cp.extended_trans);
            ignore_bins not_r_data_ext2 = binsof(MOSI_cp.write_data) && binsof(SS_n_cp.extended_trans);
            ignore_bins not_r_data_ext3 = binsof(MOSI_cp.read_address) && binsof(SS_n_cp.extended_trans);
            ignore_bins not_r_data_ext4 = binsof(MOSI_cp.read_data) && binsof(SS_n_cp.normal_trans);
        }
    endgroup

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

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

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

SPI_slave_coverage.sv

```
class SPI_slave_coverage extends  uvm_scoreboard;

    covergroup covgrp;

        rx_data_cp: coverpoint seq_item_cov.rx_data[9:8] {
            bins rx_data_all_values[] = {[0:3]};
            bins rx_data_trans[] = (0 => 1), (1 => 3), (1 => 0), (0 => 2),
                                   (2 => 3), (2 => 0), (3 => 1), (3 => 2);
        }

        SS_n_cp: coverpoint seq_item_cov.SS_n {
            bins normal_trans = (1 => 0[*13] => 1);
            bins extended_trans = (1 => 0[*23] => 1);
        }

        MOSI_cp: coverpoint seq_item_cov.MOSI{
            bins write_address = (0 => 0 => 0);
            bins write_data    = (0 => 0 => 1);
            bins read_address  = (1 => 1 => 0);
            bins read_data     = (1 => 1 => 1);
        }

        SS_n_MOSI_cr: cross SS_n_cp, MOSI_cp{
            ignore_bins not_r_data_ext1 = binsof(MOSI_cp.write_address) && binsof(SS_n_cp.extended_trans);
            ignore_bins not_r_data_ext2 = binsof(MOSI_cp.write_data) && binsof(SS_n_cp.extended_trans);
            ignore_bins not_r_data_ext3 = binsof(MOSI_cp.read_address) && binsof(SS_n_cp.extended_trans);
            ignore_bins not_r_data_ext4 = binsof(MOSI_cp.read_data) && binsof(SS_n_cp.normal_trans);
        }
    endgroup

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

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

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

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

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} System Verilog

Scoreboard

SPI_slave_scoreboard.sv

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

class SPI_slave_scoreboard extends  uvm_scoreboard;
    `uvm_component_utils(SPI_slave_scoreboard)

    uvm_analysis_export #(SPI_slave_seq_item) sb_export;
    uvm_tlm_analysis_fifo #(SPI_slave_seq_item) sb_fifo;
    SPI_slave_seq_item seq_item_sb;

    int rx_valid_error_count = 0;
    int rx_data_error_count = 0;
    int MISO_error_count = 0;
    int rx_valid_correct_count = 0;
    int rx_data_correct_count = 0;
    int MISO_correct_count = 0;

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

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

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

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            sb_fifo.get(seq_item_sb);
            if (seq_item_sb.rx_data !== seq_item_sb.rx_data_ref) begin
                `uvm_error("run_phase", $sformatf("Comparsion of rx_data failed, DUT:%s While rx_data_ref = %h", seq_item_sb.convert2string(), seq_item_sb.rx_data_ref));
                rx_data_error_count++;
            end
            else rx_data_correct_count++;

            if (seq_item_sb.rx_valid !== seq_item_sb.rx_valid_ref) begin
                `uvm_error("run_phase", $sformatf("comparsion of rx_valid failed, DUT:%s While rx_valid_ref = %b", seq_item_sb.convert2string(), seq_item_sb.rx_valid_ref));
                rx_valid_error_count++;
            end
            else rx_valid_correct_count++;

            if (seq_item_sb.MISO !== seq_item_sb.MISO_ref) begin
                `uvm_error("run_phase", $sformatf("comparsion of MISO failed, DUT:%s While MISO_ref = %b", seq_item_sb.convert2string(), seq_item_sb.MISO_ref));
                MISO_error_count++;
            end
            else MISO_correct_count++;
```

Ln 22, Col 16 Spaces: 4 UTF-8 CRLF {} System Verilog

```
            else MISO_correct_count++;
        end
    endtask

    function void report_phase (uvm_phase phase);
        super.report_phase(phase);
        `uvm_info("report_phase", "SPI SLAVE SCOREBOARD", UVM_LOW);
        `uvm_info("report_phase", $sformatf("Total successful rx_data: %d", rx_data_correct_count), UVM_LOW);
        `uvm_info("report_phase", $sformatf("Total failed rx_data: %d", rx_data_error_count), UVM_LOW);
        `uvm_info("report_phase", $sformatf("Total successful rx_valid: %d", rx_valid_correct_count), UVM_LOW);
        `uvm_info("report_phase", $sformatf("Total failed rx_valid: %d", rx_valid_error_count), UVM_LOW);
        `uvm_info("report_phase", $sformatf("Total successful MISO: %d", MISO_correct_count), UVM_LOW);
        `uvm_info("report_phase", $sformatf("Total failed MISO: %d", MISO_error_count), UVM_LOW);
    endfunction
endclass
endpackage
```

Ln 22, Col 16 Spaces: 4 UTF-8 CRLF {} System Verilog

Agent

SPI_slave_agent.sv

```
package SPI_slave_agent_pkg;
import SPI_slave_seq_item_pkg::*;
import SPI_slave_driver_pkg::*;
import SPI_slave_monitor_pkg::*;
import SPI_slave_sequencer_pkg::*;
import SPI_slave_config_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class SPI_slave_agent extends uvm_agent;
    `uvm_component_utils(SPI_slave_agent);

    SPI_slave_driver driver;
    SPI_slave_monitor monitor;
    SPI_slave_sequencer sqr;
    SPI_slave_config slave_cfg;
    uvm_analysis_port #(SPI_slave_seq_item) agt_ap;

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

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        if(!uvm_config_db#(SPI_slave_config)::get(this,"","CFG_SLAVE",slave_cfg))
            `uvm_fatal("build_phase","Agent - unable to get configuration object");

        if(slave_cfg.is_active == UVM_ACTIVE) begin
            driver = SPI_slave_driver::type_id::create("driver",this);
            sqr = SPI_slave_sequencer::type_id::create("sqr",this);
        end
        monitor = SPI_slave_monitor::type_id::create("monitor",this);
        agt_ap = new("agt_ap",this);
    endfunction

    function void connect_phase(uvm_phase phase);
        super.connect_phase(phase);
        if(slave_cfg.is_active == UVM_ACTIVE) begin
            driver.slave_if = slave_cfg.slave_if;
            driver.seq_item_port.connect(sqr.seq_item_export);
        end
        monitor.slave_if = slave_cfg.slave_if;
        monitor.mon_ap.connect(agt_ap);
    endfunction

endclass
endpackage
```

Driver

SPI_slave_driver.sv

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

class SPI_slave_driver extends uvm_driver #(SPI_slave_seq_item);
    `uvm_component_utils(SPI_slave_driver);

    virtual SPI_slave_if slave_if;
    SPI_slave_seq_item stim_seq_item;

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

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            stim_seq_item = SPI_slave_seq_item::type_id::create("stim_seq_item");
            seq_item_port.get_next_item(stim_seq_item);
            slave_if.rst_n = stim_seq_item.rst_n;
            slave_if.SS_n = stim_seq_item.SS_n;
            slave_if.tx_valid = stim_seq_item.tx_valid;
            slave_if.tx_data = stim_seq_item.tx_data;
            slave_if.MOSI = stim_seq_item.MOSI;
            @(negedge slave_if.clk);
            seq_item_port.item_done();
            `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH);
        end
    endtask
endclass

endpackage
```

Monitor

SPI_slave_monitor.sv

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

class SPI_slave_monitor extends uvm_monitor;
    `uvm_component_utils(SPI_slave_monitor);

    SPI_slave_seq_item rsp_seq_item;
    virtual SPI_slave_if slave_if;
    uvm_analysis_port #(SPI_slave_seq_item) mon_ap;

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

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

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

Sequencer

SPI_slave_sequencer.sv

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

class SPI_slave_sequencer extends uvm_sequencer #(SPI_slave_seq_item);
    `uvm_component_utils(SPI_slave_sequencer);

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

endpackage
```

Sequence

reset

SPI_slave_reset_sequence.sv

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

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

    SPI_slave_seq_item seq_item;

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

    task body;
        seq_item = SPI_slave_seq_item::type_id::create("seq_item");
        start_item(seq_item);
        seq_item.rst_n = 0;
        seq_item.SS_n = 0;
        seq_item.MOSI = 0;
        seq_item.tx_valid = 0;
        seq_item.tx_data = 0;
        finish_item(seq_item);
    endtask
endclass
endpackage
```

main

SPI_slave_main_sequence.sv

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

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

    SPI_slave_seq_item seq_item;

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

    task body;
        seq_item = SPI_slave_seq_item::type_id::create("seq_item");
        repeat(20000) begin
            start_item(seq_item);
            assert(seq_item.randomize());
            finish_item(seq_item);
        end
    endtask
endclass
endpackage
```

Sequence Item

SPI_slave_seq_item.sv

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

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

    rand bit [10:0] MOSI_array;
    static bit [1:0] state = 2'b00;
    int count;

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

    function string convert2string();
        return $sformatf("%s rst_n = 0b%0b , MOSI = 0b%0b , SS_n = 0b%0b , tx_valid = 0b%0b , MISO = 0b%0b , tx_data = 0x%02h , rx_valid = 0b%0b , rx_data = 0x%03h",
            super.convert2string(), rst_n, MOSI, SS_n, tx_valid, tx_data, MISO, rx_valid, rx_data);
    endfunction

    function string convert2string_stimulus();
        return $sformatf("%s rst_n = 0b%0b , MOSI = 0b%0b , SS_n = 0b%0b , tx_valid = 0b%0b , tx_data = 0x%02h",
            super.convert2string(), rst_n, MOSI, SS_n, tx_valid, tx_data);
    endfunction

    // SLAVE_1
    constraint rst_n_const{
        rst_n dist {0:/1, 1:/99};
    }

    // SLAVE_2
    constraint SS_n_const{
        if(MOSI_array[10:8] == 3'b111){
            if(count == 23) SS_n == 1;
            else SS_n == 0;
        }
        else{
            if(count == 13) SS_n == 1;
            else SS_n == 0;
        }
    }

    // SLAVE_3
    constraint MOSI_array_const{
        MOSI_array[10] == state[1];
        MOSI_array[9:8] == state;
    }

    // SLAVE_4
    constraint MOSI_const{
        if (count > 0 && count < 12) MOSI == MOSI_array[11 - count];
```

```
    // SLAVE_4
    constraint MOSI_const{
        if (count > 0 && count < 12) MOSI == MOSI_array[11 - count];
    }

    // SLAVE_5
    constraint tx_valid_const{
        if(MOSI_array[10:8] == 3'b111 && count > 13 && count < 24) tx_valid == 1;
        else tx_valid == 0;
    }


    function void pre_randomize();
        if(count==0) MOSI_array.rand_mode(1);
        else MOSI_array.rand_mode(0);

        if(MOSI_array[10:8] == 3'b111 && count == 12) tx_data.rand_mode(1);
        else tx_data.rand_mode(0);

    endfunction

    function void post_randomize();
        if(SS_n || !rst_n) count = 0;
        else count++;
        if(!rst_n) state = 0;
        else if(count == 0) state++;
    endfunction

endclass

endpackage
```

Do File

run_spi.do

```
vlib work
vlog +define+SIM -f src_list.list +cover -covercells
vsim -voptargs=+acc work.SPI_slave_top -classdebug -uvmcontrol=all -cover

add wave /SPI_slave_top/slave_if/*

add wave -position insertpoint  \
sim:/SPI_slave_top/DUT/cs  \
sim:/SPI_slave_top/DUT/ns  \
sim:/SPI_slave_top/DUT/counter  \
sim:/SPI_slave_top/GOLDEN/cnt  \
sim:/SPI_slave_top/DUT/received_address  \
sim:/SPI_slave_top/GOLDEN/rx_type

coverage exclude -src SPI_slave.sv -line 38 -code s
coverage exclude -src SPI_slave.sv -line 37 -code b
coverage exclude -src SPI_slave.sv -line 128 -code b


run -all

coverage exclude -cvgpath {/SPI_slave_coverage_pkg/SPI_slave_coverage/covgrp/\/SPI_slave_coverage_pkg::SPI_slave_coverage::covgrp /rx_data_cp/rx_data_trans[0=>2]}


coverage save SLAVE.ucdb -onexit
```

3. BUG REPORT

Bug #1: should be '!received_address' instead to enter READ_ADD state

before

```
43                  if (received_address)
44                      ns = READ_ADD;
45                  else
46                      ns = READ_DATA;
47              end
```

after

```
43                  if (!received_address)
44                      ns = READ_ADD;
45                  else
46                      ns = READ_DATA;
47              end
```

Bug #2: counter should be 9 since it takes one extra cycle to receive data from RAM

before

```
121                 else begin
122                     rx_valid <= 1;
123                     counter <= 8;
124                 end
```

after

```
122                 else begin
123                     rx_valid <= 1;
124                     counter <= 9;
125                 end
```

Bug #3: Counter should be Low when the RST signal is active

before

```
71  always @(posedge clk) begin
72      if (~rst_n) begin
73          rx_data <= 0;
74          rx_valid <= 0;
75          received_address <= 0;
76          MISO <= 0;
77      end
```

after

```
71  always @(posedge clk) begin
72      if (~rst_n) begin
73          rx_data <= 0;
74          rx_valid <= 0;
75          received_address <= 0;
76          MISO <= 0;
77          counter <= 0;
78      end
```

4. COVERAGE

CODE COVERAGE

BRANCH

SLAVE_cvr.txt - Notepad

File Edit Format View Help

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        38        38         0   100.00%

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

Branch Coverage for instance /SPI_slave_top/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File SPI_slave.sv
------------------------------------IF Branch------------------------------------
    20                                      5359     Count coming in to IF
    20              1                        215         if (~rst_n) begin

    23              1                       5144         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    29                                     15029     Count coming in to CASE
    30              1                       2599             IDLE : begin

    36              1                       1992             CHK_CMD : begin

    50              1                       5062             WRITE : begin

    56              1                       2436             READ_ADD : begin

    62              1                       2939             READ_DATA : begin

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

------------------------------------IF Branch------------------------------------
    31                                      2599     Count coming in to IF
    31              1                       1157                 if (SS_n)

    33              1                       1442                 else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    37                                      1992     Count coming in to IF
    39              1                       1992                 else begin

Branch totals: 1 hit of 1 branch = 100.00%

------------------------------------IF Branch------------------------------------
    40                                      1992     Count coming in to IF
    40              1                       1046                     if (~MOSI)

    42              1                        946                     else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    43                                       946     Count coming in to IF
    43              1                        678                         if (!received_address)
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

SLAVE_cvr.txt - Notepad

File Edit Format View Help

```
------------------------------------IF Branch------------------------------------
    43                                       946     Count coming in to IF
    43              1                        678                         if (!received_address)

    45              1                        268                         else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    51                                      5062     Count coming in to IF
    51              1                        674                 if (SS_n)

    53              1                       4388                 else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    57                                      2436     Count coming in to IF
    57              1                        272                 if (SS_n)

    59              1                       2164                 else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                      2939     Count coming in to IF
    63              1                        211                 if (SS_n)

    65              1                       2728                 else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    72                                     20001     Count coming in to IF
    72              1                        215         if (~rst_n) begin

    78              1                      19786         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    79                                     19786     Count coming in to CASE
    80              1                       1345                 IDLE : begin

    83              1                       1334                 CHK_CMD : begin

    86              1                       8549                 WRITE : begin

    95              1                       3404                 READ_ADD : begin

   105              1                       5154                 READ_DATA : begin

Branch totals: 5 hits of 5 branches = 100.00%

------------------------------------IF Branch------------------------------------
    87                                      8549     Count coming in to IF
    87              1                       7207                     if (counter > 0) begin

    91              1                       1342                     else begin
```

Ln 177, Col 84 | 100% | Windows (CRLF) | UTF-8

SLAVE_cvr.txt - Notepad

File Edit Format View Help

```
    105             1                       5154                READ_DATA : begin

Branch totals: 5 hits of 5 branches = 100.00%

------------------------------------IF Branch------------------------------------
    87                                      8549    Count coming in to IF
    87              1                       7207                    if (counter > 0) begin

    91              1                       1342                    else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    96                                      3404    Count coming in to IF
    96              1                       2863                    if (counter > 0) begin

    100             1                        541                    else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    106                                     5154    Count coming in to IF
    106             1                       2200                    if (tx_valid) begin

    116             1                       2954                    else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    108                                     2200    Count coming in to IF
    108             1                       1782                        if (counter > 0) begin

    112             1                        418                        else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    117                                     2954    Count coming in to IF
    117             1                       2718                        if (counter > 0) begin

    122             1                        236                        else begin

Branch totals: 2 hits of 2 branches = 100.00%
```

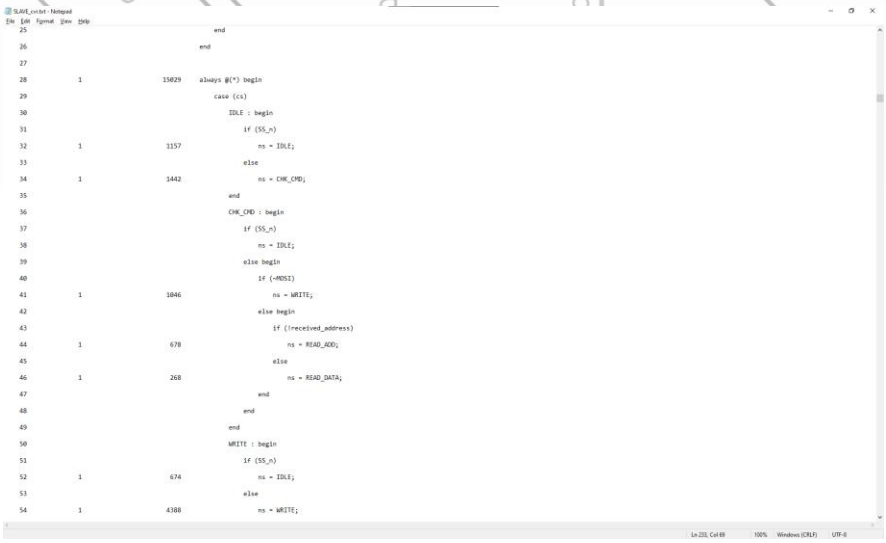
STATEMENT

SLAVE_cvr.txt - Notepad

File Edit Format View Help

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      38        38         0   100.00%

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

Statement Coverage for instance /SPI_slave_top/DUT --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File SPI_slave.sv
    1                                               module SLAVE (MOSI,MISO,SS_n,clk,rst_n,rx_data,rx_valid,tx_data,tx_valid);

    2

    3                                               localparam IDLE      = 3'b000;

    4                                               localparam CHK_CMD   = 3'b001;

    5                                               localparam WRITE     = 3'b010;

    6                                               localparam READ_ADD  = 3'b011;

    7                                               localparam READ_DATA = 3'b100;

    8

    9                                               input           MOSI, clk, rst_n, SS_n, tx_valid;

    10                                              input      [7:0] tx_data;

    11                                              output reg [9:0] rx_data;

    12                                              output reg       rx_valid, MISO;

    13

    14                                              reg [3:0] counter;

    15                                              reg       received_address;

    16

    17                                              reg [2:0] cs, ns;

    18

    19              1                       5359    always @(posedge clk) begin

    20                                                  if (~rst_n) begin

    21              1                        215            cs <= IDLE;

    22                                                  end

    23                                                  else begin

    24              1                       5144            cs <= ns;
```

Ln 233, Col 69 | 100% | Windows (CRLF) | UTF-8

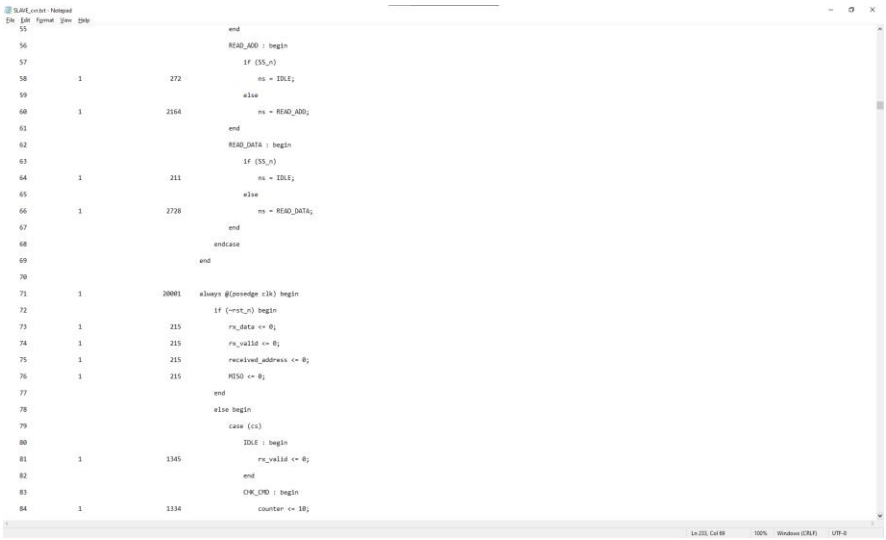


SLAVE_cnt.txt - Notepad

File Edit Format View Help

```
                                    end
                                end

1               15029   always @(*) begin
                                    case (cs)
                                        IDLE : begin
                                            if (SS_n)
1               1157                        ns = IDLE;
                                            else
1               1442                        ns = CHK_CMD;
                                        end
                                        CHK_CMD : begin
                                            if (SS_n)
                                                ns = IDLE;
                                            else begin
                                                if (~MOSI)
1               1046                            ns = WRITE;
                                                else begin
                                                    if (!received_address)
1               678                                 ns = READ_ADD;
                                                    else
1               268                                 ns = READ_DATA;
                                                end
                                            end
                                        end
                                        WRITE : begin
                                            if (SS_n)
1               674                             ns = IDLE;
                                            else
1               4388                            ns = WRITE;
```

