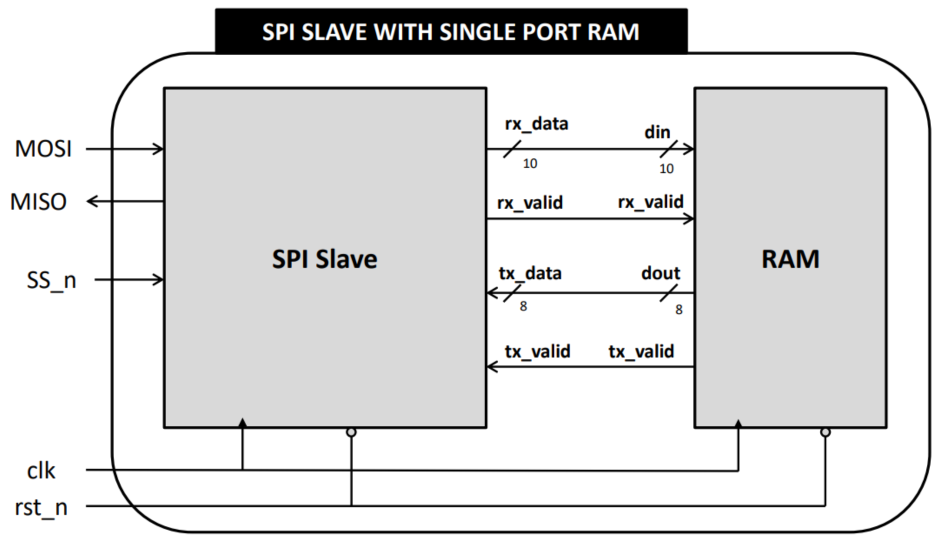
SPI Slave with Single Port RAM



Prepared by: Ahmed Tarek Hassanien

Table of Contents

1. **Overview of SPI protocol** 6
   1.1 SPI Slave with Single Port RAM 7
2. **SPI slave** 12
   2.1 **Design** 12
      2.1.1 Verilog code 12
      2.1.2 RTL elaboration 15
      2.1.3 Synthesis schematic 15
      2.1.4 Static-timing analysis and Utilization reports after synthesis 16
      2.1.5 Device Snippet 16
      2.1.6 Static-timing analysis and Utilization reports after implementation 17
   2.2 **Verification** 18
      2.2.1 UVM Hierarchy 18
      2.2.2 Buggy Design 19
      2.2.3 UVM config_item 23
      2.2.4 UVM sequence_item 23
      2.2.5 UVM reset_sequence 25
      2.2.6 UVM main_sequence 26
      2.2.7 UVM sequencer 27
      2.2.8 UVM driver 28
      2.2.9 UVM monitor 29
      2.2.10 UVM agent 30
      2.2.11 UVM scoreboard 31
      2.2.12 UVM coverage 32
      2.2.13 UVM environment 34
      2.2.14 UVM test 35
      2.2.15 Slave interface 36
      2.2.16 Slave shared package 36
      2.2.17 Slave assertions 37
      2.2.18 Slave Golden model 38
      2.2.19 UVM top 40
      2.2.20 Run.do 40

2.2.21 Bugs Found after slave verification 41
2.2.22 Simulation results 43
2.2.23 Assertions results 45
2.2.24 Coverage results 46

3 **SPI ram** 49

3.1 **Design** 49

3.1.1 Verilog code 49
3.1.2 RTL elaboration 50
3.1.3 Synthesis schematic 50
3.1.4 Static-timing analysis and Utilization reports after synthesis 51
3.1.5 Device Snippet 51
3.1.6 Static-timing analysis and Utilization reports after implementation 52

3.2 **Verification** 53

3.2.1 UVM Hierarchy 53
3.2.2 Buggy Design 54
3.2.3 UVM config_item 54
3.2.4 UVM sequence_item 55
3.2.5 UVM reset_sequence 56
3.2.6 UVM write_only_sequence 57
3.2.7 UVM read_only_sequence 58
3.2.8 UVM write_read_sequence 59
3.2.9 UVM sequencer 60
3.2.10 UVM driver 60
3.2.11 UVM monitor 61
3.2.12 UVM agent 62
3.2.13 UVM scoreboard 63
3.2.14 UVM coverage 64
3.2.15 UVM environment 65
3.2.16 UVM test 66
3.2.17 Slave interface 67
3.2.18 Slave shared package 68
3.2.19 Ram assertions 68
3.2.20 Ram Golden model 69

3.2.21 UVM top 70
3.2.22 Run.do 70
3.2.23 Bugs Found after ram verification 71
3.2.24 Simulation results....72
3.2.25 Assertions results 75
3.2.26 Coverage results....76
4 **SPI Wrapper**....79
4.1 **Design**....79
4.1.1 Verilog code 79
4.1.2 RTL elaboration 80
4.1.3 Synthesis schematic....80
4.1.4 Static-timing analysis and Utilization reports after synthesis 81
4.1.5 Device Snippet 81
4.1.6 Static-timing analysis and Utilization reports after implementation 82
4.2 **Verification**....83
4.2.1 UVM Hierarchy 84
4.2.2 Buggy Design....85
4.2.3 UVM config_item 85
4.2.4 UVM sequence_item....86
4.2.5 UVM reset_sequence....87
4.2.6 UVM write_only_sequence 88
4.2.7 UVM read_only_sequence....89
4.2.8 UVM write_read_sequence....90
4.2.9 UVM sequencer 91
4.2.10 UVM driver....91
4.2.11 UVM monitor 92
4.2.12 UVM agent....93
4.2.13 UVM scoreboard....94
4.2.14 UVM coverage....95
4.2.15 UVM environment 96
4.2.16 UVM test....97
4.2.17 Wrapper interface....99
4.2.18 Wrapper shared package 99

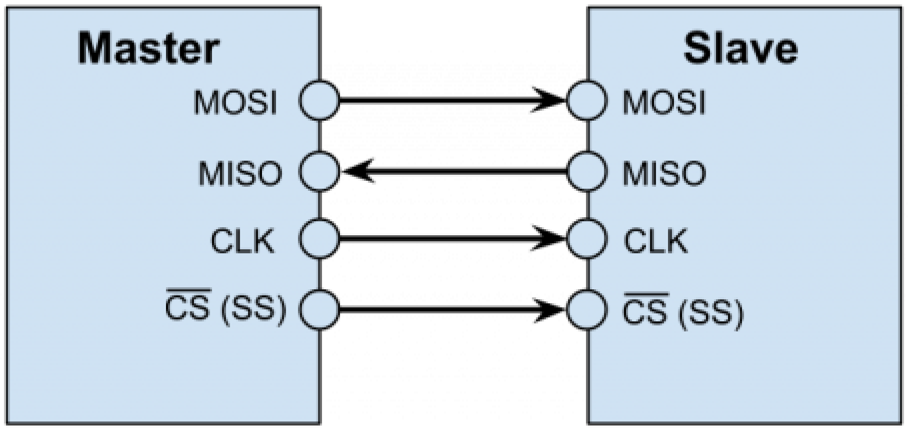
4.2.19 Wrapper assertions 100
4.2.20 Wrapper Golden model 101
4.2.21 UVM top 102
4.2.22 Run.do 103
4.2.23 Bugs Found after wrapper verification 104
4.2.24 Simulation results 106
4.2.25 Assertions results 109
4.2.26 Coverage results 110

1. Overview of SPI protocol

The Serial Peripheral Interface (SPI) is one of the most widely used synchronous serial communication interfaces.

It operates using four primary signals:

- MOSI (Master-Out-Slave-In)
- MISO (Master-In-Slave-Out)
- SCK (Serial Clock)
- SS_n (Slave Select)



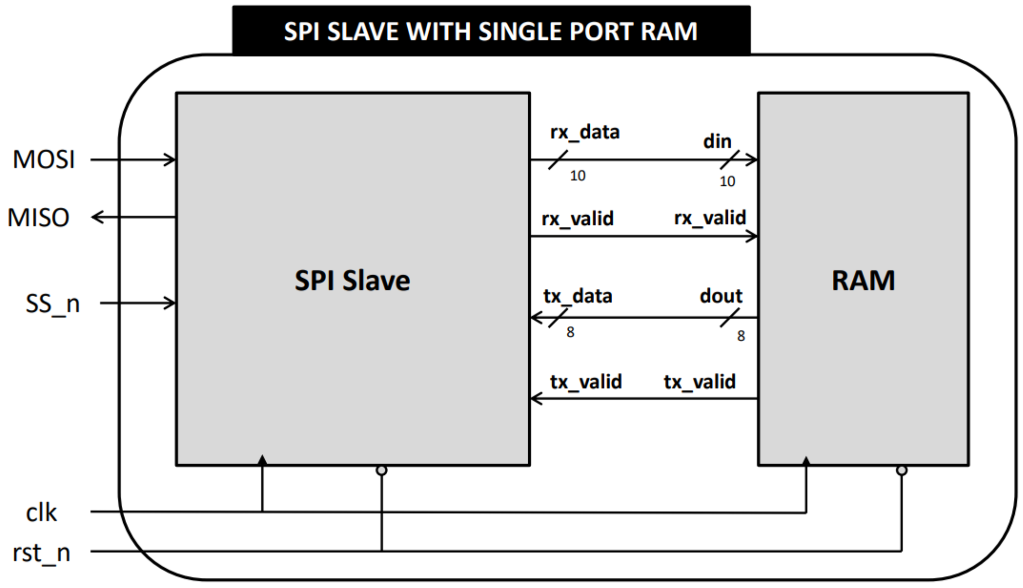
Communication is initiated when the Master pulls SS_n low.

Data is then exchanged serially on MOSI and MISO lines, synchronized by the SCK.

When communication is complete, SS_n is de-asserted high.

1.1 SPI Slave with Single Port RAM

In this project, an SPI slave interacts with a single-port RAM module.



- **Wrapper ports:**

Port	Direction	Size	Description
MOSI	**Input**	1 bit	Serial data from SPI master
SS_n		1 bit	Active low slave select
clk		1 bit	System clock
rst_n		1 bit	Active-low synchronous reset
MISO	**Output**	1 bit	Serial data to SPI master

- ## Slave ports:

Port	Direction	Size	Description
MOSI	**Input**	1 bit	Serial data from master
SS_n		1 bit	Selects and enables communication
clk		1 bit	System clock
rst_n		1 bit	Active-low synchronous reset
tx_data		8 bits	Data received from RAM
tx_valid		1 bit	Valid signal for tx_data
MISO	**Output**	1 bit	Serial data to master
rx_data		10 bits	Parallel data to RAM
rx_valid		1 bit	Indicates valid rx_data

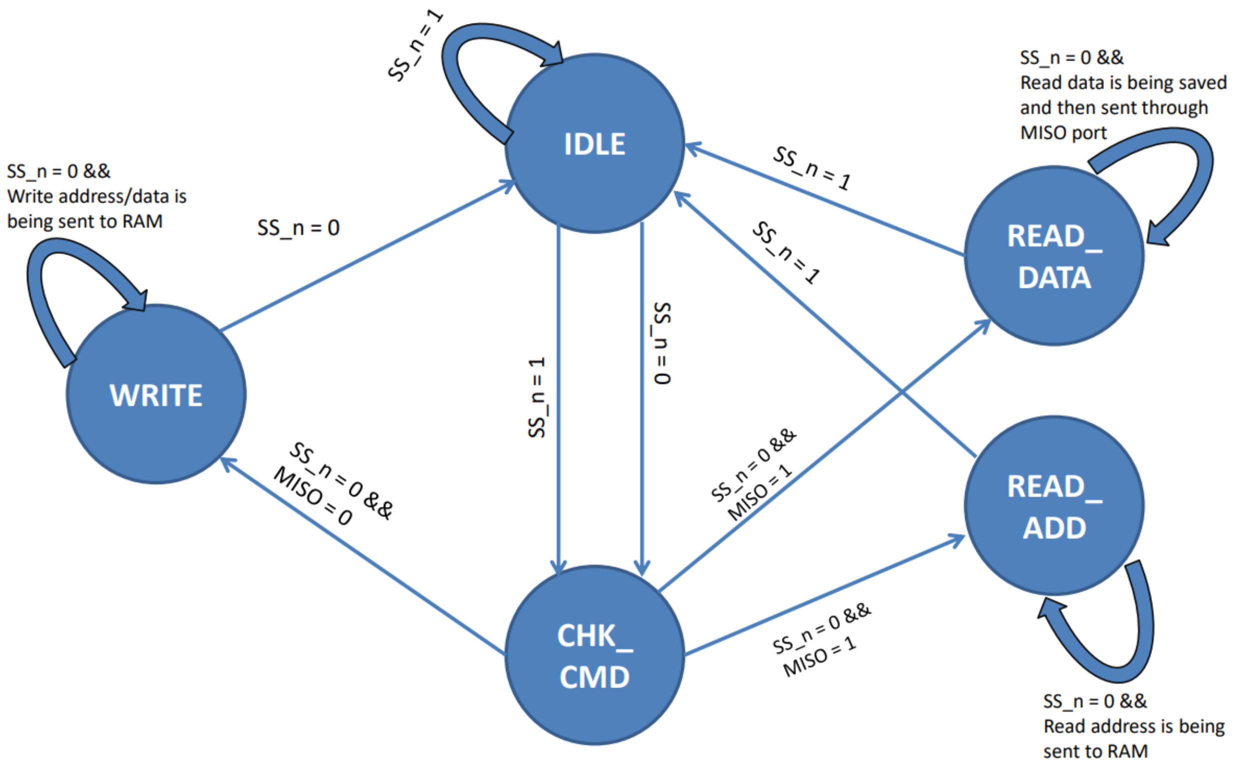
- ## Ram ports:

Name	Type	Size	Description
din	**Input**	10 bits	Data Input
clk		1 bit	Clock
rst_n		1 bit	Active low synchronous reset
rx_valid		1 bit	If HIGH: accept din[7:0] to save the write/read address internally or write a memory word depending on the most significant 2 bits din[9:8]
dout	**Output**	8 bits	Data Output
tx_valid		1 bit	Whenever the command is memory read the tx_valid should be HIGH

Most significant din bits “din[9:8]” determine if it’s a write or read command.

Port	din[9:8]	Command	Description
din	00	Write	Hold din[7:0] internally as write address
	01		Write din[7:0] in the memory with write address held previously
	10	Read	Hold din[7:0] internally as read address
	11		Read the memory with read address held previously, tx_valid should be HIGH, dout holds the word read from the memory, ignore din[7:0]

- **SPI Slave State Machine:**


IDLE
WRITE
CHK_ CMD
READ_ DATA
READ_ ADD
SS_n = 1
SS_n = 0 && Write address/data is being sent to RAM
SS_n = 0
SS_n = 0 && Read data is being saved and then sent through MISO port
SS_n = 1
SS_n = 1
SS_n = 1
SS_n = 0
SS_n = 0 && MISO = 0
SS_n = 0 && MISO = 1
SS_n = 0 && MISO = 1
SS_n = 0 && Read address is being sent to RAM

Note: An internal signal is needed to allow SPI slave memorize if the read address is received or not to decide for READ_ADD or READ_DATA transition.

- **RAM Write Command – Write Address** rx_data[9:8] = din[9:8] = 2’b00

    1. Master will start the write command by sending the write address value, rx_data[9:8] = din[9:8] = 2’b00
    2. SS_n = 0 to tell the SPI Slave that the master will begin communication
    3. SPI Slave check the first received bit on MOSI port ‘0’ which is a control bit to let the slave determine which operation will take place “write in this case”. SPI Slave then expects to receive 10 more bits, the first 2 bits are “00” on two clock cycles and then the wr_address will be sent on 8 more clock cycles
    4. Now the data is converted from serial “MOSI” to parallel after writing the rx_data[9:0] bus. rx_data[9] holds the first bit received from the MOSI port and rx_data[0] holds the last bit received.
    5. rx_valid will be HIGH to inform the RAM that it should expect data on din bus
    6. din takes the value of rx_data
    7. RAM checks on din[9:8] and find that they hold “00”
    8. RAM stores din[7:0] in the internal write address bus
    9. SS_n = 1 to end communication from Master side

- **RAM Write Command – Write Data** rx_data[9:8] = din[9:8] = 2’b01

    1. Master will continue the write command by sending the write data value, rx_data[9:8] = din[9:8] = 2’b01
    2. SS_n = 0 to tell the SPI Slave that the master will begin communication
    3. SPI Slave check the first received bit on MOSI port ‘0’ which is a control bit to let the slave determine which operation will take place “write in this case”. SPI Slave then expects to receive 10 more bits, the first 2 bits are “01” on two clock cycles and then the wr_data will be sent on 8 more clock cycles
    4. Now the data is converted from serial “MOSI” to parallel after writing the rx_data[9:0] bus. rx_data[9] holds the first bit received from the MOSI port and rx_data[0] holds the last bit received.
    5. rx_valid will be HIGH to inform the RAM that it should expect data on din bus
    6. din takes the value of rx_data
    7. RAM checks on din[9:8] and find that they hold “01”
    8. RAM stores din[7:0] in the RAM with wr_address previously held
    9. SS_n = 1 to end communication from Master side

- **RAM Read Command – Read Address** rx_data[9:8] = din[9:8] = 2’b10

1. Master will start the read command by sending the read address value, rx_data[9:8] = din[9:8] = 2’b10
2. SS_n = 0 to tell the SPI Slave that the master will begin communication
3. SPI Slave check the first received bit on MOSI port ‘1’ which is a control bit to let the slave determine which operation will take place “read in this case”. SPI Slave then expects to receive 10 more bits, the first 2 bits are “10” on two clock cycles and then the rd_address will be sent on 8 more clock cycles
4. Now the data is converted from serial “MOSI” to parallel after writing the rx_data[9:0] bus. rx_data[9] holds the first bit received from the MOSI port and rx_data[0] holds the last bit received.
5. rx_valid will be HIGH to inform the RAM that it should expect data on din bus
6. din takes the value of rx_data
7. RAM checks on din[9:8] and find that they hold “10”
8. RAM stores din[7:0] in the internal read address bus
9. SS_n = 1 to end communication from Master side

- **RAM Read Command – Read Data** rx_data[9:8] = din[9:8] = 2’b11

1. Master will continue the read command by sending the read data command, rx_data[9:8] = din[9:8] = 2’b11
2. SS_n = 0 to tell the SPI Slave that the master will begin communication
3. SPI Slave check the first received bit on MOSI port ‘1’ which is a control bit to let the slave determine which operation will take place “read in this case”. SPI Slave then expects to receive 10 more bits, the first 2 bits are “11” on two clock cycles and the dummy data will be sent and ignored since the master is waiting for the data to be sent from slave side
4. Now the data is converted from serial “MOSI” to parallel after writing the rx_data[9:0] bus. rx_data[9] holds the first bit received from the MOSI port and rx_data[0] holds the last bit received.
5. din takes the value of rx_data
6. RAM checks din[9:8] and find that they hold “11”
7. RAM will read from the memory with rd_address previously held
8. RAM will assert tx_valid to inform slave that data out is ready
9. Slave reads tx_data and convert it into serial out data on MISO port starting with the most significant bit.
10. SS_n = 1, Master ends communication after receiving data “8 clock cycles”

2. SPI slave

2.1 Design

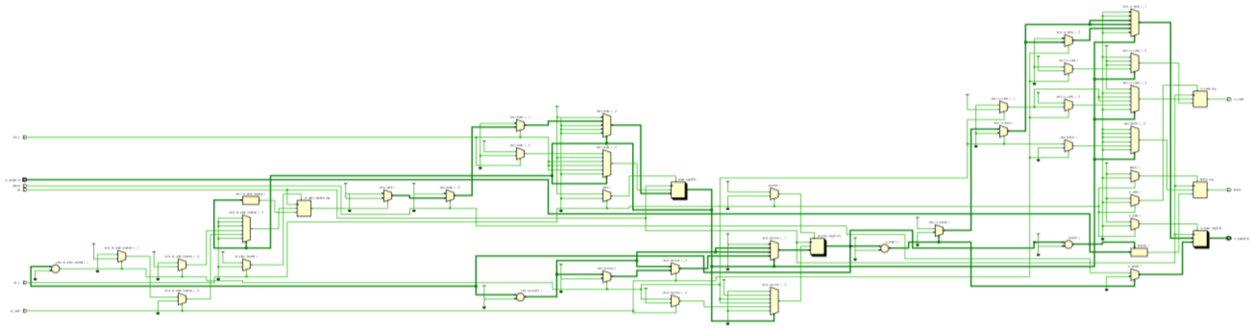
2.1.1 Verilog code

```
module slave(
    input  logic             clk,
    input  logic             MOSI,
    input  logic             rst_n,
    input  logic             SS_n,
    input  logic             tx_valid,
    input  logic [7:0]       tx_data,
    output logic [9:0]       rx_data,
    output logic             rx_valid,
    output logic             MISO
);
    typedef enum logic [2:0] {
        IDLE      = 3'b000,
        WRITE     = 3'b001,
        CHK_CMD   = 3'b010,
        READ_ADD  = 3'b011,
        READ_DATA = 3'b100
    } SPI_slave_state_e;

    // FSM signals
    (*fsm_encoding ="sequential"*)
    SPI_slave_state_e   state          ,next_state;
    reg                                 next_rx_valid;
    reg [9:0]                           next_rx_data;
    reg                                 next_MISO;
    reg                 rd_addr_loaded ,next_rd_addr_loaded;
    int                 counter        ,next_counter;

    // state memory
    always @ (posedge clk) begin
        if (~rst_n) begin
            state           <= IDLE;
            rd_addr_loaded  <= 0;
            rx_valid        <= 0;
            MISO            <= 0;
            rx_data         <= 0;
            counter         <= 0;
        end
        else begin
            state           <= next_state;
            rd_addr_loaded  <= next_rd_addr_loaded;
            rx_valid        <= next_rx_valid;
            MISO            <= next_MISO;
            rx_data         <= next_rx_data;
            counter         <= next_counter;
        end
    end
```

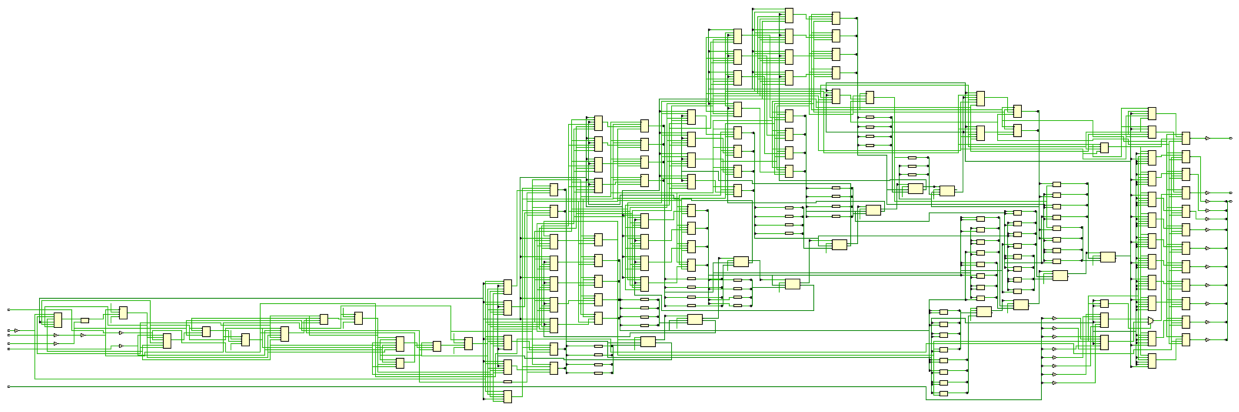
```
// next state logic
always @(*) begin
    // default
    next_state          = state;
    next_rd_addr_loaded = rd_addr_loaded;
    next_rx_valid       = rx_valid;
    next_MISO           = MISO;
    next_rx_data        = rx_data;
    next_counter        = counter;

    case (state)
        IDLE: begin
            next_rx_valid = 0;

            if (~SS_n) next_state = CHK_CMD;
        end

        CHK_CMD: begin
            if (SS_n) next_state = IDLE;
            else if (~MOSI) next_state = WRITE;
            else begin
                if  (rd_addr_loaded) next_state = READ_DATA;
                else next_state = READ_ADD;
            end
            next_counter = 10;
        end

        WRITE: begin
            if (SS_n) next_state = IDLE;

            if (counter > 0) begin
                next_rx_data [next_counter-1] = MOSI;
                next_counter = counter - 1;
            end

            else next_rx_valid = 1;
        end

        READ_ADD: begin
            if (SS_n) next_state = IDLE;

            if (counter > 0) begin
                next_rx_data [next_counter-1] = MOSI;
                next_counter = counter - 1;
            end
            else begin
                next_rx_valid = 1;
                next_rd_addr_loaded = 1;
            end
        end
```

```
            READ_DATA: begin
                if (SS_n) next_state = IDLE;

                if (tx_valid) //Data is ready to be serialed out on MISO
                begin
                    next_rx_valid = 0;
                    if (counter > 0) begin
                        next_counter = counter - 1;
                        next_MISO = tx_data[counter-1];
                    end
                    else next_rd_addr_loaded = 0;
                end
                else begin
                    if (counter > 0) begin
                        next_rx_data [next_counter-1] = MOSI;
                        next_counter = counter - 1;
                    end
                    else begin
                        next_rx_valid = 1;
                        next_counter = 9;
                        //added a cycle delay before sending MISO data
                    end
                end
            end
        endcase
    end
endmodule
```

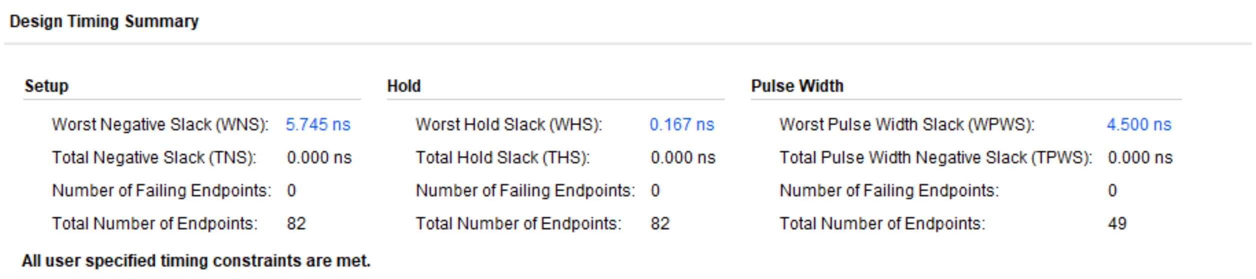
2.1.2 RTL elaboration



2.1.3 Synthesis schematic



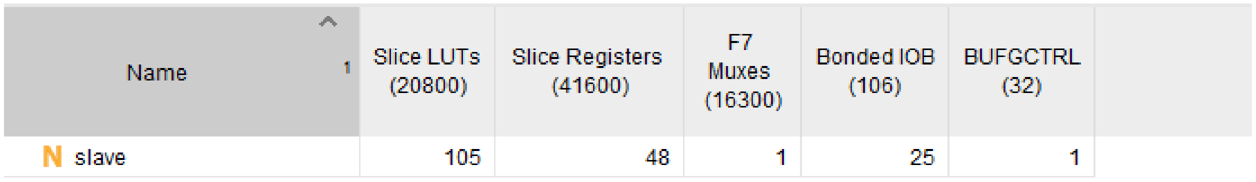
2.1.4 Static-timing analysis and Utilization reports after synthesis



Design Timing Summary

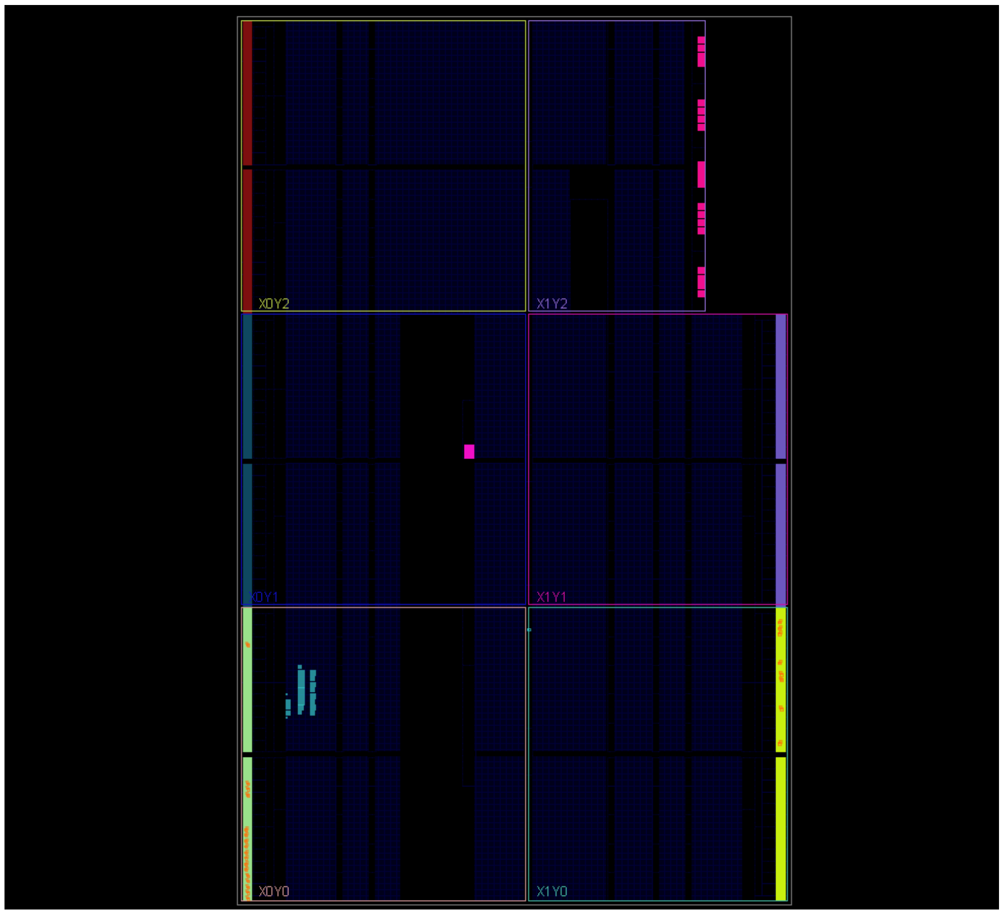
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.745 ns	Worst Hold Slack (WHS):	0.167 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	82	Total Number of Endpoints:	82	Total Number of Endpoints:	49

All user specified timing constraints are met.

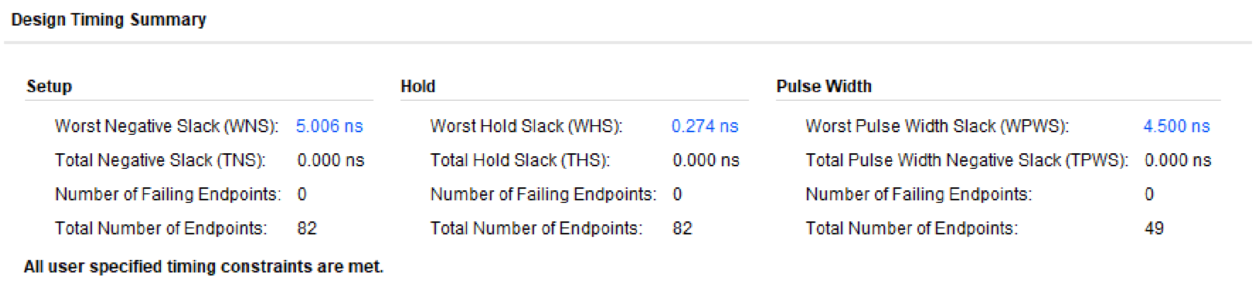


Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Bonded IOB (106)	BUFGCTRL (32)
N slave	105	48	1	25	1

2.1.5 Device Snippet



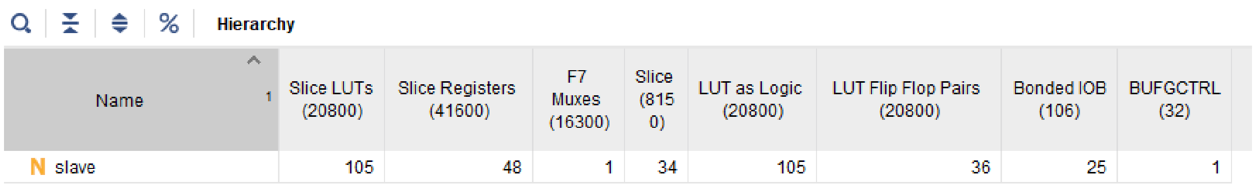
2.1.6 Static-timing analysis and Utilization reports after implementation



Design Timing Summary

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	5.006 ns	Worst Hold Slack (WHS):	0.274 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	82	Total Number of Endpoints:	82	Total Number of Endpoints:	49

All user specified timing constraints are met.