Ln 233, Col 69 100% Windows (CRLF) UTF-8

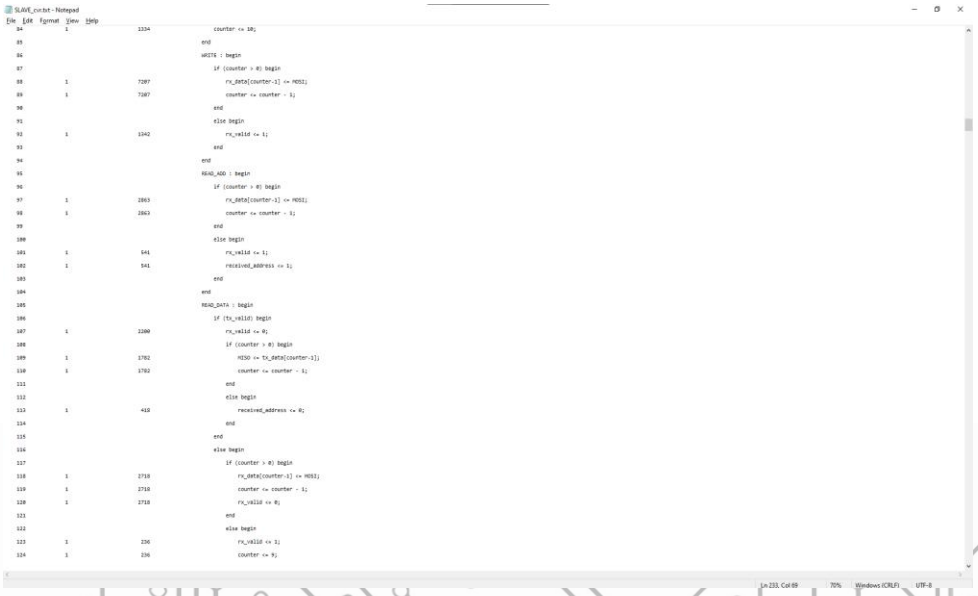


SLAVE_cnt.txt - Notepad

File Edit Format View Help

```
                                        end
                                        READ_ADD : begin
                                            if (SS_n)
1               272                             ns = IDLE;
                                            else
1               2164                            ns = READ_ADD;
                                        end
                                        READ_DATA : begin
                                            if (SS_n)
1               211                             ns = IDLE;
                                            else
1               2728                            ns = READ_DATA;
                                        end
                                    endcase
                                end

1               20001   always @(posedge clk) begin
                                    if (~rst_n) begin
1               215                     rx_data <= 0;
1               215                     rx_valid <= 0;
1               215                     received_address <= 0;
1               215                     MISO <= 0;
                                    end
                                    else begin
                                        case (cs)
                                            IDLE : begin
1               1345                            rx_valid <= 0;
                                            end
                                            CHK_CMD : begin
1               1334                            counter <= 10;
```

Ln 233, Col 69 100% Windows (CRLF) UTF-8



SLAVE_cnt.txt - Notepad

File Edit Format View Help

```
1               1334                counter <= 10;
                                    end
                                    WRITE : begin
                                        if (counter > 0) begin
1               7207                    rx_data[counter-1] <= MOSI;
1               7207                    counter <= counter - 1;
                                        end
                                        else begin
1               1342                    rx_valid <= 1;
                                        end
                                    end
                                    READ_ADD : begin
                                        if (counter > 0) begin
1               2863                    rx_data[counter-1] <= MOSI;
1               2863                    counter <= counter - 1;
                                        end
                                        else begin
1               541                     rx_valid <= 1;
1               541                     received_address <= 1;
                                        end
                                    end
                                    READ_DATA : begin
                                        if (tx_valid) begin
1               2200                    rx_valid <= 0;
                                        if (counter > 0) begin
1               1702                        MISO <= tx_data[counter-1];
1               1702                        counter <= counter - 1;
                                        end
                                        else begin
1               418                         received_address <= 0;
                                        end
                                    end
                                    else begin
                                        if (counter > 0) begin
1               1718                        rx_data[counter-1] <= MOSI;
1               1718                        counter <= counter - 1;
1               1718                        rx_valid <= 0;
                                        end
                                        else begin
1               236                         rx_valid <= 1;
1               236                         counter <= 9;
```

Ln 233, Col 69 70% Windows (CRLF) UTF-8

TOGGLE

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

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

Toggle Coverage for instance /SPI_slave_top/DUT --

                                              Node      1H->0L      0L->1H  "Coverage"
                                              ----------------------------------------
                                              MISO           1           1      100.00
                                              MOSI           1           1      100.00
                                              SS_n           1           1      100.00
                                               clk           1           1      100.00
                                     counter[0-3]           1           1      100.00
                                          cs[0-2]           1           1      100.00
                                          ns[0-2]           1           1      100.00
                                 received_address           1           1      100.00
                                             rst_n           1           1      100.00
                                     rx_data[0-9]           1           1      100.00
                                        rx_valid            1           1      100.00
                                     tx_data[0-7]           1           1      100.00
                                        tx_valid            1           1      100.00

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

Toggle Coverage      =     100.00% (72 of 72 bins)
```

FSM

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

--------------------------------FSM Details--------------------------------

FSM Coverage for instance /SPI_slave_top/DUT --

FSM_ID: cs
    Current State Object : cs
    ---------------------
    State Value MapInfo :
    ---------------------
Line           State Name               Value
----           ----------               -----
  30                 IDLE                   0
  36              CHK_CMD                   1
  62            READ_DATA                   4
  56             READ_ADD                   3
  50                WRITE                   2
    Covered States :
    ----------------
                    State           Hit_count
                    -----           ---------
                     IDLE                1359
                  CHK_CMD                1345
                READ_DATA                 531
                 READ_ADD                 599
                    WRITE                1525
    Covered Transitions :
    ---------------------
Line            Trans_ID          Hit_count          Transition
----            --------          ---------          ----------
  34                   0               1345          IDLE -> CHK_CMD
  46                   1                266          CHK_CMD -> READ_DATA
  44                   2                302          CHK_CMD -> READ_ADD
  41                   3                766          CHK_CMD -> WRITE
  38                   4                 11          CHK_CMD -> IDLE
  64                   5                266          READ_DATA -> IDLE
  58                   6                302          READ_ADD -> IDLE
  52                   7                765          WRITE -> IDLE


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

Functional Coverage

SLAVE_cvr.txt - Notepad

File Edit Format View Help

```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------------
Covergroup                                                Metric       Goal       Bins    Status
----------------------------------------------------------------------------------------------------------------
 TYPE /SPI_slave_coverage_pkg/SPI_slave_coverage/covgrp
                                                         100.00%        100          -    Covered
    covered/total bins:                                       21         21          -
    missing/total bins:                                        0         21          -
    % Hit:                                               100.00%        100          -
    Coverpoint rx_data_cp                                 91.66%        100          -    Uncovered
        covered/total bins:                                   11         12          -
        missing/total bins:                                    1         12          -
        % Hit:                                            91.66%        100          -
    Coverpoint SS_n_cp                                   100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                           100.00%        100          -
    Coverpoint MOSI_cp                                   100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                           100.00%        100          -
    Cross SS_n_MOSI_cr                                   100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                           100.00%        100          -
 Covergroup instance \/SPI_slave_coverage_pkg::SPI_slave_coverage::covgrp
                                                         100.00%        100          -    Covered
    covered/total bins:                                       21         21          -
    missing/total bins:                                        0         21          -
    % Hit:                                               100.00%        100          -
    Coverpoint rx_data_cp                                100.00%        100          -    Covered
        covered/total bins:                                   11         11          -
        missing/total bins:                                    0         11          -
        % Hit:                                           100.00%        100          -
        bin rx_data_all_values[0]                           6202          1          -    Covered
        bin rx_data_all_values[1]                           4298          1          -    Covered
        bin rx_data_all_values[2]                           5314          1          -    Covered
        bin rx_data_all_values[3]                           4187          1          -    Covered
        bin rx_data_trans[0=>1]                              344          1          -    Covered
        bin rx_data_trans[1=>3]                              297          1          -    Covered
        bin rx_data_trans[1=>0]                              149          1          -    Covered
        bin rx_data_trans[2=>3]                              259          1          -    Covered
        bin rx_data_trans[2=>0]                              150          1          -    Covered
        bin rx_data_trans[3=>1]                              102          1          -    Covered
        bin rx_data_trans[3=>2]                              409          1          -    Covered
    Coverpoint SS_n_cp                                   100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                           100.00%        100          -
        bin normal_trans                                     774          1          -    Covered
        bin extended_trans                                   224          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_address                                   2946          1          -    Covered
        bin write_data                                      2541          1          -    Covered
        bin read_address                                    2470          1          -    Covered
        bin read_data                                       2672          1          -    Covered
    Cross SS_n_MOSI_cr                                   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 <extended_trans,read_data>                    26          1          -    Covered
            bin <normal_trans,read_address>                  107          1          -    Covered
            bin <normal_trans,write_data>                     99          1          -    Covered
            bin <normal_trans,write_address>                 104          1          -    Covered
        Illegal and Ignore Bins:
            ignore_bin not_r_data_ext4                        82                     -    Occurred
            ignore_bin not_r_data_ext3                        27                     -    Occurred
            ignore_bin not_r_data_ext2                        36                     -    Occurred
            ignore_bin not_r_data_ext1                        22                     -    Occurred

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

Ln 233, Col 69 | 80% | Windows (CRLF) | UTF-8

Assertion Coverage

```
DIRECTIVE COVERAGE:
-------------------------------------------------------------------------------------------------------------
Name                                      Design Design   Lang File(Line)      Hits Status
                                          Unit   UnitType
-------------------------------------------------------------------------------------------------------------
/SPI_slave_top/DUT/cover__received_address_p2
                                          SLAVE  Verilog  SVA  SPI_slave.sv(253)
                                                                                412 Covered
/SPI_slave_top/DUT/cover__received_address_p1
                                          SLAVE  Verilog  SVA  SPI_slave.sv(252)
                                                                                537 Covered
/SPI_slave_top/DUT/cover__counter_write_p
                                          SLAVE  Verilog  SVA  SPI_slave.sv(251)
                                                                              14409 Covered
/SPI_slave_top/DUT/cover__counter_rst_p   SLAVE  Verilog  SVA  SPI_slave.sv(250)
                                                                               1321 Covered
/SPI_slave_top/DUT/cover__cs_p9           SLAVE  Verilog  SVA  SPI_slave.sv(249)
                                                                               4889 Covered
/SPI_slave_top/DUT/cover__cs_p8           SLAVE  Verilog  SVA  SPI_slave.sv(248)
                                                                               3107 Covered
/SPI_slave_top/DUT/cover__cs_p7           SLAVE  Verilog  SVA  SPI_slave.sv(247)
                                                                               7790 Covered
/SPI_slave_top/DUT/cover__cs_p6           SLAVE  Verilog  SVA  SPI_slave.sv(246)
                                                                                297 Covered
/SPI_slave_top/DUT/cover__cs_p4           SLAVE  Verilog  SVA  SPI_slave.sv(245)
                                                                                265 Covered
/SPI_slave_top/DUT/cover__cs_p3           SLAVE  Verilog  SVA  SPI_slave.sv(244)
                                                                                759 Covered
/SPI_slave_top/DUT/cover__cs_p2           SLAVE  Verilog  SVA  SPI_slave.sv(243)
                                                                               1334 Covered
/SPI_slave_top/DUT/cover__cs_p1           SLAVE  Verilog  SVA  SPI_slave.sv(242)
                                                                               1132 Covered
/SPI_slave_top/DUT/cover__rx_valid_p      SLAVE  Verilog  SVA  SPI_slave.sv(241)
                                                                                830 Covered
/SPI_slave_top/DUT/cover__read_seq_p      SLAVE  Verilog  SVA  SPI_slave.sv(240)
                                                                                562 Covered
/SPI_slave_top/DUT/cover__write_seq_p     SLAVE  Verilog  SVA  SPI_slave.sv(239)
                                                                                551 Covered
/SPI_slave_top/DUT/cover__rst_p           SLAVE  Verilog  SVA  SPI_slave.sv(238)
                                                                                215 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 16
```

```
ASSERTION RESULTS:
-----------------------------------------------------------------------
Name                File(Line)                   Failure      Pass
                                                 Count        Count
-----------------------------------------------------------------------
/SPI_slave_top/DUT/assert__received_address_p2
                    SPI_slave.sv(236)                  0           1
/SPI_slave_top/DUT/assert__received_address_p1
                    SPI_slave.sv(235)                  0           1
/SPI_slave_top/DUT/assert__counter_write_p
                    SPI_slave.sv(234)                  0           1
/SPI_slave_top/DUT/assert__counter_rst_p
                    SPI_slave.sv(233)                  0           1
/SPI_slave_top/DUT/assert__cs_p9
                    SPI_slave.sv(232)                  0           1
/SPI_slave_top/DUT/assert__cs_p8
                    SPI_slave.sv(231)                  0           1
/SPI_slave_top/DUT/assert__cs_p7
                    SPI_slave.sv(230)                  0           1
/SPI_slave_top/DUT/assert__cs_p6
                    SPI_slave.sv(229)                  0           1
/SPI_slave_top/DUT/assert__cs_p4
                    SPI_slave.sv(228)                  0           1
/SPI_slave_top/DUT/assert__cs_p3
                    SPI_slave.sv(227)                  0           1
/SPI_slave_top/DUT/assert__cs_p2
                    SPI_slave.sv(226)                  0           1
/SPI_slave_top/DUT/assert__cs_p1
                    SPI_slave.sv(225)                  0           1
/SPI_slave_top/DUT/assert__rx_valid_p
                    SPI_slave.sv(224)                  0           1
/SPI_slave_top/DUT/assert__read_seq_p
                    SPI_slave.sv(223)                  0           1
/SPI_slave_top/DUT/assert__write_seq_p
                    SPI_slave.sv(222)                  0           1
/SPI_slave_top/DUT/assert__rst_p
                    SPI_slave.sv(221)                  0           1
/SPI_slave_main_sequence_pkg/SPI_slave_main_sequence/body/#ublk#7241691$#17/immed__19
                    SPI_slave_main_sequence.sv(19)     0           1
```

NOTES:

1- ADDED A DEFAULT CASE THEN EXCLUDE IT TO ACHIEVE 100% CODE COVERAGE

2- EXCLUSION DONE TO THE BIN RX_DATA_TRANS (2=>0) TO ACHIEVE 100 % FUNCTIONAL COVERAGE (IT WILL BE FIXED IN THE WRAPPER)

5. ASSERTIONS TABLE

Feature	Assertion
Checks that when rst_n is asserted, MISO & rx_valid & rx_data & received_address are all low and cs turns back to IDLE state	`@(posedge clk) (!rst_n) |=> (!MISO && !rx_valid && (rx_data == 10'b00_0000_0000) && (cs == IDLE) && !received_address)`
Checks that when SS_n is deasserted and after 2 clk cycles MOSI is low, the MOSI should be low after another one cycle to have a right sequence of MOSI that drives the SPI to function a write operation correctly with RAM	`@(posedge clk) disable iff (!rst_n) (SS_n == 1 ##2 MOSI == 0) |=> MOSI == 0`
Checks that when SS_n is deasserted and after 2 clk cycles MOSI is high, the MOSI should be high after another one cycle to have a right sequence of MOSI that drives the SPI to function a read operation correctly with RAM	`@(posedge clk) disable iff (!rst_n) (SS_n == 1 ##2 MOSI == 1) |=> MOSI == 1`

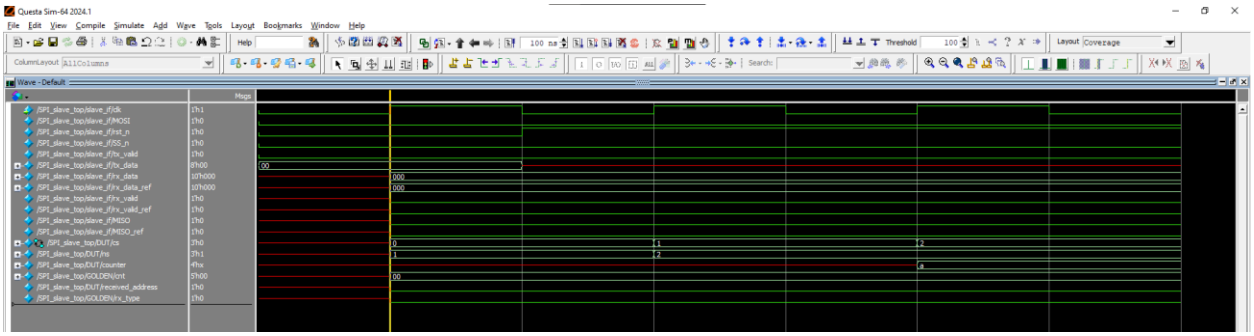
Checks that when SS_n is deasserted and is followed by right sequence of MOSI, the rx_valid is asserted after 10 clk cycles to transmit the data to RAM and SS_n is asserted eventually after that to end communication with the master	`sequence write1_seq_s;`
	`  SS_n == 1 ##2 MOSI == 0 ##1 MOSI == 0 ##1 MOSI == 0;`
	`endsequence`
	
	`sequence read1_seq_s;`
	`  SS_n == 1 ##2 MOSI == 1 ##1 MOSI == 1 ##1 MOSI == 0;`
	`endsequence`
	
	`sequence write2_seq_s;`
	`  SS_n == 1 ##2 MOSI == 0 ##1 MOSI == 0 ##1 MOSI == 1;`
	`endsequence`
	
	`sequence read2_seq_s;`
	`  SS_n == 1 ##2 MOSI == 1 ##1 MOSI == 1 ##1 MOSI == 1;`
	`endsequence`
	
	`sequence rx_ss_n_s;`
	`  ##10 rx_valid ##[1:$] SS_n;`
	`endsequence`
	
	`@(posedge clk) disable iff (!rst_n)`
	`  (write1_seq_s or read1_seq_s or write2_seq_s or read2_seq_s) |-> rx_ss_n_s`

Checks that when SS_n is deasserted, the current state turns back to IDLE state to end communication and be ready for a new one	`@(posedge clk) disable iff (!rst_n) (SS_n) |=> (cs == IDLE)`
Checks that when SS_n is asserted & the current state is IDLE, the next current state is CHK_CMD	`@(posedge clk) disable iff (!rst_n) (cs == IDLE && !SS_n) |=> (cs == CHK_CMD)`

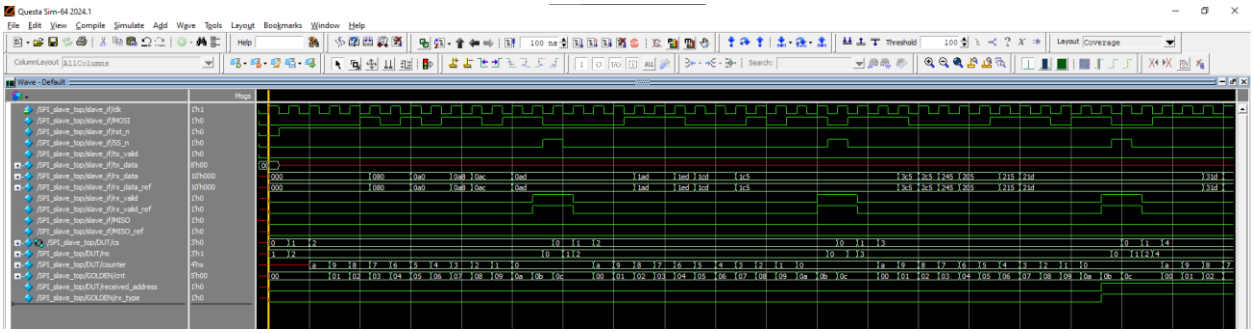
Checks that when SS_n is asserted & the current state is IDLE, if the MOSI == 0 then the next current state is WRITE	`@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && !MOSI) |=> (cs == WRITE)`
Checks that when SS_n is asserted & the current state is IDLE, if the MOSI == 1 and the received_address is high then the next current state is READ_DATA	`@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && MOSI && received_address) |=> (cs == READ_DATA)`
Checks that when SS_n is asserted & the current state is IDLE, if the MOSI == 1 and the received_address is low then the next current state is READ_ADD	`@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD && !SS_n && MOSI && !received_address) |=> (cs == READ_ADD)`
Checks that when SS_n is deasserted & the current state is WRITE, the current state stays WRITE for the next cycle	`@(posedge clk) disable iff (!rst_n) (cs == WRITE && !SS_n) |=> (cs == WRITE)`
Checks that when SS_n is deasserted & the current state is READ_ADD, the current state stays READ_ADD for the next cycle	`@(posedge clk) disable iff (!rst_n) (cs == READ_ADD && !SS_n) |=> (cs == READ_ADD)`
Checks that when SS_n is deasserted & the current state is READ_DATA, the current state stays READ_DATA for the next cycle	`@(posedge clk) disable iff (!rst_n) (cs == READ_DATA && !SS_n) |=> (cs == READ_DATA)`
Checks that when the current state is CHK_CMD, the counter resets to 10 to start a new operation	`@(posedge clk) disable iff (!rst_n) (cs == CHK_CMD) |=> (counter == 4'b1010)`
Checks that when the current state is a normal operation, the counter should decrement every clk cycle	`@(posedge clk) disable iff (!rst_n) ((cs == WRITE || cs == READ_ADD || cs == READ_DATA) && counter > 0) |=> (counter == $past(counter) - 1'b1)`
Checks that when the current state is a READ_ADD & the counter is zero, the received_address should be high to make the next read state READ_DATA	`@(posedge clk) disable iff (!rst_n) (cs == READ_ADD && counter == 0) |=> (received_address)`
Checks that when the current state is a READ_DATA & the counter is zero & tx_valid is assertes, the received_address should be low to make the next read state READ_ADD	`@(posedge clk) disable iff (!rst_n) (cs == READ_DATA && tx_valid && counter == 0) |=> (!received_address)`

6. SIMULATION SNIPPETS

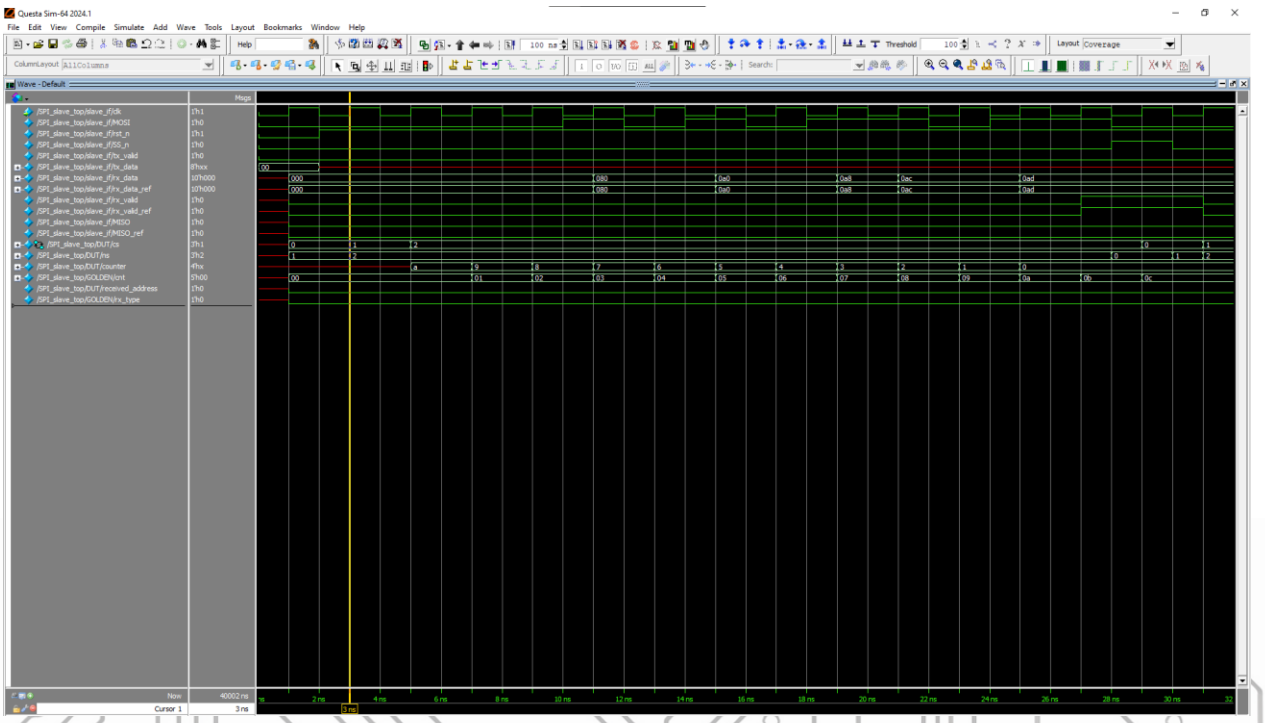
RESET SEQUENCE

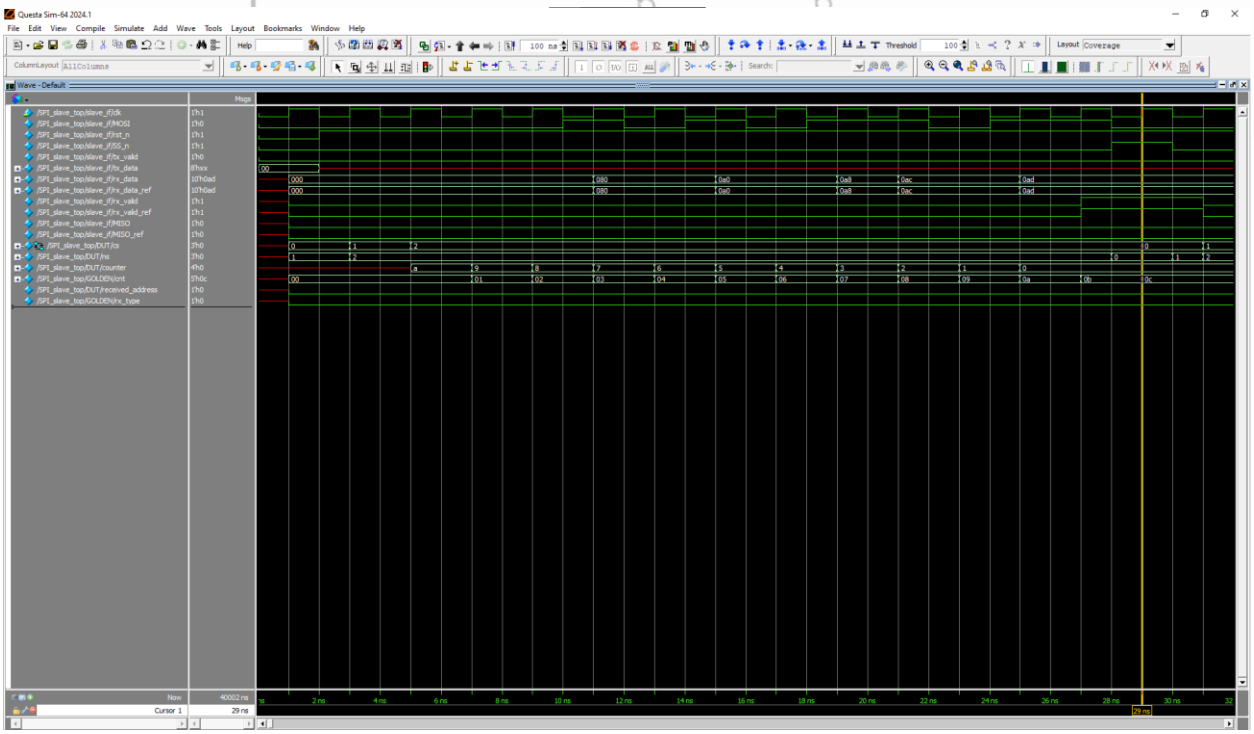


MAIN SEQUENCE

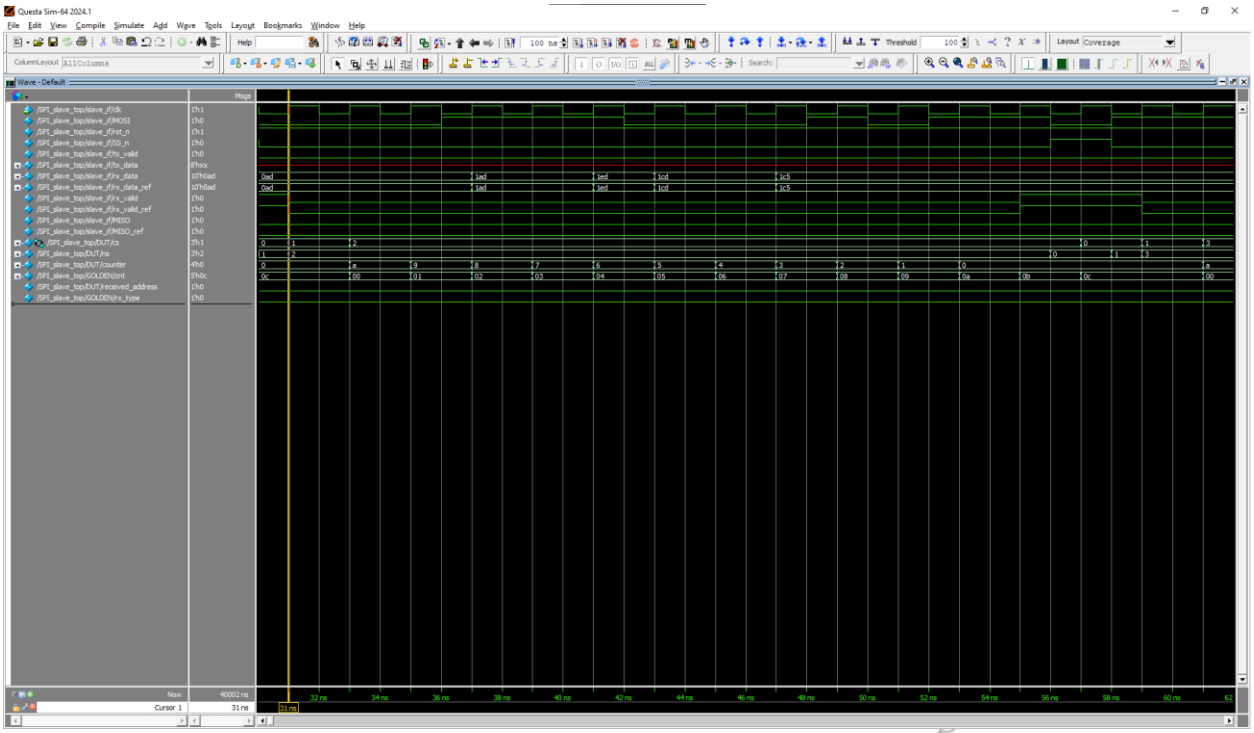


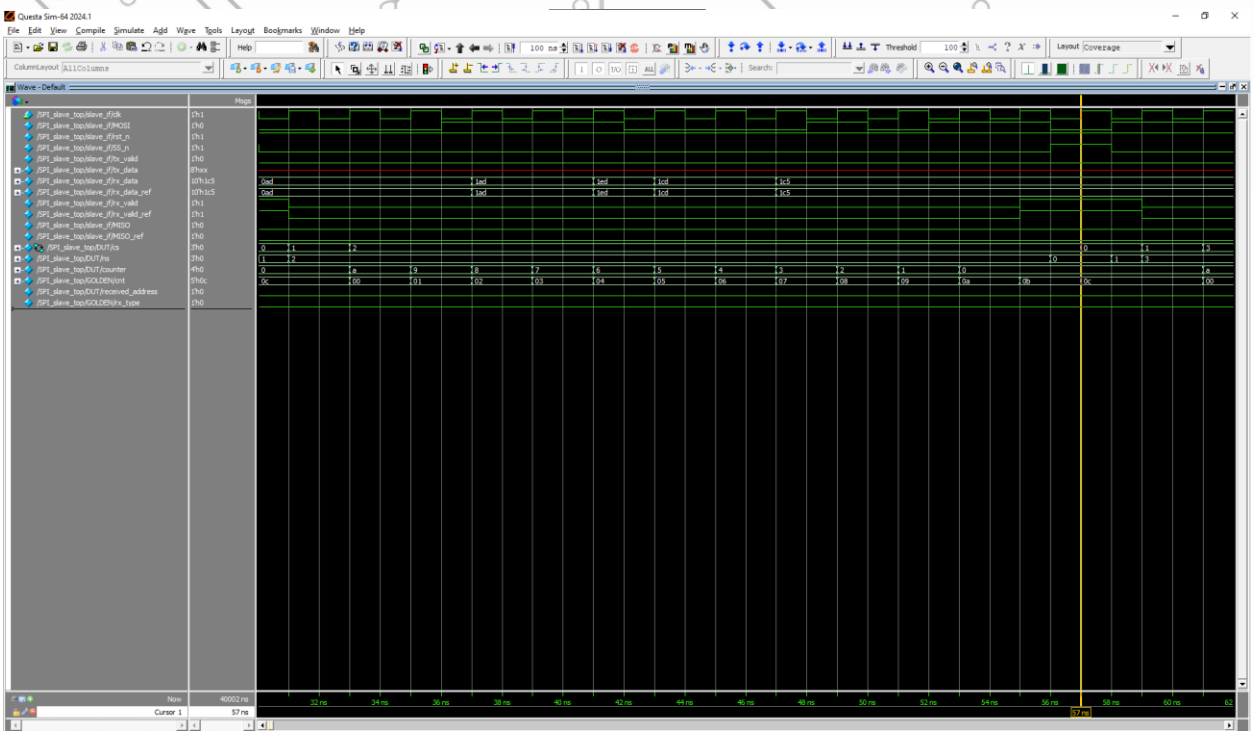
WRITE ADDRESS: (FROM 3NS TO 29 NS, 13 CLK CYCLES)



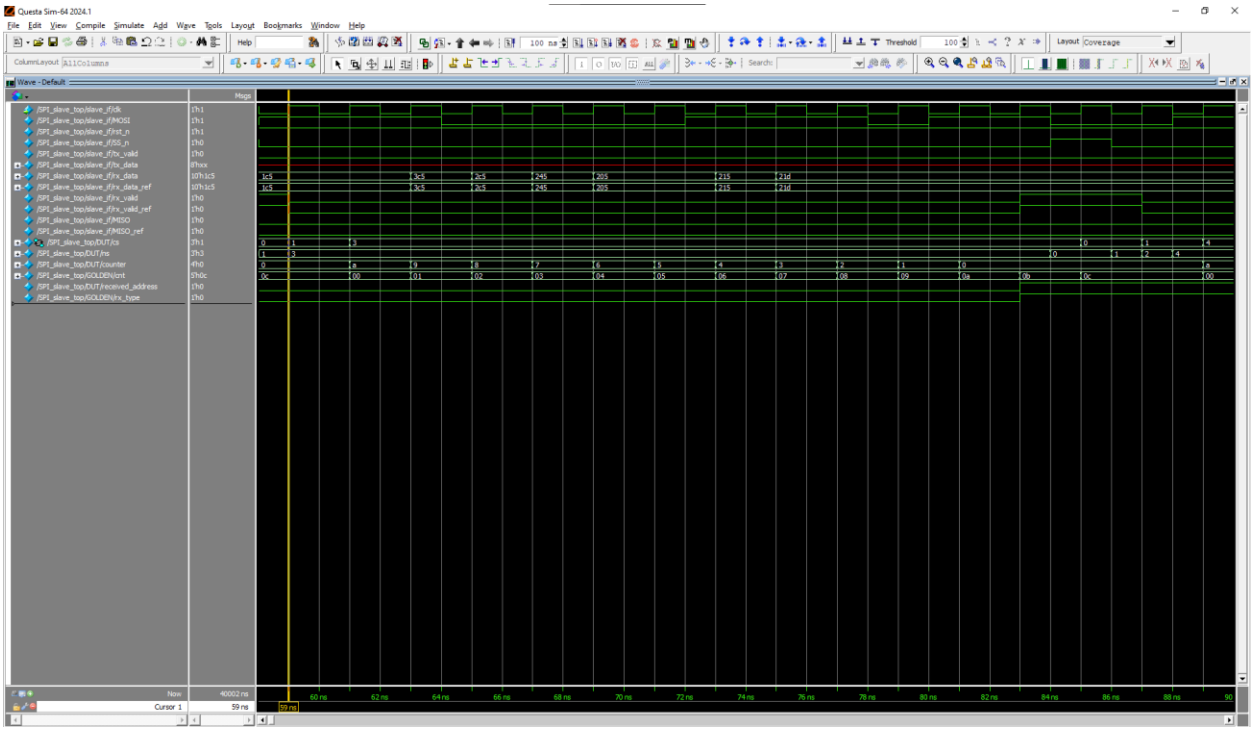


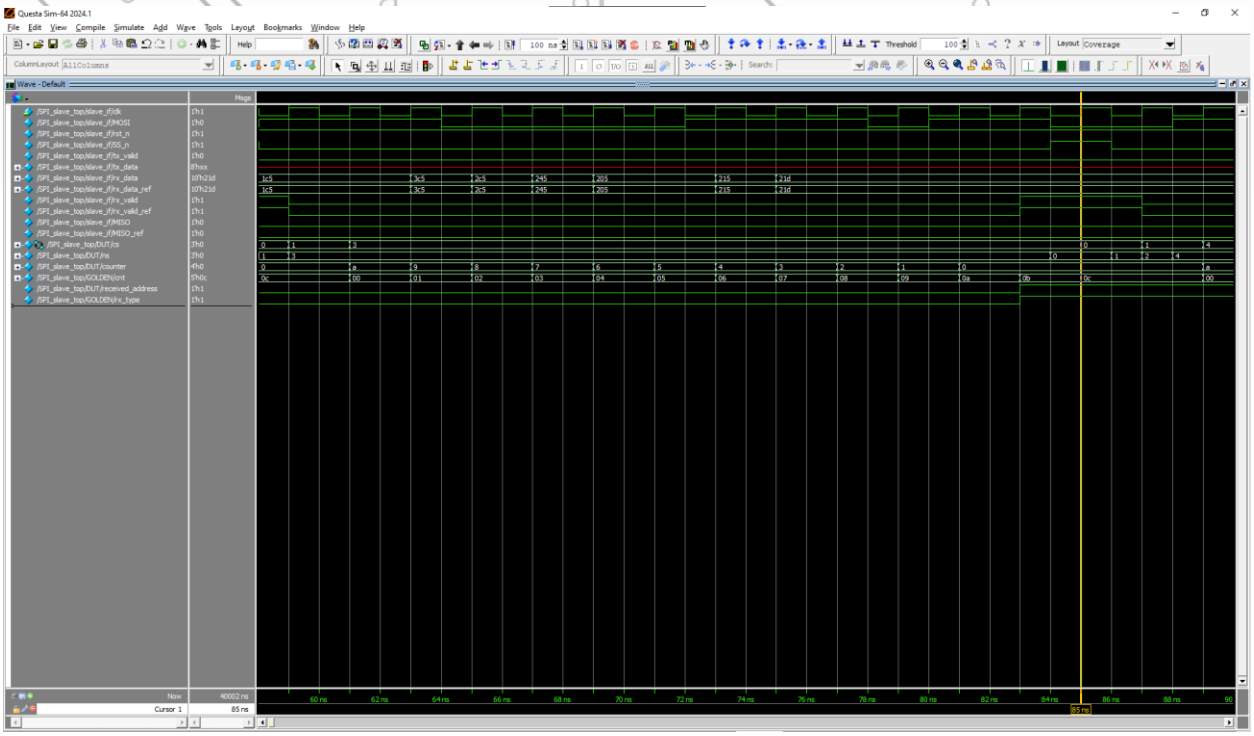
WRITE DATA: (FROM 31 NS TO 57 NS, 13 CLK CYCLES)



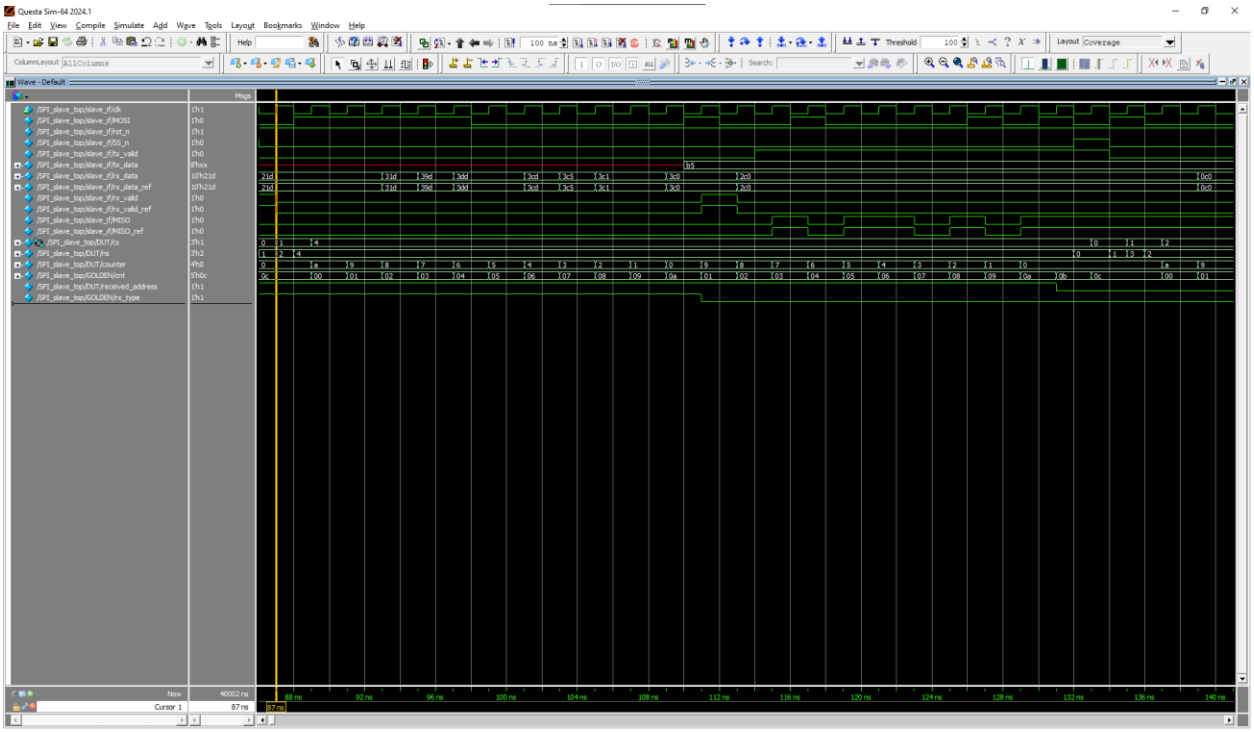


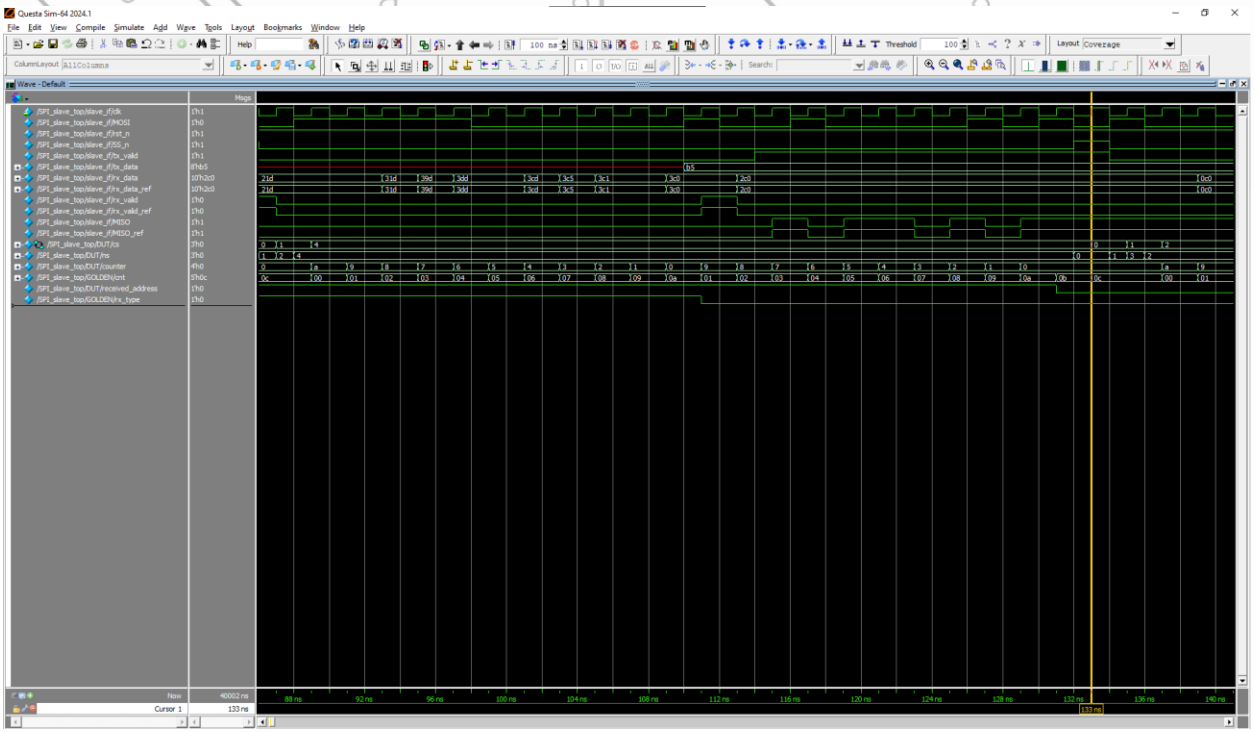
READ ADDRESS: (FROM 59 NS TO 85 NS, 13 CLK CYCLES)





READ DATA: (FROM 87 NS (33 NS, 23 CLK CYCLES)