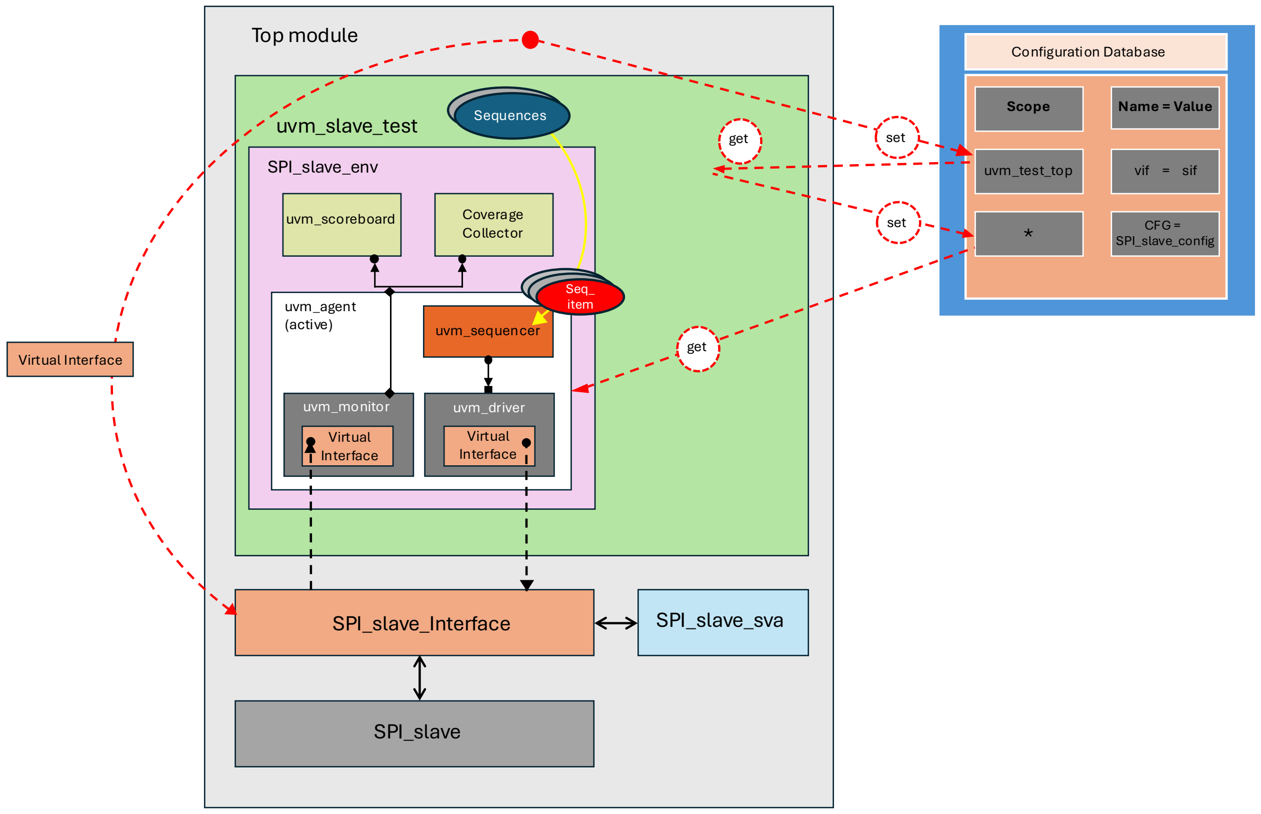
Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
slave	105	48	1	34	105	36	25	1

2.2 Verification

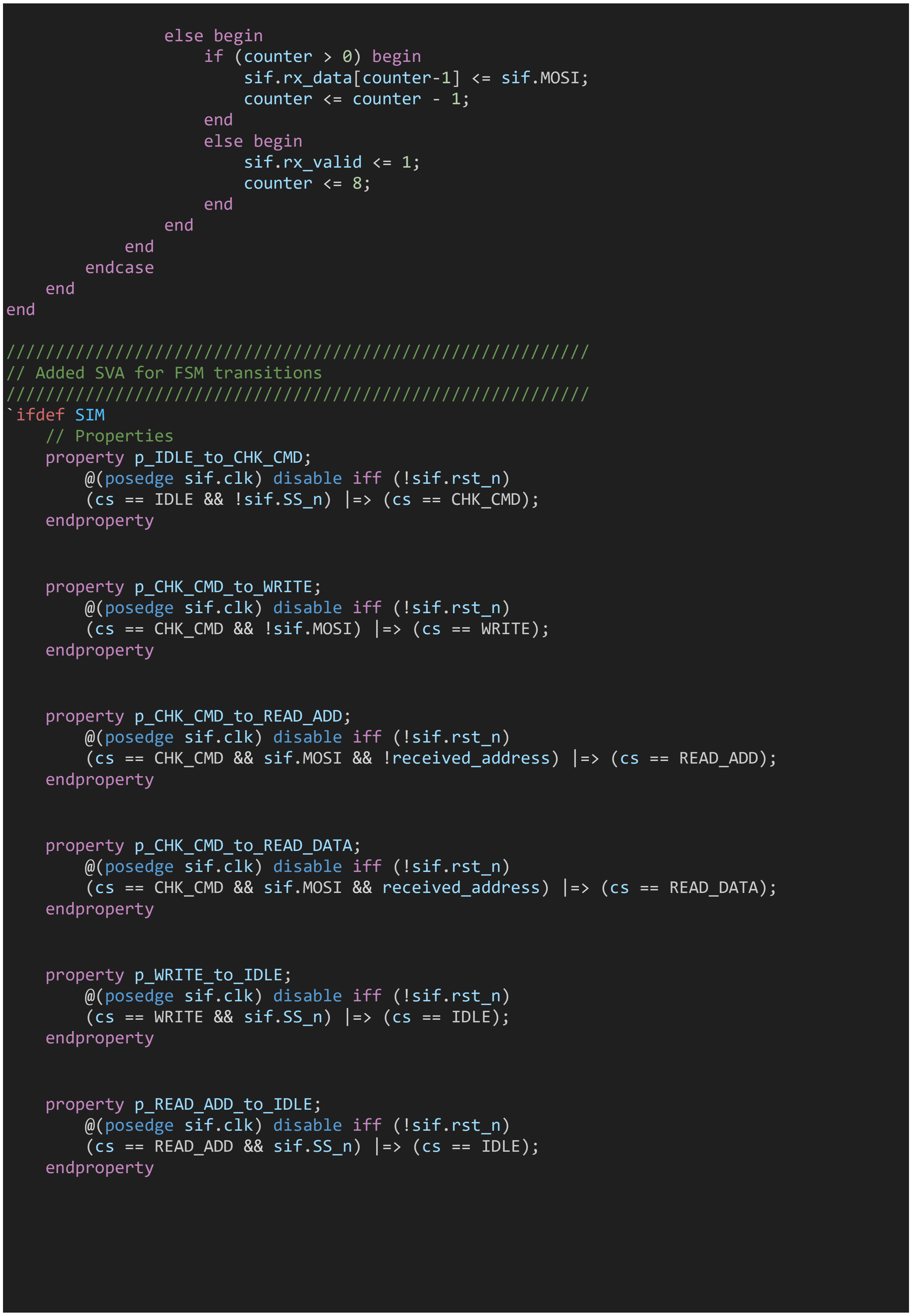
- Verification will be performed on a buggy design, with the primary objective of identifying and reporting its bugs.
- This verification will be accomplished by comparing the design's output against a golden model “the design above”.

2.2.1 UVM Hierarchy



2.2.2 Buggy Design

```
module SLAVE (SPI_slave_if.DUT sif);
import SPI_slave_shared_pkg::*;

SPI_slave_state_e cs, ns;

int       counter;
reg       received_address;


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

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

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



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

///////////////////////////////////////////////////////////
// Added SVA for FSM transitions
///////////////////////////////////////////////////////////
`ifdef SIM
    // Properties
    property p_IDLE_to_CHK_CMD;
        @(posedge sif.clk) disable iff (!sif.rst_n)
        (cs == IDLE && !sif.SS_n) |=> (cs == CHK_CMD);
    endproperty


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


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


    property p_CHK_CMD_to_READ_DATA;
        @(posedge sif.clk) disable iff (!sif.rst_n)
        (cs == CHK_CMD && sif.MOSI && received_address) |=> (cs == READ_DATA);
    endproperty


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


    property p_READ_ADD_to_IDLE;
        @(posedge sif.clk) disable iff (!sif.rst_n)
        (cs == READ_ADD && sif.SS_n) |=> (cs == IDLE);
    endproperty
    property p_READ_DATA_to_IDLE;
        @(posedge sif.clk) disable iff (!sif.rst_n)
        (cs == READ_DATA && sif.SS_n) |=> (cs == IDLE);
    endproperty


    // Assertions
    assert property (p_IDLE_to_CHK_CMD)
    else $error("Illegal FSM transition: IDLE must only go to CHK_CMD");
    assert property (p_CHK_CMD_to_READ_ADD)
    else $error("Illegal FSM transition: CHK_CMD must only go to READ_AD");
    assert property (p_CHK_CMD_to_READ_DATA)
    else $error("Illegal FSM transition: CHK_CMD must only go toREAD_DATA");
    assert property (p_CHK_CMD_to_WRITE)
    else $error("Illegal FSM transition: CHK_CMD must only go to WRITE");
    assert property (p_WRITE_to_IDLE)
    else $error("Illegal FSM transition: WRITE must return to IDLE");
    assert property (p_READ_ADD_to_IDLE)
    else $error("Illegal FSM transition: READ_ADD must return to IDLE");
    assert property (p_READ_DATA_to_IDLE)
    else $error("Illegal FSM transition: READ_DATA must return to IDLE");

    //Coverage
    cover property (p_IDLE_to_CHK_CMD);
    cover property (p_CHK_CMD_to_READ_ADD);
    cover property (p_CHK_CMD_to_READ_DATA);
    cover property (p_CHK_CMD_to_WRITE);
    cover property (p_WRITE_to_IDLE);
    cover property (p_READ_ADD_to_IDLE);
    cover property (p_READ_DATA_to_IDLE);

`endif
Endmodule
```

2.2.3 UVM config_item

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

    class SPI_slave_cfg extends uvm_object;
        `uvm_object_utils(SPI_slave_cfg)
        virtual SPI_slave_if Slave_vif;
        uvm_active_passive_enum is_active;

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


    endclass : SPI_slave_cfg
endpackage
```

2.2.4 UVM sequence_item

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

    class SPI_slave_seq_item extends uvm_sequence_item;
        `uvm_object_utils(SPI_slave_seq_item)

        rand bit            rst_n, SS_n, tx_valid, MOSI;
        rand bit [7:0]      tx_data;
        rand bit [10:0]     array_for_MOSI;
        bit      [9:0]      rx_data, rx_data_ref;
        bit                 rx_valid,rx_valid_ref, MISO, MISO_ref;
        int                 counter = 0;
        SPI_slave_state_e   cs;

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

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

        constraint c_rst_n { rst_n dist {0 := 1, 1 := 40}; }

        constraint c_SS_n {
            if (cs == SPI_slave_state_e'(READ_DATA)) SS_n == (counter % 24 == 0);
            else SS_n == (counter % 14 == 0);
            }
        constraint c_tx_valid {
            if (cs == SPI_slave_state_e'(READ_DATA) && counter >= 14)
                 tx_valid == 1;
            else tx_valid == 0;
            }
        constraint c_array_for_MOSI {
            array_for_MOSI[10:8] inside {3'b000, 3'b001, 3'b110, 3'b111};
            }

        function void post_randomize();
        if (rst_n == 0) counter = 0;
        else if (SS_n == 1) counter = 1;
        else counter++;
        endfunction : post_randomize

    endclass : SPI_slave_seq_item
endpackage
```

2.2.5 UVM reset_sequence

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

    class SPI_slave_reset_seq extends uvm_sequence #(SPI_slave_seq_item);
        `uvm_object_utils(SPI_slave_reset_seq)
        `uvm_declare_p_sequencer (SPI_slave_sequencer)
        /*This macro gives a handle to the sequencer it’s running on.
          Lets the sequence access custom fields/methods on the sequencer.*/
        SPI_slave_seq_item item;

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

        task body();
            item = SPI_slave_seq_item::type_id::create("item");
                start_item(item);
                item.rst_n = 0;
                item.SS_n = 1;
                item.MOSI = 0;
                item.tx_valid = 0;
                item.tx_data = 0;
                item.array_for_MOSI = 0;
                finish_item(item);

        endtask : body
    endclass : SPI_slave_reset_seq

endpackage
```

2.2.6 UVM main_sequence

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

    class SPI_slave_main_seq extends uvm_sequence #(SPI_slave_seq_item);
        `uvm_object_utils(SPI_slave_main_seq)
        `uvm_declare_p_sequencer (SPI_slave_sequencer)
        //This macro gives a handle to the sequencer it’s running on.
        //Lets the sequence access custom fields/methods on the sequencer.
        SPI_slave_seq_item item;
        function new(string name = "SPI_slave_main_seq");
            super.new(name);
        endfunction : new

        task body();
            item = SPI_slave_seq_item::type_id::create("item");
            repeat (1000) begin
                for (int i=10; i>=0; i--) begin
                    start_item(item);
                    //get current SPI_slave state
                    item.cs = p_sequencer.sequencer_vif.cs;

                    if (item.tx_valid) item.tx_data.rand_mode(0);
                    else item.tx_data.rand_mode(1);

                    if (item.SS_n == 1) begin
                        // enable randomization for new array when SS_n is high
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize());
                        i = 11;
                    end
                    else if (item.cs == IDLE) begin
                        item.array_for_MOSI.rand_mode(0);
                        assert(item.randomize());
                        i = 11;
                    end
                    else begin
                        // disable randomization to keep array values stable
                        item.array_for_MOSI.rand_mode(0);
                        // send array bits [10:0] sequentially to MOSI
                        assert(item.randomize()
                                with {item.MOSI == item.array_for_MOSI[i];});
                    end
                    finish_item(item);
                end
            end
        endtask : body
    endclass : SPI_slave_main_seq
endpackag
```

2.2.7 UVM sequencer

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

    class SPI_slave_sequencer extends uvm_sequencer #(SPI_slave_seq_item);
        `uvm_component_utils(SPI_slave_sequencer)
        SPI_slave_cfg cfg;
        virtual SPI_slave_if sequencer_vif;

        function new(string name, uvm_component parent);
            super.new(name, parent);
        endfunction : new

    endclass : SPI_slave_sequencer

endpackage
```

2.2.8 UVM driver

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

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

        virtual SPI_slave_if driver_vif;
        SPI_slave_seq_item stim_seq_item;

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

        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);

                driver_vif.SS_n        = stim_seq_item.SS_n;
                driver_vif.rst_n       = stim_seq_item.rst_n;
                driver_vif.MOSI        = stim_seq_item.MOSI;
                driver_vif.tx_data     = stim_seq_item.tx_data;
                driver_vif.tx_valid    = stim_seq_item.tx_valid;
                driver_vif.array_for_MOSI = stim_seq_item.array_for_MOSI;
                @(negedge driver_vif.clk);

                seq_item_port.item_done();
                `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask : run_phase
    endclass: SPI_slave_driver
endpackage
```

2.2.9 UVM monitor

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

    class SPI_slave_monitor extends uvm_monitor;
        `uvm_component_utils(SPI_slave_monitor)

        virtual SPI_slave_if monitor_vif;
        SPI_slave_seq_item rsp_seq_item;
        uvm_analysis_port #(SPI_slave_seq_item) monitor_ap;

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

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

        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 monitor_vif.clk);
                rsp_seq_item.SS_n           = monitor_vif.SS_n;
                rsp_seq_item.rst_n          = monitor_vif.rst_n;
                rsp_seq_item.MOSI           = monitor_vif.MOSI;
                rsp_seq_item.tx_data        = monitor_vif.tx_data;
                rsp_seq_item.tx_valid       = monitor_vif.tx_valid;
                rsp_seq_item.rx_data        = monitor_vif.rx_data;
                rsp_seq_item.rx_data_ref    = monitor_vif.rx_data_ref;
                rsp_seq_item.MISO           = monitor_vif.MISO;
                rsp_seq_item.MISO_ref       = monitor_vif.MISO_ref;
                rsp_seq_item.rx_valid       = monitor_vif.rx_valid;
                rsp_seq_item.rx_valid_ref   = monitor_vif.rx_valid_ref;
                rsp_seq_item.cs             = monitor_vif.cs;

                monitor_ap.write(rsp_seq_item);
                `uvm_info("run_phase", rsp_seq_item.convert2string(), UVM_HIGH)

            end
        endtask : run_phase
    endclass: SPI_slave_monitor

endpackage
```

2.2.10 UVM agent

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

    class SPI_slave_agent extends uvm_agent;
        `uvm_component_utils(SPI_slave_agent)
        SPI_slave_driver driver;
        SPI_slave_monitor monitor;
        SPI_slave_sequencer sequencer;
        SPI_slave_cfg SPI_slave_config;
        uvm_analysis_port #(SPI_slave_seq_item) agent_ap;

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

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

            if (!uvm_config_db#(SPI_slave_cfg)::get(this, "", "CFG", SPI_slave_config))
            begin
                `uvm_fatal("NOVIF", "Cannot get cfg from uvm_config_db")
            end
            // only build driver and sequencer if agent is active
            if (SPI_slave_config.is_active == UVM_ACTIVE) begin
                driver      = SPI_slave_driver::type_id::create("driver", this);
                sequencer   = SPI_slave_sequencer::type_id::create("sequencer", this);
            end

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

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            // only connect driver and sequencer if agent is active
            if (SPI_slave_config.is_active == UVM_ACTIVE) begin
                driver.seq_item_port.connect(sequencer.seq_item_export);
                driver.driver_vif       = SPI_slave_config.Slave_vif;
                sequencer.sequencer_vif = SPI_slave_config.Slave_vif;
            end
            monitor.monitor_ap.connect(agent_ap);
            monitor.monitor_vif = SPI_slave_config.Slave_vif;
        endfunction : connect_phase

    endclass: SPI_slave_agent
endpackage
```

2.2.11 UVM scoreboard

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

    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 error_count =0; int 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);
            forever begin
                sb_fifo.get(seq_item_sb);
                if (seq_item_sb.rx_data !== seq_item_sb.rx_data_ref ||
                    seq_item_sb.MISO !== seq_item_sb.MISO_ref ||
                    seq_item_sb.rx_valid !== seq_item_sb.rx_valid_ref)
                begin
                    if (seq_item_sb.rx_data !== seq_item_sb.rx_data_ref)
                        `uvm_error("run_phase", $sformatf("Mismatch in rx_data:
                                  expected=%0d, actual=%0d, %s",
                                  seq_item_sb.rx_data_ref, seq_item_sb.rx_data,
                                  seq_item_sb.convert2string_stimulus()))

                    else if (seq_item_sb.MISO !== seq_item_sb.MISO_ref)
                        `uvm_error("run_phase", $sformatf("Mismatch in MISO:
                                  expected=%0b, actual=%0b, %s",
                                  seq_item_sb.MISO_ref, seq_item_sb.MISO,
                                  seq_item_sb.convert2string_stimulus()))

                    else if (seq_item_sb.rx_valid !== seq_item_sb.rx_valid_ref)
                        `uvm_error("run_phase", $sformatf("Mismatch in rx_valid:
                                  expected=%0b, actual=%0b, %s",
                                  seq_item_sb.rx_valid_ref, seq_item_sb.rx_valid,
                                  seq_item_sb.convert2string_stimulus()))

                    error_count++;
                end
```

```
                 else begin
                    `uvm_info("run_phase", $sformatf("Match: rx_data=%0d, MISO=%0b,
                               rx_valid=%0b", seq_item_sb.rx_data, seq_item_sb.MISO,
                               seq_item_sb.rx_valid), UVM_HIGH)
                    correct_count++;
                end
            end
        endtask

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Total Correct: %0d, Total Errors: %0d",
                       correct_count, error_count), UVM_MEDIUM)
        endfunction
    endclass: SPI_slave_scoreboard

endpackage
```

2.2.12 UVM coverage

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

    class SPI_slave_coverage extends uvm_component;
        `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 cov_item;

        // Functional Coverage
        covergroup cg_SPI_slave;
            cp_rxdata: coverpoint cov_item.rx_data[9:8] {
                bins all_values[] = {[0:3]};
                bins all_transitions[] = (0 => 0), (0 => 1), (0 => 2), (0 => 3),
                                         (1 => 0), (1 => 1), (1 => 2), (1 => 3),
                                         (2 => 0), (2 => 1), (2 => 2), (2 => 3),
                                         (3 => 0), (3 => 1), (3 => 2), (3 => 3);
                //Ignore invalid transactions as they require multiple bit toggles at same
                  cycle
                ignore_bins rx_data_bins_ignored[] = (0 => 3),(2 => 1),(1 => 2);
            }

            cp_SS_n: coverpoint cov_item.SS_n {
                bins SS_n_OTHER     = (1=> 0[*13] => 1);
                bins SS_n_READ_DATA = (1=> 0[*23] => 1);
                bins SS_n_one2zero  = (1 => 0);
            }
            cp_MOSI: coverpoint cov_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);
            }
```

```
          cross_MOSI_SS_n: cross cp_MOSI, cp_SS_n {
                option.cross_auto_bin_max = 0;
                bins Read_data_SS_n_one2zero =
                     binsof(cp_MOSI.read_data) && binsof(cp_SS_n.SS_n_one2zero);
                bins Write_data_SS_n_one2zero =
                     binsof(cp_MOSI.write_data) && binsof(cp_SS_n.SS_n_one2zero);
                bins Read_address_SS_n_one2zero =
                     binsof(cp_MOSI.read_address) && binsof(cp_SS_n.SS_n_one2zero);
                bins Write_address_SS_n_one2zero =
                     binsof(cp_MOSI.write_address) && binsof(cp_SS_n.SS_n_one2zero);
            }
        endgroup : cg_SPI_slave

        function new(string name = "SPI_slave_coverage", uvm_component parent = null);
            super.new(name, parent);
            cg_SPI_slave = 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 : build_phase

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(cov_item);
                cg_SPI_slave.sample();
            end
        endtask : run_phase
    endclass: SPI_slave_coverage

endpackage
```

2.2.13 UVM environment

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

    class SPI_slave_env extends uvm_env;
        `uvm_component_utils(SPI_slave_env)

        SPI_slave_agent agent;
        SPI_slave_scoreboard scoreboard;
        SPI_slave_coverage coverage;

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

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

            agent       = SPI_slave_agent::type_id::create("agent", this);
            scoreboard  = SPI_slave_scoreboard::type_id::create("scoreboard", this);
            coverage    = SPI_slave_coverage::type_id::create("coverage", this);
        endfunction : build_phase

        virtual function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agent.agent_ap.connect(scoreboard.sb_export);
            agent.agent_ap.connect(coverage.cov_export);
        endfunction : connect_phase
    endclass: SPI_slave_env
endpackage
```

2.2.14 UVM test

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

    class SPI_slave_test extends uvm_test;
        `uvm_component_utils(SPI_slave_test)

        SPI_slave_env env;
        SPI_slave_reset_seq reset_sequence;
        SPI_slave_main_seq main_sequence;
        SPI_slave_cfg SPI_slave_config;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = SPI_slave_env::type_id::create("env", this);
            reset_sequence   = SPI_slave_reset_seq::type_id::create("reset_sequence");
            main_sequence    = SPI_slave_main_seq::type_id::create("main_sequence");
            SPI_slave_config = SPI_slave_cfg::type_id::create("SPI_slave_config");

            if(!uvm_config_db#(virtual SPI_slave_if)::get(this, "", "vif",
               SPI_slave_config.SPI_vif))
                `uvm_fatal("NOVIF", "Virtual interface must be set for:")

            SPI_slave_config.is_active = UVM_ACTIVE;

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

        endfunction : build_phase

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

            // Start with reset sequence
            `uvm_info("run_phase", "Starting reset sequence", UVM_LOW)
            reset_sequence.start(env.agent.sequencer);
            `uvm_info("run_phase", "Completed reset sequence", UVM_LOW)

            // Then run main sequence
            `uvm_info("run_phase", "Starting main sequence", UVM_LOW)
            main_sequence.start(env.agent.sequencer);
            `uvm_info("run_phase", "Completed main sequence", UVM_LOW)

            phase.drop_objection(this);
        endtask : run_phase
    endclass: SPI_slave_test
endpackage
```

2.2.15 Slave interface

```
interface SPI_slave_if (clk);
import SPI_slave_shared_pkg::*;
    input logic                     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;
          SPI_slave_state_e         cs, cs_ref;
          bit                 [10:0]  array_for_MOSI;

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

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

2.2.16 Slave shared package

```
package SPI_slave_shared_pkg;
    typedef enum logic [2:0] {
        IDLE      = 3'b000,
        WRITE     = 3'b001,
        CHK_CMD = 3'b010,
        READ_ADD  = 3'b011,
        READ_DATA = 3'b100
    } SPI_slave_state_e;

endpackage : SPI_slave_shared_pkg
```

2.2.17 Slave assertions

```
module SPI_slave_sva(SPI_slave_if.DUT sif);
    //Sequences
    sequence write_add_seq;
        (sif.SS_n == 1) ##1 (sif.SS_n == 0) ##1 (sif.MOSI == 0) ##1 (sif.MOSI == 0)
        ##1 (sif.MOSI == 0);
    endsequence
    sequence read_add_seq;
        (sif.SS_n == 1) ##1 (sif.SS_n == 0) ##1 (sif.MOSI == 1) ##1 (sif.MOSI == 1)
        ##1 (sif.MOSI == 0);
    endsequence
    sequence read_data_seq;
        (sif.SS_n == 1) ##1 (sif.SS_n == 0) ##1 (sif.MOSI == 1) ##1 (sif.MOSI == 1)
        ##1 (sif.MOSI == 1);
    endsequence
    sequence write_data_seq;
        (sif.SS_n == 1) ##1 (sif.SS_n == 0) ##1 (sif.MOSI == 0) ##1 (sif.MOSI == 0)
        ##1 (sif.MOSI == 1);
    endsequence

    //Properties
    property reset_check;
        @(posedge sif.clk) !sif.rst_n |=> (!sif.rx_valid && !sif.rx_data && !sif.MISO);
    endproperty

    property rx_valid_after_10_cycles;
        @(posedge sif.clk) disable iff (!sif.rst_n)
            (write_add_seq or read_add_seq or read_data_seq or write_data_seq)
            |-> ##10 $rose(sif.rx_valid);
    endproperty

    property SS_n_eventually_after_10_cycles;
        @(posedge sif.clk) disable iff (!sif.rst_n)
            (write_add_seq or read_add_seq or read_data_seq or write_data_seq)
            |-> ##[10:$] $rose(sif.SS_n);
    endproperty

    //Assertions
    assert property (reset_check)
    else $error("Reset check failed: rx_valid, rx_data,
                 and MISO expected low during reset.");
    assert property (rx_valid_after_10_cycles)
    else $error("rx_valid not asserted within 10 cycles.");
    assert property (SS_n_eventually_after_10_cycles)
    else $error("SS_n not asserted eventually within 10 cycles.");

    //Coverage
    cover property (reset_check);
    cover property (rx_valid_after_10_cycles);
    cover property (SS_n_eventually_after_10_cycles);
endmodule : SPI_slave_sva
```

2.2.18 Slave Golden model

```
module Golden_slave(SPI_slave_if.Golden sif);
    import SPI_slave_shared_pkg::*;

    // FSM signals
    SPI_slave_state_e   state          ,next_state;
    reg                                 next_rx_valid;
    reg [9:0]                           next_rx_data;
    reg                                 next_MISO;
    reg                 rd_addr_loaded ,next_rd_addr_loaded;
    int                 counter        ,next_counter;

    // state memory
    always @ (posedge sif.clk) begin
        if (~sif.rst_n) begin
            state            <= IDLE;
            rd_addr_loaded   <= 0;
            sif.rx_valid_ref <= 0;
            sif.MISO_ref     <= 0;
            sif.rx_data_ref  <= 0;
            counter          <= 0;
        end
        else begin
            state            <= next_state;
            rd_addr_loaded   <= next_rd_addr_loaded;
            sif.rx_valid_ref <= next_rx_valid;
            sif.MISO_ref     <= next_MISO;
            sif.rx_data_ref  <= next_rx_data;
            counter          <= next_counter;
        end
    end

    // next state logic
    always @(*) begin
        // default
        next_state           = state;
        next_rd_addr_loaded  = rd_addr_loaded;
        next_rx_valid        = sif.rx_valid_ref;
        next_MISO            = sif.MISO_ref;
        next_rx_data         = sif.rx_data_ref;
        next_counter         = counter;

        case (state)
            IDLE: begin
                next_rx_valid   = 0;

                if (~sif.SS_n) next_state = CHK_CMD;
            end

            CHK_CMD: begin
                if (sif.SS_n) next_state = IDLE;
                else if (~sif.MOSI) next_state = WRITE;
                else begin
                    if  (rd_addr_loaded) next_state = READ_DATA;
                    else next_state = READ_ADD;
                end
                next_counter = 10;
            end
            WRITE: begin
                if (sif.SS_n) next_state = IDLE;

                if (counter > 0) begin
                    next_rx_data [next_counter-1] = sif.MOSI;
                    next_counter = counter - 1;
                end

                else next_rx_valid = 1;
            end

            READ_ADD: begin
                if (sif.SS_n) next_state = IDLE;

                if (counter > 0) begin
                    next_rx_data [next_counter-1] = sif.MOSI;
                    next_counter = counter - 1;
                end
                else begin
                    next_rx_valid = 1;
                    next_rd_addr_loaded = 1;
                end
            end

            READ_DATA: begin
                if (sif.SS_n) next_state = IDLE;

                // Data is ready to be serialed out on MISO
                if (sif.tx_valid)
                begin
                    next_rx_valid = 0;
                    if (counter > 0) begin
                        next_counter = counter - 1;
                        next_MISO = sif.tx_data[counter-1];
                    end
                    else next_rd_addr_loaded = 0;
                end
                else begin
                    if (counter > 0) begin
                        next_rx_data [next_counter-1] = sif.MOSI;
                        next_counter = counter - 1;
                    end
                    else begin
                        next_rx_valid = 1;
                        next_counter = 8;
                    end
                end
            end
        endcase
    end
endmodule
```

2.2.19 UVM top

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

module top ();
    bit clk;

    always #5 clk = ~clk;

    SPI_slave_if sif(clk);

    SLAVE DUT (sif);

    Golden_slave golden_model (sif);

    assign sif.cs = DUT.cs;
    assign sif.cs_ref = golden_model.state;

    bind SLAVE SPI_slave_sva sva (sif);

    initial begin
        uvm_config_db#(virtual SPI_slave_if)::set(null, "uvm_test_top", "vif", sif);
        run_test("SPI_slave_test");
    end
endmodule
```

2.2.20 Run.do

```
vlib work
vlog -f src_files.list +cover -covercells +define+SIM
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover
do wave.do
coverage save slave_top.ucdb -onexit
coverage exclude -src SPI_slave.sv -line 29 -code s
coverage exclude -src SPI_slave.sv -line 28 -code b
coverage exclude -src SPI_slave.sv -line 71 -code b
coverage exclude -du SPI_slave_if -togglenode cs
coverage exclude -du SPI_slave_if -togglenode cs_ref
run -all
```

2.2.21 Bugs Found after slave verification

- **Bug 1:**
  At received_address high, ns = READ_DATA not READ_ADD.

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

Fix:

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

- **Bug 2:**
  counter isn’t reset to a known value at rst_n = 0.