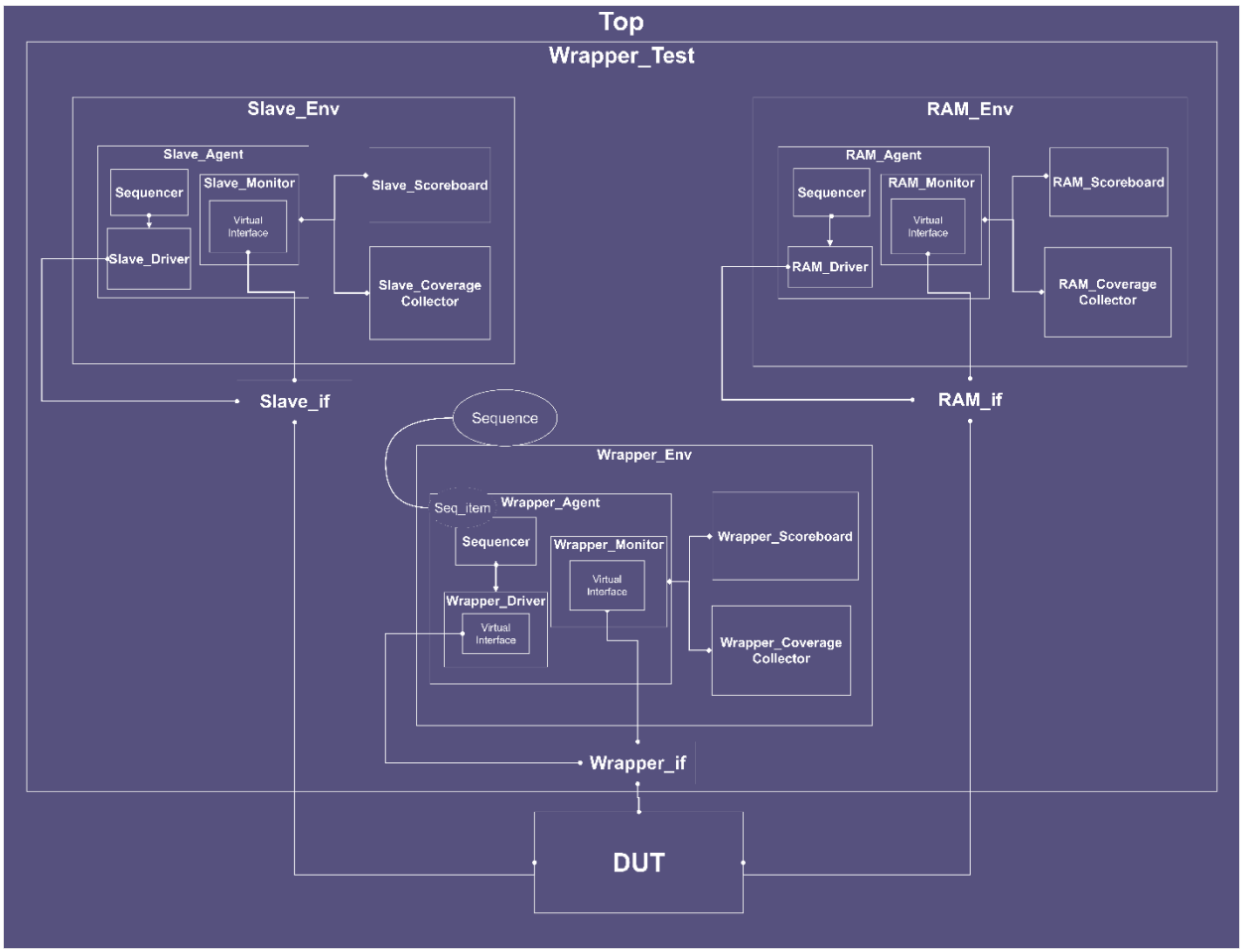


Transcript snippets

```
# (C) 2011-2013 Cypress Semiconductor Corp.
# ----------------------------------------------------------------
#
#    ***********       IMPORTANT RELEASE NOTES         ************
#
#    You are using a version of the UVM library that has been compiled
#    with `UVM_NO_DEPRECATED undefined.
#    See http://www.eda.org/svdb/view.php?id=3313 for more details.
#
#    You are using a version of the UVM library that has been compiled
#    with `UVM_OBJECT_MUST_HAVE_CONSTRUCTOR undefined.
#    See http://www.eda.org/svdb/view.php?id=3770 for more details.
#
#        (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test SPI_slave_test...
# UVM_INFO SPI_slave_test.sv(40) @ 0: uvm_test_top [run_phase] Reset Asserted
# ********************************************************************************
# * Questa UVM Transaction Recording Turned ON.                                  *
# * recording_detail has been set.                                               *
# *  To turn off, set 'recording_detail' to off:                                 *
# *  uvm_config_db#(int)            ::set(null, "", "recording_detail", 0);      *
# *  uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0);      *
# ********************************************************************************
# UVM_INFO SPI_slave_test.sv(42) @ 2: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO SPI_slave_test.sv(44) @ 2: uvm_test_top [run_phase] Main Sequence Started
# UVM_INFO SPI_slave_test.sv(46) @ 40002: uvm_test_top [run_phase] Main Sequence Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 40002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO SPI_slave_scoreboard.sv(61) @ 40002: uvm_test_top.env.sb [report_phase] SPI SLAVE SCOREBOARD
# UVM_INFO SPI_slave_scoreboard.sv(62) @ 40002: uvm_test_top.env.sb [report_phase] Total successful rx_data:       20001
# UVM_INFO SPI_slave_scoreboard.sv(63) @ 40002: uvm_test_top.env.sb [report_phase] Total failed rx_data:           0
# UVM_INFO SPI_slave_scoreboard.sv(64) @ 40002: uvm_test_top.env.sb [report_phase] Total successful rx_valid:     20001
# UVM_INFO SPI_slave_scoreboard.sv(65) @ 40002: uvm_test_top.env.sb [report_phase] Total failed rx_valid:           0
# UVM_INFO SPI_slave_scoreboard.sv(66) @ 40002: uvm_test_top.env.sb [report_phase] Total successful MISO:      20001
# UVM_INFO SPI_slave_scoreboard.sv(67) @ 40002: uvm_test_top.env.sb [report_phase] Total failed MISO:           0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   15
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     7
# [run_phase]     4
# ** Note: $finish    : C:/questasim64_2024.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 40002 ns  Iteration: 61  Instance: /SPI_slave_top
```

SECTION #3: VERIFICATION OF SPI WRAPPER

1. UVM STRUCTURE



UVM Testbench Overview

THE TOP MODULE INSTANTIATES THE DESIGN UNDER VERIFICATION (DUV) MODULES (SPI WRAPPER, SPI WRAPPER REFERENCE, SPI SLAVE, AND RAM) AND CREATES THEIR CORRESPONDING INTERFACES, WHICH ARE THEN PASSED TO THE RESPECTIVE MODULES. IT ALSO GENERATES THE SYSTEM CLOCK AND CONNECTS IT TO ALL INTERFACES. FINALLY, IT SETS THE CREATED INTERFACES INSIDE THE UVM CONFIGURATION DATABASE AND STARTS THE WRAPPER TEST.

IN THE BUILD PHASE, THE WRAPPER TEST CREATES THE ENVIRONMENTS FOR ALL DESIGN MODULES (SPI WRAPPER, SPI SLAVE, AND RAM), AS WELL AS A CONFIGURATION OBJECT FOR EACH ENVIRONMENT AND THE RANDOMIZATION SEQUENCES USED FOR VERIFICATION. THE TEST THEN RETRIEVES THE INTERFACES FROM THE TOP MODULE, ASSIGNS THEM TO THEIR RESPECTIVE CONFIGURATION OBJECTS, AND STORES THESE OBJECTS IN THE UVM CONFIGURATION DATABASE. THE SPI SLAVE AND RAM AGENTS ARE SET AS PASSIVE AGENTS.

DURING THE RUN PHASE, THE TEST STARTS EACH RANDOMIZATION SEQUENCE CREATED IN THE BUILD PHASE.

EACH ENVIRONMENT, IN ITS BUILD PHASE, CREATES THE AGENT, SCOREBOARD, AND COVERAGE COLLECTOR. IN THE CONNECT PHASE, IT CONNECTS THE AGENT'S ANALYSIS PORTS TO BOTH THE SCOREBOARD AND THE COVERAGE COLLECTOR.

EACH AGENT, IN ITS BUILD PHASE, CREATES ITS CONFIGURATION OBJECT, RETRIEVES THE ONE PASSED FROM THE TEST, AND THEN CREATES THE MONITOR. IF THE AGENT IS ACTIVE, IT ALSO CREATES THE DRIVER AND SEQUENCER. IN THE CONNECT PHASE, THE AGENT CONNECTS THE DRIVER AND MONITOR VIRTUAL INTERFACES TO THE ONES IN THE CONFIGURATION OBJECT (WHICH POINT TO THE ACTUAL INTERFACES IN THE TOP MODULE). IT ALSO CONNECTS ITS ANALYSIS PORT TO THE MONITOR PORT TO RECEIVE MONITORED TRANSACTIONS FOR THE SCOREBOARD AND COVERAGE COLLECTOR.

THE SEQUENCER RECEIVES RANDOMIZED SEQUENCE ITEMS FROM THE SEQUENCE STARTED IN THE TEST AND PASSES THEM TO THE DRIVER.

THE DRIVER RECEIVES THESE SEQUENCE ITEMS FROM THE SEQUENCER AND DRIVES THE DUT'S INPUT SIGNALS THROUGH THE INTERFACE.

IN THE BUILD PHASE, THE MONITOR CREATES ITS OWN SEQUENCE ITEM, SAMPLES SIGNAL VALUES FROM THE DUT VIA THE VIRTUAL INTERFACE, FILLS THE SEQUENCE ITEM WITH THESE VALUES, AND SENDS IT BACK TO THE AGENT.

AFTER THE AGENT RECEIVES THE MONITORED SEQUENCE ITEM FROM THE MONITOR, IT FORWARDS IT TO BOTH THE SCOREBOARD AND THE COVERAGE COLLECTOR.

THE SCOREBOARD RECEIVES THE MONITORED SEQUENCE ITEM, COMPARES IT WITH THE EXPECTED (GOLDEN) MODEL OUTPUT, AND REPORTS AN ERROR IF A MISMATCH OCCURS.

THE COVERAGE COLLECTOR DEFINES THE COVERGROUPS REQUIRED TO MEASURE THE FUNCTIONAL COVERAGE OF THE DESIGN. IT RECEIVES THE MONITORED SEQUENCE ITEMS AND USES THE COVERGROUP TO SAMPLE DATA AND EVALUATE DESIGN FUNCTIONALITY COVERAGE.

2. VERIFICATION PLAN

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
WRAPPER_1	When the reset is asserted, the output (MISO) should be zero	Directed at the start of the simulation then randomized with constrains to be off most of the time	-	The outputs are checked using both golden model and assertions
WRAPPER_2	In the write only sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 000, then the slave sends the incoming values of the MOSI to the ram then store it as write address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI value to be 000 or 001 in the next first 3 cycles after SS_n equal to zero	-	The outputs are checked using both golden model and assertions
WRAPPER_3	In the write only sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 001, then the slave sends the incoming values of the MOSI to the ram then it stores them in the memory using the write address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI value to be 000 or 001 in the next first 3 cycles after SS_n equal to zero	-	The outputs are checked using both golden model and assertions
WRAPPER_4	In the read only sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 110, then the slave sends the incoming values of the MOSI to the ram then store it as read address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 110 then the next values will be 111 and if its previous values was 111 then the next values will be 110, if none then it will be 110	-	The outputs are checked using both golden model and assertions
WRAPPER_5	In the read only sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 110, then the slave sends the incoming values of the MOSI to the ram then sends the data of the memory at read address to the slave then the slave output this data bit by bit on the MISO	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 110 then the next values will be 111 and if its previous values was 111 then the next values will be 110, if none then it will be 111	-	The outputs are checked using both golden model and assertions
	In the write read sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 000, then the slave sends the incoming values of the MOSI to the ram then store it as write address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 000 then the next values will be 001 or 000 and if its previous	-	The outputs are checked using both golden model and assertions

Plan | Assertions

	A	B	C	D	E	F	G	H	I
7	WRAPPER_6	In the write read sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 000, then the slave sends the incoming values of the MOSI to the ram then store it as write address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 000 then the next values will be 001 or 000 and if its previous values was 001 then the next values will be 60% 110 or 40% 000, if its previous values was 110 then the next values will be 111 and if its previous values was 111 then the next values will be 60% 000 or 40% 110,	-	The outputs are checked using both golden model and assertions				
8	WRAPPER_7	In the write read sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 001, then the slave sends the incoming values of the MOSI to the ram then it stores them in the memory using the write address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 000 then the next values will be 001 or 000 and if its previous values was 001 then the next values will be 60% 110 or 40% 000, if its previous values was 110 then the next values will be 111 and if its previous values was 111 then the next values will be 60% 000 or 40% 110,	-	The outputs are checked using both golden model and assertions				
9	WRAPPER_8	In the write read sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 110, then the slave sends the incoming values of the MOSI to the ram then store it as read address	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 000 then the next values will be 001 or 000 and if its previous values was 001 then the next values will be 60% 110 or 40% 000, if its previous values was 110 then the next values will be 111 and if its previous values was 111 then the next values will be 60% 000 or 40% 110,	-	The outputs are checked using both golden model and assertions				
10	WRAPPER_9	In the write read sequence, when the SS_n is low and the value of MOSI in the following 3 cylces is 110, then the slave sends the incoming values of the MOSI to the ram then sends the data of the memory at read address to the slave then the slave output this data bit by bit on the MISO	Randomization with constraints on SS_n to be one every 13 cycle if the internal state of the slave is not READ_DATA and to be one every 23 cycles if the internal state of the slave is READ_DATA, and constraints on MOSI in the next first 3 cycles after SS_n equal to zero that if its previous values was 000 then the next values will be 001 or 000 and if its previous values was 001 then the next values will be 60% 110 or 40% 000, if its previous values was 110 then the next values will be 111 and if its previous values was 111 then the next values will be 60% 000 or 40% 110,	-	The outputs are checked using both golden model and assertions				

Plan | Assertions | +

3. CODE SNIPPETS

INTERFACE

WRAPPER_if.sv

```sv
interface WRAPPER_IF(clk);
    input           clk;
    //Module Signals
    logic   MOSI, SS_n, rst_n;
    logic   MISO;
    logic   MISO_ref;
endinterface
```

DESIGN

SPI_wrapper.v

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

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

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

RAM   RAM_instance   (
    .din(rx_data_din),
    .clk(clk),
    .rst_n(rst_n),
    .rx_valid(rx_valid),
    .dout(tx_data_dout),
    .tx_valid(tx_valid)
);
SLAVE SLAVE_instance (
    .MOSI(MOSI),
    .MISO(MISO),
    .SS_n(SS_n),
    .clk(clk),
    .rst_n(rst_n),
    .rx_data(rx_data_din),
    .rx_valid(rx_valid),
    .tx_data(tx_data_dout),
    .tx_valid(tx_valid)
);

endmodule
```

Golen Model

SPI_wrapper_ref.v

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

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

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

RAM_ref   RAM_instance   (
    .clk(clk),
    .rst_n(rst_n),
    .rx_valid(rx_valid),
    .rx_data(rx_data_din),
    .tx_valid(tx_valid),
    .tx_data(tx_data_dout)
);
SLAVE_golden_model SLAVE_instance (
    .clk(clk),
    .rst_n(rst_n),
    .SS_n(SS_n),
    .MOSI(MOSI),
    .tx_valid(tx_valid),
    .tx_data(tx_data_dout),
    .MISO(MISO),
    .rx_valid(rx_valid),
    .rx_data(rx_data_din)
);

endmodule
```

SVA

WRAPPER_sva.sv

```
module WRAPPER_sva(MOSI,MISO,SS_n,clk,rst_n,rx_data_din,rx_valid,tx_valid,operation);
    input  MOSI, SS_n, clk, rst_n;
    input MISO;
    input [9:0] rx_data_din;
    input       rx_valid;
    input       tx_valid;
    input [0:2] operation;

    property rst_wrapper_p;
        @(posedge clk) (!rst_n) |-> (!MISO && (rx_data_din == 10'b00_0000_0000) && !rx_valid);
    endproperty

    property MISO_wrapper_p;
        @(posedge clk) disable iff(!rst_n) (operation != 3'b100) |=> (MISO == $past(MISO));
    endproperty

    assert property (rst_wrapper_p);
    assert property (MISO_wrapper_p);

    cover property (rst_wrapper_p);
    cover property (MISO_wrapper_p);
endmodule
```

Config

WRAPPER_config.sv

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

        virtual WRAPPER_IF wrapper_if;
        uvm_active_passive_enum is_active;

        function new(string name = "WRAPPER_config");
            super.new(name);
        endfunction //new()
    endclass //WRAPPER_config extends uvm_object
endpackage
```

Top

WRAPPER_top.sv

```
import uvm_pkg::*;
`include"uvm_macros.svh"
import WRAPPER_test_pkg::*;
module WRAPPER_top;
    logic clk = 0;
    initial begin
        $readmemb("mem.dat", DUT_WRAPPER.RAM_instance.MEM);
        $readmemb("mem.dat", GM_WRAPPER.RAM_instance.mem);

        forever #1 clk = ~clk;
    end

    WRAPPER_IF wrapper_if(clk);
    RAM_IF ram_if(clk);
    SPI_slave_if slave_if(clk);

    WRAPPER DUT_WRAPPER (
        .clk(wrapper_if.clk),
        .rst_n(wrapper_if.rst_n),
        .MOSI(wrapper_if.MOSI),
        .SS_n(wrapper_if.SS_n),
        .MISO(wrapper_if.MISO)
    );

    WRAPPER_ref GM_WRAPPER(
        .clk(wrapper_if.clk),
        .rst_n(wrapper_if.rst_n),
        .MOSI(wrapper_if.MOSI),
        .SS_n(wrapper_if.SS_n),
        .MISO(wrapper_if.MISO_ref)
    );

//RAM Inputs
    assign ram_if.rst_n = DUT_WRAPPER.rst_n;
    assign ram_if.rx_valid = DUT_WRAPPER.rx_valid;
    assign ram_if.din = DUT_WRAPPER.rx_data_din;
    assign ram_if.dout = DUT_WRAPPER.tx_data_dout;
    assign ram_if.tx_valid = DUT_WRAPPER.tx_valid;
    assign ram_if.dout_ref = GM_WRAPPER.tx_data_dout;
    assign ram_if.tx_valid_ref = GM_WRAPPER.tx_valid;


//Slave Inputs
    assign slave_if.rst_n = DUT_WRAPPER.rst_n;
    assign slave_if.SS_n = DUT_WRAPPER.SS_n;
    assign slave_if.tx_valid = DUT_WRAPPER.tx_valid;
    assign slave_if.tx_data = DUT_WRAPPER.tx_data_dout;
    assign slave_if.MOSI = DUT_WRAPPER.MOSI;
    assign slave_if.rx_data = DUT_WRAPPER.rx_data_din;
    assign slave_if.rx_data_ref = GM_WRAPPER.rx_data_din;
    assign slave_if.MISO = DUT_WRAPPER.MISO;
    assign slave_if.MISO_ref = GM_WRAPPER.MISO;
    assign slave_if.rx_valid = DUT_WRAPPER.rx_valid;
    assign slave_if.rx_valid_ref = GM_WRAPPER.rx_valid;

endmodule
```

WRAPPER_top.sv

```
//RAM Inputs
    assign ram_if.rst_n = DUT_WRAPPER.rst_n;
    assign ram_if.rx_valid = DUT_WRAPPER.rx_valid;
    assign ram_if.din = DUT_WRAPPER.rx_data_din;
    assign ram_if.dout = DUT_WRAPPER.tx_data_dout;
    assign ram_if.tx_valid = DUT_WRAPPER.tx_valid;
    assign ram_if.dout_ref = GM_WRAPPER.tx_data_dout;
    assign ram_if.tx_valid_ref = GM_WRAPPER.tx_valid;


//Slave Inputs
    assign slave_if.rst_n = DUT_WRAPPER.rst_n;
    assign slave_if.SS_n = DUT_WRAPPER.SS_n;
    assign slave_if.tx_valid = DUT_WRAPPER.tx_valid;
    assign slave_if.tx_data = DUT_WRAPPER.tx_data_dout;
    assign slave_if.MOSI = DUT_WRAPPER.MOSI;
    assign slave_if.rx_data = DUT_WRAPPER.rx_data_din;
    assign slave_if.rx_data_ref = GM_WRAPPER.rx_data_din;
    assign slave_if.MISO = DUT_WRAPPER.MISO;
    assign slave_if.MISO_ref = GM_WRAPPER.MISO;
    assign slave_if.rx_valid = DUT_WRAPPER.rx_valid;
    assign slave_if.rx_valid_ref = GM_WRAPPER.rx_valid;


    bind RAM RAM_sva RAM_sva_inst(
        .clk(ram_if.clk),
        .rst_n(ram_if.rst_n),
        .rx_valid(ram_if.rx_valid),
        .din(ram_if.din),
        .tx_valid(ram_if.tx_valid),
        .dout(ram_if.dout)
    );

    bind WRAPPER WRAPPER_sva WRAPPER_sva_inst(
        .clk(wrapper_if.clk),
        .rst_n(wrapper_if.rst_n),
        .MOSI(wrapper_if.MOSI),
        .SS_n(wrapper_if.SS_n),
        .MISO(wrapper_if.MISO),
        .rx_data_din(WRAPPER.rx_data_din),
        .rx_valid(WRAPPER.rx_valid),
        .tx_valid(WRAPPER.tx_valid),
        .operation(WRAPPER.SLAVE_instance.cs)
    );

    initial begin
        uvm_config_db #(virtual WRAPPER_IF)::set(null,"uvm_test_top","WRAPPER_IF", wrapper_if);
        uvm_config_db #(virtual SPI_slave_if)::set(null,"uvm_test_top","SLAVE_IF", slave_if);
        uvm_config_db #(virtual RAM_IF)::set(null,"uvm_test_top","RAM_IF", ram_if);
        run_test("WRAPPER_test");
    end

endmodule
```

Ln 71, Col 32 Spaces: 4 UTF-8 CRLF {} System Verilog

Test

WRAPPER_test.sv

```
package WRAPPER_test_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import shared_pkg::*;
    import WRAPPER_config_pkg::*;
    import WRAPPER_sequence_pkg::*;
    import WRAPPER_seq_item_pkg::*;
    import WRAPPER_env_pkg::*;

    import SPI_slave_env_pkg::*;
    import SPI_slave_config_pkg::*;

    import RAM_env_pkg::*;
    import RAM_config_pkg::*;

    class WRAPPER_test extends uvm_test;
        `uvm_component_utils(WRAPPER_test)

        //ENV
        WRAPPER_env wrapper_env;
        RAM_env ram_env;
        SPI_slave_env slave_env;

        //Config Obj
        WRAPPER_config wrapper_cfg;
        RAM_config ram_cfg;
        SPI_slave_config slave_cfg;

        //Sequences
        WRAPPER_reset_seq wrapper_reset;
        WRAPPER_write_only_seq wrapper_write_only;
        WRAPPER_read_only_seq wrapper_read_only;
        WRAPPER_write_read_seq wrapper_write_read;

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


        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            //ENV Creation
            wrapper_env = WRAPPER_env::type_id::create("wrapper_env",this);
            slave_env = SPI_slave_env::type_id::create("slave_env",this);
            ram_env = RAM_env::type_id::create("ram_env",this);

            //Config Obj Creation
            wrapper_cfg = WRAPPER_config::type_id::create("wrapper_cfg", this);
            slave_cfg = SPI_slave_config::type_id::create("slave_cfg", this);
            ram_cfg = RAM_config::type_id::create("ram_cfg", this);

            //Sequences Creation
            wrapper_reset = WRAPPER_reset_seq::type_id::create("wrapper_reset");
            wrapper_write_only = WRAPPER_write_only_seq::type_id::create("wrapper_write_only");
            wrapper_read_only = WRAPPER_read_only_seq::type_id::create("wrapper_read_only");
            wrapper_write_read = WRAPPER_write_read_seq::type_id::create("wrapper_write_read");

            wrapper_cfg.is_active = UVM_ACTIVE;
            if (!uvm_config_db #(virtual WRAPPER_IF)::get(this,"","WRAPPER_IF",wrapper_cfg.wrapper_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the WRAPPER interface");
            uvm_config_db #(WRAPPER_config)::set(this, "*", "CFG_WRAPPER", wrapper_cfg);

            slave_cfg.is_active = UVM_PASSIVE;
            if (!uvm_config_db #(virtual SPI_slave_if)::get(this,"","SLAVE_IF",slave_cfg.slave_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the SLAVE interface");
            uvm_config_db #(SPI_slave_config)::set(this, "*", "CFG_SLAVE", slave_cfg);
```

WRAPPER_test.sv

```
package WRAPPER_test_pkg;
    class WRAPPER_test extends uvm_test;
        function void build_phase(uvm_phase phase);
            //Config Obj Creation
            wrapper_cfg = WRAPPER_config::type_id::create("wrapper_cfg", this);
            slave_cfg = SPI_slave_config::type_id::create("slave_cfg", this);
            ram_cfg = RAM_config::type_id::create("ram_cfg", this);

            //Sequences Creation
            wrapper_reset = WRAPPER_reset_seq::type_id::create("wrapper_reset");
            wrapper_write_only = WRAPPER_write_only_seq::type_id::create("wrapper_write_only");
            wrapper_read_only = WRAPPER_read_only_seq::type_id::create("wrapper_read_only");
            wrapper_write_read = WRAPPER_write_read_seq::type_id::create("wrapper_write_read");

            wrapper_cfg.is_active = UVM_ACTIVE;
            if (!uvm_config_db #(virtual WRAPPER_IF)::get(this,"","WRAPPER_IF",wrapper_cfg.wrapper_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the WRAPPER interface");
            uvm_config_db #(WRAPPER_config)::set(this, "*", "CFG_WRAPPER", wrapper_cfg);

            slave_cfg.is_active = UVM_PASSIVE;
            if (!uvm_config_db #(virtual SPI_slave_if)::get(this,"","SLAVE_IF",slave_cfg.slave_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the SLAVE interface");
            uvm_config_db #(SPI_slave_config)::set(this, "*", "CFG_SLAVE", slave_cfg);

            ram_cfg.is_active = UVM_PASSIVE;
            if (!uvm_config_db #(virtual RAM_IF)::get(this,"","RAM_IF",ram_cfg.ram_if))
                `uvm_fatal("build_phase", "Error - Test cannot retrieve the RAM interface");
            uvm_config_db #(RAM_config)::set(this, "*", "CFG_RAM", ram_cfg);

        endfunction

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

            `uvm_info("run_phase", "Reset Asserted", UVM_LOW);
            wrapper_reset.start(wrapper_env.wrapper_agt.wrapper_sqr);
            `uvm_info("run_phase", "Reset Deasserted", UVM_LOW);

            `uvm_info("run_phase", "Write Only Seq Started", UVM_LOW);
            wrapper_write_only.start(wrapper_env.wrapper_agt.wrapper_sqr);
            `uvm_info("run_phase", "Write Only Seq Ended", UVM_LOW);

            `uvm_info("run_phase", "Reset Asserted", UVM_LOW);
            wrapper_reset.start(wrapper_env.wrapper_agt.wrapper_sqr);
            `uvm_info("run_phase", "Reset Deasserted", UVM_LOW);

            `uvm_info("run_phase", "Read Only Seq Started", UVM_LOW);
            is_readOnly = 1;
            wrapper_read_only.start(wrapper_env.wrapper_agt.wrapper_sqr);
            is_readOnly = 0;
            `uvm_info("run_phase", "Read Only Seq Ended", UVM_LOW);

            `uvm_info("run_phase", "Reset Asserted", UVM_LOW);
            wrapper_reset.start(wrapper_env.wrapper_agt.wrapper_sqr);
            `uvm_info("run_phase", "Reset Deasserted", UVM_LOW);

            `uvm_info("run_phase", "Write Read Seq Started", UVM_LOW);
            wrapper_write_read.start(wrapper_env.wrapper_agt.wrapper_sqr);
            `uvm_info("run_phase", "Write Read Seq Ended", UVM_LOW);

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

Environment

WRAPPER_env.sv

```
package WRAPPER_env_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import WRAPPER_agent_pkg::*;
    import WRAPPER_scoreboard_pkg::*;
    import WRAPPER_coverage_pkg::*;
    class WRAPPER_env extends uvm_env;
        `uvm_component_utils(WRAPPER_env)

        WRAPPER_agent wrapper_agt;
        WRAPPER_scoreboard wrapper_sb;
        WRAPPER_coverage wrapper_cvg;

        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);
            wrapper_agt = WRAPPER_agent::type_id::create("wrapper_agt", this);
            wrapper_sb= WRAPPER_scoreboard::type_id::create("wrapper_sb", this);
            wrapper_cvg = WRAPPER_coverage::type_id::create("wrapper_cvg", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            wrapper_agt.agt_ap.connect(wrapper_sb.sb_axp);
            wrapper_agt.agt_ap.connect(wrapper_cvg.cvg_axp);
        endfunction
    endclass
endpackage
```

Coverage

WRAPPER_coverage.sv

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

    import WRAPPER_seq_item_pkg::*;
    import shared_pkg::*;
    class WRAPPER_coverage extends uvm_component;
        `uvm_component_utils(WRAPPER_coverage)

        uvm_analysis_export #(WRAPPER_seq_item) cvg_axp;
        uvm_tlm_analysis_fifo #(WRAPPER_seq_item) cvg_fifo;

        WRAPPER_seq_item cvg_seq_item;

        //Cover Groups
        covergroup WRAPPER_CVG;
            SS_n_cp: coverpoint cvg_seq_item.SS_n {
                bins normal_trans = {1 => 0[*13] => 1};
                bins extended_trans = {1 => 0[*23] => 1};
            }

            MOSI_cp: coverpoint cvg_seq_item.MOSI{
                bins write_address = {0 => 0 => 0};
                bins write_data    = {0 => 0 => 1};
                bins read_address  = {1 => 1 => 0};
                bins read_data     = {1 => 1 => 1};
            }

            SS_n_MOSI_cr: cross SS_n_cp, MOSI_cp{
                ignore_bins not_r_data_ext1 = binsof(MOSI_cp.write_address) && binsof(SS_n_cp.extended_trans);
                ignore_bins not_r_data_ext2 = binsof(MOSI_cp.write_data) && binsof(SS_n_cp.extended_trans);
                ignore_bins not_r_data_ext3 = binsof(MOSI_cp.read_address) && binsof(SS_n_cp.extended_trans);
                ignore_bins not_r_data_ext4 = binsof(MOSI_cp.read_data) && binsof(SS_n_cp.normal_trans);
            }
        endgroup
        function new (string name = "WRAPPER_coverage", uvm_component parent = null);
            super.new(name, parent);
            WRAPPER_CVG = new;
        endfunction

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

            cvg_axp = new ("cvg_axp", this);
            cvg_fifo = new ("cvg_fifo", this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cvg_axp.connect(cvg_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cvg_fifo.get(cvg_seq_item);
                WRAPPER_CVG.sample();
            end
        endtask


    endclass
endpackage
```

Ln 17, Col 13 Spaces: 4 UTF-8 CRLF System Verilog

Scoreboard

WRAPPER_scoreboard.sv

```
package WRAPPER_scoreboard_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import WRAPPER_seq_item_pkg::*;
    class WRAPPER_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(WRAPPER_scoreboard)


        uvm_analysis_export #(WRAPPER_seq_item) sb_axp;
        uvm_tlm_analysis_fifo #(WRAPPER_seq_item) sb_fifo;

        WRAPPER_seq_item sb_seq_item;

        int error_cnt = 0, correct_cnt = 0;

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

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

        endfunction

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(sb_seq_item);
                if (
                    sb_seq_item.MISO != sb_seq_item.MISO_ref
                ) begin
                    error_cnt++;
                    `uvm_error("run_phase", $sformatf("Comparison Failed - Transaction Received DUT : %s, Ref Model : MISO = %b", sb_seq_item.convert2string, sb_seq_item.MISO_ref));
                end
                else begin
                    correct_cnt++;
                    `uvm_info("run_phase", $sformatf("Correct RAM Comparison: %s", sb_seq_item.convert2string),UVM_HIGH);
                end
            end
        endtask

        function void report_phase (uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Correct Tests = %d, Error Tests = %d", correct_cnt, error_cnt),UVM_MEDIUM)
        endfunction
    endclass
endpackage
```

Ln 52, Col 20 Spaces: 4 UTF-8 CRLF System Verilog

Agent

WRAPPER_agent.sv

```
package WRAPPER_agent_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import WRAPPER_sequencer_pkg::*;
    import WRAPPER_monitor_pkg::*;
    import WRAPPER_driver_pkg::*;
    import WRAPPER_seq_item_pkg::*;
    import WRAPPER_config_pkg::*;
    class WRAPPER_agent extends uvm_agent;
        `uvm_component_utils(WRAPPER_agent)

        WRAPPER_config wrapper_cfg;

        WRAPPER_sequencer wrapper_sqr;
        WRAPPER_monitor wrapper_mon;
        WRAPPER_driver wrapper_dvr;

        uvm_analysis_port #(WRAPPER_seq_item) agt_ap;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            if (!uvm_config_db #(WRAPPER_config)::get(this,"","CFG_WRAPPER",wrapper_cfg))
                `uvm_fatal("build_phase", "Error - Wrapper Agent cannot retrieve the config object");

            wrapper_mon = WRAPPER_monitor::type_id::create("wrapper_mon", this);

            if (wrapper_cfg.is_active == UVM_ACTIVE) begin
                wrapper_dvr = WRAPPER_driver::type_id::create("wrapper_dvr", this);
                wrapper_sqr = WRAPPER_sequencer::type_id::create("wrapper_sqr", this);
            end

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

        function void connect_phase (uvm_phase phase);
            super.connect_phase(phase);
            if(wrapper_cfg.is_active == UVM_ACTIVE) begin
                wrapper_dvr.wrapper_dvr_if = wrapper_cfg.wrapper_if;
                wrapper_dvr.seq_item_port.connect(wrapper_sqr.seq_item_export);
            end
            wrapper_mon.wrapper_mon_if = wrapper_cfg.wrapper_if;
            wrapper_mon.mon_ap.connect(agt_ap);
        endfunction
    endclass
endpackage
```


Driver

WRAPPER_driver.sv

```
package WRAPPER_driver_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import WRAPPER_seq_item_pkg::*;
    class WRAPPER_driver extends uvm_driver #(WRAPPER_seq_item);
        `uvm_component_utils(WRAPPER_driver)

        WRAPPER_seq_item dvr_seq_item;
        virtual WRAPPER_IF wrapper_dvr_if;

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

        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                dvr_seq_item = WRAPPER_seq_item::type_id::create("dvr_seq_item");
                seq_item_port.get_next_item(dvr_seq_item);
                //Drive Signals
                wrapper_dvr_if.rst_n = dvr_seq_item.rst_n;
                wrapper_dvr_if.MOSI = dvr_seq_item.MOSI;
                wrapper_dvr_if.SS_n = dvr_seq_item.SS_n;
                repeat(1)@(negedge wrapper_dvr_if.clk);
                seq_item_port.item_done();
                `uvm_info("run_phase", dvr_seq_item.convert2string_stimulus, UVM_HIGH);
            end
        endtask
    endclass
endpackage
```

Monitor

WRAPPER_monitor.sv

```
package WRAPPER_monitor_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import WRAPPER_seq_item_pkg::*;
    class WRAPPER_monitor extends uvm_monitor;
        `uvm_component_utils(WRAPPER_monitor)

        virtual WRAPPER_IF wrapper_mon_if;
        WRAPPER_seq_item mon_seq_item;

        uvm_analysis_port #(WRAPPER_seq_item) mon_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                mon_seq_item = WRAPPER_seq_item::type_id::create("mon_seq_item");
                //Monitor the signals
                repeat(1)@(negedge wrapper_mon_if.clk);
                mon_seq_item.rst_n = wrapper_mon_if.rst_n;
                mon_seq_item.MOSI = wrapper_mon_if.MOSI;
                mon_seq_item.SS_n = wrapper_mon_if.SS_n;
                mon_seq_item.MISO = wrapper_mon_if.MISO;
                mon_seq_item.MISO_ref = wrapper_mon_if.MISO_ref;
                mon_ap.write(mon_seq_item);
                `uvm_info("run_phase", mon_seq_item.convert2string, UVM_HIGH);
            end
        endtask


    endclass
endpackage
```

Sequencer

WRAPPER_sequencer.sv

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

Sequence

WRAPPER_sequence.sv

```
package WRAPPER_sequence_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import shared_pkg::*;
    import WRAPPER_seq_item_pkg::*;
    class WRAPPER_reset_seq extends uvm_sequence #(WRAPPER_seq_item);
        `uvm_object_utils(WRAPPER_reset_seq)

        WRAPPER_seq_item wrapper_seq_item;

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

        virtual task body();
            wrapper_seq_item = WRAPPER_seq_item::type_id::create("wrapper_seq_item");
            start_item(wrapper_seq_item);
            wrapper_seq_item.rst_n = 0;
            wrapper_seq_item.MOSI = 0;
            wrapper_seq_item.SS_n = 0;
            finish_item(wrapper_seq_item);
        endtask
    endclass
    class WRAPPER_write_only_seq extends uvm_sequence #(WRAPPER_seq_item);
        `uvm_object_utils(WRAPPER_write_only_seq)

        WRAPPER_seq_item wrapper_seq_item;

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

        virtual task body();
                wrapper_seq_item = WRAPPER_seq_item::type_id::create("wrapper_seq_item");
                wrapper_seq_item.read_only_const.constraint_mode(0);
                wrapper_seq_item.write_read_const.constraint_mode(0);
            repeat (1000) begin
                start_item(wrapper_seq_item);
                assert(wrapper_seq_item.randomize);
                finish_item(wrapper_seq_item);
            end
        endtask
    endclass
    class WRAPPER_read_only_seq extends uvm_sequence #(WRAPPER_seq_item);
        `uvm_object_utils(WRAPPER_read_only_seq)

        WRAPPER_seq_item wrapper_seq_item;

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

        virtual task body();
                wrapper_seq_item = WRAPPER_seq_item::type_id::create("wrapper_seq_item");
                wrapper_seq_item.write_only_const.constraint_mode(0);
```

WRAPPER_sequence.sv

```
package WRAPPER_sequence_pkg;
    class WRAPPER_write_only_seq extends uvm_sequence #(WRAPPER_seq_item);
        virtual task body();
            end
        endtask
    endclass
    class WRAPPER_read_only_seq extends uvm_sequence #(WRAPPER_seq_item);
        `uvm_object_utils(WRAPPER_read_only_seq)

        WRAPPER_seq_item wrapper_seq_item;

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

        virtual task body();
                wrapper_seq_item = WRAPPER_seq_item::type_id::create("wrapper_seq_item");
                wrapper_seq_item.write_only_const.constraint_mode(0);
                wrapper_seq_item.read_only_const.constraint_mode(1);
                wrapper_seq_item.write_read_const.constraint_mode(0);
            for(int i = 0; i<1000; i++) begin
                start_item(wrapper_seq_item);
                if(i == 1) start_read = 1;
                else start_read = 0;
                assert(wrapper_seq_item.randomize);
                finish_item(wrapper_seq_item);
            end
        endtask
    endclass
    class WRAPPER_write_read_seq extends uvm_sequence #(WRAPPER_seq_item);
        `uvm_object_utils(WRAPPER_write_read_seq)

        WRAPPER_seq_item wrapper_seq_item;

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

        virtual task body();
                wrapper_seq_item = WRAPPER_seq_item::type_id::create("wrapper_seq_item");
                wrapper_seq_item.write_only_const.constraint_mode(0);
                wrapper_seq_item.read_only_const.constraint_mode(0);
                wrapper_seq_item.write_read_const.constraint_mode(1);
            for(int i = 0; i<1000; i++) begin
                start_item(wrapper_seq_item);
                if(i == 1) start_read = 1;
                else start_read = 0;
                assert(wrapper_seq_item.randomize);
                finish_item(wrapper_seq_item);
            end
        endtask
    endclass
endpackage
```

Ln 90, Col 11 Spaces: 4 UTF-8 CRLF {} System Verilog

Sequence Item

WRAPPER_seq_item.sv

```
package WRAPPER_seq_item_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import shared_pkg::*;
    class WRAPPER_seq_item extends uvm_sequence_item;
        `uvm_object_utils(WRAPPER_seq_item)
    // Signals Delarations
        rand bit   MOSI, SS_n,rst_n;;
        bit   MISO;
        bit   MISO_ref;

        rand bit [1:0] state = 2'b00;
        static bit [1:0] state_old = 2'b00;
        int count = 0;
        bit SS_n_HIGH = 1;
        rand bit [10:0] MOSI_array;

        static bit rand_ok, read_add_ok, read_type;

    // Functions
        function new(string name = "WRAPPER_seq_item");
            super.new(name);
        endfunction

        function string convert2string();
            return $sformatf("%s - rst_n = %b , SS_n = %b, MOSI = %b, OUT: MISO = %b", super.convert2string, rst_n, SS_n, MOSI, MISO);
        endfunction
        function string convert2string_stimulus();
            return $sformatf("%s - rst_n = %b , SS_n = %b, MOSI = %b", super.convert2string, rst_n, SS_n, MOSI);
        endfunction
        function void pre_randomize();
            if (count == 0 || !rst_n) begin
                MOSI_array.rand_mode(1);
                state.rand_mode(1);
                rand_ok = 1;
            end
            else begin
                MOSI_array.rand_mode(0);
                state.rand_mode(0);
                rand_ok = 0;
            end
        endfunction
        function void post_randomize();
            begin
                if (!rst_n || SS_n) begin
                    count = 0;
                    SS_n_HIGH = 0;
                end
                else count ++;

                if (count > 12 && count < 23) begin
                    if (state != 2'b11) begin
                        SS_n_HIGH = 1;
                    end
                    else SS_n_HIGH = 0;
```


WRAPPER_seq_item.sv

```
package WRAPPER_seq_item_pkg;
    // Functions
        function void pre_randomize();
            end
        endfunction
        function void post_randomize();
            begin
                if (!rst_n || SS_n) begin
                    count = 0;
                    SS_n_HIGH = 0;
                end
                else count ++;

                if (count > 12 && count < 23) begin
                    if (state != 2'b11) begin
                        SS_n_HIGH = 1;
                    end
                    else SS_n_HIGH = 0;
                end else if (count == 23) begin
                    if (state == 2'b11) begin
                        SS_n_HIGH = 1;
                    end
                    else SS_n_HIGH = 0;
                end
                else SS_n_HIGH = 0;

                state_old = state;

                if(state_old == 2'b01) read_type = 0;
                else if(state_old == 2'b11) read_type = 1;
                if(!rst_n && (state_old[1] == 1)) read_add_ok = 1;
                else read_add_ok = 0;
            end
        endfunction

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

    constraint MOSI_array_const{
        MOSI_array[10] == state[1];
        MOSI_array[9:8] == state;
    };

    constraint MOSI_const{
        if (count > 0 && count < 12) MOSI == MOSI_array [11 - (count)]; // 10
    };
    constraint SS_n_const{
        SS_n == SS_n_HIGH;
    };
    constraint write_only_const{
        state inside {2'b00, 2'b01};
    };
    constraint read_only_const{
```

WRAPPER_seq_item.sv

```
package WRAPPER_seq_item_pkg;
    // Functions
        function void post_randomize();
            begin
                if(state_old == 2'b01) read_type = 0;
                else if(state_old == 2'b11) read_type = 1;
                if(!rst_n && (state_old[1] == 1)) read_add_ok = 1;
                else read_add_ok = 0;
            end
        endfunction

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

    constraint MOSI_array_const{
        MOSI_array[10] == state[1];
        MOSI_array[9:8] == state;
    };

    constraint MOSI_const{
        if (count > 0 && count < 12) MOSI == MOSI_array [11 - (count)]; // 10
    };
    constraint SS_n_const{
        SS_n == SS_n_HIGH;
    };
    constraint write_only_const{
        state inside {2'b00, 2'b01};
    };
    constraint read_only_const{
        if (start_read || read_add_ok ||  (state_old == 2'b11 && rand_ok)) { // Read data
            state == 2'b10;
        }
        else if (state_old == 2'b10 && rand_ok) { //read Addr
            state == 2'b11;
        }
    };
    constraint write_read_const{
        if (start_read || read_add_ok || (state_old == 2'b00 && rand_ok)) {
            state inside {2'b00, 2'b01};
        }
        else if (state_old == 2'b01 && rand_ok) {
            state dist {2'b10:/60, 2'b00:/40};
        }
        else if (state_old == 2'b11 && rand_ok) {
            state dist {2'b00:/60, 2'b10:/40};
        }
        else if (state_old == 2'b10 && rand_ok) {
            state inside {2'b11};
        }
    };
    endclass
endpackage
```

Do File

run.do