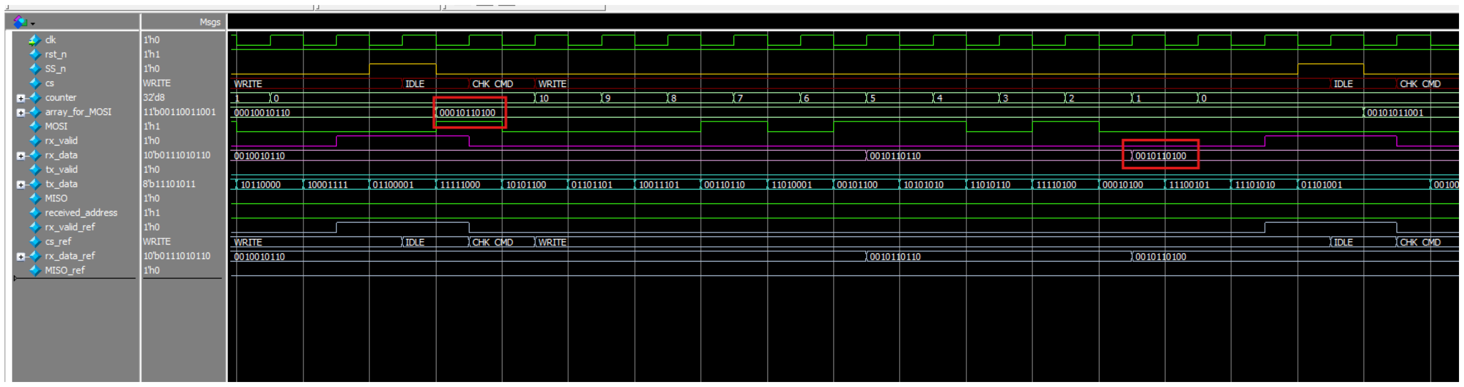
```
if (~sif.rst_n) begin
    sif.rx_data <= 0;
    sif.rx_valid <= 0;
    received_address <= 0;
    sif.MISO <= 0;
end
```

Fix:

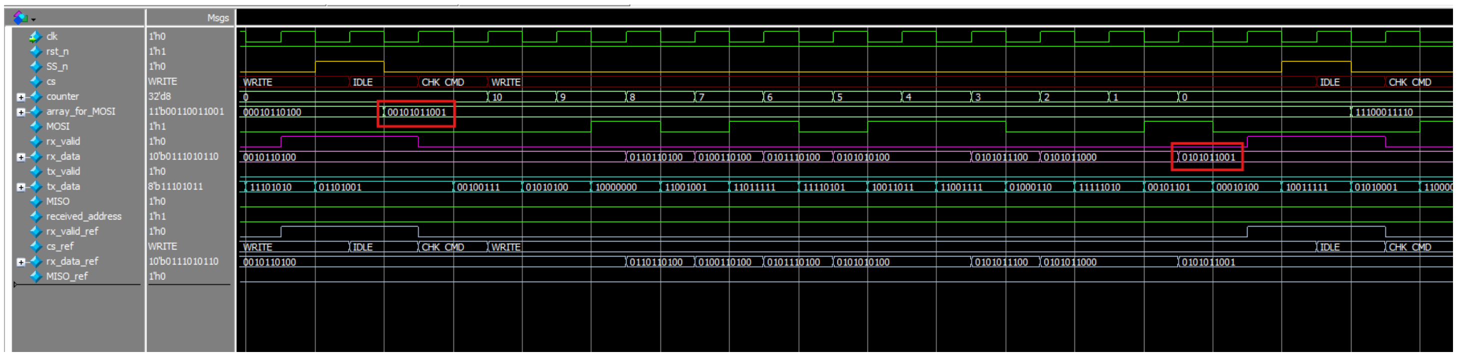
```
if (~sif.rst_n) begin
    sif.rx_data <= 0;
    sif.rx_valid <= 0;
    received_address <= 0;
    sif.MISO <= 0;
    counter <= 0; // bug: reset counter on reset
end
```

2.2.22 Simulation results

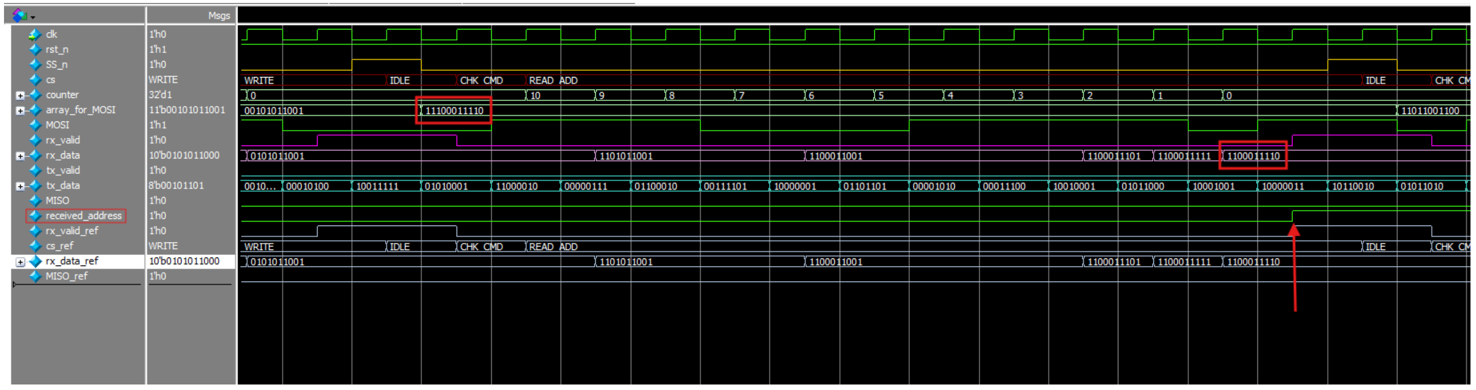
- Write Address



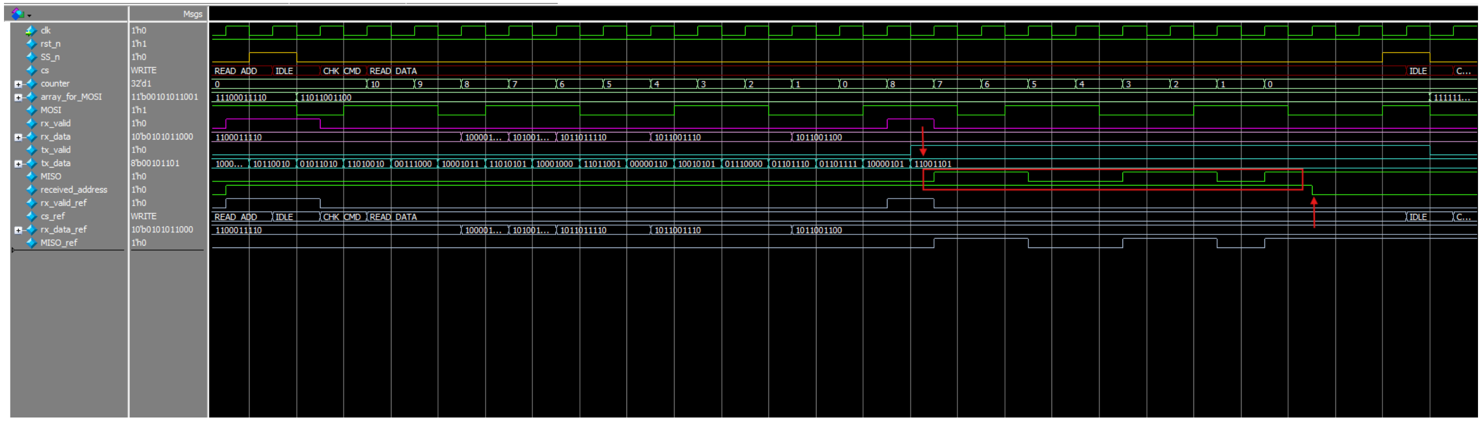
- Write Data

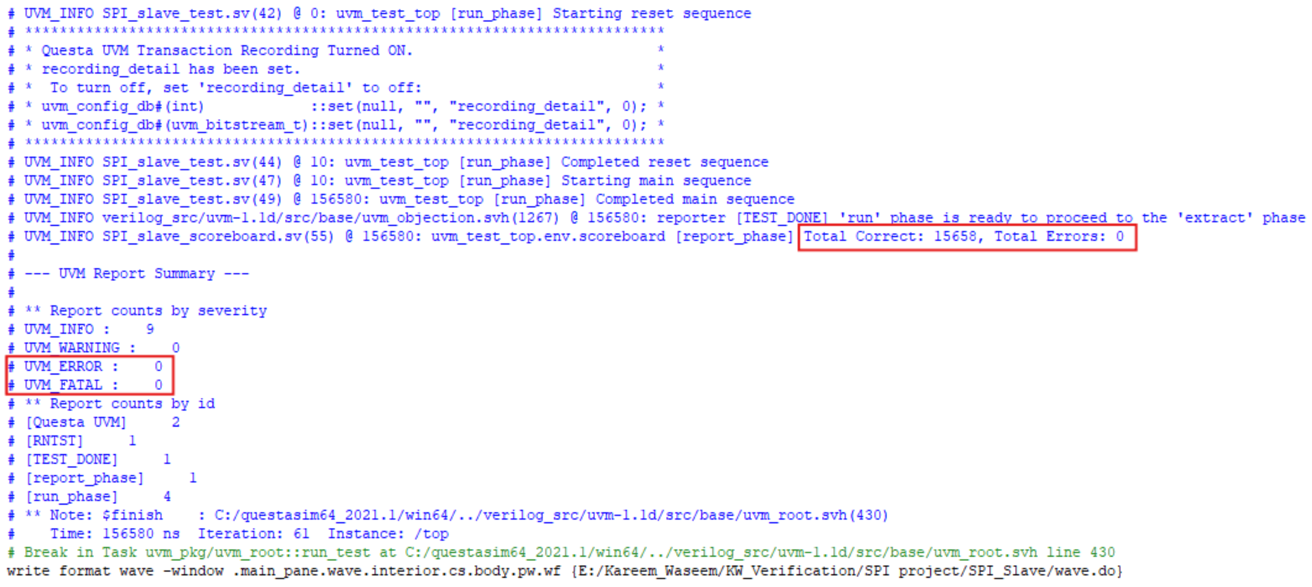


- Read Address



- Read Data

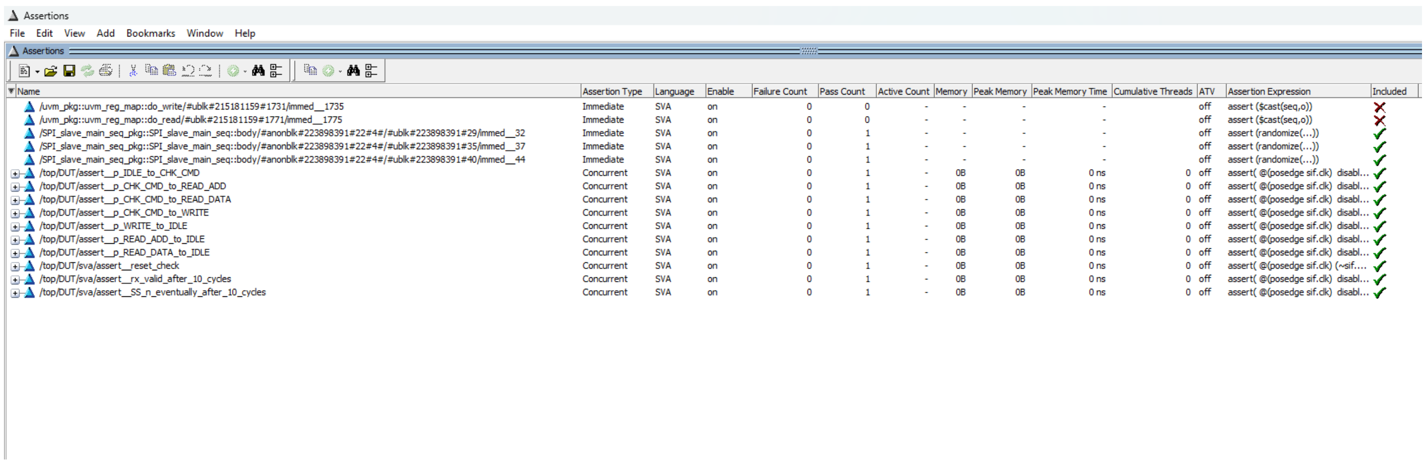




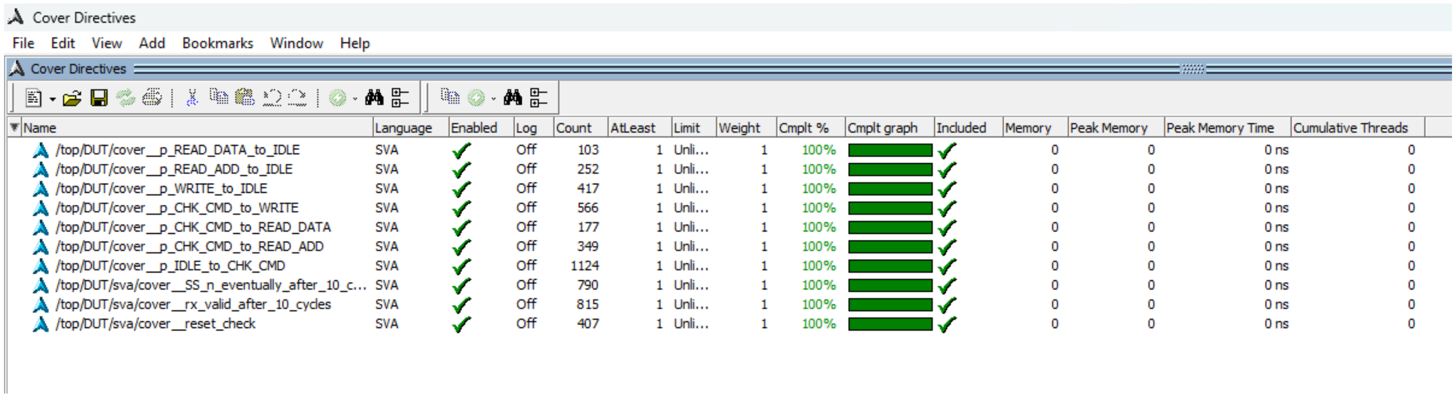
```
# UVM_INFO SPI_slave_test.sv(42) @ 0: uvm_test_top [run_phase] Starting reset sequence
# ************************************************************************
# * Questa UVM Transaction Recording Turned ON.                          *
# * recording_detail has been set.                                       *
# *  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(44) @ 10: uvm_test_top [run_phase] Completed reset sequence
# UVM_INFO SPI_slave_test.sv(47) @ 10: uvm_test_top [run_phase] Starting main sequence
# UVM_INFO SPI_slave_test.sv(49) @ 156580: uvm_test_top [run_phase] Completed main sequence
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 156580: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO SPI_slave_scoreboard.sv(55) @ 156580: uvm_test_top.env.scoreboard [report_phase] Total Correct: 15658, Total Errors: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :    9
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     1
# [run_phase]     4
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 156580 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
write format wave -window .main_pane.wave.interior.cs.body.pw.wf {E:/Kareem_Waseem/KW_Verification/SPI project/SPI_Slave/wave.do}
```

There are no errors

2.2.23 Assertions results



Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/SPI_slave_main_seq_pkg::SPI_slave_main_seq::body/#anonblk#223898391#22#4#/#ublk#223898391#29/immed__32	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/SPI_slave_main_seq_pkg::SPI_slave_main_seq::body/#anonblk#223898391#22#4#/#ublk#223898391#35/immed__37	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/SPI_slave_main_seq_pkg::SPI_slave_main_seq::body/#anonblk#223898391#22#4#/#ublk#223898391#40/immed__44	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/DUT/assert__p_IDLE_to_CHK_CMD	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/assert__p_CHK_CMD_to_READ_ADD	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/assert__p_CHK_CMD_to_READ_DATA	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/assert__p_CHK_CMD_to_WRITE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/assert__p_WRITE_to_IDLE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/assert__p_READ_ADD_to_IDLE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/assert__p_READ_DATA_to_IDLE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/sva/assert__reset_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) (~sif....	✓
/top/DUT/sva/assert__rx_valid_after_10_cycles	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/DUT/sva/assert__SS_n_eventually_after_10_cycles	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓

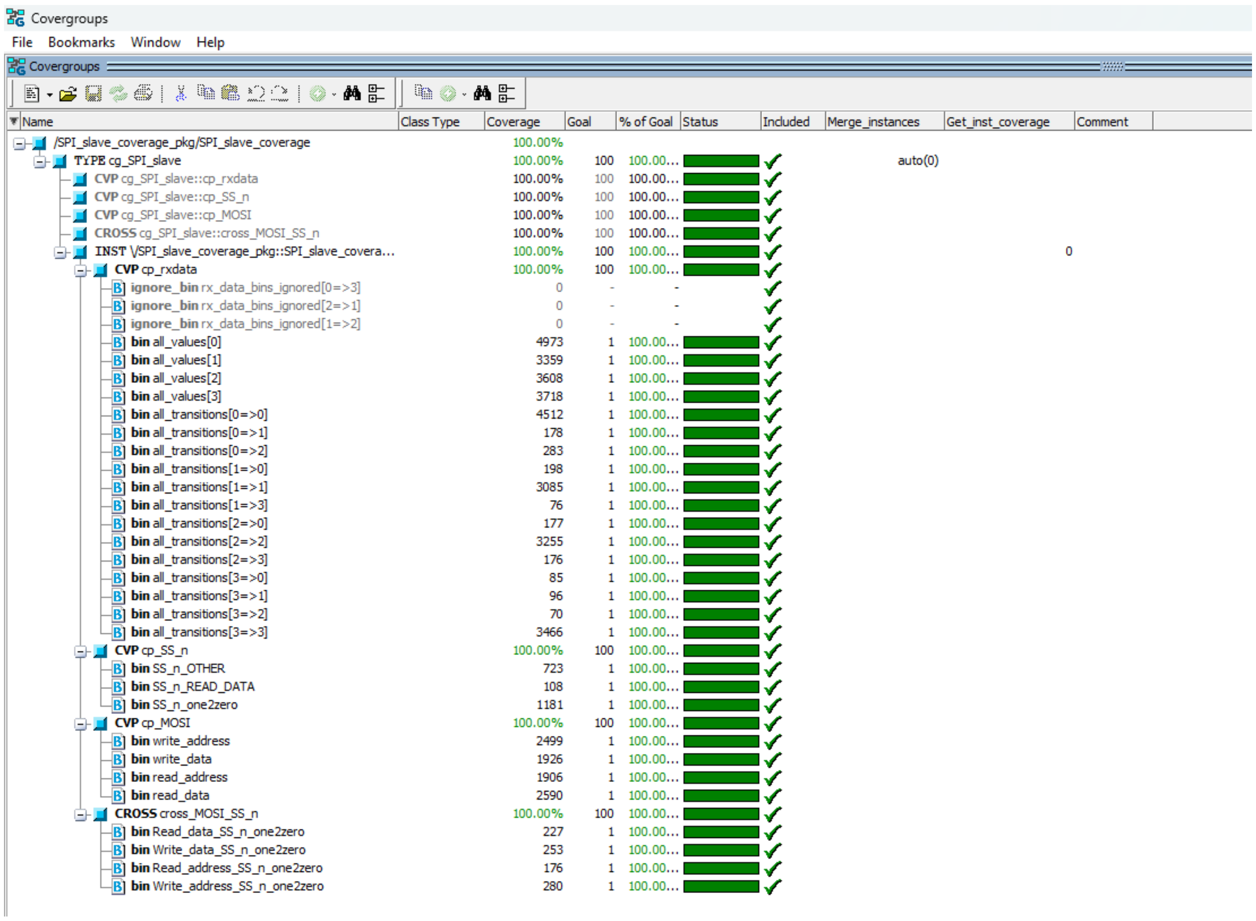


Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/cover__p_READ_DATA_to_IDLE	SVA	✓	Off	103	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/cover__p_READ_ADD_to_IDLE	SVA	✓	Off	252	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/cover__p_WRITE_to_IDLE	SVA	✓	Off	417	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/cover__p_CHK_CMD_to_WRITE	SVA	✓	Off	566	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/cover__p_CHK_CMD_to_READ_DATA	SVA	✓	Off	177	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/cover__p_CHK_CMD_to_READ_ADD	SVA	✓	Off	349	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/cover__p_IDLE_to_CHK_CMD	SVA	✓	Off	1124	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__SS_n_eventually_after_10_c...	SVA	✓	Off	790	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__rx_valid_after_10_cycles	SVA	✓	Off	815	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__reset_check	SVA	✓	Off	407	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0

There are no assertion errors

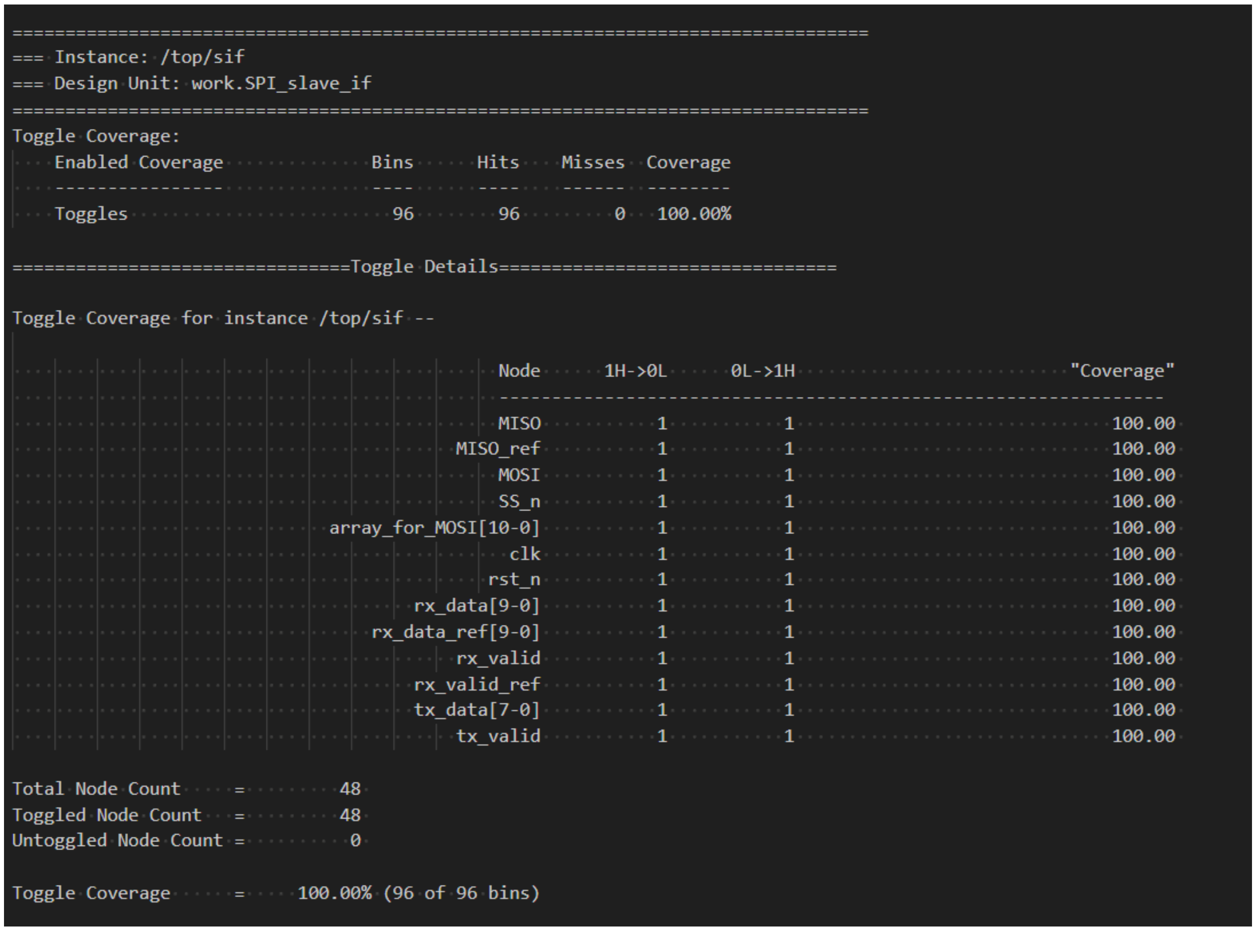
2.2.24 Coverage results

- Functional coverage



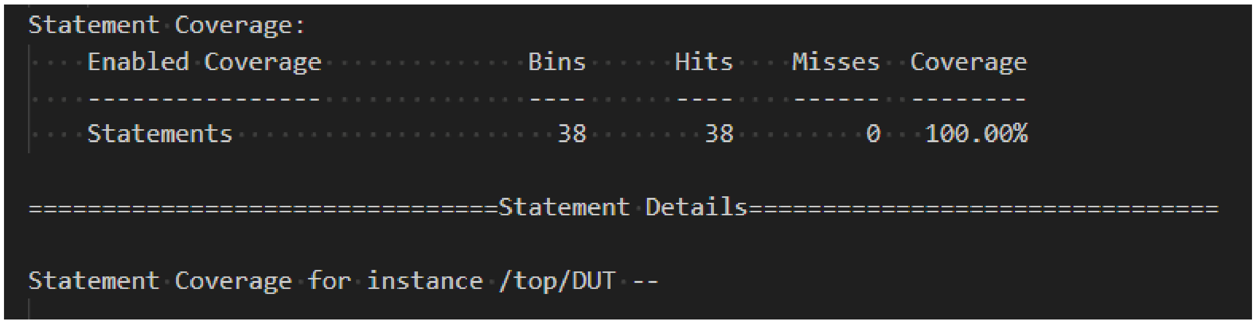
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/SPI_slave_coverage_pkg/SPI_slave_coverage		100.00%							
TYPE cg_SPI_slave		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP cg_SPI_slave::cp_rxdata		100.00%	100	100.00...	🟩	✓			
CVP cg_SPI_slave::cp_SS_n		100.00%	100	100.00...	🟩	✓			
CVP cg_SPI_slave::cp_MOSI		100.00%	100	100.00...	🟩	✓			
CROSS cg_SPI_slave::cross_MOSI_SS_n		100.00%	100	100.00...	🟩	✓			
INST \/SPI_slave_coverage_pkg::SPI_slave_covera...		100.00%	100	100.00...	🟩	✓		0	
CVP cp_rxdata		100.00%	100	100.00...	🟩	✓			
ignore_bin rx_data_bins_ignored[0=>3]		0	-	-		✓			
ignore_bin rx_data_bins_ignored[2=>1]		0	-	-		✓			
ignore_bin rx_data_bins_ignored[1=>2]		0	-	-		✓			
bin all_values[0]		4973	1	100.00...	🟩	✓			
bin all_values[1]		3359	1	100.00...	🟩	✓			
bin all_values[2]		3608	1	100.00...	🟩	✓			
bin all_values[3]		3718	1	100.00...	🟩	✓			
bin all_transitions[0=>0]		4512	1	100.00...	🟩	✓			
bin all_transitions[0=>1]		178	1	100.00...	🟩	✓			
bin all_transitions[0=>2]		283	1	100.00...	🟩	✓			
bin all_transitions[1=>0]		198	1	100.00...	🟩	✓			
bin all_transitions[1=>1]		3085	1	100.00...	🟩	✓			
bin all_transitions[1=>3]		76	1	100.00...	🟩	✓			
bin all_transitions[2=>0]		177	1	100.00...	🟩	✓			
bin all_transitions[2=>2]		3255	1	100.00...	🟩	✓			
bin all_transitions[2=>3]		176	1	100.00...	🟩	✓			
bin all_transitions[3=>0]		85	1	100.00...	🟩	✓			
bin all_transitions[3=>1]		96	1	100.00...	🟩	✓			
bin all_transitions[3=>2]		70	1	100.00...	🟩	✓			
bin all_transitions[3=>3]		3466	1	100.00...	🟩	✓			
CVP cp_SS_n		100.00%	100	100.00...	🟩	✓			
bin SS_n_OTHER		723	1	100.00...	🟩	✓			
bin SS_n_READ_DATA		108	1	100.00...	🟩	✓			
bin SS_n_one2zero		1181	1	100.00...	🟩	✓			
CVP cp_MOSI		100.00%	100	100.00...	🟩	✓			
bin write_address		2499	1	100.00...	🟩	✓			
bin write_data		1926	1	100.00...	🟩	✓			
bin read_address		1906	1	100.00...	🟩	✓			
bin read_data		2590	1	100.00...	🟩	✓			
CROSS cross_MOSI_SS_n		100.00%	100	100.00...	🟩	✓			
bin Read_data_SS_n_one2zero		227	1	100.00...	🟩	✓			
bin Write_data_SS_n_one2zero		253	1	100.00...	🟩	✓			
bin Read_address_SS_n_one2zero		176	1	100.00...	🟩	✓			
bin Write_address_SS_n_one2zero		280	1	100.00...	🟩	✓			

- Code coverage
    - Toggle Coverage



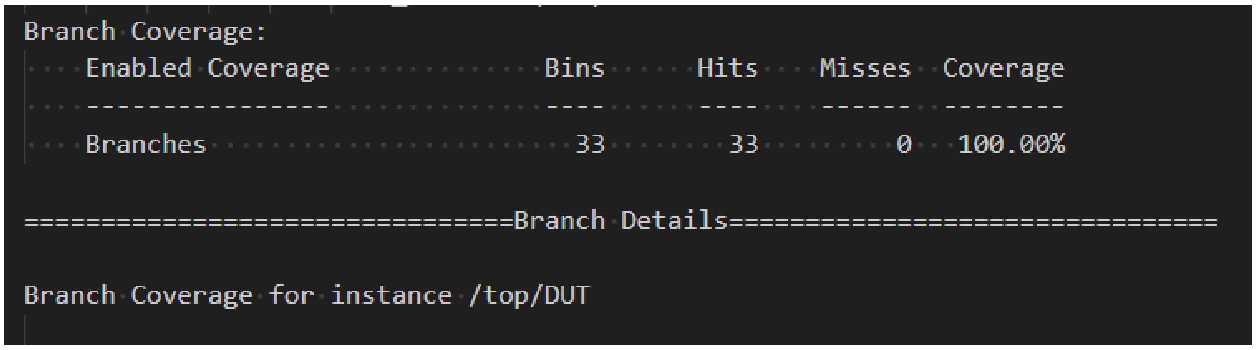
```
==================================================================================
=== Instance: /top/sif
=== Design Unit: work.SPI_slave_if
==================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         96        96         0   100.00%

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

Toggle Coverage for instance /top/sif --

                                              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
                              array_for_MOSI[10-0]           1           1                            100.00
                                               clk           1           1                            100.00
                                             rst_n           1           1                            100.00
                                      rx_data[9-0]           1           1                            100.00
                                  rx_data_ref[9-0]           1           1                            100.00
                                          rx_valid           1           1                            100.00
                                      rx_valid_ref           1           1                            100.00
                                      tx_data[7-0]           1           1                            100.00
                                          tx_valid           1           1                            100.00

Total Node Count     =         48
Toggled Node Count   =         48
Untoggled Node Count =          0

Toggle Coverage      =    100.00% (96 of 96 bins)
```

- Statement Coverage



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

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

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

- Branch coverage



```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        33        33         0   100.00%

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

Branch Coverage for instance /top/DUT
```

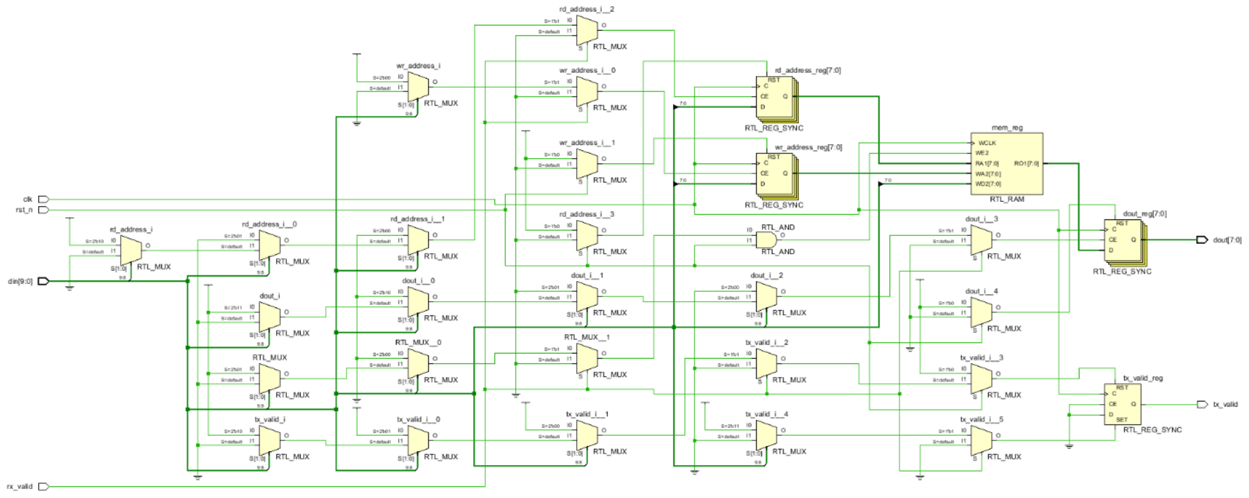
3 SPI ram

3.1 Design

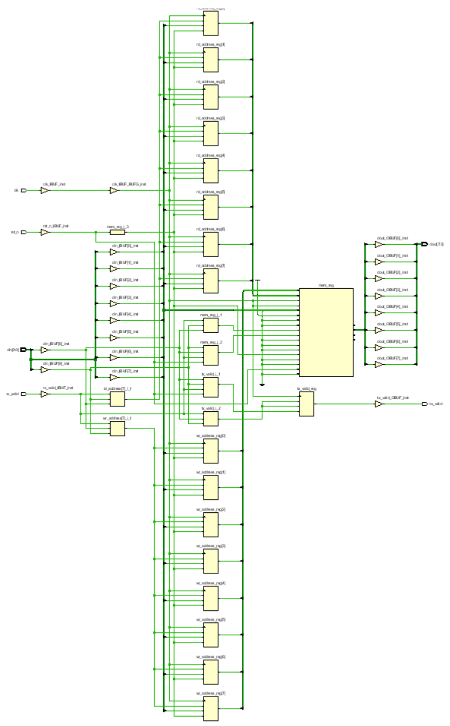
3.1.1 Verilog code

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

    reg [7:0] mem [255:0];
    reg [7:0] rd_address, wr_address;