```
vlib work
vlog +define+SIM  -f src_files.list +cover -covercells
vsim -voptargs=+acc work.WRAPPER_top -classdebug -uvmcontrol=all -cover
add wave -position insertpoint sim:/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/*
add wave -position insertpoint  \
sim:/WRAPPER_top/DUT_WRAPPER/RAM_instance/din \
sim:/WRAPPER_top/DUT_WRAPPER/RAM_instance/dout \
sim:/WRAPPER_top/DUT_WRAPPER/RAM_instance/MEM  \
sim:/WRAPPER_top/DUT_WRAPPER/RAM_instance/Rd_Addr \
sim:/WRAPPER_top/DUT_WRAPPER/RAM_instance/Wr_Addr
coverage save wrapper.ucdb -onexit
run -all
```

4. COVERAGE

Code Coverage

TOGGLE

```
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         12        12         0   100.00%

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

Toggle Coverage for instance /WRAPPER_top/wrapper_if --

                                    Node      1H->0L      0L->1H                          "Coverage"
                                    ----------------------------------------------------------------
                                    MISO           1           1                             100.00
                                MISO_ref           1           1                             100.00
                                    MOSI           1           1                             100.00
                                    SS_n           1           1                             100.00
                                     clk           1           1                             100.00
                                   rst_n           1           1                             100.00

Total Node Count     =          6
Toggled Node Count   =          6
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (12 of 12 bins)
```

Functional Coverage

WRAPPER_cvr.txt - Notepad

File Edit Format View Help

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------------
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_reset_txValid_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(36)    29 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_reset_dout_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(37)    29 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_addr_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(39)   870 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_data_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(40)   696 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_addr_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(41)   305 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_data_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(43)     9 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_addr_to_wr_data_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(45)    78 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_addr_to_rd_data_cvr
                                        RAM_sva Verilog  SVA  RAM_sva.sv(46)    72 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__received_address_p2
                                        SLAVE  Verilog  SVA  SPI_slave.sv(253)
                                                                                65 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__received_address_p1
                                        SLAVE  Verilog  SVA  SPI_slave.sv(252)
                                                                                73 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__counter_write_p
                                        SLAVE  Verilog  SVA  SPI_slave.sv(251)
                                                                              2172 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__counter_rst_p
                                        SLAVE  Verilog  SVA  SPI_slave.sv(250)
                                                                               195 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p9
                                        SLAVE  Verilog  SVA  SPI_slave.sv(249)
                                                                               730 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p8
                                        SLAVE  Verilog  SVA  SPI_slave.sv(248)
                                                                               427 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p7
                                        SLAVE  Verilog  SVA  SPI_slave.sv(247)
                                                                              1224 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p6
                                        SLAVE  Verilog  SVA  SPI_slave.sv(246)
                                                                                42 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p4
                                        SLAVE  Verilog  SVA  SPI_slave.sv(245)
                                                                                36 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p3
                                        SLAVE  Verilog  SVA  SPI_slave.sv(244)
                                                                               117 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p2
                                        SLAVE  Verilog  SVA  SPI_slave.sv(243)
                                                                               197 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p1
                                        SLAVE  Verilog  SVA  SPI_slave.sv(242)
                                                                               172 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__rx_valid_p
                                        SLAVE  Verilog  SVA  SPI_slave.sv(241)
                                                                               131 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__read_seq_p
                                        SLAVE  Verilog  SVA  SPI_slave.sv(240)
```

Ln 1, Col 1 90% Windows (CRLF) UTF-8

WRAPPER_cvr.txt - Notepad

File Edit Format View Help

```
                                RAM_sva Verilog  SVA  RAM_sva.sv(41)   305 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_data_cvr
                                RAM_sva Verilog  SVA  RAM_sva.sv(43)     9 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_addr_to_wr_data_cvr
                                RAM_sva Verilog  SVA  RAM_sva.sv(45)    78 Covered
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_addr_to_rd_data_cvr
                                RAM_sva Verilog  SVA  RAM_sva.sv(46)    72 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__received_address_p2
                                SLAVE  Verilog  SVA  SPI_slave.sv(253)
                                                                        65 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__received_address_p1
                                SLAVE  Verilog  SVA  SPI_slave.sv(252)
                                                                        73 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__counter_write_p
                                SLAVE  Verilog  SVA  SPI_slave.sv(251)
                                                                      2172 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__counter_rst_p
                                SLAVE  Verilog  SVA  SPI_slave.sv(250)
                                                                       195 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p9
                                SLAVE  Verilog  SVA  SPI_slave.sv(249)
                                                                       730 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p8
                                SLAVE  Verilog  SVA  SPI_slave.sv(248)
                                                                       427 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p7
                                SLAVE  Verilog  SVA  SPI_slave.sv(247)
                                                                      1224 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p6
                                SLAVE  Verilog  SVA  SPI_slave.sv(246)
                                                                        42 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p4
                                SLAVE  Verilog  SVA  SPI_slave.sv(245)
                                                                        36 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p3
                                SLAVE  Verilog  SVA  SPI_slave.sv(244)
                                                                       117 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p2
                                SLAVE  Verilog  SVA  SPI_slave.sv(243)
                                                                       197 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__cs_p1
                                SLAVE  Verilog  SVA  SPI_slave.sv(242)
                                                                       172 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__rx_valid_p
                                SLAVE  Verilog  SVA  SPI_slave.sv(241)
                                                                       131 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__read_seq_p
                                SLAVE  Verilog  SVA  SPI_slave.sv(240)
                                                                        68 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__write_seq_p
                                SLAVE  Verilog  SVA  SPI_slave.sv(239)
                                                                       101 Covered
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/cover__rst_p
                                SLAVE  Verilog  SVA  SPI_slave.sv(238)
                                                                        29 Covered
/WRAPPER_top/DUT_WRAPPER/WRAPPER_sva_inst/cover__MISO_wrapper_p
                                WRAPPER_sva Verilog  SVA  WRAPPER_sva.sv(21)
                                                                      2183 Covered
/WRAPPER_top/DUT_WRAPPER/WRAPPER_sva_inst/cover__rst_wrapper_p
                                WRAPPER_sva Verilog  SVA  WRAPPER_sva.sv(20)
                                                                        29 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 26
```

Ln 1, Col 1 | 90% | Windows (CRLF) | UTF-8

Assertion Coverage

WRAPPER_cvr.txt - Notepad

File Edit Format View Help

```
ASSERTION RESULTS:
-------------------------------------------------------------------
Name                File(Line)                   Failure      Pass
                                                 Count        Count
-------------------------------------------------------------------
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_reset_txValid
                    RAM_sva.sv(23)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_reset_dout
                    RAM_sva.sv(24)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_addr
                    RAM_sva.sv(26)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_data
                    RAM_sva.sv(27)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_addr
                    RAM_sva.sv(28)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_data
                    RAM_sva.sv(30)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_wr_addr_to_wr_data
                    RAM_sva.sv(32)               0            1
/WRAPPER_top/DUT_WRAPPER/RAM_instance/RAM_sva_inst/a_rd_addr_to_rd_data
                    RAM_sva.sv(33)               0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__received_address_p2
                    SPI_slave.sv(236)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__received_address_p1
                    SPI_slave.sv(235)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__counter_write_p
                    SPI_slave.sv(234)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__counter_rst_p
                    SPI_slave.sv(233)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p9
                    SPI_slave.sv(232)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p8
                    SPI_slave.sv(231)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p7
                    SPI_slave.sv(230)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p6
                    SPI_slave.sv(229)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p4
                    SPI_slave.sv(228)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p3
                    SPI_slave.sv(227)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p2
                    SPI_slave.sv(226)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__cs_p1
                    SPI_slave.sv(225)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__rx_valid_p
                    SPI_slave.sv(224)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__read_seq_p
                    SPI_slave.sv(223)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__write_seq_p
                    SPI_slave.sv(222)            0            1
/WRAPPER_top/DUT_WRAPPER/SLAVE_instance/assert__rst_p
                    SPI_slave.sv(221)            0            1
/WRAPPER_top/DUT_WRAPPER/WRAPPER_sva_inst/assert__MISO_wrapper_p
                    WRAPPER_sva.sv(18)           0            1
/WRAPPER_top/DUT_WRAPPER/WRAPPER_sva_inst/assert__rst_wrapper_p
                    WRAPPER_sva.sv(17)           0            1
/WRAPPER_sequence_pkg/WRAPPER_write_only_seq/body/#ublk#73704343#37/immed__39
                    WRAPPER_sequence.sv(39)      0            1
/WRAPPER_sequence_pkg/WRAPPER_read_only_seq/body/#anonblk#73704343#58#4#/#ublk#73704343#58/immed__62
                    WRAPPER_sequence.sv(62)      0            1
/WRAPPER_sequence_pkg/WRAPPER_write_read_seq/body/#anonblk#73704343#81#4#/#ublk#73704343#81/immed__85
```

Ln 1, Col 1 | 90% | Windows (CRLF) | UTF-8

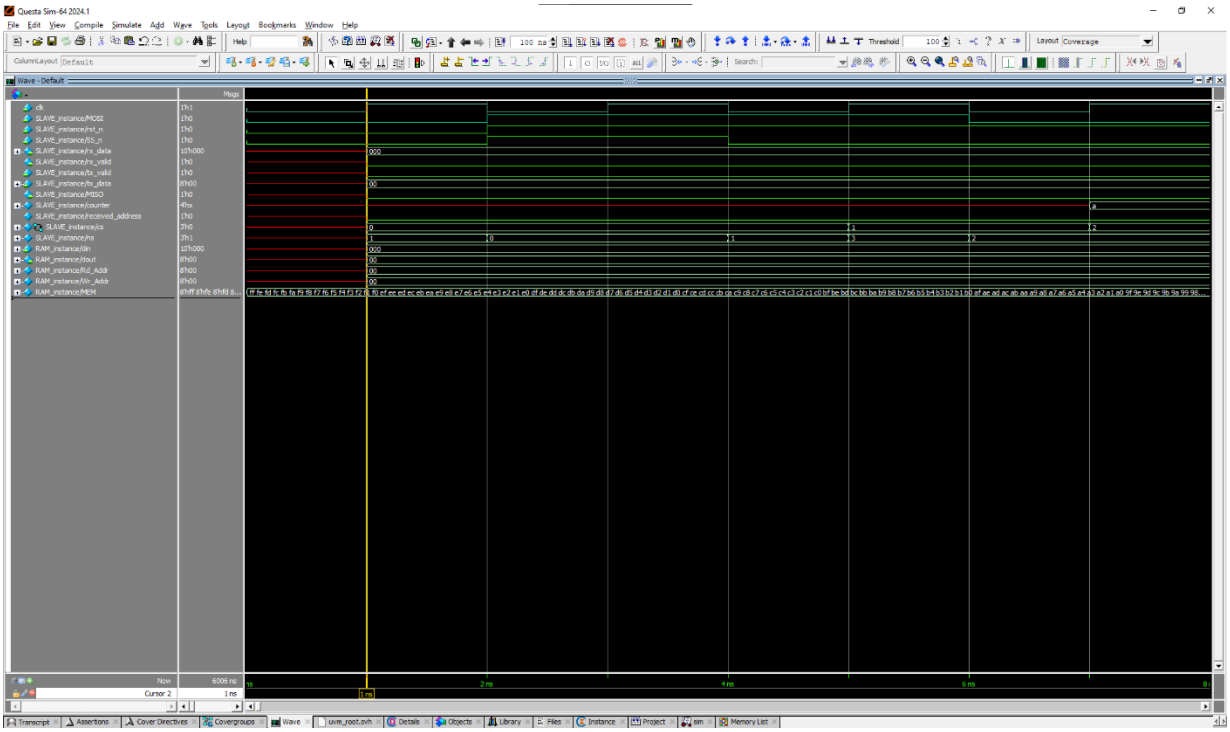
```
                    WRAPPER_sequence.sv(62)      0            1
/WRAPPER_sequence_pkg/WRAPPER_write_read_seq/body/#anonblk#73704343#81#4#/#ublk#73704343#81/immed__85
                    WRAPPER_sequence.sv(85)      0            1
```

5. ASSERTIONS TABLE

	A	B	C	D	E	F	G
1	**Feature**	**Assertion**					
2	When the reset is asserted, after 1 clk cycle MISO is low, din is low and rx_valid is low	(@(posedge clk) (!rst_n) |=> (!MISO && (rx_data_din == 10'b00_0000_0000) && !rx_valid))					
3	When the reset is inactive, if the internal state of the slave is not READ_DATA, then MISO is not changed	(@(posedge clk) disable iff(!rst_n) (operation != 3'b100) |=> (MISO == $past(MISO)))					

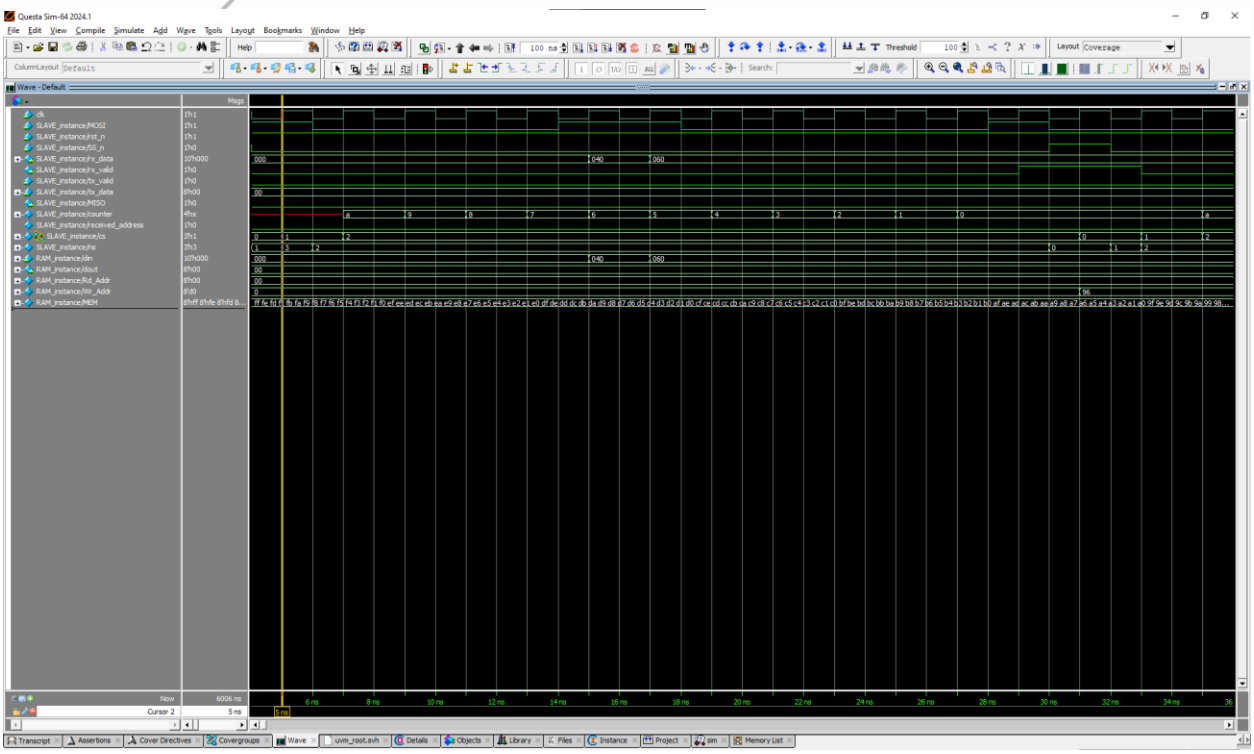
6. SIMULATION SNIPPETS

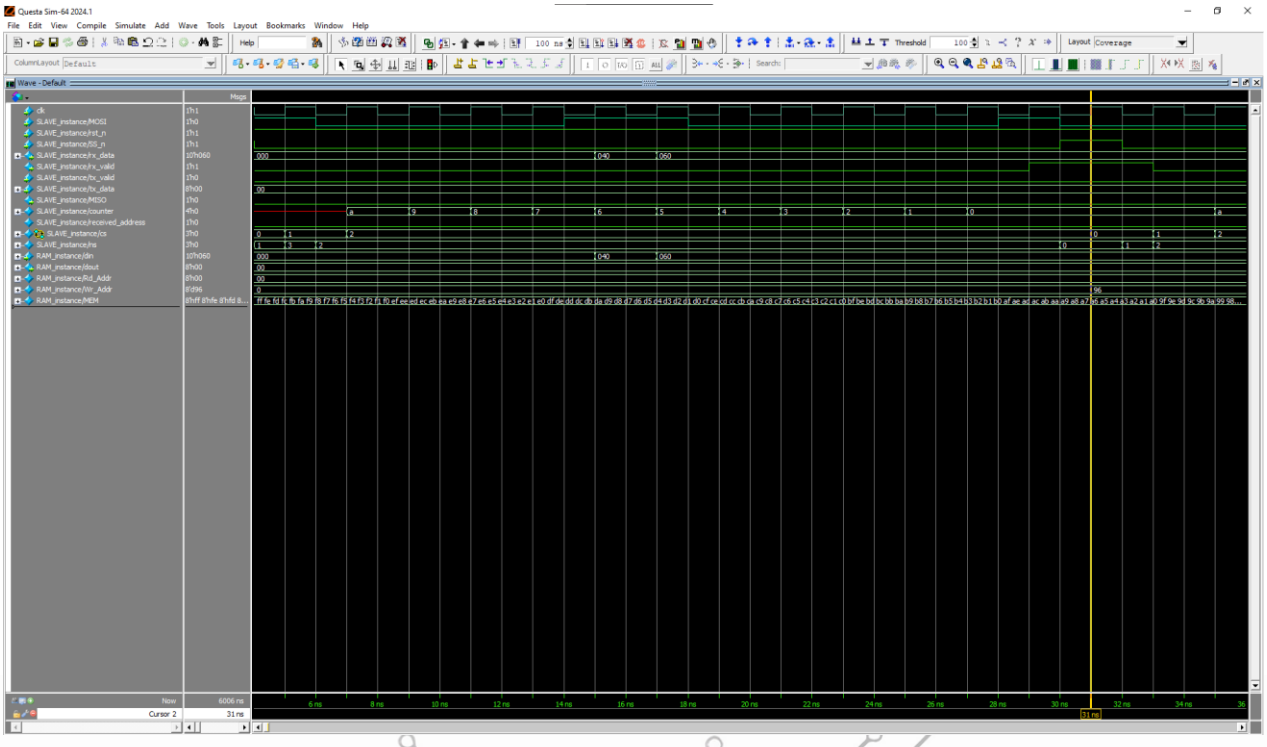
RESET SEQUENCE



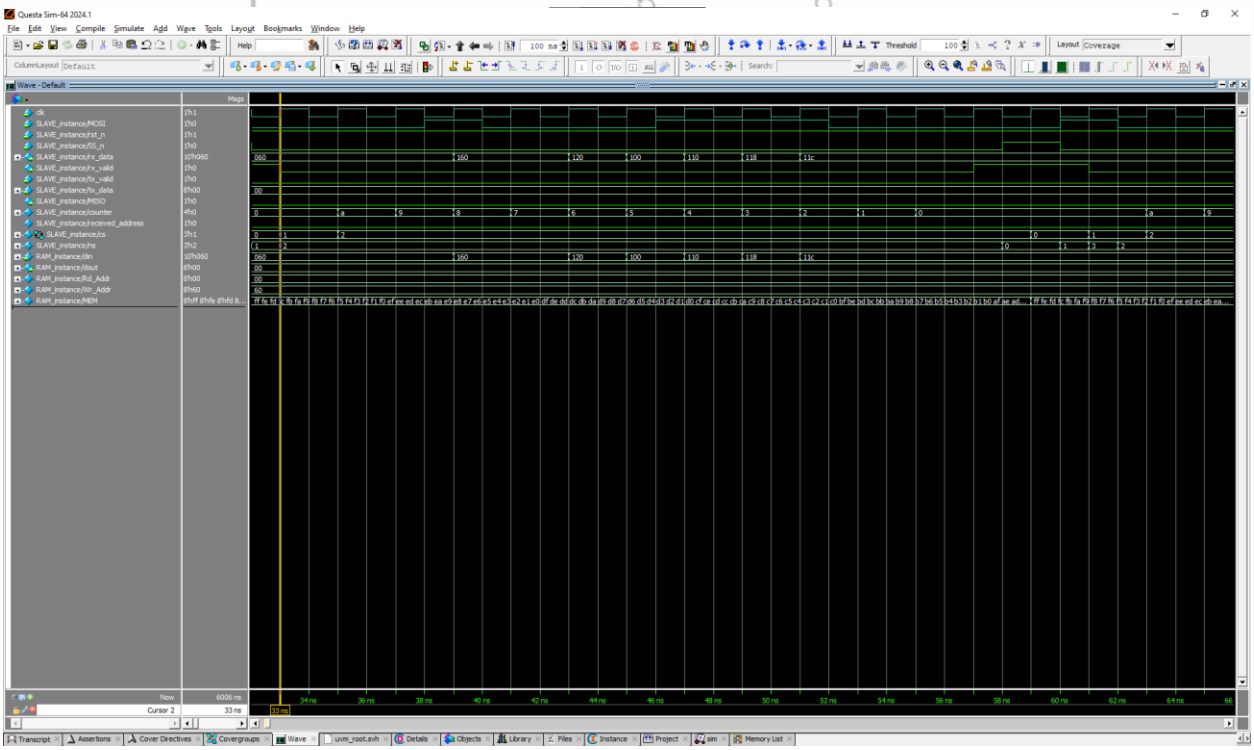
Write Only Sequence

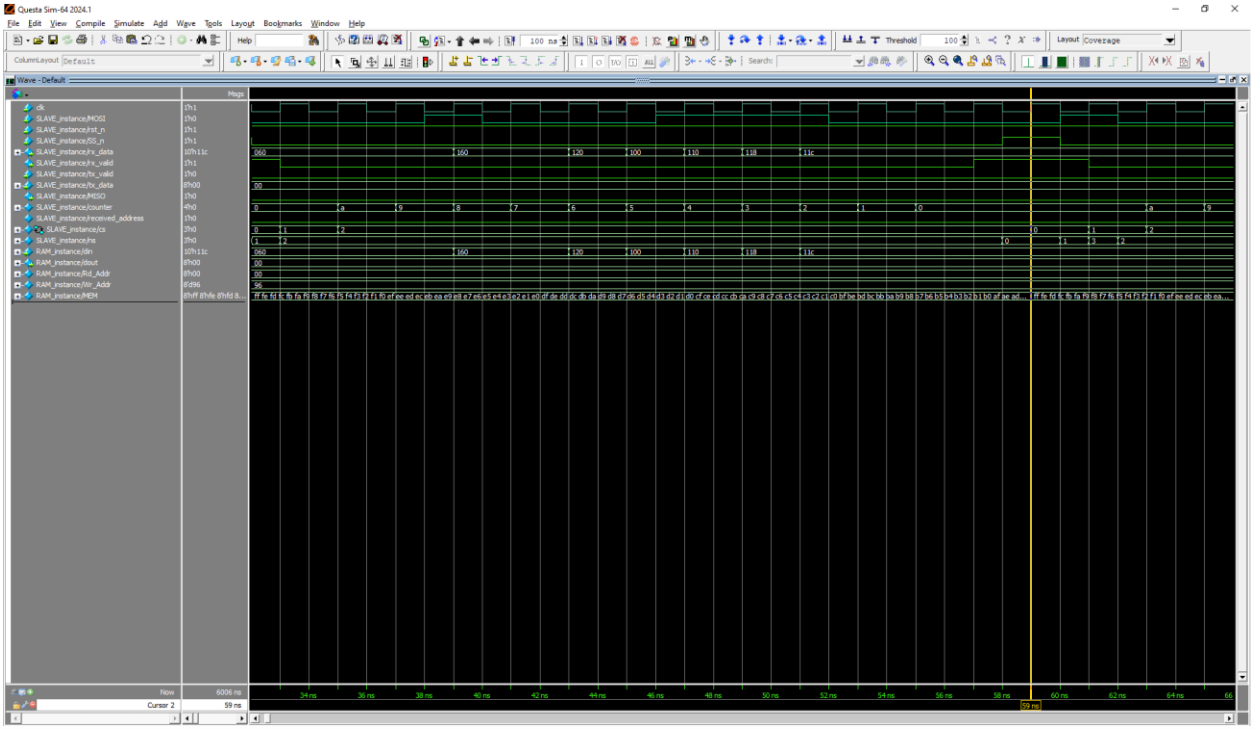
write addr: (from 5 ns to 31 ns, 13 clk cycles)

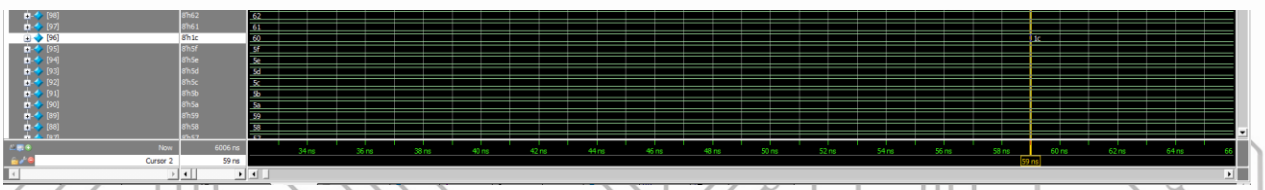




WRITE DATA :(FROM 33NS TO 59NS, 13 CLK CYCLES)

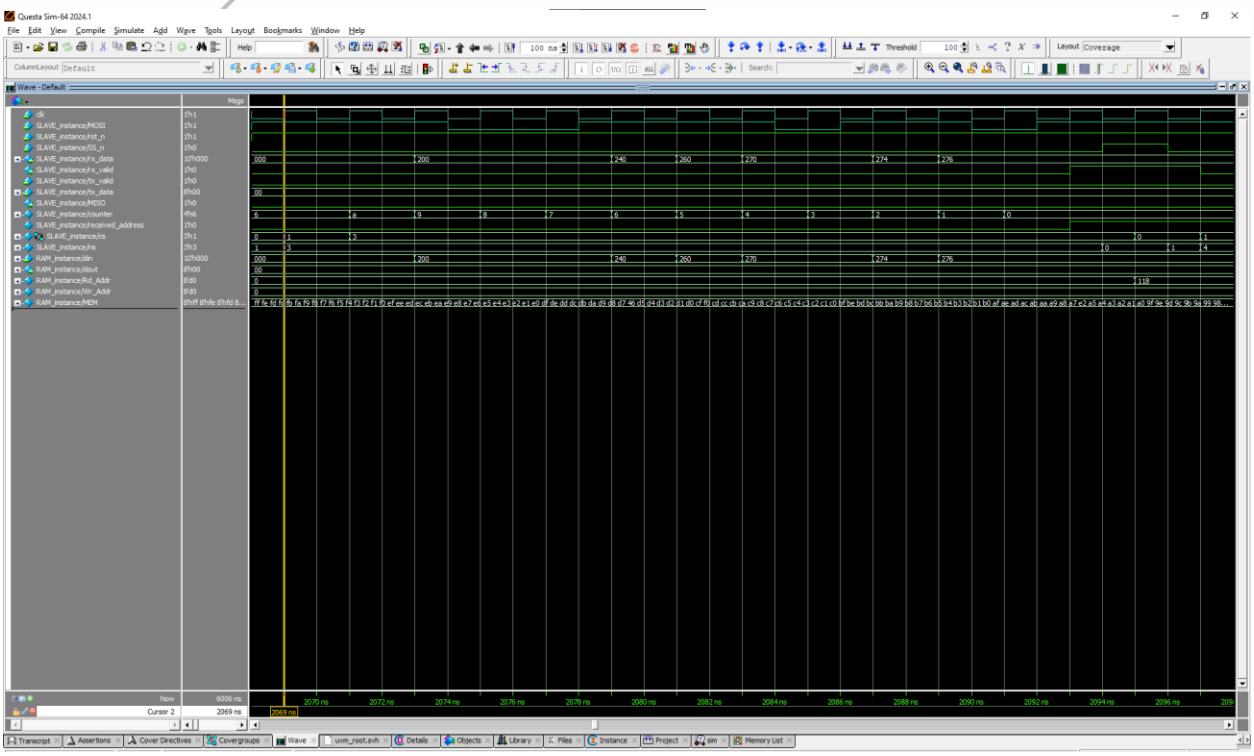


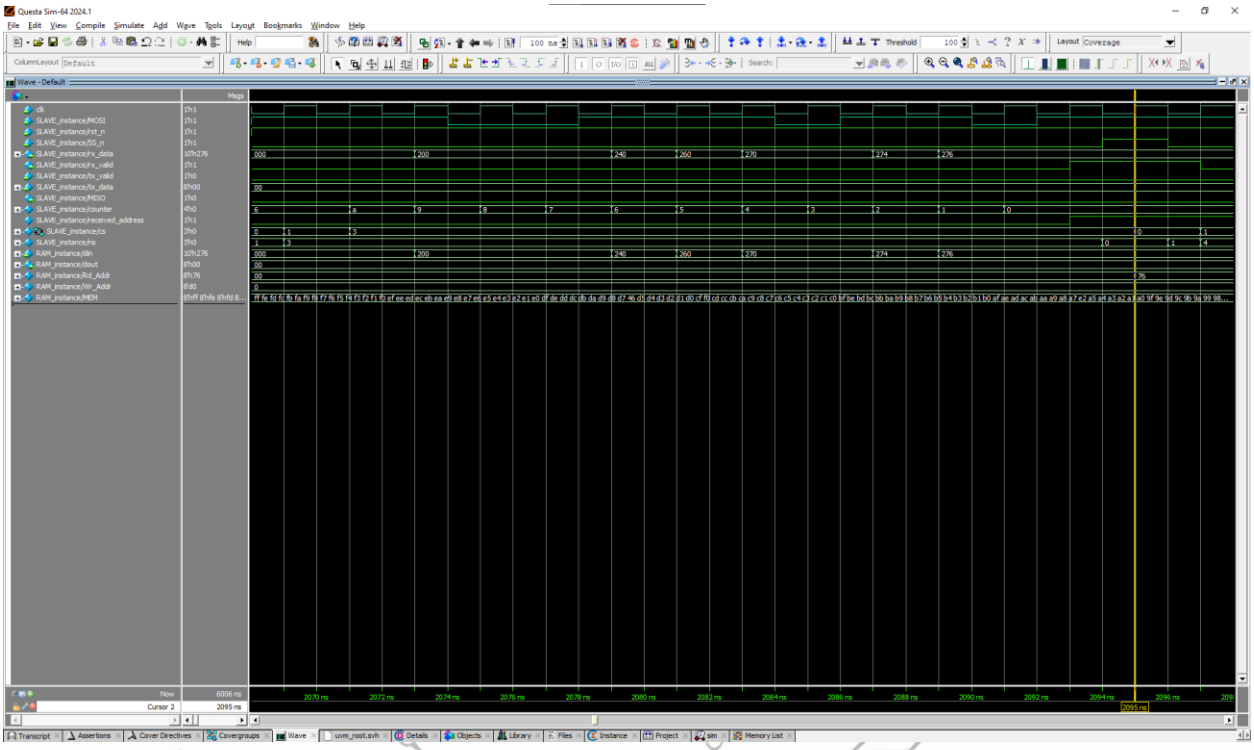




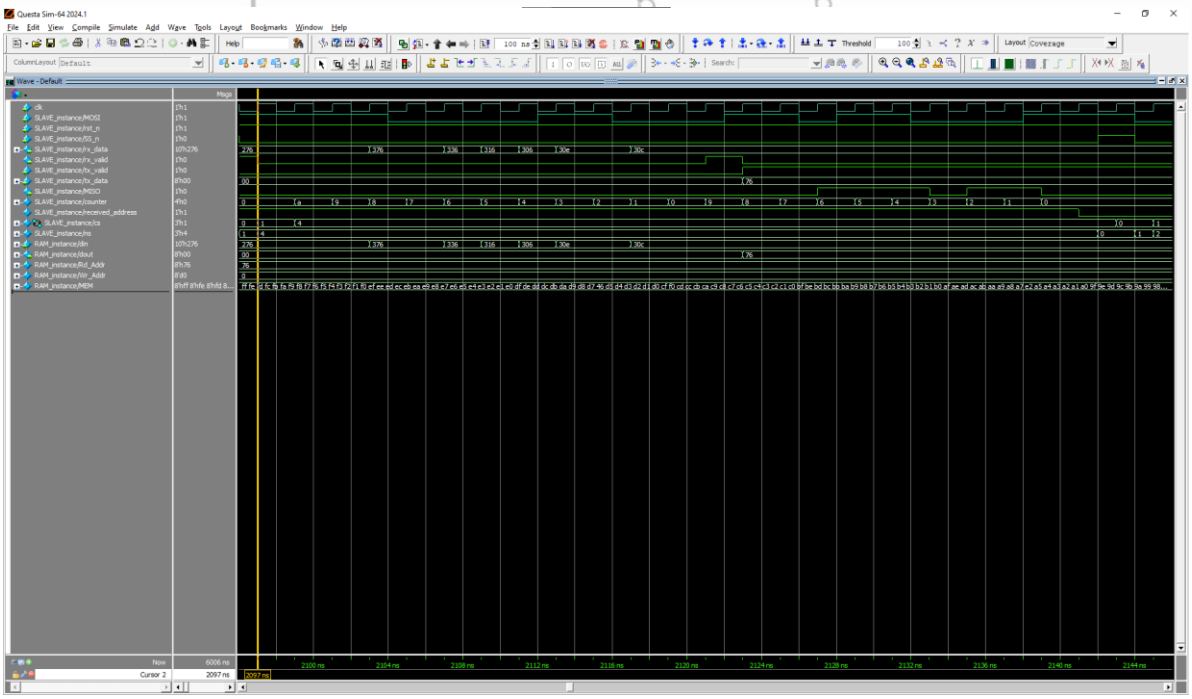
Read Only Sequence

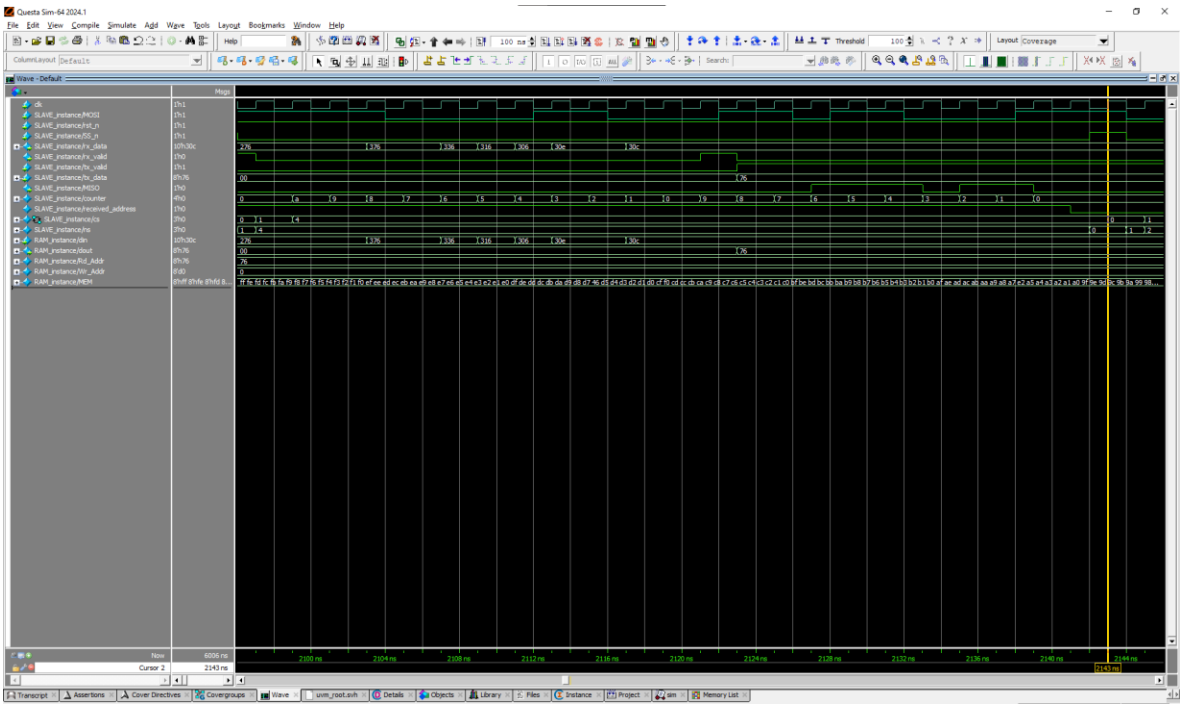
Read Address: (from 2069ns to 2095ns, 13 clk cycles)

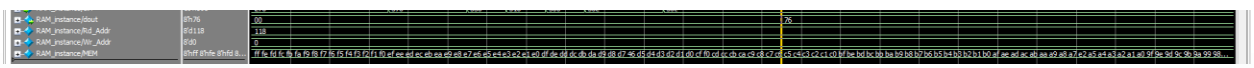


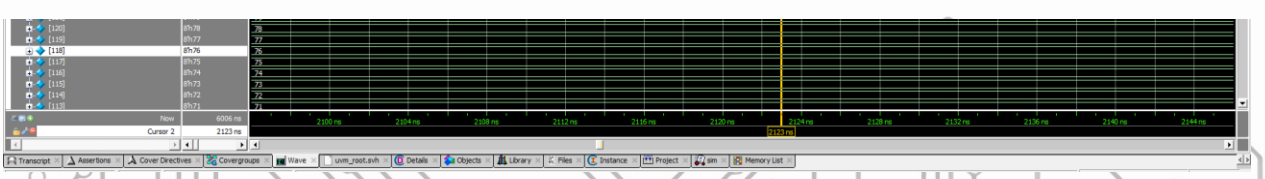


READ DATA: (FROM 2097NS TO 2143NS, 23 CLK CYCLES)