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

endmodule
```

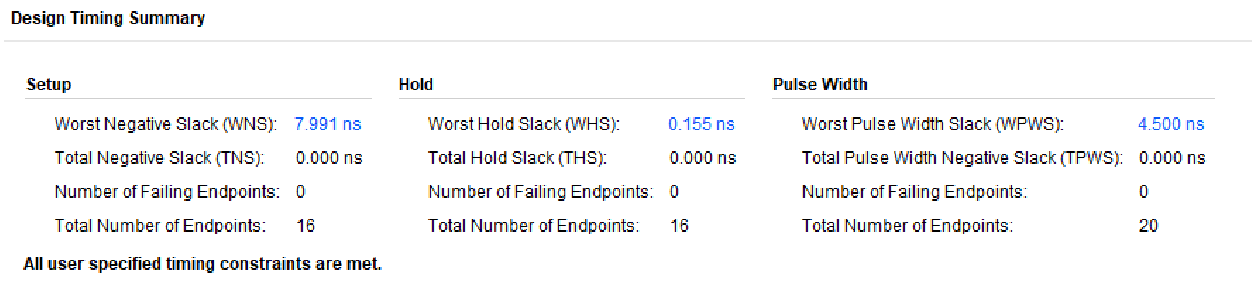
3.1.2 RTL elaboration



3.1.3 Synthesis schematic



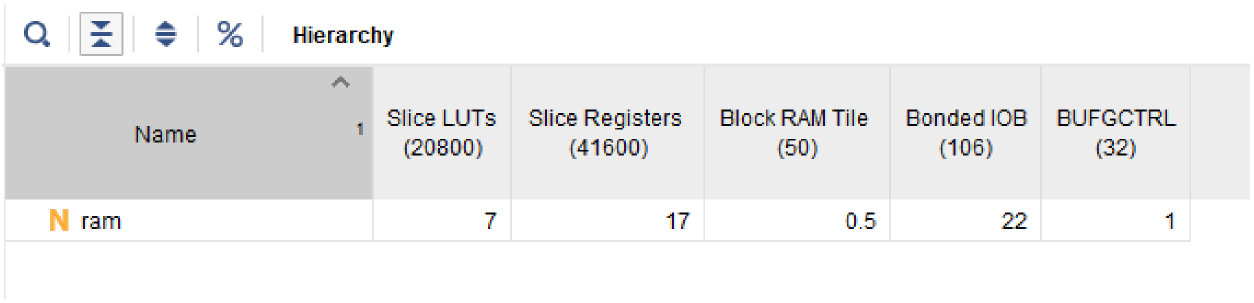
3.1.4 Static-timing analysis and Utilization reports after synthesis



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.991 ns	Worst Hold Slack (WHS):	0.155 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	16	Total Number of Endpoints:	16	Total Number of Endpoints:	20

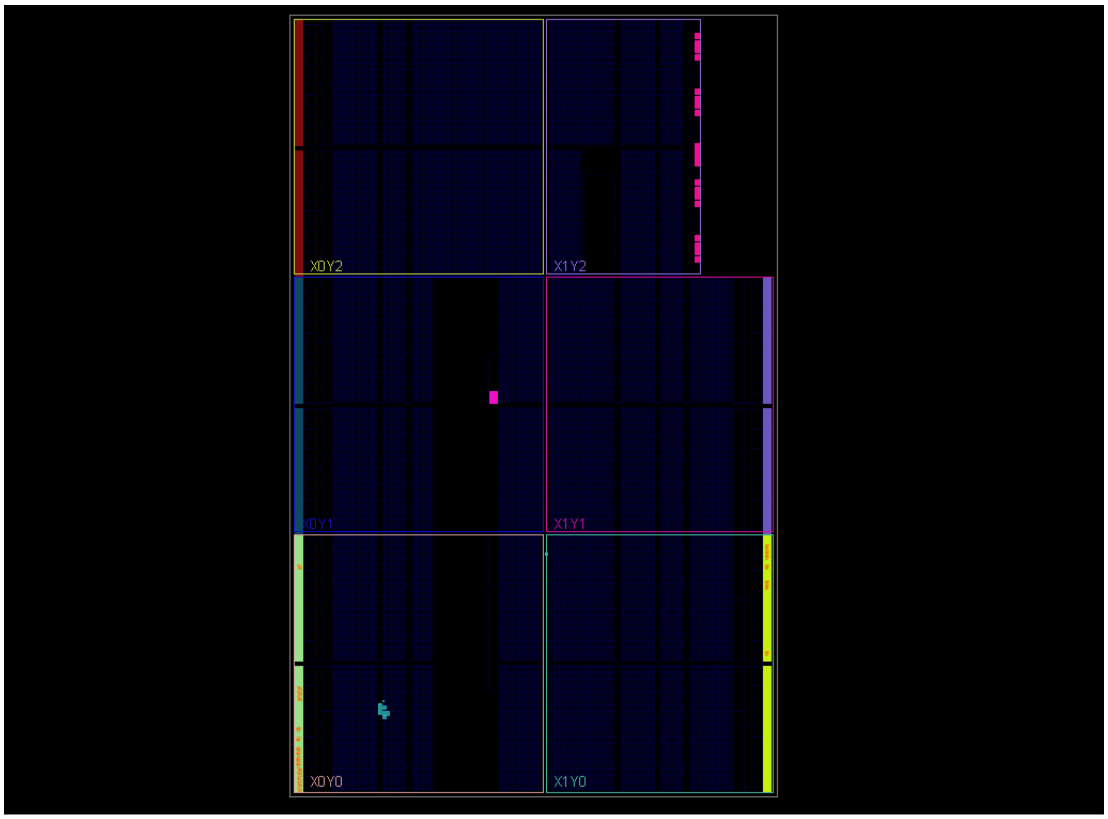
All user specified timing constraints are met.



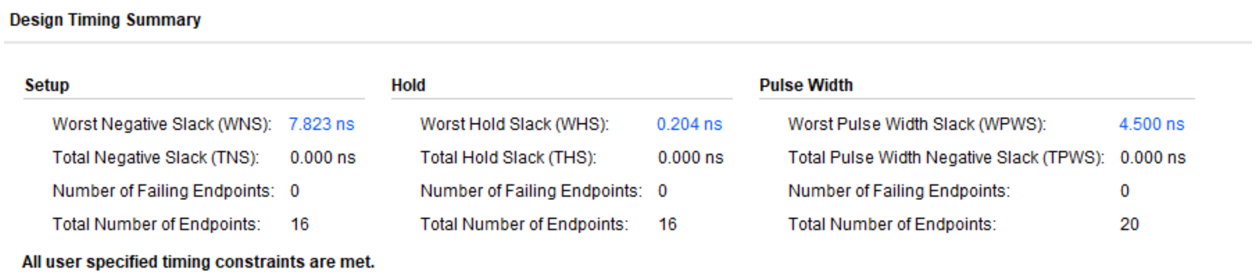
Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
N ram	7	17	0.5	22	1

3.1.5 Device Snippet



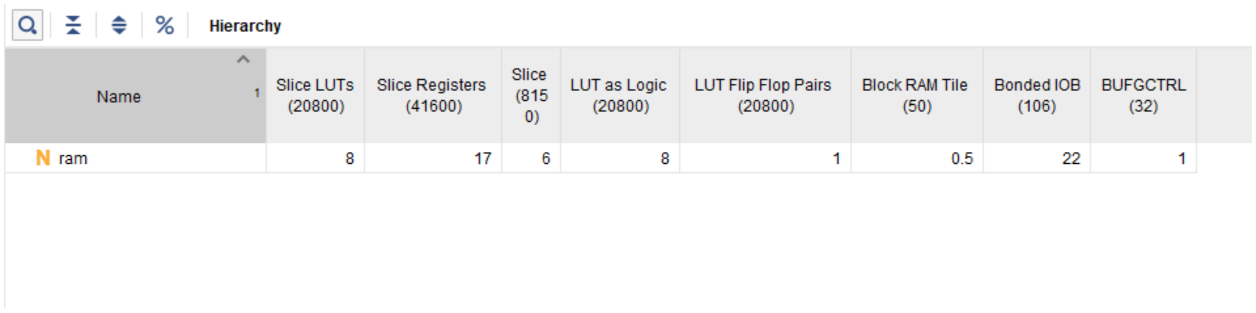
3.1.6 Static-timing analysis and Utilization reports after implementation



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.823 ns	Worst Hold Slack (WHS):	0.204 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	16	Total Number of Endpoints:	16	Total Number of Endpoints:	20

All user specified timing constraints are met.



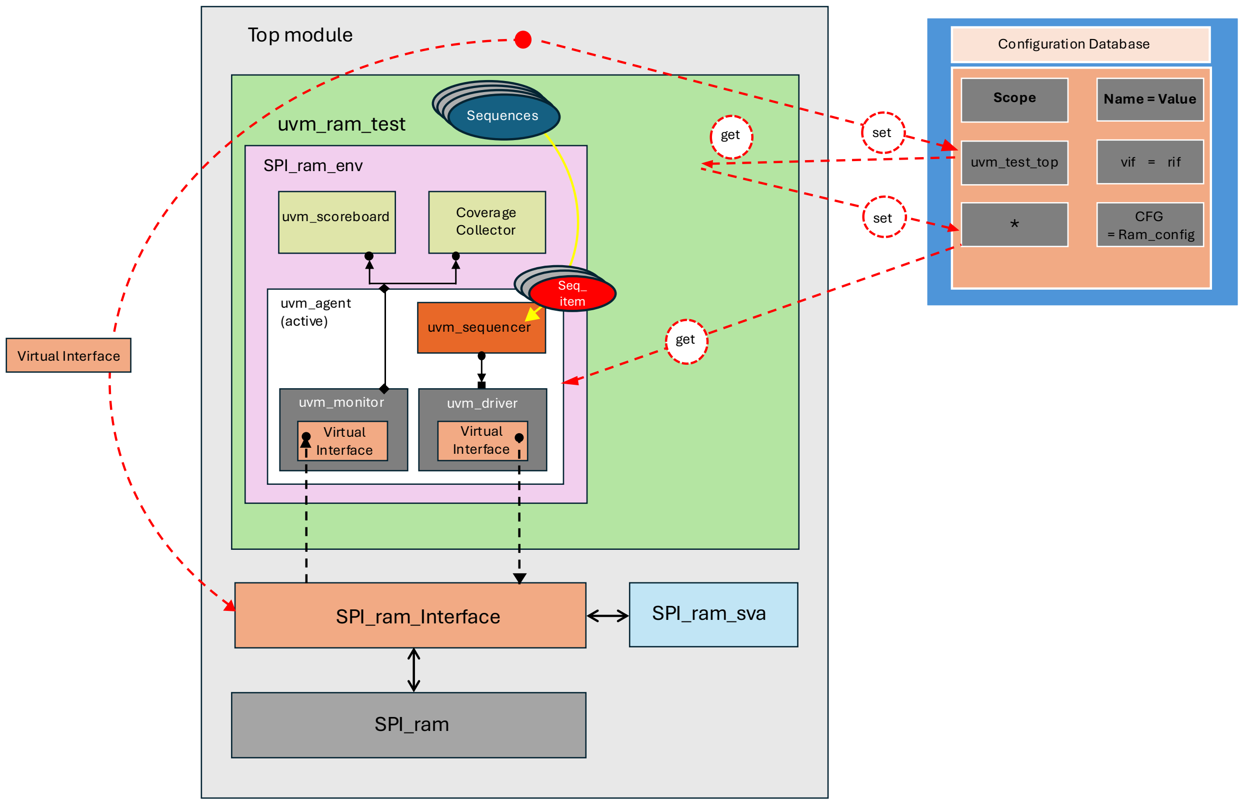
Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
N ram	8	17	6	8	1	0.5	22	1

3.2 Verification

- Verification will be performed on a buggy design, with the primary objective of identifying and reporting its bugs.
- This verification will be accomplished by comparing the design's output against a golden model “the design above”.

3.2.1 UVM Hierarchy



3.2.2 Buggy Design

```
module RAM (Ram_if.DUT rif);

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

reg [7:0] Rd_Addr, Wr_Addr;

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

3.2.3 UVM config_item

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

    class Ram_cfg extends uvm_object;
        `uvm_object_utils(Ram_cfg)
        virtual Ram_if Ram_vif;
        uvm_active_passive_enum is_active; // UVM_ACTIVE or UVM_PASSIVE

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

    endclass : Ram_cfg
endpackage
```

3.2.4 UVM sequence_item

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

    class Ram_seq_item extends uvm_sequence_item;
        `uvm_object_utils(Ram_seq_item)

        rand bit            rst_n, rx_valid;
        rand bit [9:0]      din;
        bit                 tx_valid, tx_valid_ref;
        bit      [7:0]      dout, dout_ref;
        bit      [1:0]      old_operation;

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

        function string convert2string();
            return $sformatf("%s rst_n=%b, din=0x%0h, tx_valid=%b, dout=0x%0h,
                             rx_valid=%b, dout_ref=0x%0h, tx_valid_ref=%b",
                             super.convert2string(), rst_n, din, tx_valid, dout,
                             rx_valid, dout_ref, tx_valid_ref);
        endfunction : convert2string

        function string convert2string_stimulus();
            return $sformatf("rst_n=%b, din=0x%0h, tx_valid=%b",
                             rst_n, din, tx_valid);
        endfunction : convert2string_stimulus

        constraint c_rst_n    { rst_n dist {0 := 1, 1 := 9}; }

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

        constraint c_din_write_only {
                din[9:8] inside {WRITE_DATA, WRITE_ADDR};
            }

        constraint c_din_read_only {
                din[9:8] inside {READ_DATA, READ_ADDR};

                if (old_operation == READ_ADDR)
                    din[9:8] == READ_DATA;
                else
                    din[9:8] == READ_ADDR;
            }
        constraint c_din_read_write {
            if      (old_operation == WRITE_ADDR)
                din[9:8] inside {WRITE_DATA, WRITE_ADDR};

            else if (old_operation == WRITE_DATA)
                din[9:8] dist {READ_ADDR := 60, WRITE_ADDR := 40};

            else if (old_operation == READ_ADDR)
                din[9:8] == READ_DATA;

            else
                din[9:8] dist {WRITE_ADDR := 60, READ_ADDR := 40};
            }

        function void post_randomize();
            old_operation = din[9:8];
        endfunction : post_randomize

    endclass : Ram_seq_item
endpackage
```

3.2.5 UVM reset_sequence

```
package Ram_reset_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_seq_item_pkg::*;
    import Ram_sequencer_pkg::*;
    import Ram_shared_pkg::*;

    class Ram_reset_seq extends uvm_sequence #(Ram_seq_item);
        `uvm_object_utils(Ram_reset_seq)
        Ram_seq_item item;

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

        task body();
            item = Ram_seq_item::type_id::create("item");
            start_item(item);
            item.rst_n = 0;
            item.din = '0;
            item.rx_valid = 0;
            finish_item(item);
        endtask : body
    endclass : Ram_reset_seq

endpackage
```

3.2.6 UVM write_only_sequence

```
package Ram_write_only_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_seq_item_pkg::*;
    import Ram_sequencer_pkg::*;
    import Ram_shared_pkg::*;

    class Ram_write_only_seq extends uvm_sequence #(Ram_seq_item);
        `uvm_object_utils(Ram_write_only_seq)
        Ram_seq_item item;

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

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

            repeat (1000) begin
                start_item(item);
                item.constraint_mode(1);
                item.c_din_read_only.constraint_mode(0);
                item.c_din_read_write.constraint_mode(0);
                assert(item.randomize());
                finish_item(item);
            end
        endtask : body
    endclass : Ram_write_only_seq

endpackage
```

3.2.7 UVM read_only_sequence

```
package Ram_read_only_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_seq_item_pkg::*;
    import Ram_sequencer_pkg::*;
    import Ram_shared_pkg::*;

    class Ram_read_only_seq extends uvm_sequence #(Ram_seq_item);
        `uvm_object_utils(Ram_read_only_seq)
        Ram_seq_item item;

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

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

            repeat (1000) begin
                start_item(item);
                item.constraint_mode(1);
                item.c_din_write_only.constraint_mode(0);
                item.c_din_read_write.constraint_mode(0);
                assert(item.randomize());
                finish_item(item);
            end
        endtask : body
    endclass : Ram_read_only_seq

endpackage
```

3.2.8 UVM write_read_sequence

```
package Ram_write_read_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_seq_item_pkg::*;
    import Ram_sequencer_pkg::*;
    import Ram_shared_pkg::*;

    class Ram_write_read_seq extends uvm_sequence #(Ram_seq_item);
        `uvm_object_utils(Ram_write_read_seq)
        Ram_seq_item item;

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

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

            repeat (1000) begin
                start_item(item);
                item.constraint_mode(1);
                item.c_din_write_only.constraint_mode(0);
                item.c_din_read_only.constraint_mode(0);
                assert(item.randomize());
                finish_item(item);
            end
        endtask : body
    endclass : Ram_write_read_seq

endpackage
```

3.2.9 UVM sequencer

```
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, uvm_component parent);
            super.new(name, parent);
        endfunction : new

    endclass : Ram_sequencer

endpackage
```

3.2.10 UVM driver

```
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)

        virtual Ram_if driver_vif;
        Ram_seq_item stim_seq_item;

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                stim_seq_item = Ram_seq_item::type_id::create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);

                driver_vif.rst_n    = stim_seq_item.rst_n;
                driver_vif.rx_valid = stim_seq_item.rx_valid;
                driver_vif.din      = stim_seq_item.din;
                @(negedge driver_vif.clk);

                seq_item_port.item_done();
                `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask : run_phase
    endclass: Ram_driver
endpackage
```

3.2.11 UVM monitor

```
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 monitor_vif;
        Ram_seq_item rsp_seq_item;
        uvm_analysis_port #(Ram_seq_item) monitor_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = Ram_seq_item::type_id::create("rsp_seq_item");

                @(negedge monitor_vif.clk);
                rsp_seq_item.rst_n        = monitor_vif.rst_n;
                rsp_seq_item.rx_valid     = monitor_vif.rx_valid;
                rsp_seq_item.din          = monitor_vif.din;
                rsp_seq_item.tx_valid     = monitor_vif.tx_valid;
                rsp_seq_item.tx_valid_ref = monitor_vif.tx_valid_ref;
                rsp_seq_item.dout         = monitor_vif.dout;
                rsp_seq_item.dout_ref     = monitor_vif.dout_ref;

                monitor_ap.write(rsp_seq_item);
                `uvm_info("run_phase", rsp_seq_item.convert2string(), UVM_HIGH)

            end
        endtask : run_phase
    endclass: Ram_monitor

endpackage
```

3.2.12 UVM agent

```
package Ram_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_driver_pkg::*;
    import Ram_monitor_pkg::*;
    import Ram_sequencer_pkg::*;
    import Ram_config_pkg::*;
    import Ram_seq_item_pkg::*;

    class Ram_agent extends uvm_agent;
        `uvm_component_utils(Ram_agent)
        Ram_driver driver;
        Ram_monitor monitor;
        Ram_sequencer sequencer;
        Ram_cfg Ram_config;
        uvm_analysis_port #(Ram_seq_item) agent_ap;

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

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

            if (!uvm_config_db#(Ram_cfg)::get(this, "", "CFG", Ram_config)) begin
                `uvm_fatal("NOVIF", "Cannot get cfg from uvm_config_db")
            end

             // only build driver and sequencer if agent is active
            if (Ram_config.is_active == UVM_ACTIVE) begin
                driver      = Ram_driver::type_id::create("driver", this);
                sequencer   = Ram_sequencer::type_id::create("sequencer", this);
            end

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

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            // only connect driver and sequencer if agent is active
            if (Ram_config.is_active == UVM_ACTIVE) begin
                driver.seq_item_port.connect(sequencer.seq_item_export);
                driver.driver_vif = Ram_config.Ram_vif;
            end
            monitor.monitor_ap.connect(agent_ap);
            monitor.monitor_vif = Ram_config.Ram_vif;
        endfunction : connect_phase

    endclass: Ram_agent
endpackage
```

3.2.13 UVM scoreboard

```
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)
        uvm_analysis_export #(Ram_seq_item) sb_export;
        uvm_tlm_analysis_fifo #(Ram_seq_item) sb_fifo;
        Ram_seq_item seq_item_sb;

        int error_count =0; int correct_count=0;

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

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

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

        task run_phase(uvm_phase phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                if ((seq_item_sb.dout !== seq_item_sb.dout_ref) ||
                   (seq_item_sb.tx_valid !== seq_item_sb.tx_valid_ref))
                begin
                    if (seq_item_sb.dout !== seq_item_sb.dout_ref)
                        `uvm_error("run_phase", $sformatf("Dout Mismatch:
                                    dout=%0d, dout_ref=%0d, %s",
                                    seq_item_sb.dout, seq_item_sb.dout_ref,
                                    seq_item_sb.convert2string_stimulus()))
                    else if (seq_item_sb.tx_valid !== seq_item_sb.tx_valid_ref)
                        `uvm_error("run_phase", $sformatf("Tx_valid Mismatch:
                                    tx_valid=%0b, tx_valid_ref=%0b, %s",
                                    seq_item_sb.tx_valid, seq_item_sb.tx_valid_ref,
                                    seq_item_sb.convert2string_stimulus()))

                    error_count++;
                end
                else begin
                    `uvm_info("run_phase", $sformatf("Match: dout=%0d, tx_valid=%0b",
                        seq_item_sb.dout, seq_item_sb.tx_valid), UVM_HIGH)

                    correct_count++;
                end
            end
        endtask
        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Total Correct: %0d, Total Errors: %0d",
                                                 correct_count, error_count), UVM_MEDIUM)
        endfunction
    endclass: Ram_scoreboard

endpackage
```

3.2.14 UVM coverage

```
package Ram_coverage_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_seq_item_pkg::*;
    import Ram_shared_pkg::*;

    class Ram_coverage extends uvm_component;
        `uvm_component_utils(Ram_coverage)
        uvm_analysis_export #(Ram_seq_item) cov_export;
        uvm_tlm_analysis_fifo #(Ram_seq_item) cov_fifo;
        Ram_seq_item cov_item;

        // Functional Coverage
        covergroup cg_Ram;
            cp_din: coverpoint cov_item.din[9:8] {
                bins all_values[]      = {[0:3]};
                bins read_data         = {READ_DATA};
                bins wr_addr_2_wr_data = (WRITE_ADDR => WRITE_DATA);
                bins rd_addr_2_rd_addr = (READ_ADDR => READ_DATA);
                bins transition = (WRITE_ADDR => WRITE_DATA => READ_ADDR => READ_DATA);
            }
            cp_rx_valid: coverpoint cov_item.rx_valid {
                bins rx_valid_0 = {0};
                bins rx_valid_1 = {1};
            }
            cp_tx_valid: coverpoint cov_item.tx_valid {
                bins tx_valid_0 = {0};
                bins tx_valid_1 = {1};
            }
            cross_din_rx_valid: cross cp_din, cp_rx_valid {
                option.cross_auto_bin_max = 0;
                bins din_all_values_rx_valid_1 =
                     binsof (cp_din.all_values) && binsof (cp_rx_valid.rx_valid_1);
            }

            cross_din_tx_valid: cross cp_din, cp_tx_valid {
                option.cross_auto_bin_max = 0;
                bins din_read_data_tx_valid_1 =
                     binsof (cp_din.read_data) && binsof (cp_tx_valid.tx_valid_1);
            }

        endgroup : cg_Ram
        function new(string name = "Ram_coverage", uvm_component parent = null);
            super.new(name, parent);
            cg_Ram = 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 : build_phase

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(cov_item);
                cg_Ram.sample();
            end
        endtask : run_phase

    endclass: Ram_coverage
endpackage
```

3.2.15 UVM environment

```
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 agent;
        Ram_scoreboard scoreboard;
        Ram_coverage coverage;

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

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

            agent       = Ram_agent::type_id::create("agent", this);
            scoreboard  = Ram_scoreboard::type_id::create("scoreboard", this);
            coverage    = Ram_coverage::type_id::create("coverage", this);
        endfunction : build_phase
        virtual function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agent.agent_ap.connect(scoreboard.sb_export);
            agent.agent_ap.connect(coverage.cov_export);
        endfunction : connect_phase
    endclass: Ram_env
endpackage
```

3.2.16 UVM test

```
package Ram_test_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Ram_env_pkg::*;
    import Ram_config_pkg::*;
    import Ram_read_only_seq_pkg::*;
    import Ram_write_read_seq_pkg::*;
    import Ram_write_only_seq_pkg::*;
    import Ram_reset_seq_pkg::*;

    class Ram_test extends uvm_test;
        `uvm_component_utils(Ram_test)

        Ram_env env;
        Ram_read_only_seq read_only_sequence;
        Ram_write_read_seq write_read_sequence;
        Ram_write_only_seq write_only_sequence;
        Ram_reset_seq reset_sequence;
        Ram_cfg Ram_config;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            env = Ram_env::type_id::create("env", this);
            read_only_sequence = Ram_read_only_seq::type_id::create("read_only_sequence");

            write_read_sequence =
                                   Ram_write_read_seq::type_id::create("write_read_sequence");
            write_only_sequence =
                                   Ram_write_only_seq::type_id::create("write_only_sequence");
            reset_sequence      = Ram_reset_seq::type_id::create("reset_sequence");

            Ram_config = Ram_cfg::type_id::create("Ram_config");

            if(!uvm_config_db#(virtual Ram_if)::get(this, "", "vif", Ram_config.Ram_vif))
                `uvm_fatal("NOVIF", "Virtual interface must be set for:")

            Ram_config.is_active = UVM_ACTIVE; // Set agent to active mode

            uvm_config_db#(Ram_cfg)::set(this, "*", "CFG", Ram_config);

        endfunction : build_phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            phase.raise_objection(this);

            // Start with reset sequence
            `uvm_info("run_phase", "Starting reset sequence", UVM_LOW)
            reset_sequence.start(env.agent.sequencer);
            `uvm_info("run_phase", "Completed reset sequence", UVM_LOW)

            // Then run main sequences

            `uvm_info("run_phase", "Starting write only sequence", UVM_LOW)
            write_only_sequence.start(env.agent.sequencer);
            `uvm_info("run_phase", "Completed write only sequence", UVM_LOW)

            `uvm_info("run_phase", "Starting read only sequence", UVM_LOW)
            read_only_sequence.start(env.agent.sequencer);
            `uvm_info("run_phase", "Completed read only sequence", UVM_LOW)

            `uvm_info("run_phase", "Starting write read sequence", UVM_LOW)
            write_read_sequence.start(env.agent.sequencer);
            `uvm_info("run_phase", "Completed write read sequence", UVM_LOW)

            phase.drop_objection(this);
        endtask : run_phase
    endclass: Ram_test
endpackage
```

3.2.17 Slave interface

```
interface Ram_if (clk);
import Ram_shared_pkg::*;
    input logic            clk;
          logic            rst_n, rx_valid, tx_valid, tx_valid_ref;
          logic            [9:0] din;
          logic            [7:0] dout, dout_ref;

    modport DUT (
        input clk, rst_n, rx_valid, din,
        output dout, tx_valid
    );

    modport Golden (
        input clk, rst_n, rx_valid, din,
        output dout_ref, tx_valid_ref
    );
Endinterface
```

3.2.18 Slave shared package

```
package Ram_shared_pkg;

    parameter WRITE_ADDR = 2'b00;
    parameter WRITE_DATA = 2'b01;
    parameter READ_ADDR  = 2'b10;
    parameter READ_DATA  = 2'b11;

endpackage : Ram_shared_pkg
```

3.2.19 Ram assertions

```
module Ram_sva(Ram_if.DUT rif);
    import Ram_shared_pkg::*;

    //Sequences
    sequence write_addr_seq;
            (rif.rx_valid && (rif.din[9:8] == WRITE_ADDR));
    endsequence

    sequence write_data_seq;
            (rif.rx_valid && (rif.din[9:8] == WRITE_DATA));
    endsequence

    sequence read_addr_seq;
            (rif.rx_valid && (rif.din[9:8] == READ_ADDR));
    endsequence

    sequence read_data_seq;
            (rif.rx_valid && (rif.din[9:8] == READ_DATA));
    endsequence

    //Properties
    property reset;
        @(posedge rif.clk) !rif.rst_n |=> (!rif.dout) && (!rif.tx_valid);
    endproperty

    property read_data_tx_valid;
        @(posedge rif.clk) disable iff (!rif.rst_n)
            read_data_seq |=> (rif.tx_valid) |-> ##[1:$] (!rif.tx_valid);
    endproperty

    property other_tx_valid;
        @(posedge rif.clk) disable iff (!rif.rst_n)
            (write_addr_seq or write_data_seq or read_addr_seq) |=> (!rif.tx_valid);
    endproperty

    property write_addr_write_data;
        @(posedge rif.clk) disable iff (!rif.rst_n)
            write_addr_seq |-> ##[1:$] write_data_seq;
    endproperty
    property read_addr_read_data;
        @(posedge rif.clk) disable iff (!rif.rst_n)
            read_addr_seq |-> ##[1:$] read_data_seq;
    endproperty

    //Assertions
    assert property (reset)
    else $error ("Reset failed: dout and tx_valid should be zero after reset");
    assert property (read_data_tx_valid)
    else $error ("Read data tx_valid failed: tx_valid should be set after read data
                  command and cleared afterwards");
    assert property (other_tx_valid)
    else $error ("Other operations tx_valid failed: tx_valid should be zero after other
                  commands");
    assert property (write_addr_write_data)
    else $error ("Write addr-data sequence failed: write data command did not follow write
                  address command");
    assert property (read_addr_read_data)
    else $error ("Read addr-data sequence failed: read data command did not follow read
                  address command");

    // Coverage
    cover property (reset);
    cover property (read_data_tx_valid);
    cover property (other_tx_valid);
    cover property (write_addr_write_data);
    cover property (read_addr_read_data);
endmodule : Ram_sva
```

3.2.20 Ram Golden model

```
module Golden_ram (Ram_if.Golden rif);

    reg [7:0] mem [255:0];
    reg [7:0] rd_address, wr_address;

    always @(posedge rif.clk) begin
        if (!rif.rst_n) begin
            rif.dout_ref <= 0;
            rif.tx_valid_ref <= 0;
            rd_address <= 0;
            wr_address <= 0;
        end
        else if (rif.rx_valid) begin
            if (rif.din[9:8] == 2'b00) begin
                wr_address <= rif.din [7:0];
                rif.tx_valid_ref <= 0;
            end
            else if (rif.din[9:8] == 2'b01) begin
                mem[wr_address] <= rif.din [7:0];
                rif.tx_valid_ref <= 0;
            end
            else if (rif.din[9:8] == 2'b10) begin
                rd_address <= rif.din [7:0];
                rif.tx_valid_ref <= 0;
            end
            else if (rif.din[9:8] == 2'b11) begin
                rif.dout_ref <= mem[rd_address];
                rif.tx_valid_ref <= 1;
            end
        end
    end

endmodule
```

3.2.21 UVM top

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

module top ();
    bit clk;

    always #5 clk = ~clk;

    Ram_if rif(clk);

    RAM DUT (rif);

    Golden_ram golden_model (rif);

    bind RAM Ram_sva sva (rif);

    initial begin
        uvm_config_db#(virtual Ram_if)::set(null, "uvm_test_top", "vif", rif);
        run_test("Ram_test");
    end
endmodule
```

3.2.22 Run.do

```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover
do wave.do
coverage save ram_top.ucdb -onexit
coverage exclude -src RAM.sv -line 21 -code s
coverage exclude -src RAM.sv -line 21 -code b
run -all
```

3.2.23 Bugs Found after ram verification

- **Bug 1:**
  Ram reads data using write address not read address

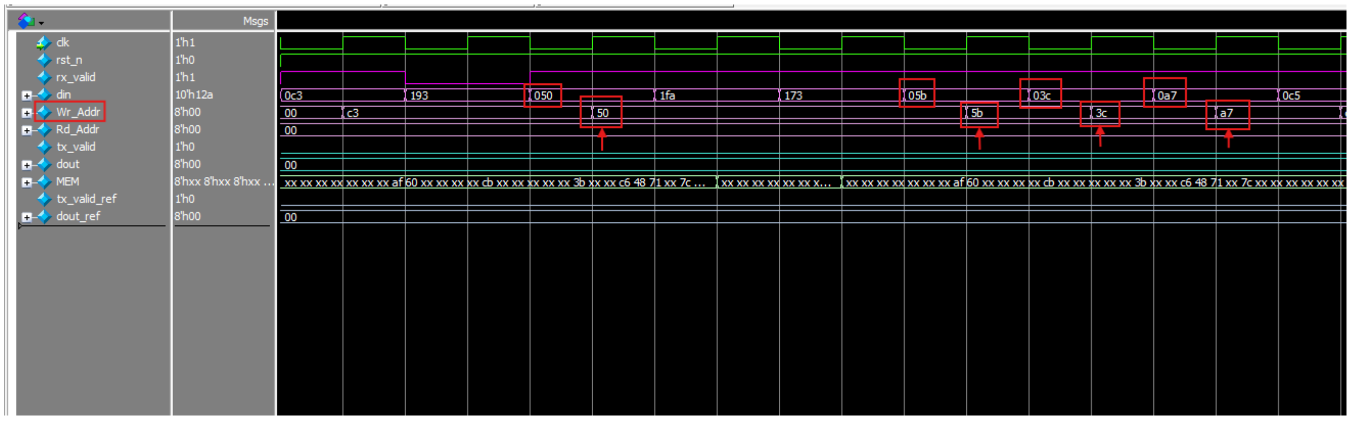
```
case (rif.din[9:8])
     2'b00   : Wr_Addr       <= rif.din[7:0];
     2'b01   : MEM[Wr_Addr]  <= rif.din[7:0];
     2'b10   : Rd_Addr       <= rif.din[7:0];
     2'b11   : rif.dout      <= MEM[Wr_Addr];
     default : rif.dout      <= 0;
 endcase
```

Fix:

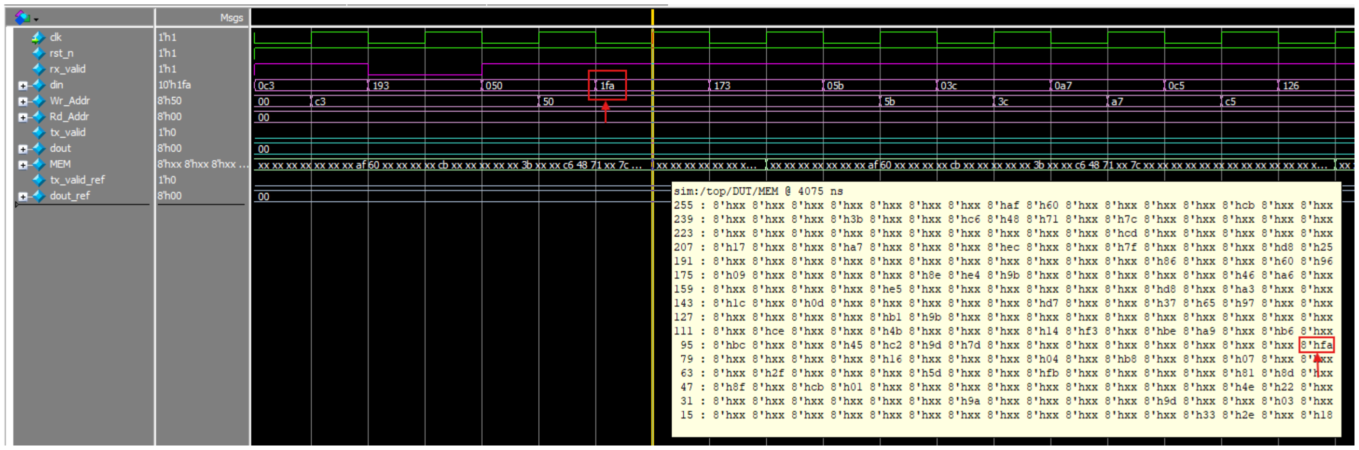
```
case (rif.din[9:8])
    2'b00   : Wr_Addr       <= rif.din[7:0];
    2'b01   : MEM[Wr_Addr]  <= rif.din[7:0];
    2'b10   : Rd_Addr       <= rif.din[7:0];
    2'b11   : rif.dout      <= MEM[Rd_Addr]; // bug: Wr_Addr ==> Rd_Addr
    default : rif.dout      <= 0;
endcase
```

3.2.24 Simulation results

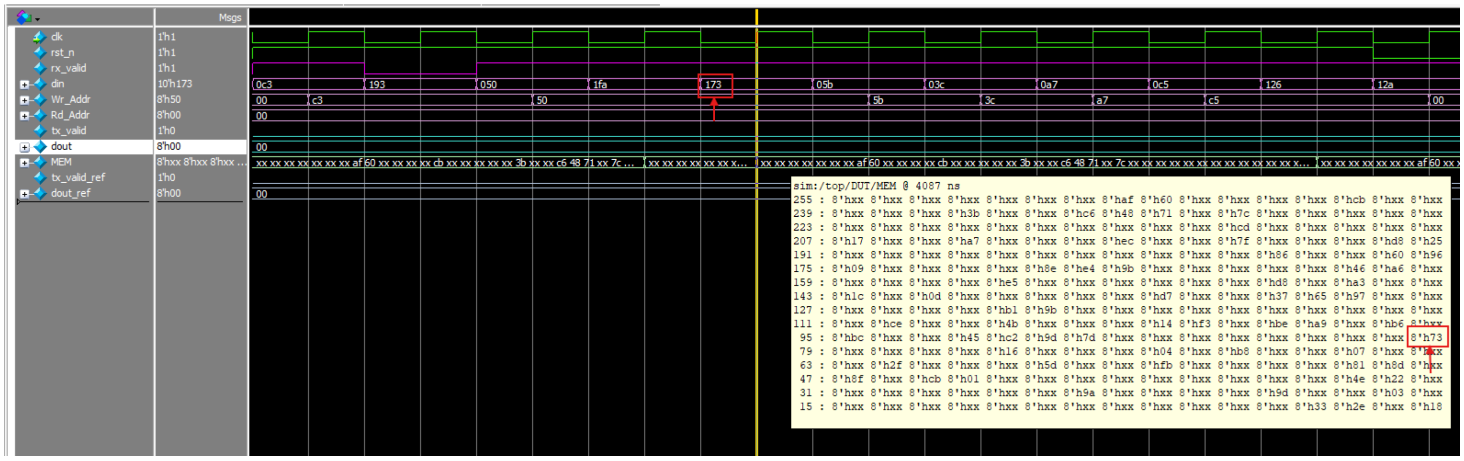
- Write Address



- Write Data

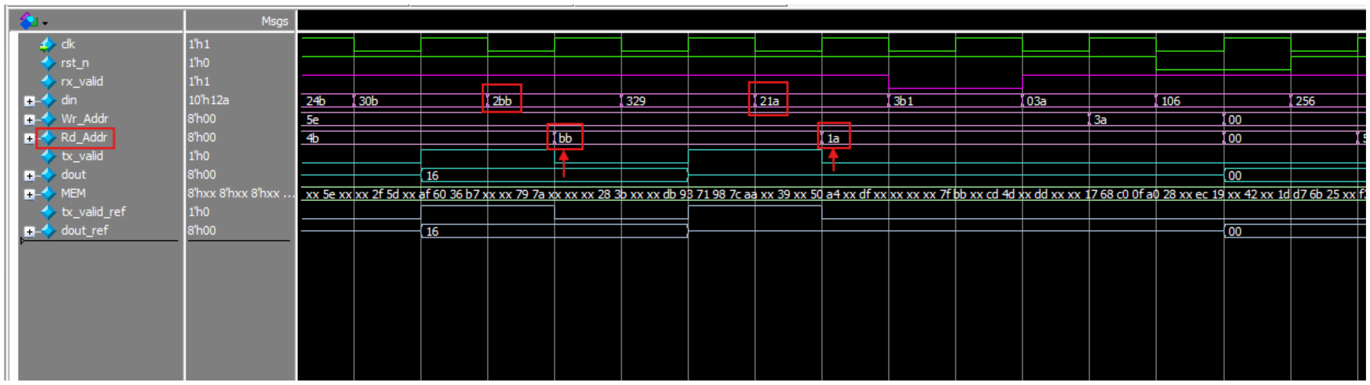


```
sim:/top/DUT/MEM @ 4075 ns
255 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'haf 8'h60 8'hxx 8'hxx 8'hxx 8'hxx 8'hcb 8'hxx 8'hxx
239 : 8'hxx 8'hxx 8'hxx 8'h3b 8'hxx 8'hxx 8'hc6 8'h48 8'h71 8'hxx 8'h7c 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
223 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hcd 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
207 : 8'h17 8'hxx 8'hxx 8'ha7 8'hxx 8'hxx 8'hxx 8'hec 8'hxx 8'hxx 8'h7f 8'hxx 8'hxx 8'hxx 8'hd8 8'h25
191 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h86 8'hxx 8'hxx 8'h60 8'h96
175 : 8'h09 8'hxx 8'hxx 8'hxx 8'hxx 8'h8e 8'he4 8'h9b 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h46 8'ha6 8'hxx
159 : 8'hxx 8'hxx 8'hxx 8'hxx 8'he5 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hd8 8'hxx 8'ha3 8'hxx 8'hxx
143 : 8'h1c 8'hxx 8'h0d 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hd7 8'hxx 8'hxx 8'h37 8'h65 8'h97 8'hxx 8'hxx
127 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hb1 8'h9b 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
111 : 8'hxx 8'hce 8'hxx 8'hxx 8'h4b 8'hxx 8'hxx 8'hxx 8'h14 8'hf3 8'hxx 8'hbe 8'ha9 8'hxx 8'hb6 8'hxx
 95 : 8'hbc 8'hxx 8'hxx 8'h45 8'hc2 8'h9d 8'h7d 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hfa
 79 : 8'hxx 8'hxx 8'hxx 8'hxx 8'h16 8'hxx 8'hxx 8'hxx 8'h04 8'hxx 8'hb8 8'hxx 8'hxx 8'h07 8'hxx 8'hxx
 63 : 8'hxx 8'h2f 8'hxx 8'hxx 8'hxx 8'h5d 8'hxx 8'hxx 8'hfb 8'hxx 8'hxx 8'hxx 8'hxx 8'h81 8'h8d 8'hxx
 47 : 8'h8f 8'hxx 8'hcb 8'h01 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h4e 8'h22 8'hxx
 31 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h9a 8'hxx 8'hxx 8'hxx 8'hxx 8'h9d 8'hxx 8'hxx 8'h03 8'hxx
 15 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h33 8'h2e 8'hxx 8'h18
```

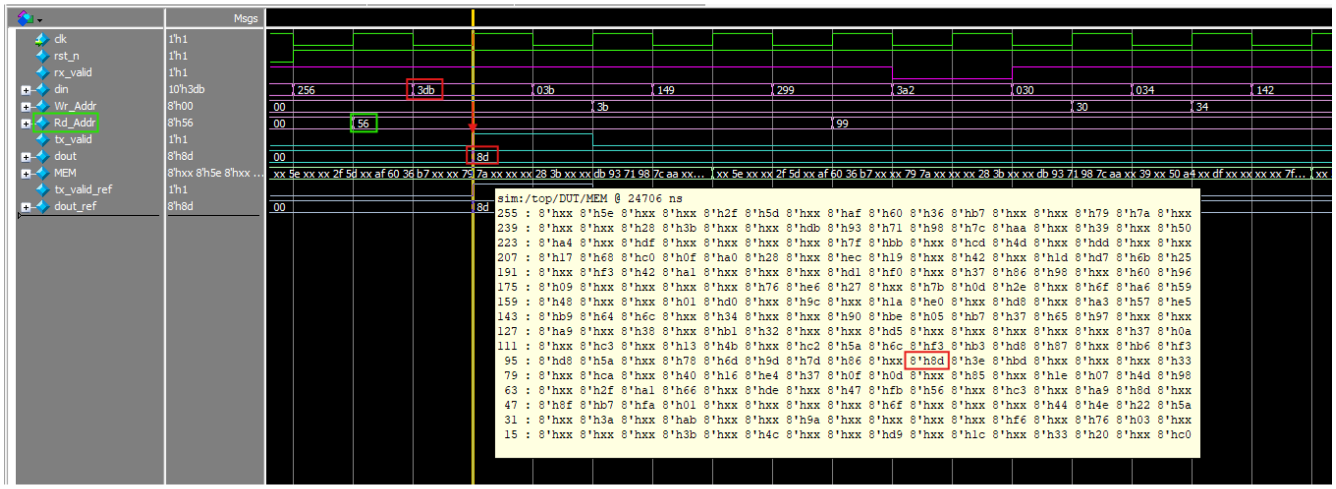


```
sim:/top/DUT/MEM @ 4087 ns
255 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'haf 8'h60 8'hxx 8'hxx 8'hxx 8'hxx 8'hcb 8'hxx 8'hxx
239 : 8'hxx 8'hxx 8'hxx 8'h3b 8'hxx 8'hxx 8'hc6 8'h48 8'h71 8'hxx 8'h7c 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
223 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hcd 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
207 : 8'h17 8'hxx 8'hxx 8'ha7 8'hxx 8'hxx 8'hxx 8'hec 8'hxx 8'hxx 8'h7f 8'hxx 8'hxx 8'hxx 8'hd8 8'h25
191 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h86 8'hxx 8'hxx 8'h60 8'h96
175 : 8'h09 8'hxx 8'hxx 8'hxx 8'hxx 8'h8e 8'he4 8'h9b 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h46 8'ha6 8'hxx
159 : 8'hxx 8'hxx 8'hxx 8'hxx 8'he5 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hd8 8'hxx 8'ha3 8'hxx 8'hxx
143 : 8'h1c 8'hxx 8'h0d 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hd7 8'hxx 8'hxx 8'h37 8'h65 8'h97 8'hxx 8'hxx
127 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hb1 8'h9b 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
111 : 8'hxx 8'hce 8'hxx 8'hxx 8'h4b 8'hxx 8'hxx 8'hxx 8'h14 8'hf3 8'hxx 8'hbe 8'ha9 8'hxx 8'hb6 8'hxx
 95 : 8'hbc 8'hxx 8'hxx 8'h45 8'hc2 8'h9d 8'h7d 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h73
 79 : 8'hxx 8'hxx 8'hxx 8'hxx 8'h16 8'hxx 8'hxx 8'hxx 8'h04 8'hxx 8'hb8 8'hxx 8'hxx 8'h07 8'hxx 8'hxx
 63 : 8'hxx 8'h2f 8'hxx 8'hxx 8'hxx 8'h5d 8'hxx 8'hxx 8'hfb 8'hxx 8'hxx 8'hxx 8'hxx 8'h81 8'h8d 8'hxx
 47 : 8'h8f 8'hxx 8'hcb 8'h01 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h4e 8'h22 8'hxx
 31 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h9a 8'hxx 8'hxx 8'hxx 8'hxx 8'h9d 8'hxx 8'hxx 8'h03 8'hxx
 15 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h33 8'h2e 8'hxx 8'h18
```

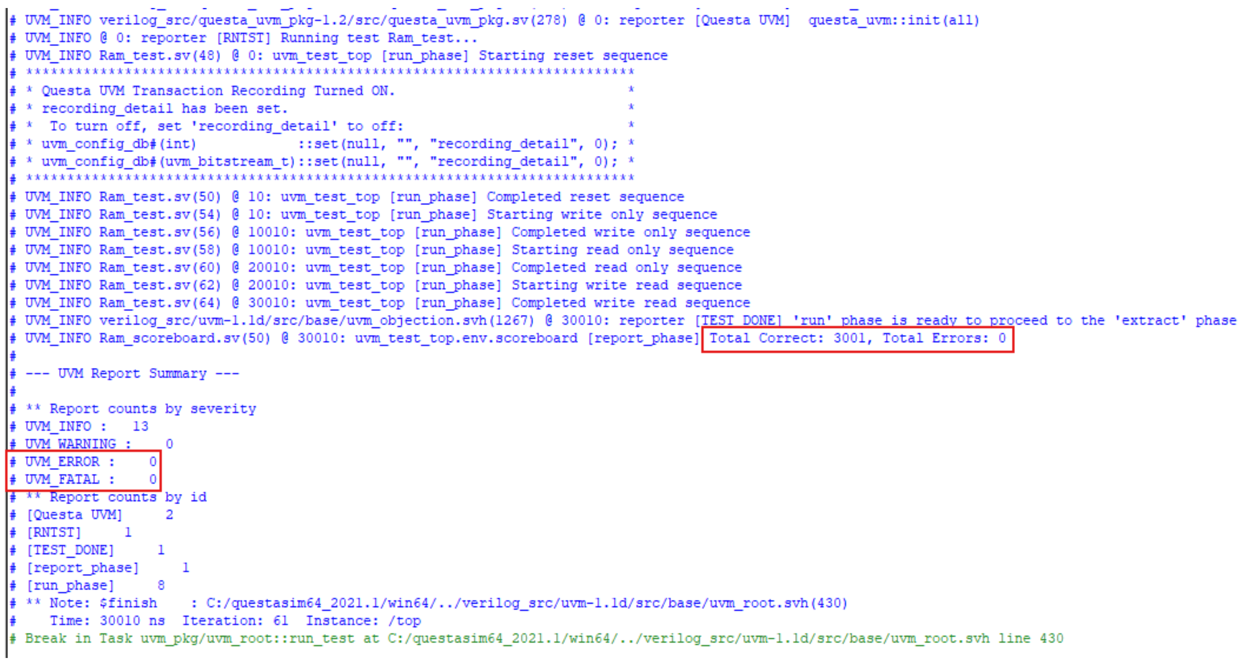
- Read Address



- Read Data



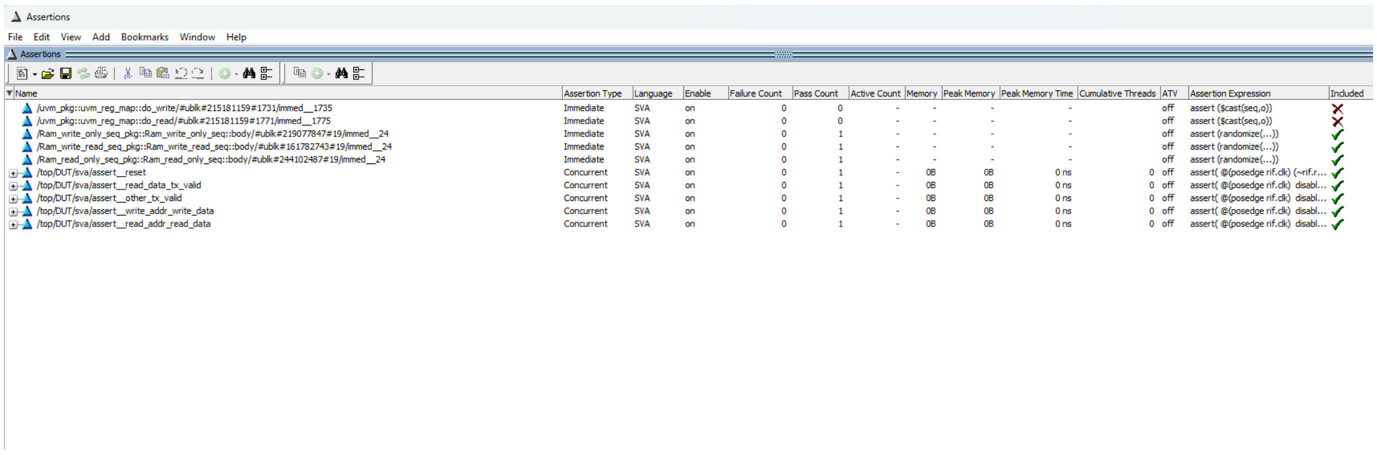
```
sim:/top/DUT/MEM @ 24706 ns
255 : 8'hxx 8'h5e 8'hxx 8'hxx 8'h2f 8'h5d 8'hxx 8'haf 8'h60 8'h36 8'hb7 8'hxx 8'hxx 8'h79 8'h7a 8'hxx
239 : 8'hxx 8'hxx 8'h28 8'h3b 8'hxx 8'hxx 8'hdb 8'h93 8'h71 8'h98 8'h7c 8'haa 8'hxx 8'h39 8'hxx 8'h50
223 : 8'ha4 8'hxx 8'hdf 8'hxx 8'hxx 8'hxx 8'hxx 8'h7f 8'hbb 8'hxx 8'hcd 8'h4d 8'hxx 8'hdd 8'hxx 8'hxx
207 : 8'h17 8'h68 8'hc0 8'h0f 8'ha0 8'h28 8'hxx 8'hec 8'h19 8'hxx 8'h42 8'hxx 8'h1d 8'hd7 8'h6b 8'h25
191 : 8'hxx 8'hf3 8'h42 8'ha1 8'hxx 8'hxx 8'hxx 8'hd1 8'hf0 8'hxx 8'h37 8'h86 8'h98 8'hxx 8'h60 8'h96
175 : 8'h09 8'hxx 8'hxx 8'hxx 8'hxx 8'h76 8'he6 8'h27 8'hxx 8'h7b 8'h0d 8'h2e 8'hxx 8'h6f 8'ha6 8'h59
159 : 8'h48 8'hxx 8'hxx 8'h01 8'hd0 8'hxx 8'h9c 8'hxx 8'h1a 8'he0 8'hxx 8'hd8 8'hxx 8'ha3 8'h57 8'he5
143 : 8'hb9 8'h64 8'h6c 8'hxx 8'h34 8'hxx 8'hxx 8'h90 8'hbe 8'h05 8'hb7 8'h37 8'h65 8'h97 8'hxx 8'hxx
127 : 8'ha9 8'hxx 8'h38 8'hxx 8'hb1 8'h32 8'hxx 8'hxx 8'hd5 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h37 8'h0a
111 : 8'hxx 8'hc3 8'hxx 8'h13 8'h4b 8'hxx 8'hc2 8'h5a 8'h6c 8'hf3 8'hb3 8'hd8 8'h87 8'hxx 8'hb6 8'hf3
 95 : 8'hd8 8'h5a 8'hxx 8'h78 8'h6d 8'h9d 8'h7d 8'h86 8'hxx 8'h8d 8'h3e 8'hbd 8'hxx 8'hxx 8'hxx 8'h33
 79 : 8'hxx 8'hca 8'hxx 8'h40 8'h16 8'he4 8'h37 8'h0f 8'h0d 8'hxx 8'h85 8'hxx 8'h1e 8'h07 8'h4d 8'h98
 63 : 8'hxx 8'h2f 8'ha1 8'h66 8'hxx 8'hde 8'hxx 8'h47 8'hfb 8'h56 8'hxx 8'hc3 8'hxx 8'ha9 8'h8d 8'hxx
 47 : 8'h8f 8'hb7 8'hfa 8'h01 8'hxx 8'hxx 8'hxx 8'hxx 8'h6f 8'hxx 8'hxx 8'hxx 8'h44 8'h4e 8'h22 8'h5a
 31 : 8'hxx 8'h3a 8'hxx 8'hab 8'hxx 8'hxx 8'h9a 8'hxx 8'hxx 8'hxx 8'hxx 8'hf6 8'hxx 8'h76 8'h03 8'hxx
 15 : 8'hxx 8'hxx 8'hxx 8'h3b 8'hxx 8'h4c 8'hxx 8'hxx 8'hd9 8'hxx 8'h1c 8'hxx 8'h33 8'h20 8'hxx 8'hc0
```



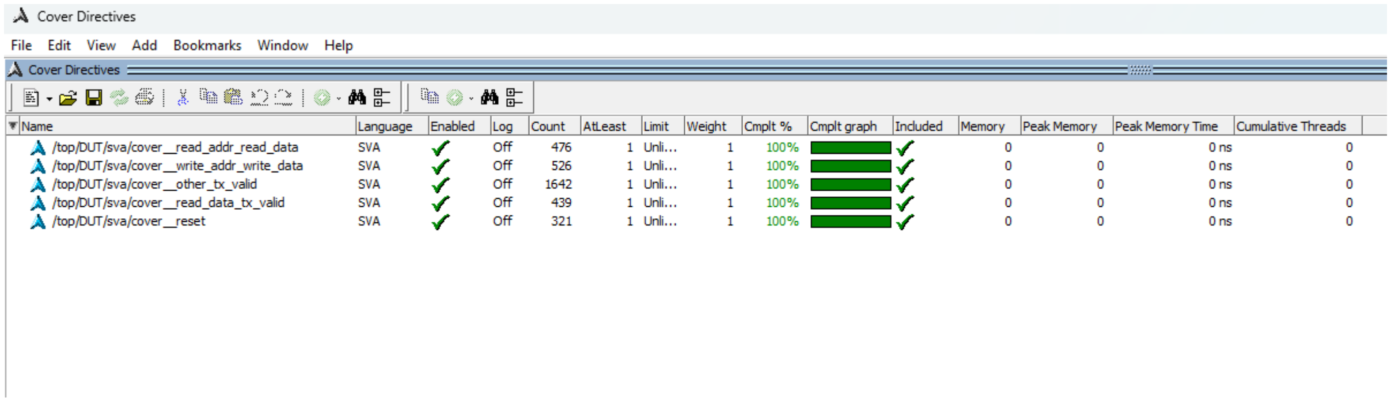
```
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test Ram_test...
# UVM_INFO Ram_test.sv(48) @ 0: uvm_test_top [run_phase] Starting reset sequence
# **************************************************************************
# * 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 Ram_test.sv(50) @ 10: uvm_test_top [run_phase] Completed reset sequence
# UVM_INFO Ram_test.sv(54) @ 10: uvm_test_top [run_phase] Starting write only sequence
# UVM_INFO Ram_test.sv(56) @ 10010: uvm_test_top [run_phase] Completed write only sequence
# UVM_INFO Ram_test.sv(58) @ 10010: uvm_test_top [run_phase] Starting read only sequence
# UVM_INFO Ram_test.sv(60) @ 20010: uvm_test_top [run_phase] Completed read only sequence
# UVM_INFO Ram_test.sv(62) @ 20010: uvm_test_top [run_phase] Starting write read sequence
# UVM_INFO Ram_test.sv(64) @ 30010: uvm_test_top [run_phase] Completed write read sequence
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 30010: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO Ram_scoreboard.sv(50) @ 30010: uvm_test_top.env.scoreboard [report_phase] Total Correct: 3001, Total Errors: 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   13
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     1
# [run_phase]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 30010 ns  Iteration: 61  Instance: /top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

There are no errors

3.2.25 Assertions results



Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/Ram_write_only_seq_pkg::Ram_write_only_seq::body/#ublk#219077847#19/immed__24	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Ram_write_read_seq_pkg::Ram_write_read_seq::body/#ublk#161782743#19/immed__24	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Ram_read_only_seq_pkg::Ram_read_only_seq::body/#ublk#244102487#19/immed__24	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/DUT/sva/assert__reset	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) (~rif.r...	✓
/top/DUT/sva/assert__read_data_tx_valid	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓
/top/DUT/sva/assert__other_tx_valid	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓
/top/DUT/sva/assert__write_addr_write_data	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓
/top/DUT/sva/assert__read_addr_read_data	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓

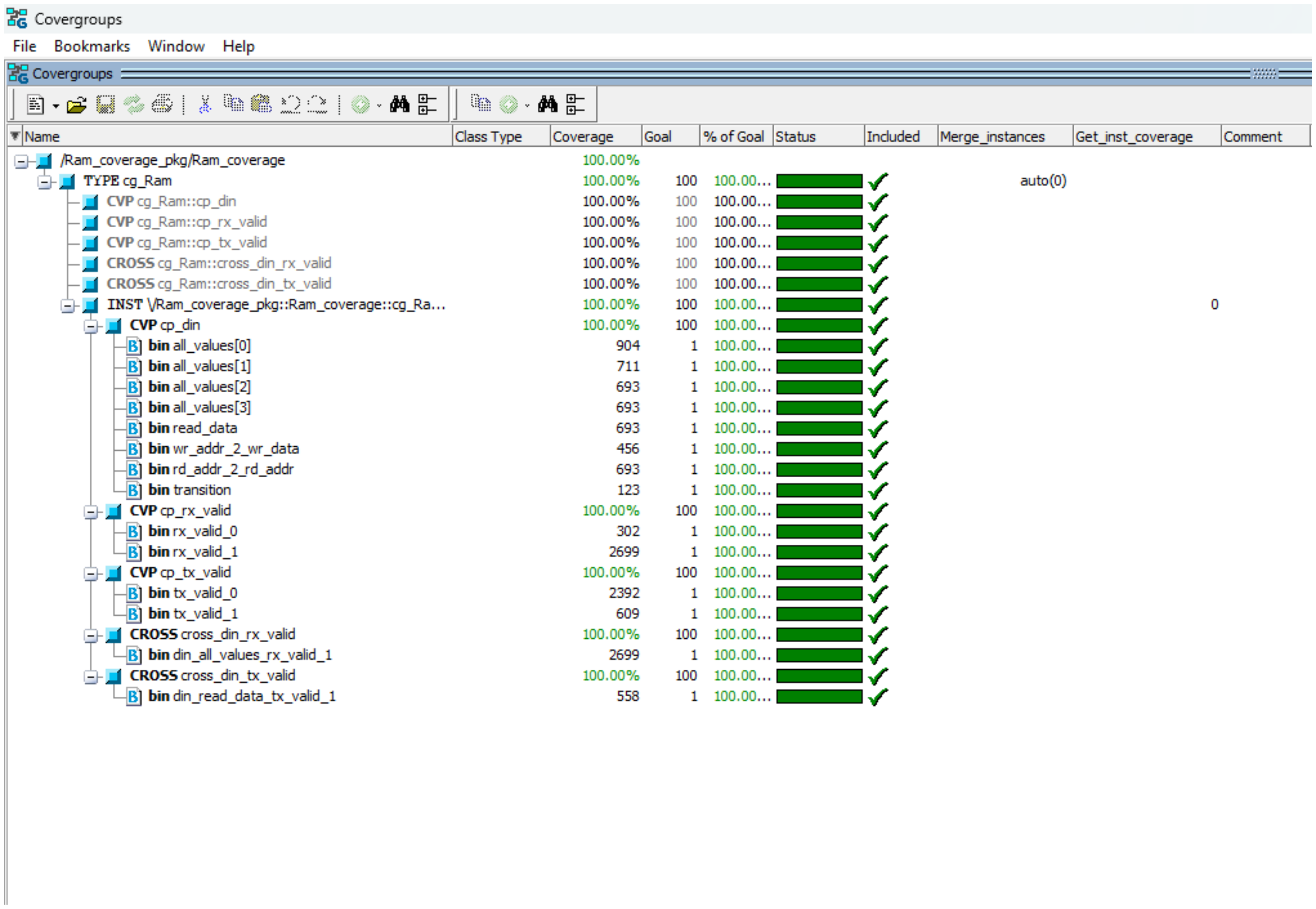


Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/sva/cover__read_addr_read_data	SVA	✓	Off	476	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__write_addr_write_data	SVA	✓	Off	526	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__other_tx_valid	SVA	✓	Off	1642	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__read_data_tx_valid	SVA	✓	Off	439	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/sva/cover__reset	SVA	✓	Off	321	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0

There are no assertion errors

3.2.26 Coverage results

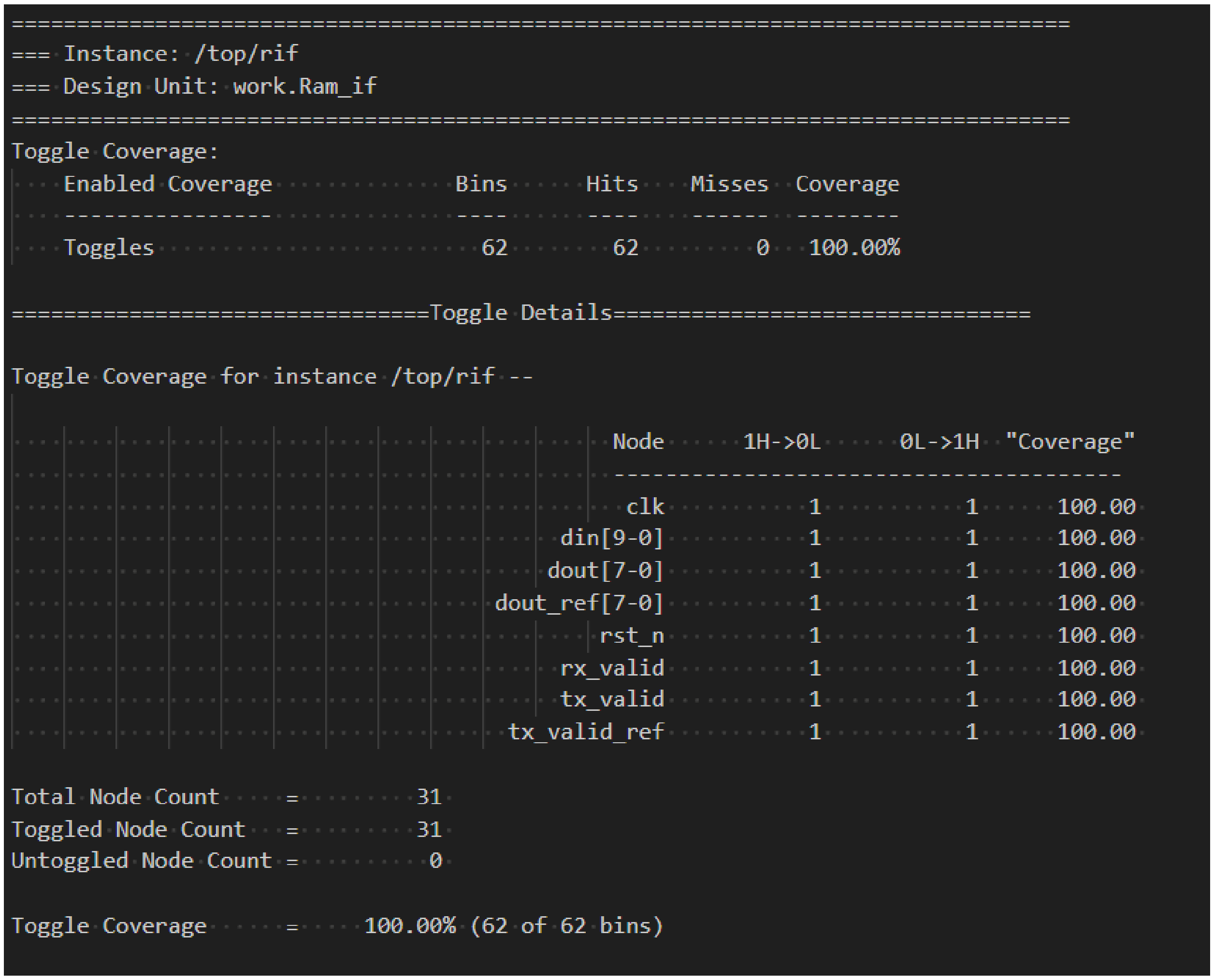
- Functional coverage



Covergroups

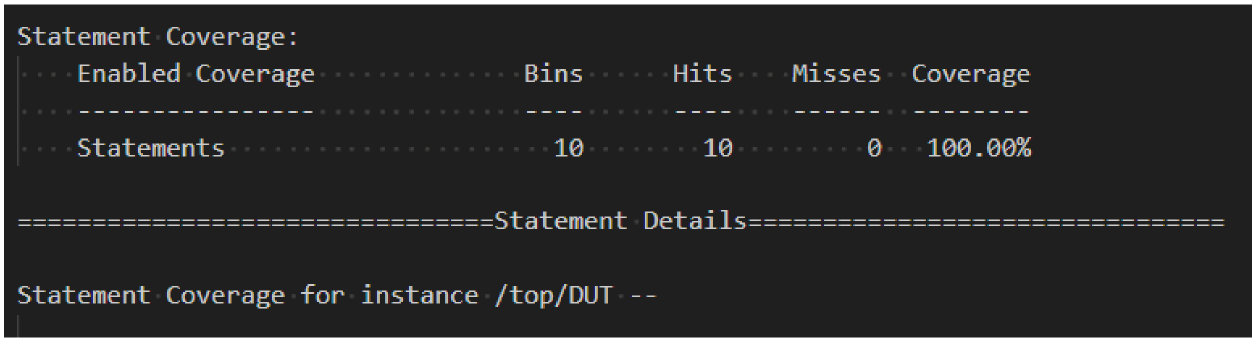
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/Ram_coverage_pkg/Ram_coverage		100.00%							
TYPE cg_Ram		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP cg_Ram::cp_din		100.00%	100	100.00...	🟩	✓			
CVP cg_Ram::cp_rx_valid		100.00%	100	100.00...	🟩	✓			
CVP cg_Ram::cp_tx_valid		100.00%	100	100.00...	🟩	✓			
CROSS cg_Ram::cross_din_rx_valid		100.00%	100	100.00...	🟩	✓			
CROSS cg_Ram::cross_din_tx_valid		100.00%	100	100.00...	🟩	✓			
INST \/Ram_coverage_pkg::Ram_coverage::cg_Ra...		100.00%	100	100.00...	🟩	✓		0	
CVP cp_din		100.00%	100	100.00...	🟩	✓			
bin all_values[0]		904	1	100.00...	🟩	✓			
bin all_values[1]		711	1	100.00...	🟩	✓			
bin all_values[2]		693	1	100.00...	🟩	✓			
bin all_values[3]		693	1	100.00...	🟩	✓			
bin read_data		693	1	100.00...	🟩	✓			
bin wr_addr_2_wr_data		456	1	100.00...	🟩	✓			
bin rd_addr_2_rd_addr		693	1	100.00...	🟩	✓			
bin transition		123	1	100.00...	🟩	✓			
CVP cp_rx_valid		100.00%	100	100.00...	🟩	✓			
bin rx_valid_0		302	1	100.00...	🟩	✓			
bin rx_valid_1		2699	1	100.00...	🟩	✓			
CVP cp_tx_valid		100.00%	100	100.00...	🟩	✓			
bin tx_valid_0		2392	1	100.00...	🟩	✓			
bin tx_valid_1		609	1	100.00...	🟩	✓			
CROSS cross_din_rx_valid		100.00%	100	100.00...	🟩	✓			
bin din_all_values_rx_valid_1		2699	1	100.00...	🟩	✓			
CROSS cross_din_tx_valid		100.00%	100	100.00...	🟩	✓			
bin din_read_data_tx_valid_1		558	1	100.00...	🟩	✓			