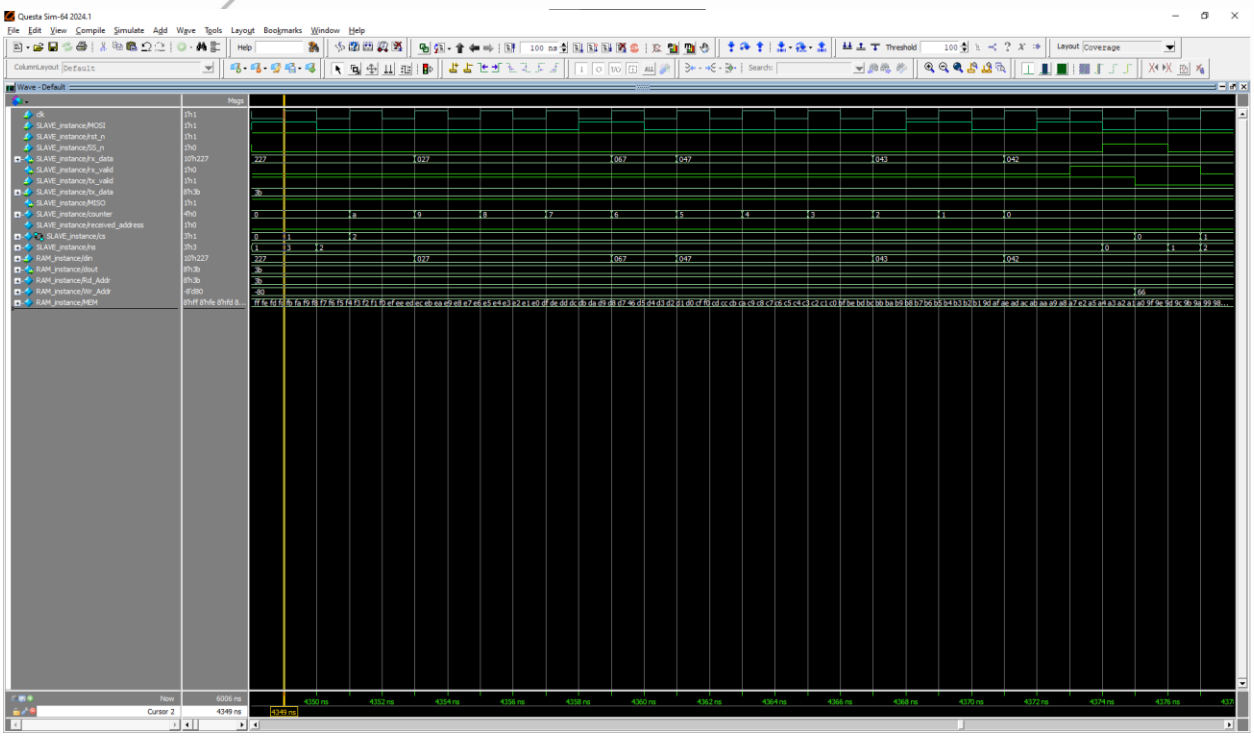


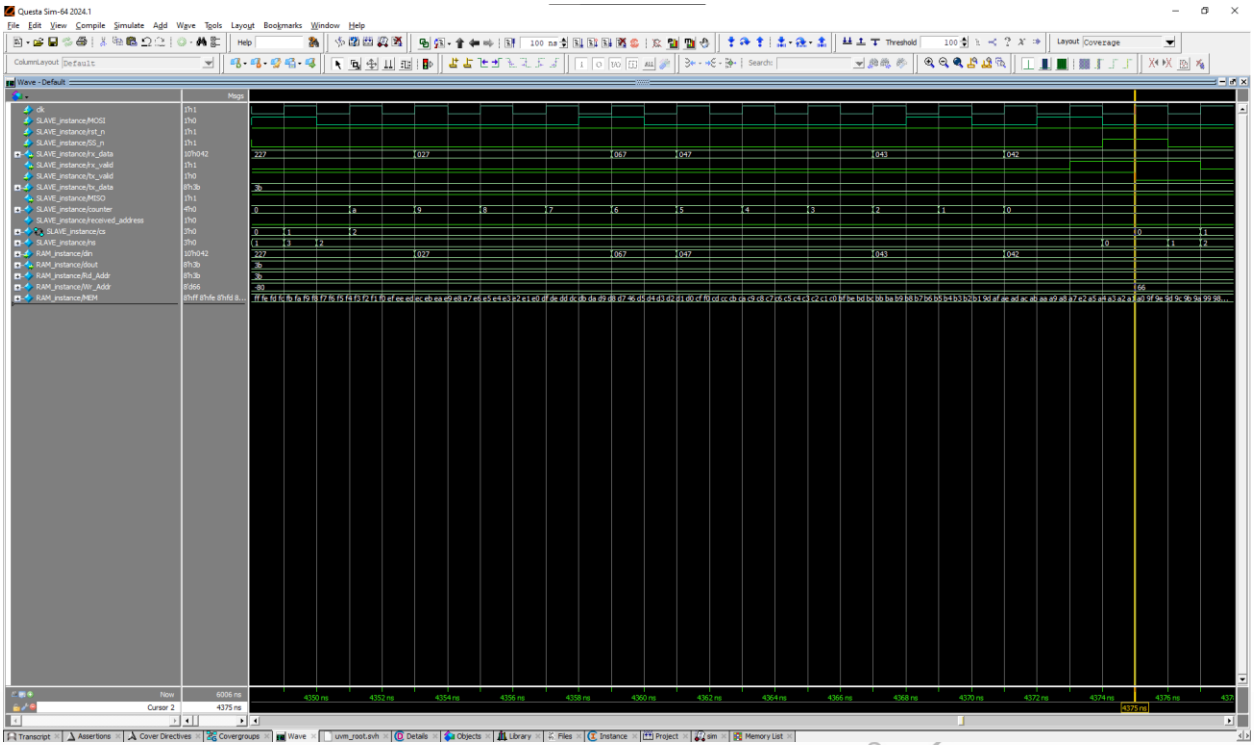




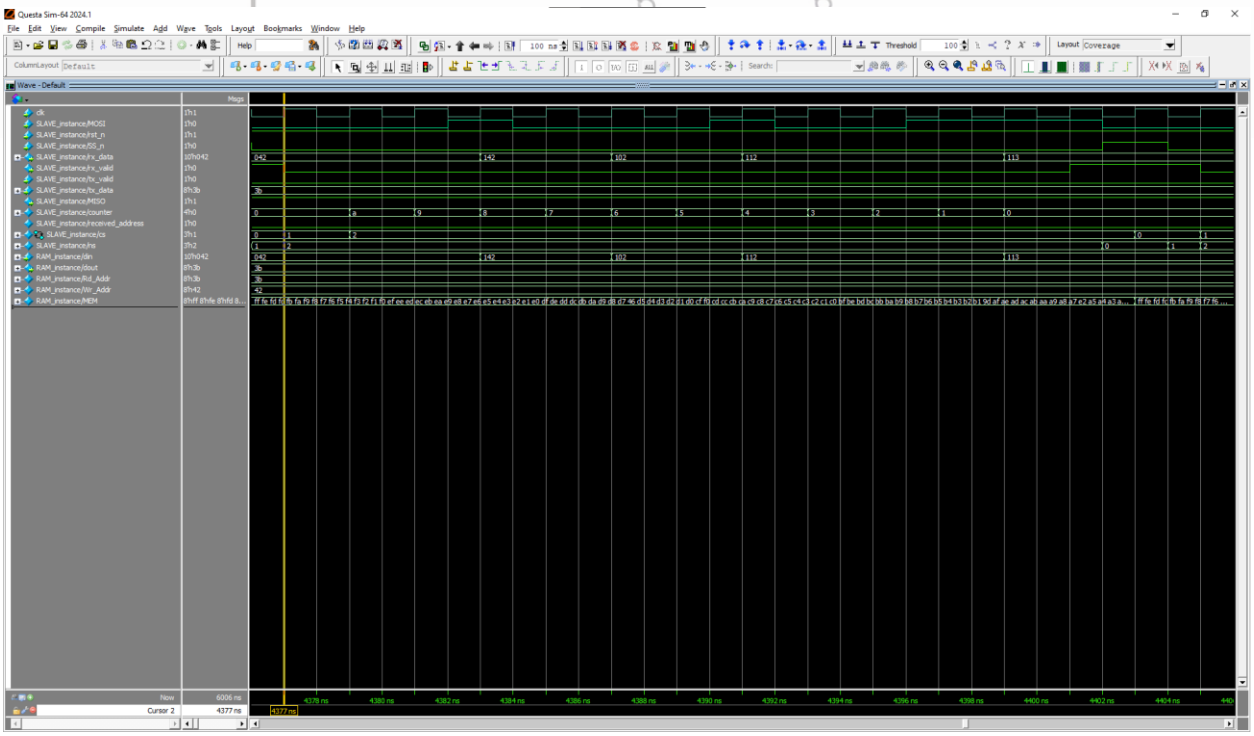
Write Read Sequence

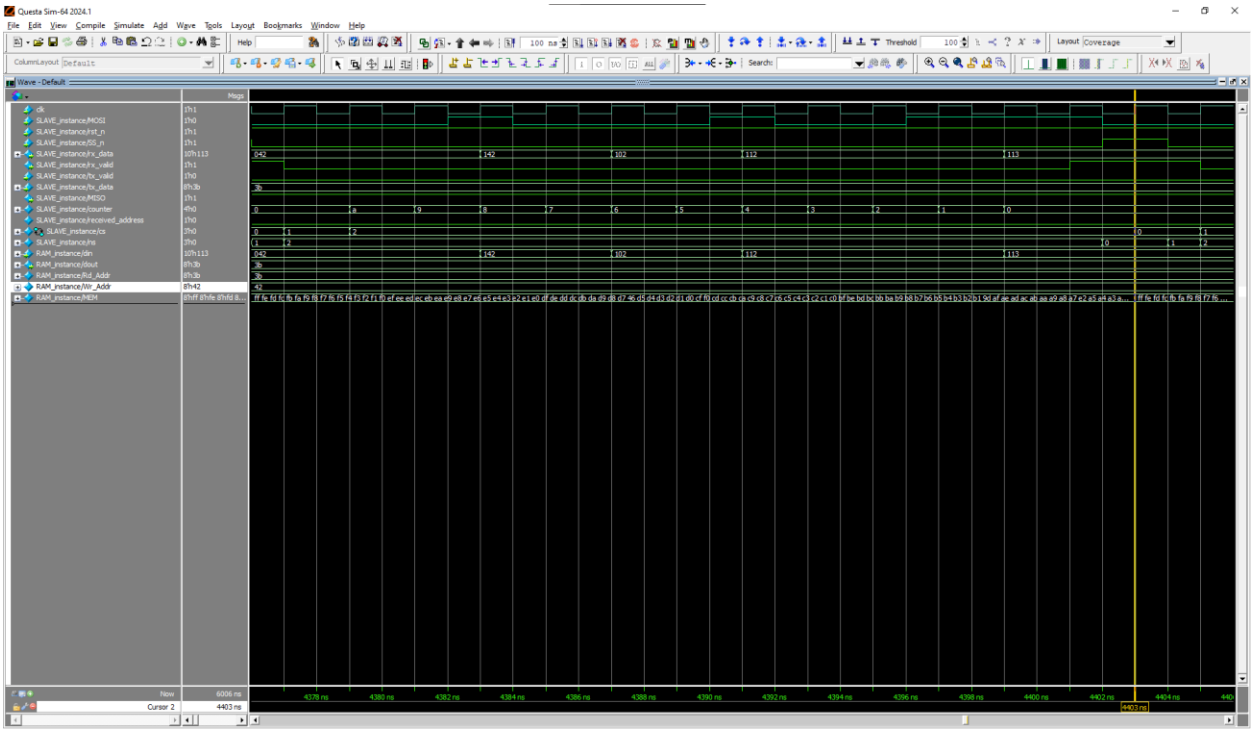
write address: (from 4349ns to 4375ns, 13 clk cycles)

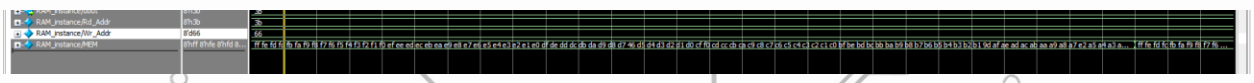


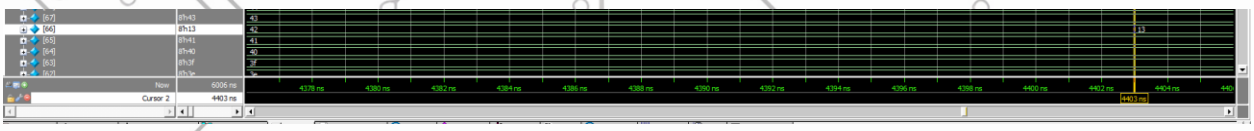


WRITE DATA: (FROM 4377NS TO 4403NS, 13 CLK CYCLES)

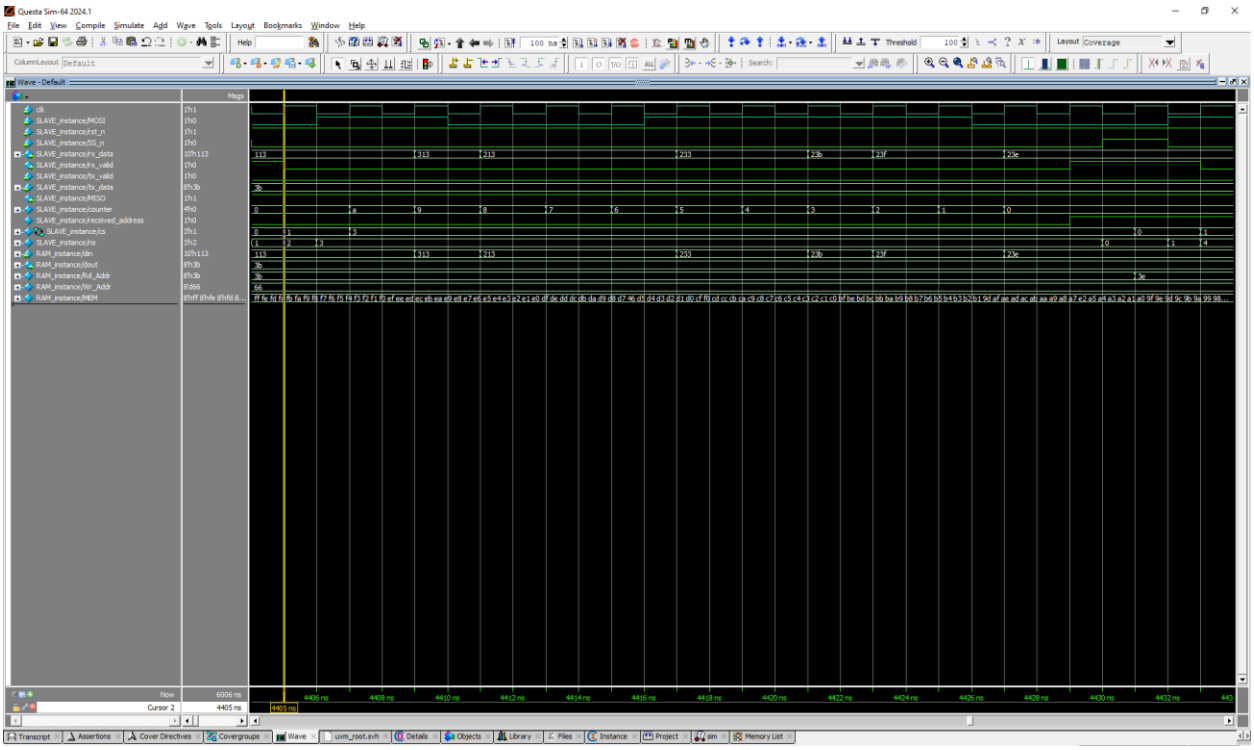


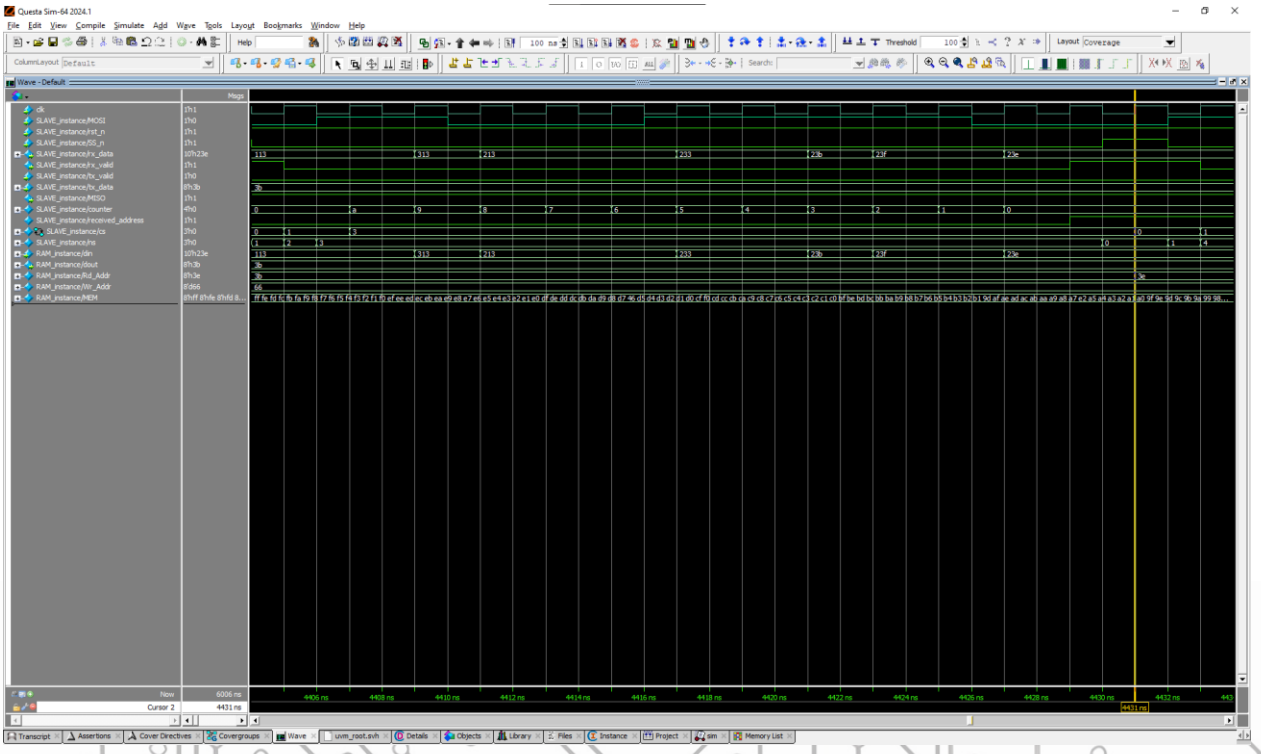




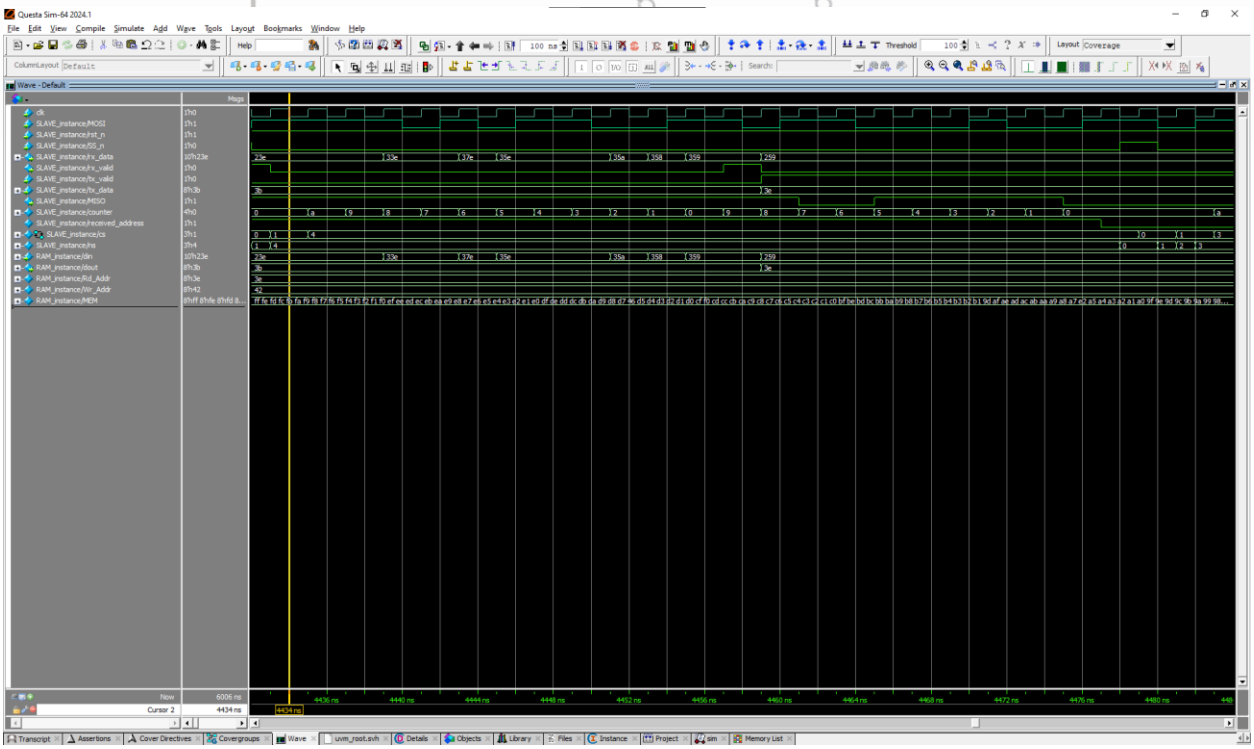


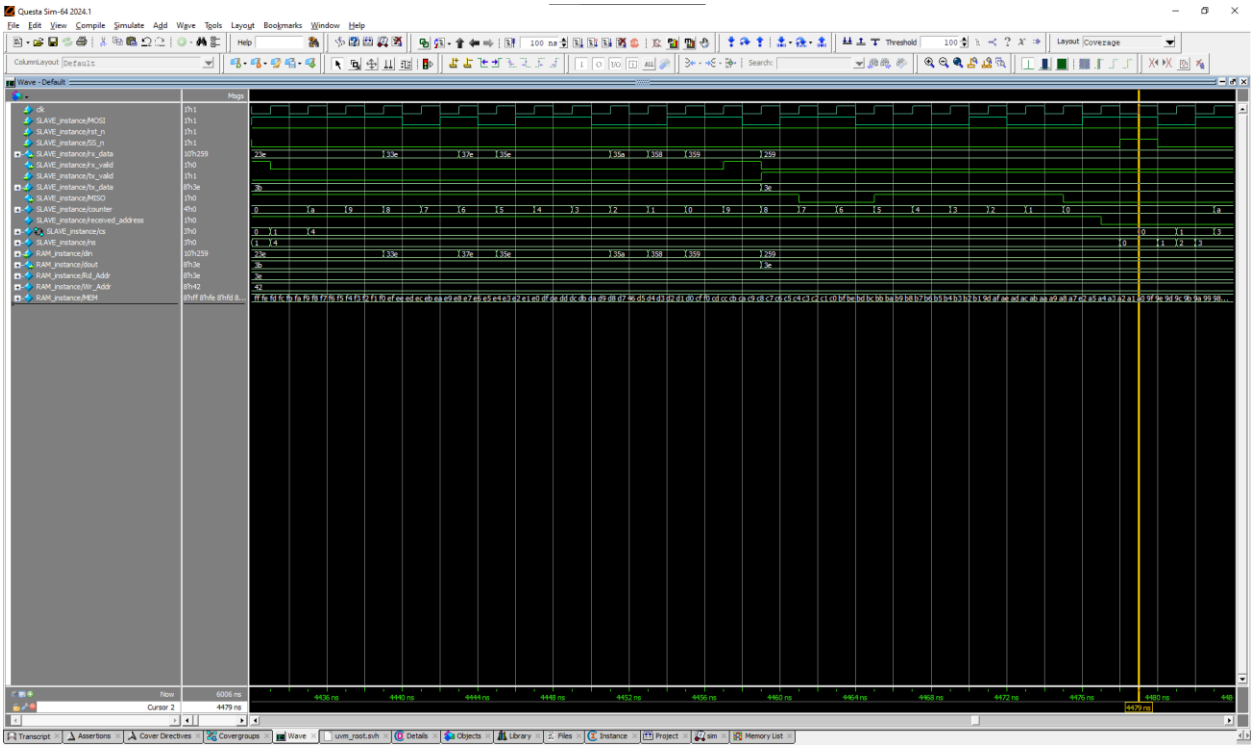
READ ADDRESS: (FROM 4405NS TO 4431NS , 13 CLK CYCLES)

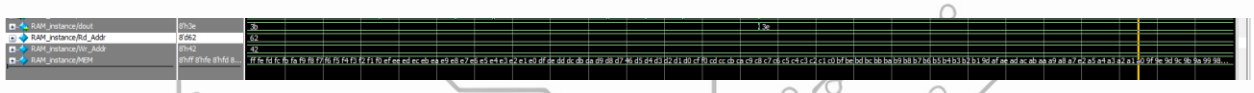


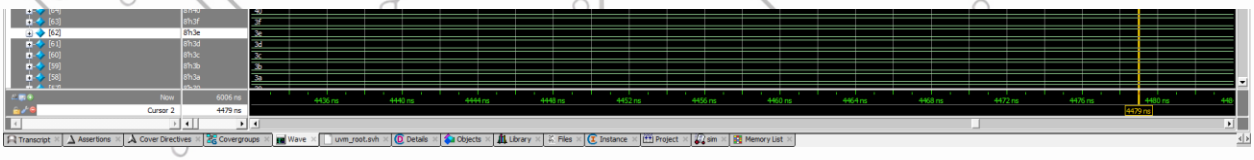


READ DATA: (FROM 4433NS TO 4479NS, 23 CLK CYCLES)









Transcript Snippets

Questa Sim-64 2024.1

```
#
#  ************       IMPORTANT RELEASE NOTES         ************
#
#  You are using a version of the UVM library that has been compiled
#  with `UVM_NO_DEPRECATED undefined.
#  See http://www.eda.org/svdb/view.php?id=3313 for more details.
#
#  You are using a version of the UVM library that has been compiled
#  with `UVM_OBJECT_MUST_HAVE_CONSTRUCTOR undefined.
#  See http://www.eda.org/svdb/view.php?id=3770 for more details.
#
#      (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test WRAPPER_test...
# UVM_INFO WRAPPER_test.sv(79) @ 0: uvm_test_top [run_phase] Reset Asserted
# ****************************************************************
# * Questa UVM Transaction Recording Turned ON.                  *
# * recording_detail has been set.                               *
# *  To turn off, set 'recording_detail' to off:                 *
# *  uvm_config_db#(int)            ::set(null, "", "recording_detail", 0); *
# *  uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0); *
# ****************************************************************
# UVM_INFO WRAPPER_test.sv(81) @ 2: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO WRAPPER_test.sv(83) @ 2: uvm_test_top [run_phase] Write Only Seq Started
# UVM_INFO WRAPPER_test.sv(85) @ 2002: uvm_test_top [run_phase] Write Only Seq Ended
# UVM_INFO WRAPPER_test.sv(87) @ 2002: uvm_test_top [run_phase] Reset Asserted
# UVM_INFO WRAPPER_test.sv(89) @ 2004: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO WRAPPER_test.sv(91) @ 2004: uvm_test_top [run_phase] Read Only Seq Started
# UVM_INFO WRAPPER_test.sv(95) @ 4004: uvm_test_top [run_phase] Read Only Seq Ended
# UVM_INFO WRAPPER_test.sv(97) @ 4004: uvm_test_top [run_phase] Reset Asserted
# UVM_INFO WRAPPER_test.sv(99) @ 4006: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO WRAPPER_test.sv(101) @ 4006: uvm_test_top [run_phase] Write Read Seq Started
# UVM_INFO WRAPPER_test.sv(103) @ 6006: uvm_test_top [run_phase] Write Read Seq Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 6006: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO RAM_scoreboard.sv(60) @ 6006: uvm_test_top.ram_env.ram_sb [report_phase] Correct Tests =        3003, Error Tests =          0
# UVM_INFO SPI_slave_scoreboard.sv(61) @ 6006: uvm_test_top.slave_env.sb [report_phase] SPI SLAVE SCOREBOARD
# UVM_INFO SPI_slave_scoreboard.sv(62) @ 6006: uvm_test_top.slave_env.sb [report_phase] Total successful rx_data:       3003
# UVM_INFO SPI_slave_scoreboard.sv(63) @ 6006: uvm_test_top.slave_env.sb [report_phase] Total failed rx_data:          0
# UVM_INFO SPI_slave_scoreboard.sv(64) @ 6006: uvm_test_top.slave_env.sb [report_phase] Total successful rx_valid:      3003
# UVM_INFO SPI_slave_scoreboard.sv(65) @ 6006: uvm_test_top.slave_env.sb [report_phase] Total failed rx_valid:         0
# UVM_INFO SPI_slave_scoreboard.sv(66) @ 6006: uvm_test_top.slave_env.sb [report_phase] Total successful MISO:       3003
# UVM_INFO SPI_slave_scoreboard.sv(67) @ 6006: uvm_test_top.slave_env.sb [report_phase] Total failed MISO:          0
# UVM_INFO WRAPPER_scoreboard.sv(51) @ 6006: uvm_test_top.wrapper_env.wrapper_sb [report_phase] Correct Tests =        3003, Error Tests =          0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   25
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     9
# [run_phase]    12
# ** Note: $finish    : C:/questasim64_2024.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 6006 ns  Iteration: 61  Instance: /WRAPPER_top
```