- Code coverage
    - Toggle Coverage



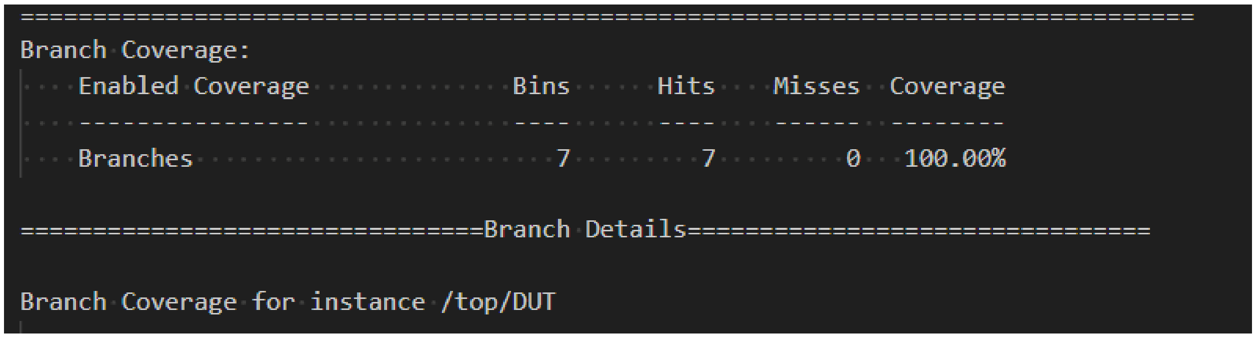
```
=================================================================================
=== Instance: /top/rif
=== Design Unit: work.Ram_if
=================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage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

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

Toggle Coverage for instance /top/rif --

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

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

Toggle Coverage      =     100.00% (62 of 62 bins)
```

- Statement Coverage



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

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

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

- Branch coverage



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

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

Branch Coverage for instance /top/DUT
```

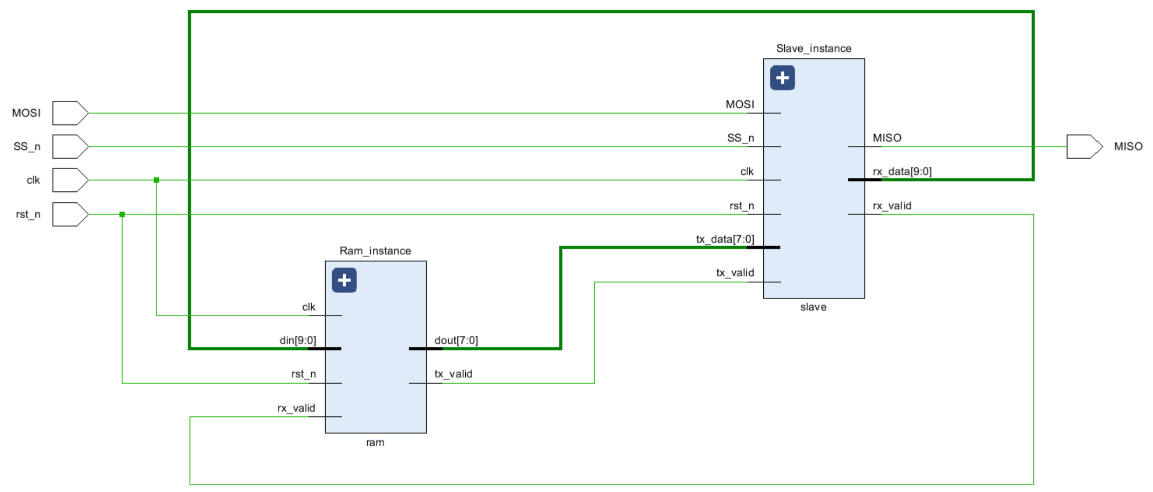
4 SPI Wrapper

4.1 Design

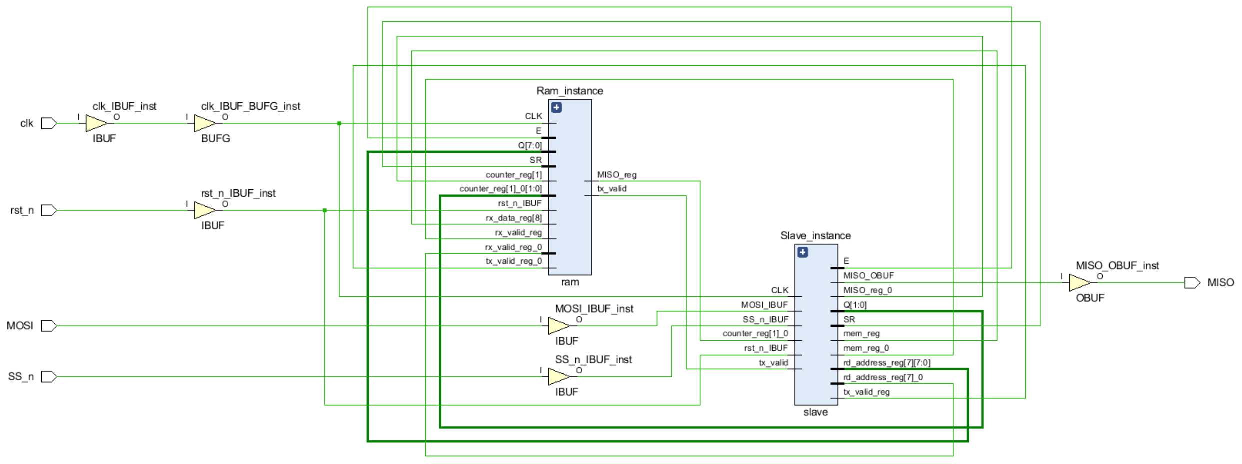
4.1.1 Verilog code

```
module wrapper  (
    input  logic       clk,
    input  logic       rst_n,
    input  logic       MOSI,
    input  logic       SS_n,
    output logic       MISO
);
    wire [9:0]  rx_data;
    wire        rx_valid, tx_valid;
    wire [7:0]  tx_data;

    slave Slave_instance (
        .clk(clk),
        .rst_n(rst_n),
        .tx_valid(tx_valid),
        .MOSI(MOSI),
        .SS_n(SS_n),
        .tx_data(tx_data),
        .rx_data(rx_data),
        .rx_valid(rx_valid),
        .MISO(MISO)
    );

    ram Ram_instance (
        .clk(clk),
        .rst_n(rst_n),
        .rx_valid(rx_valid),
        .din(rx_data),
        .dout(tx_data),
        .tx_valid(tx_valid)
    );
Endmodule
```

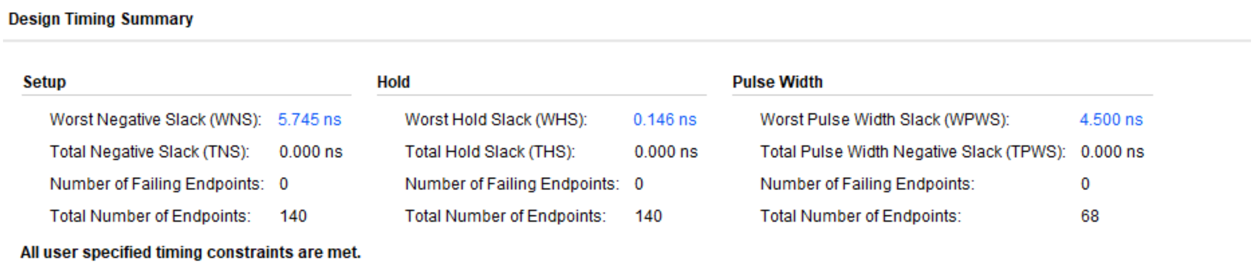
4.1.2 RTL elaboration



4.1.3 Synthesis schematic



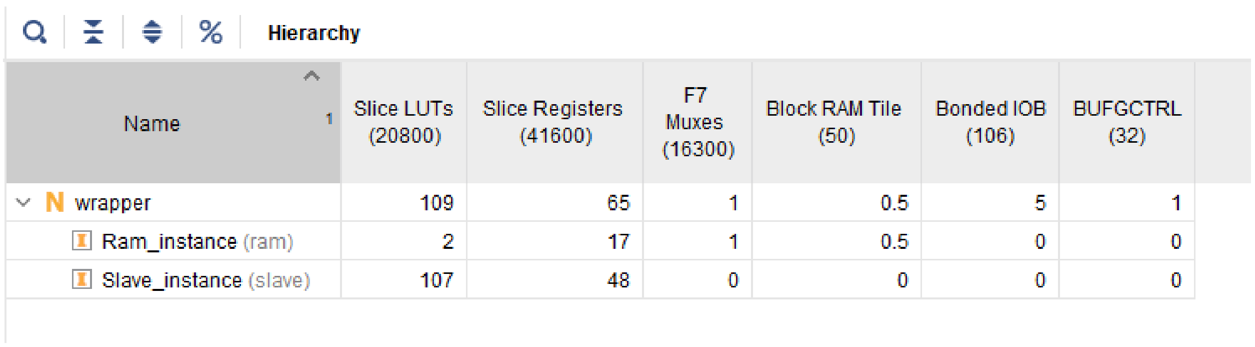
4.1.4 Static-timing analysis and Utilization reports after synthesis



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.745 ns	Worst Hold Slack (WHS):	0.146 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	140	Total Number of Endpoints:	140	Total Number of Endpoints:	68

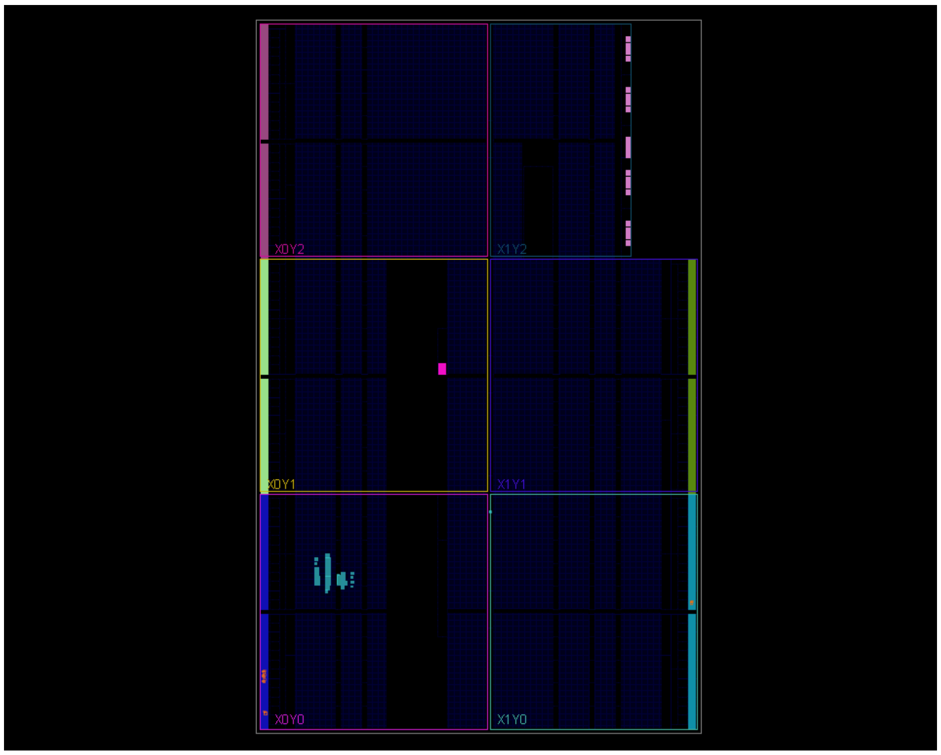
All user specified timing constraints are met.



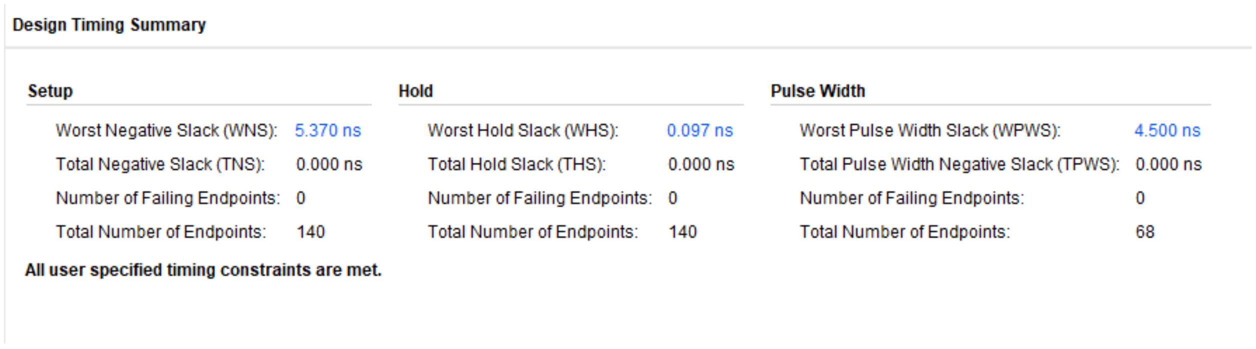
Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
wrapper	109	65	1	0.5	5	1
Ram_instance (ram)	2	17	1	0.5	0	0
Slave_instance (slave)	107	48	0	0	0	0

4.1.5 Device Snippet



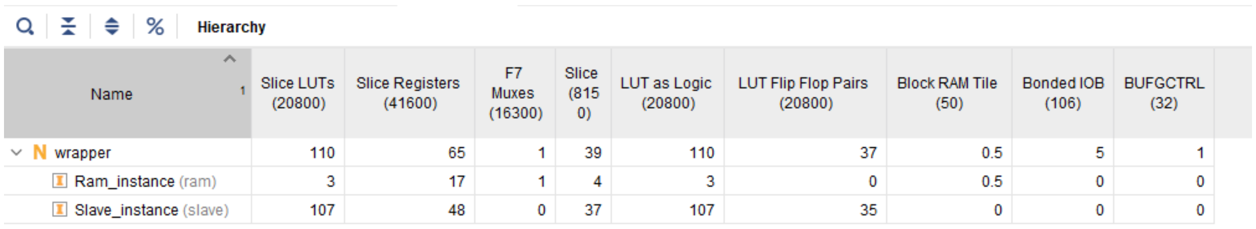
4.1.6 Static-timing analysis and Utilization reports after implementation



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.370 ns	Worst Hold Slack (WHS):	0.097 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	140	Total Number of Endpoints:	140	Total Number of Endpoints:	68

All user specified timing constraints are met.



Hierarchy

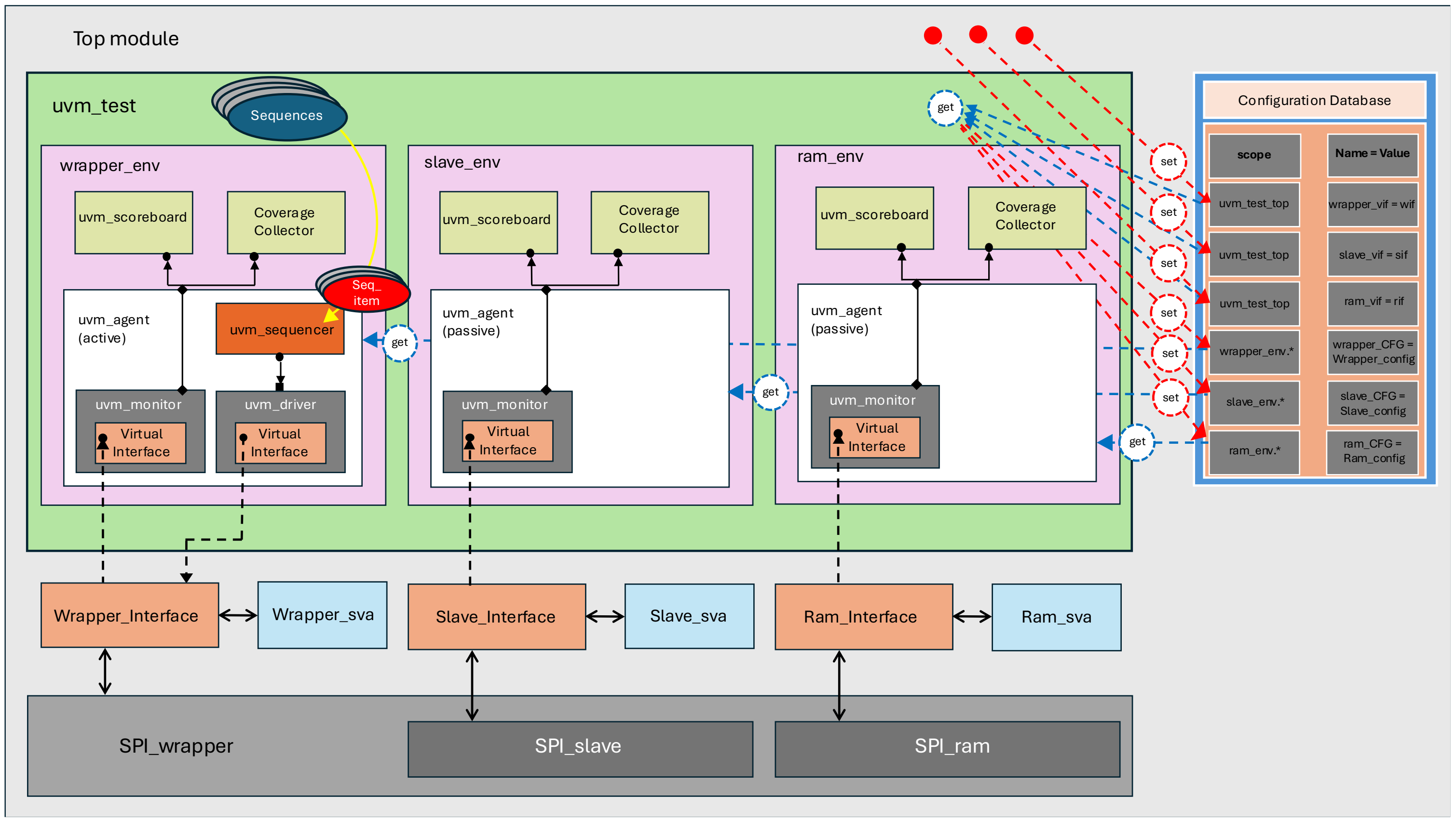
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
wrapper	110	65	1	39	110	37	0.5	5	1
Ram_instance (ram)	3	17	1	4	3	0	0.5	0	0
Slave_instance (slave)	107	48	0	37	107	35	0	0	0

4.2 Verification

- Since the Slave and Ram are now part of the SPI Wrapper, and we have 2 separate environments for both, you will add the 2 environments in the Wrapper test.
- The Wrapper environment will drive the interface of the Wrapper, and the Slave and Ram environments will be **passive** and will not drive data.
- The Slave and Ram environments will only monitor their interfaces and send the data to their scoreboards and coverage collectors.

- Verification will be performed on a buggy design, with the primary objective of identifying and reporting its bugs.
- This verification will be accomplished by comparing the design's output against a golden model “the design above”.

4.2.1 UVM Hierarchy



scope	Name = Value
uvm_test_top	wrapper_vif = wif
uvm_test_top	slave_vif = sif
uvm_test_top	ram_vif = rif
wrapper_env.*	wrapper_CFG = Wrapper_config
slave_env.*	slave_CFG = Slave_config
ram_env.*	ram_CFG = Ram_config

4.2.2 Buggy Design

```
module WRAPPER (Wrapper_if.DUT wif);
wire [9:0] rx_data_din;
wire       rx_valid, tx_valid;
wire [7:0] tx_data_dout;

    SLAVE SLAVE_instance (
        .MOSI(wif.MOSI),
        .MISO(wif.MISO),
        .SS_n(wif.SS_n),
        .clk(wif.clk),
        .rst_n(wif.rst_n),
        .rx_data(rx_data_din),
        .rx_valid(rx_valid),
        .tx_data(tx_data_dout),
        .tx_valid(tx_valid)
    );

    RAM   RAM_instance (
        .din(rx_data_din),
        .clk(wif.clk),
        .rst_n(wif.rst_n),
        .rx_valid(rx_valid),
        .dout(tx_data_dout),
        .tx_valid(tx_valid)
    );

Endmodule
```

4.2.3 UVM config_item

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

    class Wrapper_cfg extends uvm_object;
        `uvm_object_utils(Wrapper_cfg)
        virtual Wrapper_if Wrapper_vif;
        uvm_active_passive_enum is_active;

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

    endclass : Wrapper_cfg
endpackage
```

4.2.4 UVM sequence_item

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

    class Wrapper_seq_item extends uvm_sequence_item;
        `uvm_object_utils(Wrapper_seq_item)

        rand bit            SS_n, rst_n, MOSI;
        bit                 MISO, MISO_ref;
        rand bit [10:0]     array_for_MOSI;
        int                 counter;
        SPI_slave_state_e   cs;

        bit [2:0] old_operation;

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

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

        function string convert2string_stimulus();
            return $sformatf("SS_n=%b, rst_n=%b, MOSI=%b, array_for_MOSI=0x%0h",
                              SS_n, rst_n, MOSI, array_for_MOSI);
        endfunction : convert2string_stimulus

        constraint c_rst_n { rst_n dist {0 := 1, 1 := 40}; }

        constraint c_SS_n {
            if (cs == SPI_slave_state_e'(READ_DATA)) SS_n == (counter % 24 == 0);
            else SS_n == (counter % 14 == 0);
            }

        constraint c_array_for_MOSI {
            array_for_MOSI[10:8] inside {WRITE_DATA, WRITE_ADDR, READ_DATAA, READ_ADDR};
            }

        constraint c_array_for_MOSI_write_only {
                array_for_MOSI[10:8] inside {WRITE_DATA, WRITE_ADDR};
            }

        constraint c_array_for_MOSI_read_only {
            if (old_operation == READ_ADDR)
                array_for_MOSI[10:8] == READ_DATAA;
            else
                array_for_MOSI[10:8] == READ_ADDR;
            }
        constraint c_array_for_MOSI_read_write {
            if (old_operation == WRITE_ADDR)
                array_for_MOSI[10:8] inside {WRITE_DATA, WRITE_ADDR};

            else if (old_operation == WRITE_DATA)
                array_for_MOSI[10:8] dist {READ_ADDR := 60, WRITE_ADDR := 40};

            else if (old_operation == READ_ADDR)
                array_for_MOSI[10:8] == READ_DATAA;

            else
                array_for_MOSI[10:8] dist {WRITE_ADDR := 60, READ_ADDR := 40};
            }


        function void post_randomize();
            old_operation = array_for_MOSI[10:8];

            if (rst_n == 0) begin
                counter = 0;
            end
            else if (SS_n == 1) counter = 1;
            else counter++;
        endfunction : post_randomize

    endclass : Wrapper_seq_item
endpackage
```

4.2.5 UVM reset_sequence

```
package Wrapper_reset_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_seq_item_pkg::*;
    import Wrapper_sequencer_pkg::*;
    import Wrapper_shared_pkg::*;

    class Wrapper_reset_seq extends uvm_sequence #(Wrapper_seq_item);
        `uvm_object_utils(Wrapper_reset_seq)
        Wrapper_seq_item item;

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

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

endpackage
```

4.2.6 UVM write_only_sequence

```
package Wrapper_write_only_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_seq_item_pkg::*;
    import Wrapper_sequencer_pkg::*;
    import Wrapper_shared_pkg::*;

    class Wrapper_write_only_seq extends uvm_sequence #(Wrapper_seq_item);
        `uvm_object_utils(Wrapper_write_only_seq)
        `uvm_declare_p_sequencer (Wrapper_sequencer)
        Wrapper_seq_item item;

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

        task body();
            item = Wrapper_seq_item::type_id::create("item");
            repeat (1000) begin
                for (int i=10; i>=0; i--) begin
                    start_item(item);
                    item.cs = p_sequencer.sequencer_vif.cs; //get current state

                    if (item.rst_n == 0) begin
                        item.constraint_mode(0);
                        item.c_SS_n.constraint_mode(1);
                        item.c_rst_n.constraint_mode(1);
                        // enable randomization to set all array bits to 1 during reset
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize() with { item.array_for_MOSI == 11'h7FF; });
                        i = 11;
                    end
                    else if (item.SS_n == 1) begin
                        item.constraint_mode(1);
                        item.c_array_for_MOSI_write_only.constraint_mode(1);
                        item.c_array_for_MOSI_read_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_write.constraint_mode(0);
                        // enable randomization for new array when SS_n is high
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize());
                        i = 11;
                    end
                    else begin
                        item.constraint_mode(1);
                        item.c_array_for_MOSI_write_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_write.constraint_mode(0);
                        // disable randomization to keep array values stable
                        item.array_for_MOSI.rand_mode(0);
                        // send array bits [10:0] sequentially to MOSI
                        assert(item.randomize() with {item.MOSI == item.array_for_MOSI[i];});
                    end
                    finish_item(item);
                end
            end
        endtask : body
    endclass : Wrapper_write_only_seq
endpackage
```

4.2.7 UVM read_only_sequence

```
package Wrapper_read_only_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_seq_item_pkg::*;
    import Wrapper_sequencer_pkg::*;
    import Wrapper_shared_pkg::*;

    class Wrapper_read_only_seq extends uvm_sequence #(Wrapper_seq_item);
        `uvm_object_utils(Wrapper_read_only_seq)
        Wrapper_seq_item item;
        `uvm_declare_p_sequencer (Wrapper_sequencer)

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

        task body();
            item = Wrapper_seq_item::type_id::create("item");
            repeat (1000) begin
                for (int i=10; i>=0; i--) begin
                    start_item(item);
                    item.cs = p_sequencer.sequencer_vif.cs; //get current SPI_slave state

                    if (item.rst_n == 0) begin
                        item.constraint_mode(0);
                        item.c_SS_n.constraint_mode(1);
                        item.c_rst_n.constraint_mode(1);
                        // enable randomization to set all array bits to 1 during reset
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize() with { item.array_for_MOSI == 11'h7FF; });
                        i = 11;
                    end
                    else if (item.SS_n == 1) begin
                        item.constraint_mode(1);
                        item.c_array_for_MOSI_read_only.constraint_mode(1);
                        item.c_array_for_MOSI_write_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_write.constraint_mode(0);
                        // enable randomization for new array when SS_n is high
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize());
                        i = 11;
                    end
                    else begin
                        item.constraint_mode(1);
                        item.c_array_for_MOSI_write_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_write.constraint_mode(0);
                        item.c_array_for_MOSI_read_only.constraint_mode(0);
                        // disable randomization to keep array values stable
                        item.array_for_MOSI.rand_mode(0);
                        // send array bits [10:0] sequentially to MOSI
                        assert(item.randomize() with {item.MOSI == item.array_for_MOSI[i];});
                    end
                    finish_item(item);
                end
            end
        endtask : body
    endclass : Wrapper_read_only_seq
endpackage
```

4.2.8 UVM write_read_sequence

```
package Wrapper_write_read_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_seq_item_pkg::*;
    import Wrapper_sequencer_pkg::*;
    import Wrapper_shared_pkg::*;

    class Wrapper_write_read_seq extends uvm_sequence #(Wrapper_seq_item);
        `uvm_object_utils(Wrapper_write_read_seq)
        `uvm_declare_p_sequencer (Wrapper_sequencer)
        Wrapper_seq_item item;

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

        task body();
            item = Wrapper_seq_item::type_id::create("item");
            repeat (1000) begin
                for (int i=10; i>=0; i--) begin
                    start_item(item);
                    item.cs = p_sequencer.sequencer_vif.cs; //get current SPI_slave state

                    if (item.rst_n == 0) begin
                        item.constraint_mode(0);
                        item.c_SS_n.constraint_mode(1);
                        item.c_rst_n.constraint_mode(1);
                        // enable randomization to set all array bits to 1 during reset
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize() with { item.array_for_MOSI == 11'h7FF; });
                        i = 11;
                    end
                    else if (item.SS_n == 1) begin
                        item.constraint_mode(1);
                        item.c_array_for_MOSI_write_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_write.constraint_mode(1);
                        // enable randomization for new array when SS_n is high
                        item.array_for_MOSI.rand_mode(1);
                        assert(item.randomize());
                        i = 11;
                    end
                    else begin
                        item.constraint_mode(1);
                        item.c_array_for_MOSI_write_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_only.constraint_mode(0);
                        item.c_array_for_MOSI_read_write.constraint_mode(0);
                        // disable randomization to keep array values stable
                        item.array_for_MOSI.rand_mode(0);
                        // send array bits [10:0] sequentially to MOSI
                        assert(item.randomize() with {item.MOSI == item.array_for_MOSI[i];});
                    end
                    finish_item(item);
                end
            end
        endtask : body
    endclass : Wrapper_write_read_seq
endpackage
```

4.2.9 UVM sequencer

```
package Wrapper_sequencer_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_seq_item_pkg::*;
    import Wrapper_config_pkg::*;

    class Wrapper_sequencer extends uvm_sequencer #(Wrapper_seq_item);
        `uvm_component_utils(Wrapper_sequencer)
        Wrapper_cfg cfg;
        virtual Wrapper_if sequencer_vif;

        function new(string name, uvm_component parent);
            super.new(name, parent);
        endfunction : new

    endclass : Wrapper_sequencer
endpackage
```

4.2.10 UVM driver

```
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)

        virtual Wrapper_if driver_vif;
        Wrapper_seq_item stim_seq_item;

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                stim_seq_item = Wrapper_seq_item::type_id::create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);

                driver_vif.rst_n            = stim_seq_item.rst_n;
                driver_vif.SS_n             = stim_seq_item.SS_n;
                driver_vif.MOSI             = stim_seq_item.MOSI;
                driver_vif.array_for_MOSI   = stim_seq_item.array_for_MOSI;
                driver_vif.old_operation    = stim_seq_item.old_operation;
                @(negedge driver_vif.clk);

                seq_item_port.item_done();
                `uvm_info("run_phase", stim_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask : run_phase
    endclass: Wrapper_driver
endpackage
```

4.2.11 UVM monitor

```
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 monitor_vif;
        Wrapper_seq_item rsp_seq_item;
        uvm_analysis_port #(Wrapper_seq_item) monitor_ap;

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

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = Wrapper_seq_item::type_id::create("rsp_seq_item");

                @(negedge monitor_vif.clk);
                rsp_seq_item.SS_n             = monitor_vif.SS_n;
                rsp_seq_item.rst_n            = monitor_vif.rst_n;
                rsp_seq_item.MOSI             = monitor_vif.MOSI;
                rsp_seq_item.MISO             = monitor_vif.MISO;
                rsp_seq_item.MISO_ref         = monitor_vif.MISO_ref;

                monitor_ap.write(rsp_seq_item);
                `uvm_info("run_phase", rsp_seq_item.convert2string(), UVM_HIGH)

            end
        endtask : run_phase
    endclass: Wrapper_monitor

endpackage
```

4.2.12 UVM agent

```
package Wrapper_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_driver_pkg::*;
    import Wrapper_monitor_pkg::*;
    import Wrapper_sequencer_pkg::*;
    import Wrapper_config_pkg::*;
    import Wrapper_seq_item_pkg::*;

    class Wrapper_agent extends uvm_agent;
        `uvm_component_utils(Wrapper_agent)
        Wrapper_driver driver;
        Wrapper_monitor monitor;
        Wrapper_sequencer sequencer;
        Wrapper_cfg Wrapper_config;
        uvm_analysis_port #(Wrapper_seq_item) agent_ap;

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

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

            if (!uvm_config_db#(Wrapper_cfg)::get(this, "", "wrapper_CFG", Wrapper_config))
            begin
                `uvm_fatal("NOVIF", "Cannot get cfg from uvm_config_db")
            end
            if (Wrapper_config.is_active == UVM_ACTIVE) begin
                driver      = Wrapper_driver::type_id::create("driver", this);
                sequencer   = Wrapper_sequencer::type_id::create("sequencer", this);
            end
            monitor = Wrapper_monitor::type_id::create("monitor", this);
            agent_ap = new("agent_ap", this);
        endfunction : build_phase

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            if (Wrapper_config.is_active == UVM_ACTIVE) begin
                driver.seq_item_port.connect(sequencer.seq_item_export);
                driver.driver_vif       = Wrapper_config.Wrapper_vif;
                sequencer.sequencer_vif = Wrapper_config.Wrapper_vif;
            end
            monitor.monitor_ap.connect(agent_ap);
            monitor.monitor_vif         = Wrapper_config.Wrapper_vif;
        endfunction : connect_phase

    endclass: Wrapper_agent
endpackage
```

4.2.13 UVM scoreboard

```
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_export;
        uvm_tlm_analysis_fifo #(Wrapper_seq_item) sb_fifo;
        Wrapper_seq_item seq_item_sb;

        int error_count =0; int correct_count=0;

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

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

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

        task run_phase(uvm_phase phase);
            forever begin
                sb_fifo.get(seq_item_sb);
                if (seq_item_sb.MISO !== seq_item_sb.MISO_ref)
                begin
                    `uvm_error("run_phase", $sformatf("Mismatch: MISO=%0b, MISO_ref=%0b, %s",
                                seq_item_sb.MISO, seq_item_sb.MISO_ref,
                                seq_item_sb.convert2string_stimulus()))
                    error_count++;
                end
                else begin
                    `uvm_info("run_phase", $sformatf("Match: MISO=%0b, MISO_ref=%0b",
                               seq_item_sb.MISO, seq_item_sb.MISO_ref), UVM_HIGH)
                    correct_count++;
                end
            end
        endtask

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("report_phase", $sformatf("Total Correct: %0d, Total Errors: %0d",
                      correct_count, error_count), UVM_MEDIUM)
        endfunction
    endclass: Wrapper_scoreboard
endpackage
```

4.2.14 UVM coverage

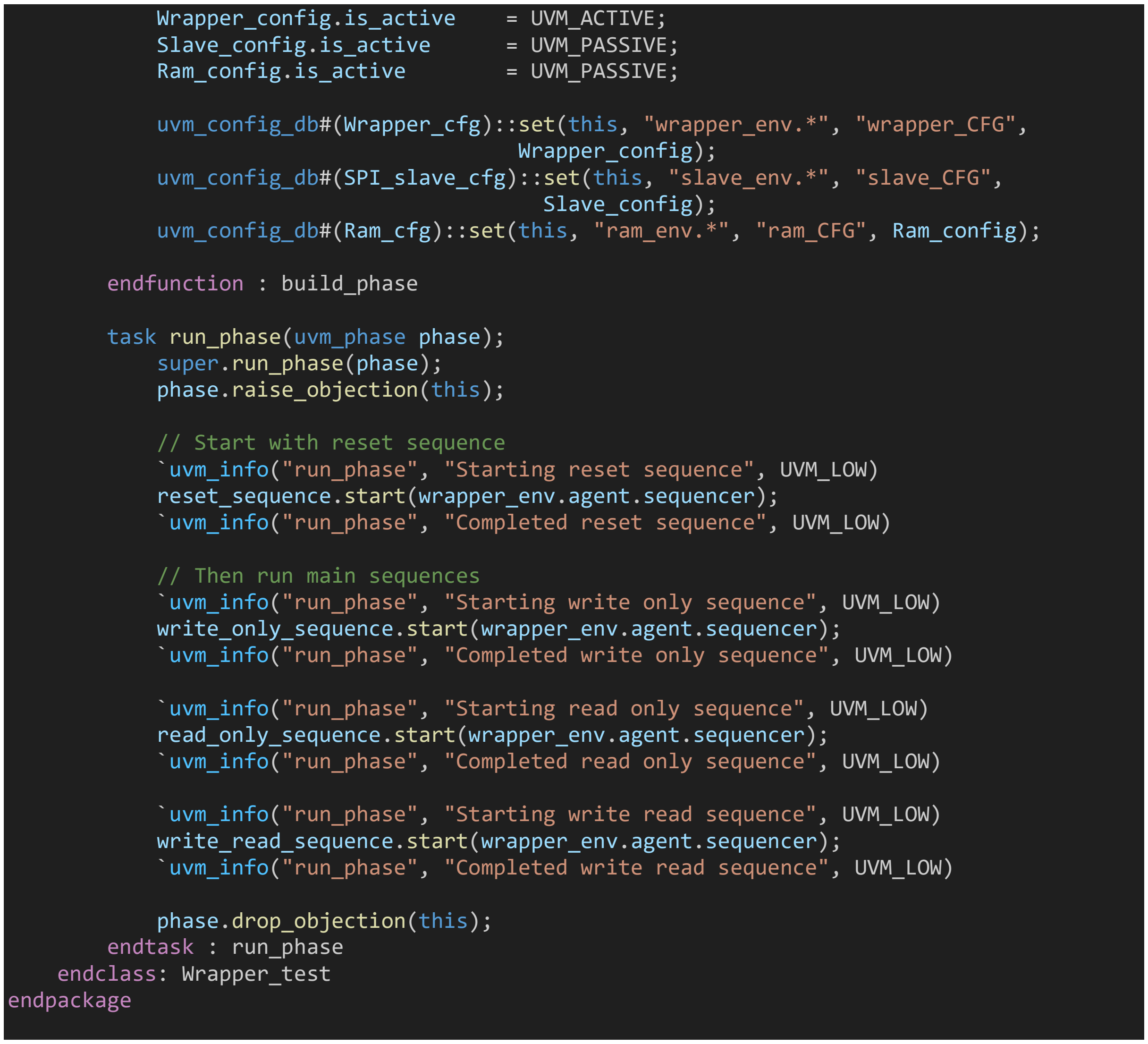
```
package Wrapper_coverage_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_seq_item_pkg::*;
    import Wrapper_shared_pkg::*;

    class Wrapper_coverage extends uvm_component;
        `uvm_component_utils(Wrapper_coverage)
        uvm_analysis_export #(Wrapper_seq_item) cov_export;
        uvm_tlm_analysis_fifo #(Wrapper_seq_item) cov_fifo;
        Wrapper_seq_item cov_item;

        // Functional Coverage
        covergroup cg_Wrapper;
            cp_SS_n: coverpoint cov_item.SS_n {
                bins SS_n_OTHER     = (1=> 0[*13] => 1);
                bins SS_n_READ_DATA = (1=> 0[*23] => 1);
                bins SS_n_one2zero  = (1 => 0);
            }
            cp_MOSI: coverpoint cov_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);
            }
            cross_MOSI_SS_n: cross cp_MOSI, cp_SS_n {
                option.cross_auto_bin_max = 0;
                bins Read_data_SS_n_READ_DATA
                    = binsof(cp_MOSI.read_data) && binsof(cp_SS_n.SS_n_READ_DATA);
                bins Write_data_SS_n_OTHER
                    = binsof(cp_MOSI.write_data) && binsof(cp_SS_n.SS_n_OTHER);
                bins Read_address_SS_n_OTHER
                    = binsof(cp_MOSI.read_address) && binsof(cp_SS_n.SS_n_OTHER);
                bins Write_address_SS_n_OTHER
                    = binsof(cp_MOSI.write_address) && binsof(cp_SS_n.SS_n_OTHER);
            }
        endgroup : cg_Wrapper

        function new(string name = "Wrapper_coverage", uvm_component parent = null);
            super.new(name, parent);
            cg_Wrapper = 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 : build_phase

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cov_export.connect(cov_fifo.analysis_export);
        endfunction : connect_phase
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(cov_item);
                cg_Wrapper.sample();
            end
        endtask : run_phase
    endclass: Wrapper_coverage
endpackage
```

4.2.15 UVM environment

```
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 agent;
        Wrapper_scoreboard scoreboard;
        Wrapper_coverage coverage;

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

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

            agent       = Wrapper_agent::type_id::create("agent", this);
            scoreboard  = Wrapper_scoreboard::type_id::create("scoreboard", this);
            coverage    = Wrapper_coverage::type_id::create("coverage", this);
        endfunction : build_phase

        virtual function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agent.agent_ap.connect(scoreboard.sb_export);
            agent.agent_ap.connect(coverage.cov_export);
        endfunction : connect_phase
    endclass: Wrapper_env
endpackage
```

4.2.16 UVM test

```
package Wrapper_test_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import Wrapper_env_pkg::*;
    import SPI_slave_env_pkg::*;
    import Ram_env_pkg::*;
    import Wrapper_config_pkg::*;
    import SPI_slave_config_pkg::*;
    import Ram_config_pkg::*;
    import Wrapper_read_only_seq_pkg::*;
    import Wrapper_write_read_seq_pkg::*;
    import Wrapper_write_only_seq_pkg::*;
    import Wrapper_reset_seq_pkg::*;

    class Wrapper_test extends uvm_test;
        `uvm_component_utils(Wrapper_test)

        Wrapper_env wrapper_env;
        SPI_slave_env slave_env;
        Ram_env ram_env;
        Wrapper_read_only_seq read_only_sequence;
        Wrapper_write_read_seq write_read_sequence;
        Wrapper_write_only_seq write_only_sequence;
        Wrapper_reset_seq reset_sequence;
        Wrapper_cfg Wrapper_config;
        SPI_slave_cfg Slave_config;
        Ram_cfg Ram_config;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            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);
            read_only_sequence   = Wrapper_read_only_seq::type_id::create("read_only_sequence");
            write_read_sequence  = Wrapper_write_read_seq::type_id::create("write_read_sequence");
            write_only_sequence  = Wrapper_write_only_seq::type_id::create("write_only_sequence");
            reset_sequence       = Wrapper_reset_seq::type_id::create("reset_sequence");
            Wrapper_config       = Wrapper_cfg::type_id::create("Wrapper_config");
            Slave_config         = SPI_slave_cfg::type_id::create("Slave_config");
            Ram_config           = Ram_cfg::type_id::create("Ram_config");

            if(!uvm_config_db#(virtual Wrapper_if)::get(this, "", "wrapper_vif",
              Wrapper_config.Wrapper_vif))
               `uvm_fatal("NOVIF", "Virtual interface must be set for:")
            if(!uvm_config_db#(virtual SPI_slave_if)::get(this, "", "slave_vif",
              Slave_config.Slave_vif))
               `uvm_fatal("NOVIF", "Virtual interface must be set for:")
            if(!uvm_config_db#(virtual Ram_if)::get(this, "", "ram_vif",
              Ram_config.Ram_vif))
               `uvm_fatal("NOVIF", "Virtual interface must be set for:")
```



```
            Wrapper_config.is_active    = UVM_ACTIVE;
            Slave_config.is_active      = UVM_PASSIVE;
            Ram_config.is_active        = UVM_PASSIVE;

            uvm_config_db#(Wrapper_cfg)::set(this, "wrapper_env.*", "wrapper_CFG",
                                         Wrapper_config);
            uvm_config_db#(SPI_slave_cfg)::set(this, "slave_env.*", "slave_CFG",
                                           Slave_config);
            uvm_config_db#(Ram_cfg)::set(this, "ram_env.*", "ram_CFG", Ram_config);

        endfunction : build_phase

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

            // Start with reset sequence
            `uvm_info("run_phase", "Starting reset sequence", UVM_LOW)
            reset_sequence.start(wrapper_env.agent.sequencer);
            `uvm_info("run_phase", "Completed reset sequence", UVM_LOW)

            // Then run main sequences
            `uvm_info("run_phase", "Starting write only sequence", UVM_LOW)
            write_only_sequence.start(wrapper_env.agent.sequencer);
            `uvm_info("run_phase", "Completed write only sequence", UVM_LOW)

            `uvm_info("run_phase", "Starting read only sequence", UVM_LOW)
            read_only_sequence.start(wrapper_env.agent.sequencer);
            `uvm_info("run_phase", "Completed read only sequence", UVM_LOW)

            `uvm_info("run_phase", "Starting write read sequence", UVM_LOW)
            write_read_sequence.start(wrapper_env.agent.sequencer);
            `uvm_info("run_phase", "Completed write read sequence", UVM_LOW)

            phase.drop_objection(this);
        endtask : run_phase
    endclass: Wrapper_test
endpackage
```

4.2.17 Wrapper interface

```
interface Wrapper_if (clk);
import Wrapper_shared_pkg::*;
    input logic                 clk;
          logic                 SS_n, rst_n, MOSI;
          logic                 MISO, MISO_ref;
          logic [10:0]          array_for_MOSI;
          SPI_slave_state_e cs;
          bit [2:0]             old_operation;

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

    modport Golden (
        input  clk, rst_n, SS_n, MOSI,
        output MISO_ref
    );
Endinterface
```

4.2.18 Wrapper shared package

```
package Wrapper_shared_pkg;
    parameter WRITE_ADDR = 3'b000;
    parameter WRITE_DATA = 3'b001;
    parameter READ_ADDR  = 3'b110;
    parameter READ_DATAA   = 3'b111;

    typedef enum logic [2:0] {
        IDLE      = 3'b000,
        WRITE     = 3'b001,
        CHK_CMD   = 3'b010,
        READ_ADD  = 3'b011,
        READ_DATA = 3'b100
    } SPI_slave_state_e;
endpackage : Wrapper_shared_pkg
```

4.2.19 Wrapper assertions

```
module Wrapper_sva(Wrapper_if.DUT wif);
//Sequences
    sequence write_add_seq;
        (wif.SS_n == 1) ##1 (wif.SS_n == 0) ##1 (wif.MOSI == 0) ##1 (wif.MOSI == 0)
        ##1 (wif.MOSI == 0);
    endsequence

    sequence read_add_seq;
        (wif.SS_n == 1) ##1 (wif.SS_n == 0) ##1 (wif.MOSI == 1) ##1 (wif.MOSI == 1)
        ##1 (wif.MOSI == 0);
    endsequence

    sequence write_data_seq;
        (wif.SS_n == 1) ##1 (wif.SS_n == 0) ##1 (wif.MOSI == 0) ##1 (wif.MOSI == 0)
        ##1 (wif.MOSI == 1);
    endsequence

//properties
    property reset_check;
        @(posedge wif.clk) !wif.rst_n |=> (!wif.MISO);
    endproperty : reset_check

    property write_add_stable_MISO;
        @(posedge wif.clk) disable iff (!wif.rst_n)
        write_add_seq |=> (($stable(wif.MISO) throughout wif.SS_n[->1]));
    endproperty : write_add_stable_MISO

    property read_add_stable_MISO;
        @(posedge wif.clk) disable iff (!wif.rst_n)
        read_add_seq |=> (($stable(wif.MISO) throughout wif.SS_n[->1]));
    endproperty : read_add_stable_MISO

    property write_data_stable_MISO;
        @(posedge wif.clk) disable iff (!wif.rst_n)
        write_data_seq |=> (($stable(wif.MISO) throughout wif.SS_n[->1]));
    endproperty : write_data_stable_MISO

//Assertions
    assert property (reset_check)
    else $error("SVA-ERROR: Reset check failed");
    assert property (write_add_stable_MISO)
    else $error("SVA-ERROR: MISO changed during write address transaction");
    assert property (read_add_stable_MISO)
    else $error("SVA-ERROR: MISO changed during read address transaction");
    assert property (write_data_stable_MISO)
    else $error("SVA-ERROR: MISO changed during write data transaction");
//Coverage
    cover property (reset_check);
    cover property (write_add_stable_MISO);
    cover property (read_add_stable_MISO);
    cover property (write_data_stable_MISO);
endmodule : Wrapper_sva
```

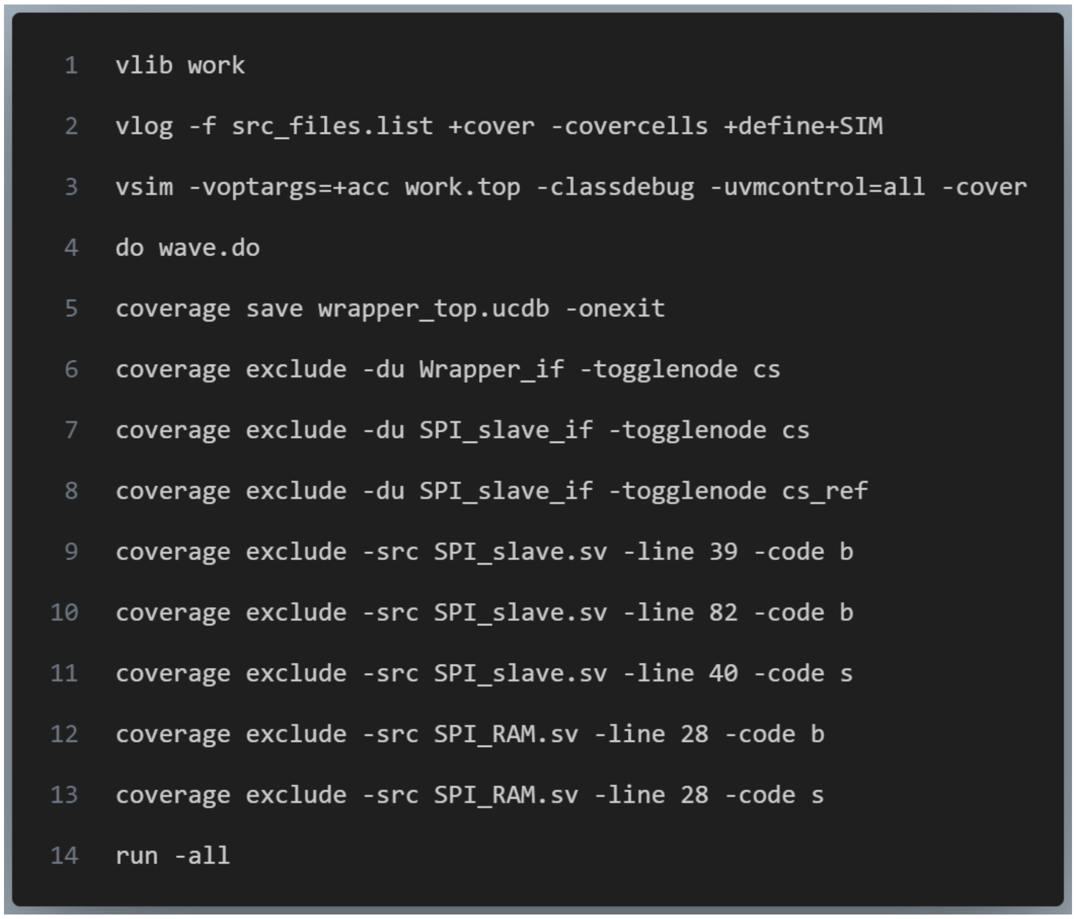
4.2.20 Wrapper Golden model

```
module Golden_wrapper  (Wrapper_if.Golden wif);
    wire [9:0]  rx_data;
    wire        rx_valid, tx_valid;
    wire [7:0]  tx_data;

    Golden_slave Slave_instance (
        .clk(wif.clk),
        .rst_n(wif.rst_n),
        .tx_valid(tx_valid),
        .MOSI(wif.MOSI),
        .SS_n(wif.SS_n),
        .tx_data(tx_data),
        .rx_data_ref(rx_data),
        .rx_valid_ref(rx_valid),
        .MISO_ref(wif.MISO_ref)
    );

    Golden_ram Ram_instance (
        .clk(wif.clk),
        .rst_n(wif.rst_n),
        .rx_valid(rx_valid),
        .din(rx_data),
        .dout_ref(tx_data),
        .tx_valid_ref(tx_valid)
    );
Endmodule
```

4.2.21 UVM top

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

module top ();

    bit clk;
    always #5 clk = ~clk;

    Ram_if rif(clk);
    SPI_slave_if sif(clk);
    Wrapper_if wif(clk);

    WRAPPER DUT (wif);
    Golden_wrapper golden_model (wif);

    assign sif.rst_n           = DUT.SLAVE_instance.rst_n;
    assign sif.SS_n            = DUT.SLAVE_instance.SS_n;
    assign sif.MOSI            = DUT.SLAVE_instance.MOSI;
    assign sif.tx_data         = DUT.SLAVE_instance.tx_data;
    assign sif.tx_valid        = DUT.SLAVE_instance.tx_valid;
    assign sif.cs              = DUT.SLAVE_instance.cs;
    assign sif.rx_valid        = DUT.SLAVE_instance.rx_valid;
    assign sif.rx_data         = DUT.SLAVE_instance.rx_data;
    assign sif.MISO            = DUT.SLAVE_instance.MISO;
    assign sif.cs_ref          = golden_model.Slave_instance.state;
    assign sif.rx_data_ref     = golden_model.Slave_instance.rx_data_ref;
    assign sif.rx_valid_ref    = golden_model.Slave_instance.rx_valid_ref;
    assign sif.MISO_ref        = golden_model.Slave_instance.MISO_ref;
    assign sif.array_for_MOSI  = wif.array_for_MOSI;

    assign rif.rst_n           = DUT.RAM_instance.rst_n;
    assign rif.rx_valid        = DUT.RAM_instance.rx_valid;
    assign rif.din             = DUT.RAM_instance.din;
    assign rif.dout            = DUT.RAM_instance.dout;
    assign rif.tx_valid        = DUT.RAM_instance.tx_valid;
    assign rif.dout_ref        = golden_model.Ram_instance.dout_ref;
    assign rif.tx_valid_ref    = golden_model.Ram_instance.tx_valid_ref;

    assign wif.cs = SPI_slave_state_e'(DUT.SLAVE_instance.cs);

    Wrapper_sva wrapper_sva (wif);
    SPI_slave_sva spi_slave_sva (sif);
    Ram_sva ram_sva (rif);

    initial begin
        uvm_config_db#(virtual Ram_if)::set(null, "uvm_test_top", "ram_vif", rif);
        uvm_config_db#(virtual SPI_slave_if)::set(null, "uvm_test_top", "slave_vif", sif);
        uvm_config_db#(virtual Wrapper_if)::set(null, "uvm_test_top", "wrapper_vif", wif);
        run_test("Wrapper_test");
    end
endmodule
```

4.2.22 Run.do



```
vlib work
vlog -f src_files.list +cover -covercells +define+SIM
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all -cover
do wave.do
coverage save wrapper_top.ucdb -onexit
coverage exclude -du Wrapper_if -togglenode cs
coverage exclude -du SPI_slave_if -togglenode cs
coverage exclude -du SPI_slave_if -togglenode cs_ref
coverage exclude -src SPI_slave.sv -line 39 -code b
coverage exclude -src SPI_slave.sv -line 82 -code b
coverage exclude -src SPI_slave.sv -line 40 -code s
coverage exclude -src SPI_RAM.sv -line 28 -code b
coverage exclude -src SPI_RAM.sv -line 28 -code s
run -all
```

4.2.23 Bugs Found after wrapper verification

- **Bug 1: In Slave module**

  A one-cycle delay is needed before the MISO data transmission begins.

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

Fix:

```
READ_DATA : begin
     if (sif.tx_valid) begin
         sif.rx_valid <= 0;
         if (counter > 0) begin
             sif.MISO <= sif.tx_data[counter-1];
             counter <= counter - 1;
         end
         else begin
             received_address <= 0;
         end
     end
     else begin
         if (counter > 0) begin
             sif.rx_data[counter-1] <= sif.MOSI;
             counter <= counter - 1;
         end
         else begin
             sif.rx_valid <= 1;
             counter <= 9; //bug: add a cycle delay before sending MISO data
         end
     end
 end
```

- **Bug 2: In Ram module**
  The tx_valid signal should only be updated when rx_valid is high, not on every clock cycle.

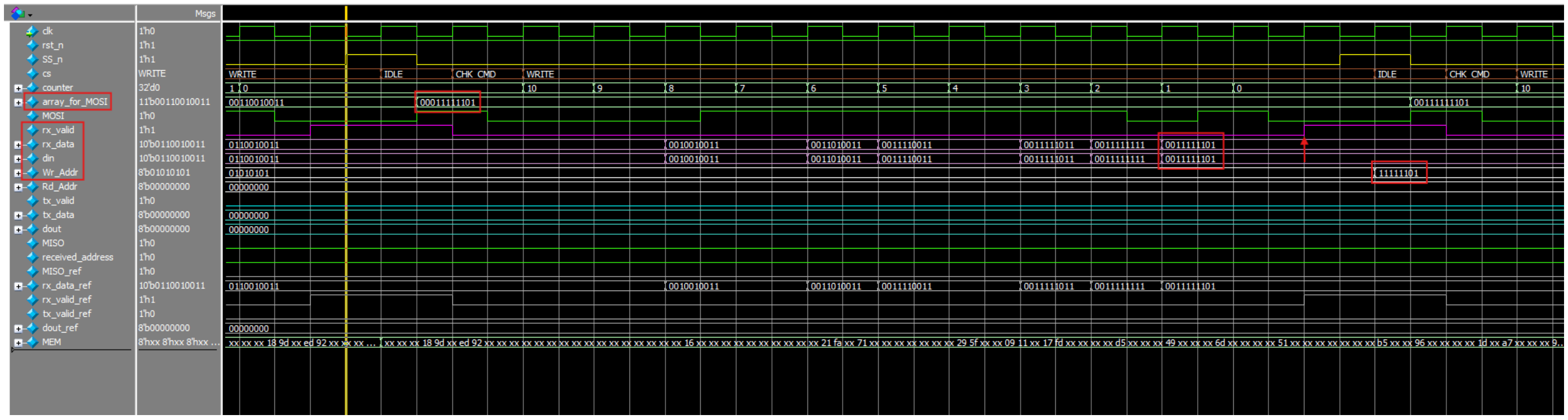
```
if (rif.rx_valid) begin
    case (rif.din[9:8])
        2'b00   : Wr_Addr       <= rif.din[7:0];
        2'b01   : MEM[Wr_Addr]  <= rif.din[7:0];
        2'b10   : Rd_Addr       <= rif.din[7:0];
        2'b11   : rif.dout      <= MEM[Rd_Addr];
        default : rif.dout      <= 0;
    endcase
end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

Fix:

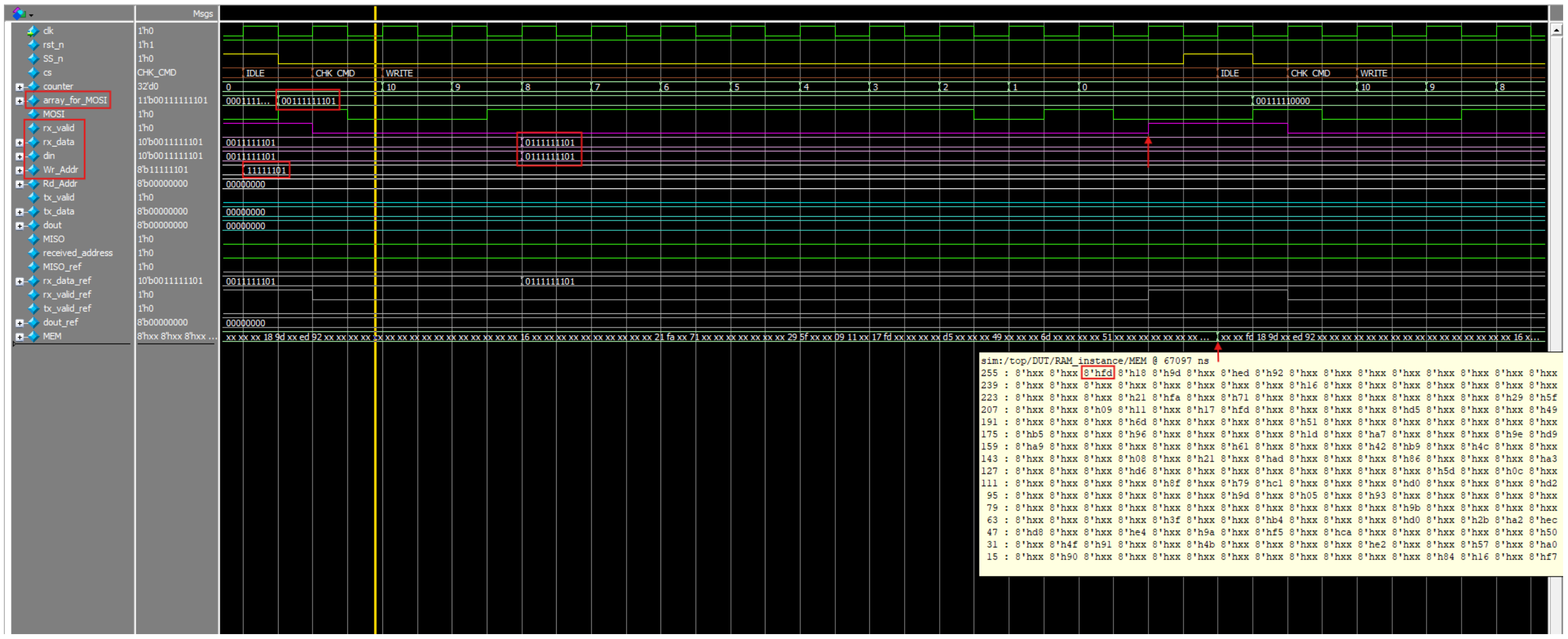
```
if (rif.rx_valid) begin
     case (rif.din[9:8])
         2'b00   : Wr_Addr       <= rif.din[7:0];
         2'b01   : MEM[Wr_Addr]  <= rif.din[7:0];
         2'b10   : Rd_Addr       <= rif.din[7:0];
         2'b11   : rif.dout      <= MEM[Rd_Addr];
         default : rif.dout      <= 0;
     endcase
     //bug: tx_valid signal should only be updated when rx_valid is high.
     rif.tx_valid <= (rif.din[9] && rif.din[8])? 1'b1 : 1'b0;
 end
```

4.2.24 Simulation results

- Write Address

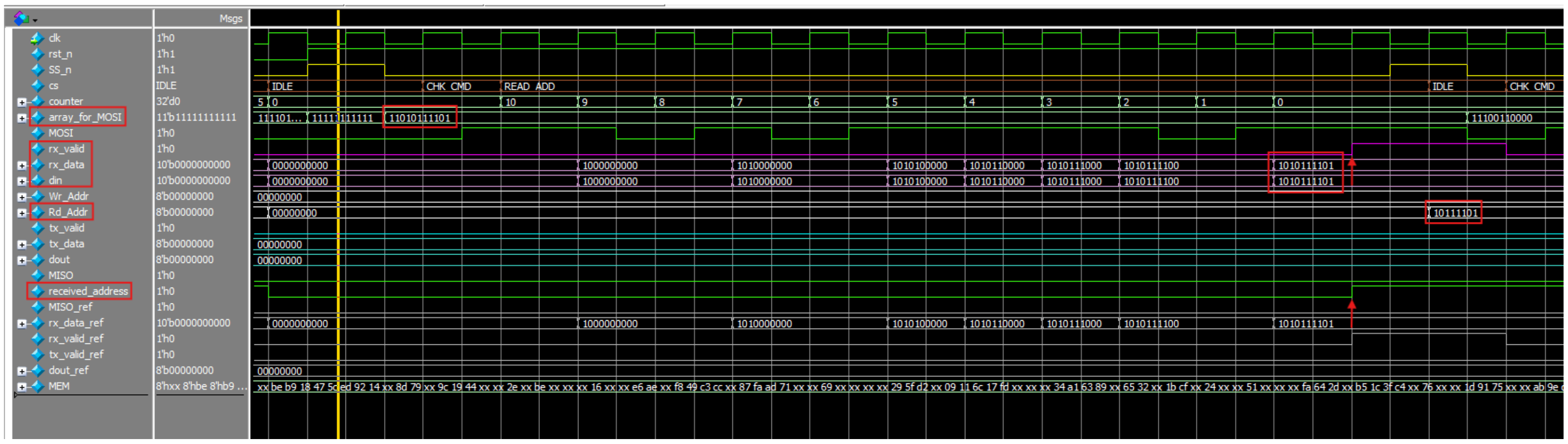


- Write Data

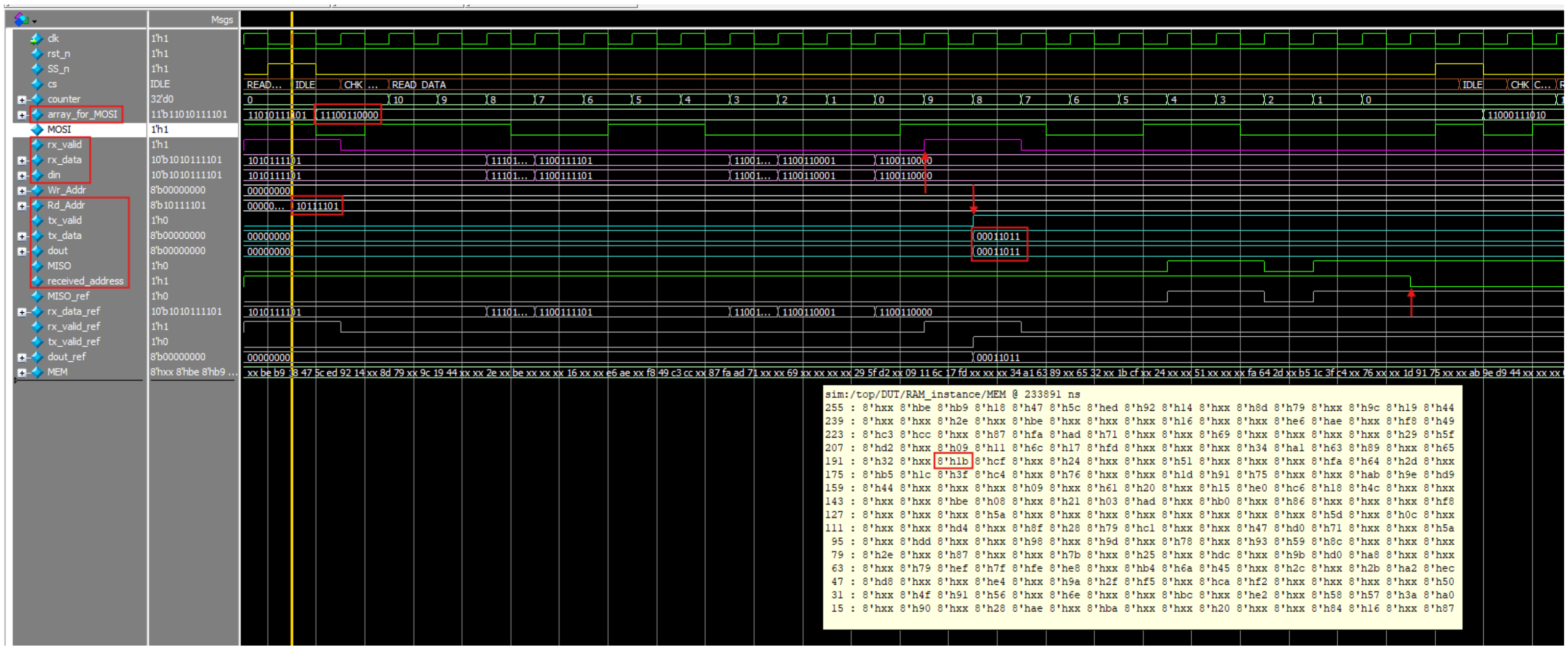


```
sim:/top/DUT/RAM_instance/MEM @ 67097 ns
255 : 8'hxx 8'hxx 8'hfd 8'h18 8'h9d 8'hxx 8'hed 8'h92 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
239 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h16 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
223 : 8'hxx 8'hxx 8'hxx 8'h21 8'hfa 8'hxx 8'h71 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h29 8'h5f
207 : 8'hxx 8'hxx 8'h09 8'h11 8'hxx 8'h17 8'hfd 8'hxx 8'hxx 8'hxx 8'hxx 8'hd5 8'hxx 8'hxx 8'hxx 8'h49
191 : 8'hxx 8'hxx 8'hxx 8'h6d 8'hxx 8'hxx 8'hxx 8'hxx 8'h51 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
175 : 8'hb5 8'hxx 8'hxx 8'h96 8'hxx 8'hxx 8'hxx 8'hxx 8'h1d 8'hxx 8'ha7 8'hxx 8'hxx 8'hxx 8'h9e 8'hd9
159 : 8'ha9 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h61 8'hxx 8'hxx 8'hxx 8'h42 8'hb9 8'hxx 8'h4c 8'hxx 8'hxx
143 : 8'hxx 8'hxx 8'hxx 8'h08 8'hxx 8'h21 8'hxx 8'had 8'hxx 8'hxx 8'hxx 8'h86 8'hxx 8'hxx 8'hxx 8'ha3
127 : 8'hxx 8'hxx 8'hxx 8'hd6 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h5d 8'hxx 8'h0c 8'hxx
111 : 8'hxx 8'hxx 8'hxx 8'hxx 8'h8f 8'hxx 8'h79 8'hc1 8'hxx 8'hxx 8'hxx 8'hd0 8'hxx 8'hxx 8'hxx 8'hd2
 95 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h9d 8'hxx 8'h05 8'hxx 8'h93 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx
 79 : 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h9b 8'hxx 8'hxx 8'hxx 8'hxx
 63 : 8'hxx 8'hxx 8'hxx 8'hxx 8'h3f 8'hxx 8'hxx 8'hb4 8'hxx 8'hxx 8'hxx 8'hd0 8'hxx 8'h2b 8'ha2 8'hec
 47 : 8'hd8 8'hxx 8'hxx 8'he4 8'hxx 8'h9a 8'hxx 8'hf5 8'hxx 8'hca 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h50
 31 : 8'hxx 8'h4f 8'h91 8'hxx 8'hxx 8'h4b 8'hxx 8'hxx 8'hxx 8'hxx 8'he2 8'hxx 8'hxx 8'h57 8'hxx 8'ha0
 15 : 8'hxx 8'h90 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h84 8'h16 8'hxx 8'hf7
```

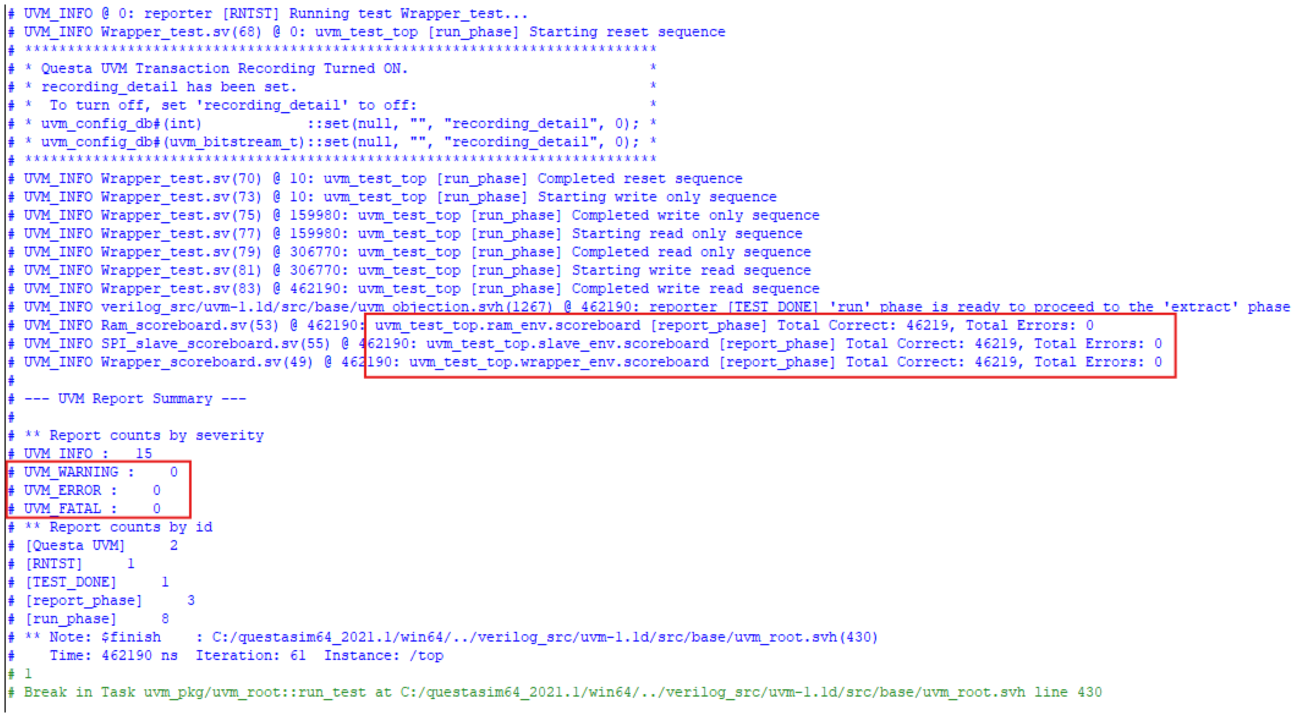
- Read Address



- Read Data



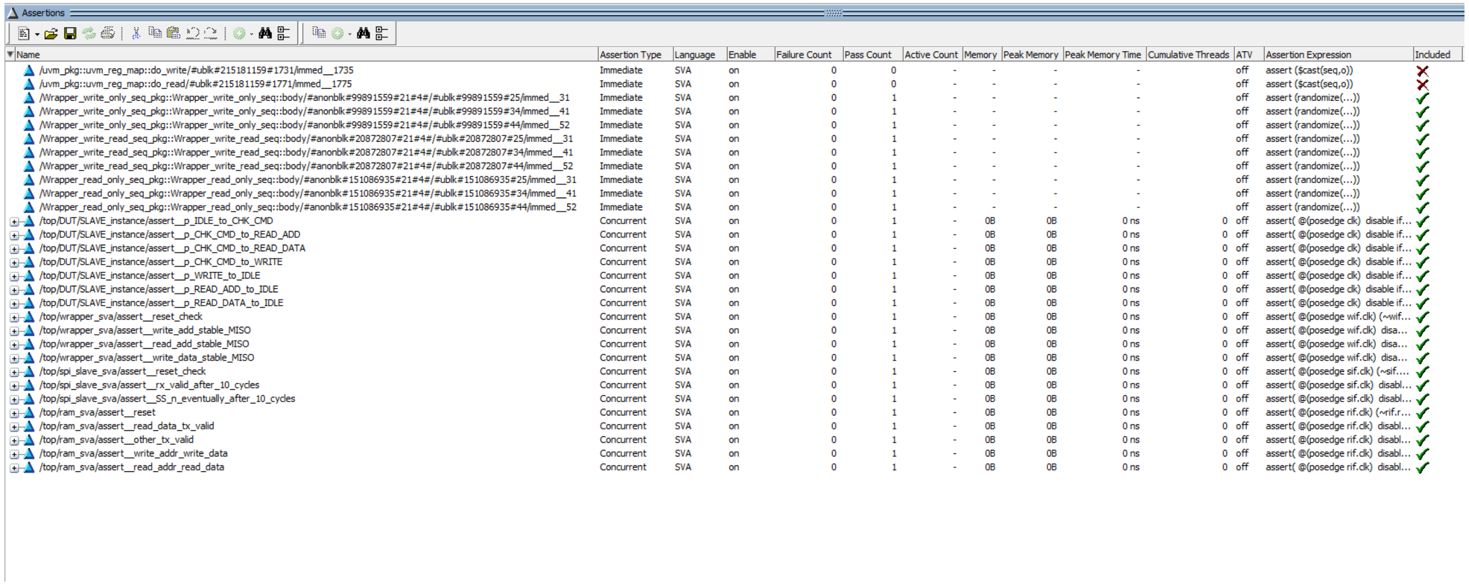
```
sim:/top/DUT/RAM_instance/MEM @ 233891 ns
255 : 8'hxx 8'hbe 8'hb9 8'h18 8'h47 8'h5c 8'hed 8'h92 8'h14 8'hxx 8'h8d 8'h79 8'hxx 8'h9c 8'h19 8'h44
239 : 8'hxx 8'hxx 8'h2e 8'hxx 8'hbe 8'hxx 8'hxx 8'hxx 8'h16 8'hxx 8'hxx 8'he6 8'hae 8'hxx 8'hf8 8'h49
223 : 8'hc3 8'hcc 8'hxx 8'h87 8'hfa 8'had 8'h71 8'hxx 8'hxx 8'h69 8'hxx 8'hxx 8'hxx 8'hxx 8'h29 8'h5f
207 : 8'hd2 8'hxx 8'h09 8'h11 8'h6c 8'h17 8'hfd 8'hxx 8'hxx 8'hxx 8'h34 8'ha1 8'h63 8'h89 8'hxx 8'h65
191 : 8'h32 8'hxx 8'h1b 8'hcf 8'hxx 8'h24 8'hxx 8'hxx 8'h51 8'hxx 8'hxx 8'hxx 8'hfa 8'h64 8'h2d 8'hxx
175 : 8'hb5 8'h1c 8'h3f 8'hc4 8'hxx 8'h76 8'hxx 8'hxx 8'h1d 8'h91 8'h75 8'hxx 8'hxx 8'hab 8'h9e 8'hd9
159 : 8'h44 8'hxx 8'hxx 8'hxx 8'h09 8'hxx 8'h61 8'h20 8'hxx 8'h15 8'he0 8'hc6 8'h18 8'h4c 8'hxx 8'hxx
143 : 8'hxx 8'hxx 8'hbe 8'h08 8'hxx 8'h21 8'h03 8'had 8'hxx 8'hb0 8'hxx 8'h86 8'hxx 8'hxx 8'hxx 8'hf8
127 : 8'hxx 8'hxx 8'hxx 8'h5a 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'hxx 8'h5d 8'hxx 8'h0c 8'hxx
111 : 8'hxx 8'hxx 8'hd4 8'hxx 8'h8f 8'h28 8'h79 8'hc1 8'hxx 8'hxx 8'h47 8'hd0 8'h71 8'hxx 8'hxx 8'h5a
 95 : 8'hxx 8'hdd 8'hxx 8'hxx 8'h98 8'hxx 8'h9d 8'hxx 8'h78 8'hxx 8'h93 8'h59 8'h8c 8'hxx 8'hxx 8'hxx
 79 : 8'h2e 8'hxx 8'h87 8'hxx 8'hxx 8'h7b 8'hxx 8'h25 8'hxx 8'hdc 8'hxx 8'h9b 8'hd0 8'ha8 8'hxx 8'hxx
 63 : 8'hxx 8'h79 8'hef 8'h7f 8'hfe 8'he8 8'hxx 8'hb4 8'h6a 8'h45 8'hxx 8'h2c 8'hxx 8'h2b 8'ha2 8'hec
 47 : 8'hd8 8'hxx 8'hxx 8'he4 8'hxx 8'h9a 8'h2f 8'hf5 8'hxx 8'hca 8'hf2 8'hxx 8'hxx 8'hxx 8'hxx 8'h50
 31 : 8'hxx 8'h4f 8'h91 8'h56 8'hxx 8'h6e 8'hxx 8'hxx 8'hbc 8'hxx 8'he2 8'hxx 8'h58 8'h57 8'h3a 8'ha0
 15 : 8'hxx 8'h90 8'hxx 8'h28 8'hae 8'hxx 8'hba 8'hxx 8'hxx 8'h20 8'hxx 8'hxx 8'h84 8'h16 8'hxx 8'h87
```



```
# UVM_INFO @ 0: reporter [RNTST] Running test Wrapper_test...
# UVM_INFO Wrapper_test.sv(68) @ 0: uvm_test_top [run_phase] Starting reset sequence
# **************************************************************************
# * 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(70) @ 10: uvm_test_top [run_phase] Completed reset sequence
# UVM_INFO Wrapper_test.sv(73) @ 10: uvm_test_top [run_phase] Starting write only sequence
# UVM_INFO Wrapper_test.sv(75) @ 159980: uvm_test_top [run_phase] Completed write only sequence
# UVM_INFO Wrapper_test.sv(77) @ 159980: uvm_test_top [run_phase] Starting read only sequence
# UVM_INFO Wrapper_test.sv(79) @ 306770: uvm_test_top [run_phase] Completed read only sequence
# UVM_INFO Wrapper_test.sv(81) @ 306770: uvm_test_top [run_phase] Starting write read sequence
# UVM_INFO Wrapper_test.sv(83) @ 462190: uvm_test_top [run_phase] Completed write read sequence
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 462190: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO Ram_scoreboard.sv(53) @ 462190: uvm_test_top.ram_env.scoreboard [report_phase] Total Correct: 46219, Total Errors: 0
# UVM_INFO SPI_slave_scoreboard.sv(55) @ 462190: uvm_test_top.slave_env.scoreboard [report_phase] Total Correct: 46219, Total Errors: 0
# UVM_INFO Wrapper_scoreboard.sv(49) @ 462190: uvm_test_top.wrapper_env.scoreboard [report_phase] Total Correct: 46219, Total Errors: 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]     3
# [run_phase]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 462190 ns  Iteration: 61  Instance: /top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

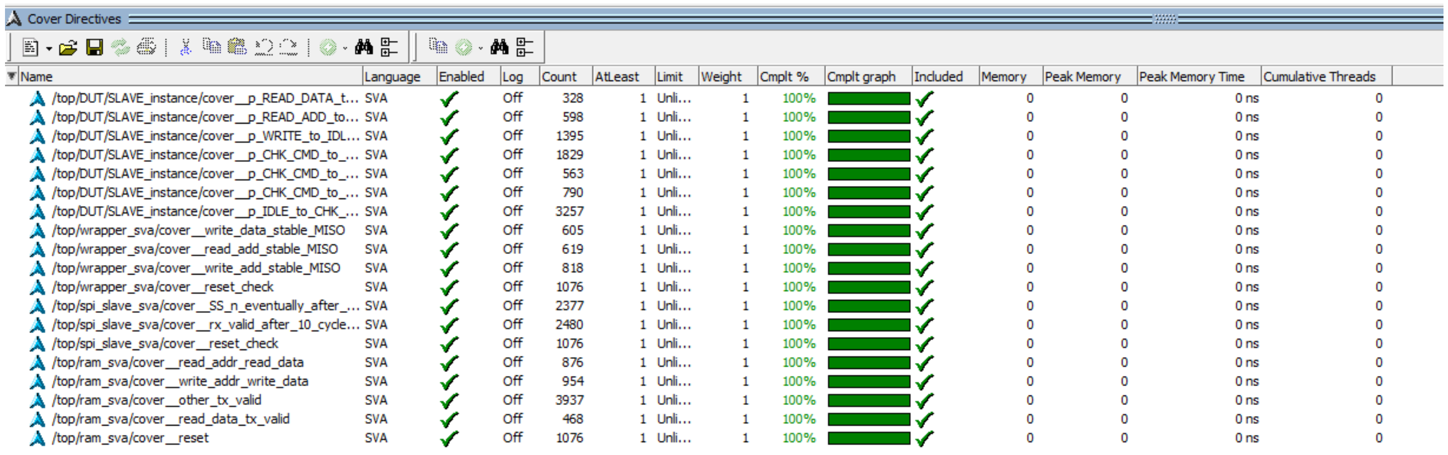
There are no errors

4.2.25 Assertions results



Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
/Wrapper_write_only_seq_pkg::Wrapper_write_only_seq::body/#anonblk#99891559#21#4#/#ublk#99891559#25/immed__31	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_write_only_seq_pkg::Wrapper_write_only_seq::body/#anonblk#99891559#21#4#/#ublk#99891559#34/immed__41	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_write_only_seq_pkg::Wrapper_write_only_seq::body/#anonblk#99891559#21#4#/#ublk#99891559#44/immed__52	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_write_read_seq_pkg::Wrapper_write_read_seq::body/#anonblk#20872807#21#4#/#ublk#20872807#25/immed__31	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_write_read_seq_pkg::Wrapper_write_read_seq::body/#anonblk#20872807#21#4#/#ublk#20872807#34/immed__41	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_write_read_seq_pkg::Wrapper_write_read_seq::body/#anonblk#20872807#21#4#/#ublk#20872807#44/immed__52	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_read_only_seq_pkg::Wrapper_read_only_seq::body/#anonblk#151086935#21#4#/#ublk#151086935#25/immed__31	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_read_only_seq_pkg::Wrapper_read_only_seq::body/#anonblk#151086935#21#4#/#ublk#151086935#34/immed__41	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/Wrapper_read_only_seq_pkg::Wrapper_read_only_seq::body/#anonblk#151086935#21#4#/#ublk#151086935#44/immed__52	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
/top/DUT/SLAVE_instance/assert__p_IDLE_to_CHK_CMD	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/DUT/SLAVE_instance/assert__p_CHK_CMD_to_READ_ADD	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/DUT/SLAVE_instance/assert__p_CHK_CMD_to_READ_DATA	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/DUT/SLAVE_instance/assert__p_CHK_CMD_to_WRITE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/DUT/SLAVE_instance/assert__p_WRITE_to_IDLE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/DUT/SLAVE_instance/assert__p_READ_ADD_to_IDLE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/DUT/SLAVE_instance/assert__p_READ_DATA_to_IDLE	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable if...	✓
/top/wrapper_sva/assert__reset_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge wif.clk) (~wif...	✓
/top/wrapper_sva/assert__write_add_stable_MISO	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge wif.clk) disa...	✓
/top/wrapper_sva/assert__read_add_stable_MISO	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge wif.clk) disa...	✓
/top/wrapper_sva/assert__write_data_stable_MISO	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge wif.clk) disa...	✓
/top/spi_slave_sva/assert__reset_check	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) (~sif....	✓
/top/spi_slave_sva/assert__rx_valid_after_10_cycles	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/spi_slave_sva/assert__SS_n_eventually_after_10_cycles	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge sif.clk) disabl...	✓
/top/ram_sva/assert__reset	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) (~rif.r...	✓
/top/ram_sva/assert__read_data_tx_valid	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓
/top/ram_sva/assert__other_tx_valid	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓
/top/ram_sva/assert__write_addr_write_data	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓
/top/ram_sva/assert__read_addr_read_data	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge rif.clk) disabl...	✓



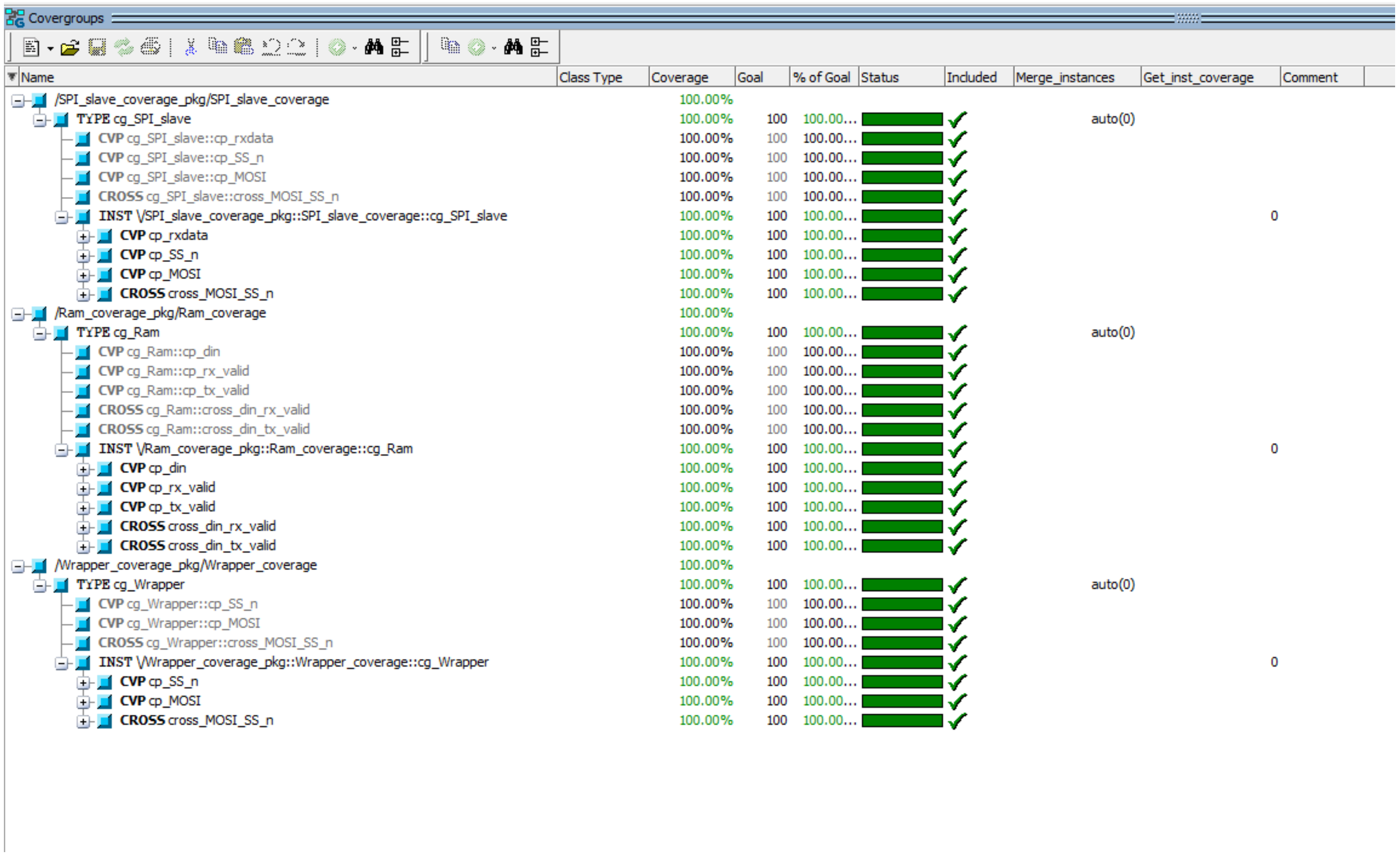
Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/DUT/SLAVE_instance/cover__p_READ_DATA_t...	SVA	✓	Off	328	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SLAVE_instance/cover__p_READ_ADD_to...	SVA	✓	Off	598	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SLAVE_instance/cover__p_WRITE_to_IDL...	SVA	✓	Off	1395	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SLAVE_instance/cover__p_CHK_CMD_to_...	SVA	✓	Off	1829	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SLAVE_instance/cover__p_CHK_CMD_to_...	SVA	✓	Off	563	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SLAVE_instance/cover__p_CHK_CMD_to_...	SVA	✓	Off	790	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/DUT/SLAVE_instance/cover__p_IDLE_to_CHK_...	SVA	✓	Off	3257	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/wrapper_sva/cover__write_data_stable_MISO	SVA	✓	Off	605	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/wrapper_sva/cover__read_add_stable_MISO	SVA	✓	Off	619	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/wrapper_sva/cover__write_add_stable_MISO	SVA	✓	Off	818	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/wrapper_sva/cover__reset_check	SVA	✓	Off	1076	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/spi_slave_sva/cover__SS_n_eventually_after_...	SVA	✓	Off	2377	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/spi_slave_sva/cover__rx_valid_after_10_cycle...	SVA	✓	Off	2480	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/spi_slave_sva/cover__reset_check	SVA	✓	Off	1076	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/ram_sva/cover__read_addr_read_data	SVA	✓	Off	876	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/ram_sva/cover__write_addr_write_data	SVA	✓	Off	954	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/ram_sva/cover__other_tx_valid	SVA	✓	Off	3937	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/ram_sva/cover__read_data_tx_valid	SVA	✓	Off	468	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0
/top/ram_sva/cover__reset	SVA	✓	Off	1076	1	Unli...	1	100%	🟩	✓	0	0	0 ns	0

There are no assertion errors

4.2.26 Coverage results

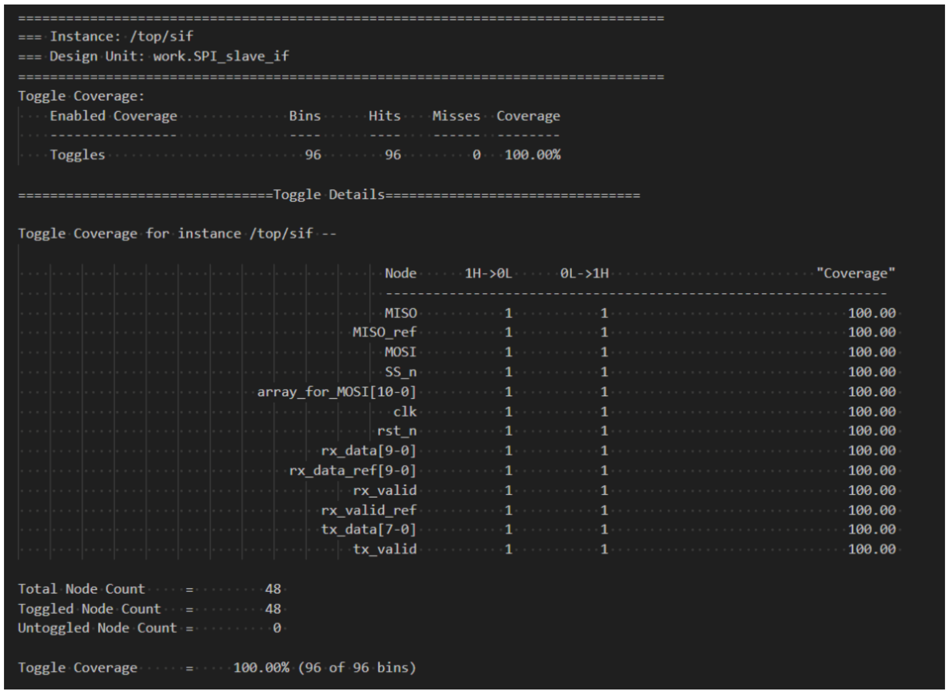
- Functional coverage



Covergroups

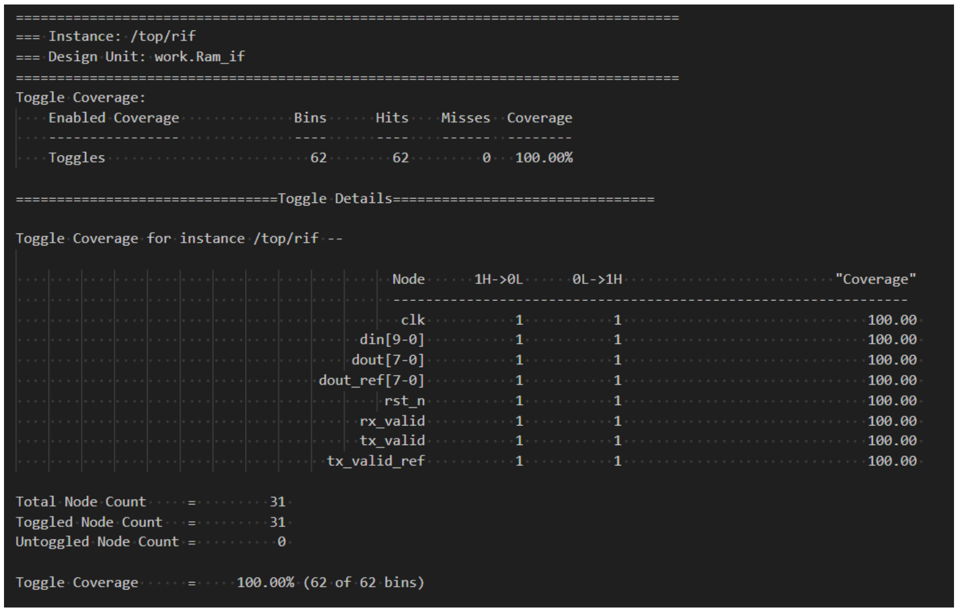
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/SPI_slave_coverage_pkg/SPI_slave_coverage		100.00%							
TYPE cg_SPI_slave		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP cg_SPI_slave::cp_rxdata		100.00%	100	100.00...	🟩	✓			
CVP cg_SPI_slave::cp_SS_n		100.00%	100	100.00...	🟩	✓			
CVP cg_SPI_slave::cp_MOSI		100.00%	100	100.00...	🟩	✓			
CROSS cg_SPI_slave::cross_MOSI_SS_n		100.00%	100	100.00...	🟩	✓			
INST \/SPI_slave_coverage_pkg::SPI_slave_coverage::cg_SPI_slave		100.00%	100	100.00...	🟩	✓		0	
CVP cp_rxdata		100.00%	100	100.00...	🟩	✓			
CVP cp_SS_n		100.00%	100	100.00...	🟩	✓			
CVP cp_MOSI		100.00%	100	100.00...	🟩	✓			
CROSS cross_MOSI_SS_n		100.00%	100	100.00...	🟩	✓			
/Ram_coverage_pkg/Ram_coverage		100.00%							
TYPE cg_Ram		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP cg_Ram::cp_din		100.00%	100	100.00...	🟩	✓			
CVP cg_Ram::cp_rx_valid		100.00%	100	100.00...	🟩	✓			
CVP cg_Ram::cp_tx_valid		100.00%	100	100.00...	🟩	✓			
CROSS cg_Ram::cross_din_rx_valid		100.00%	100	100.00...	🟩	✓			
CROSS cg_Ram::cross_din_tx_valid		100.00%	100	100.00...	🟩	✓			
INST \/Ram_coverage_pkg::Ram_coverage::cg_Ram		100.00%	100	100.00...	🟩	✓		0	
CVP cp_din		100.00%	100	100.00...	🟩	✓			
CVP cp_rx_valid		100.00%	100	100.00...	🟩	✓			
CVP cp_tx_valid		100.00%	100	100.00...	🟩	✓			
CROSS cross_din_rx_valid		100.00%	100	100.00...	🟩	✓			
CROSS cross_din_tx_valid		100.00%	100	100.00...	🟩	✓			
/Wrapper_coverage_pkg/Wrapper_coverage		100.00%							
TYPE cg_Wrapper		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP cg_Wrapper::cp_SS_n		100.00%	100	100.00...	🟩	✓			
CVP cg_Wrapper::cp_MOSI		100.00%	100	100.00...	🟩	✓			
CROSS cg_Wrapper::cross_MOSI_SS_n		100.00%	100	100.00...	🟩	✓			
INST \/Wrapper_coverage_pkg::Wrapper_coverage::cg_Wrapper		100.00%	100	100.00...	🟩	✓		0	
CVP cp_SS_n		100.00%	100	100.00...	🟩	✓			
CVP cp_MOSI		100.00%	100	100.00...	🟩	✓			
CROSS cross_MOSI_SS_n		100.00%	100	100.00...	🟩	✓			

- Code coverage
  - Toggle Coverage
    - Slave interface



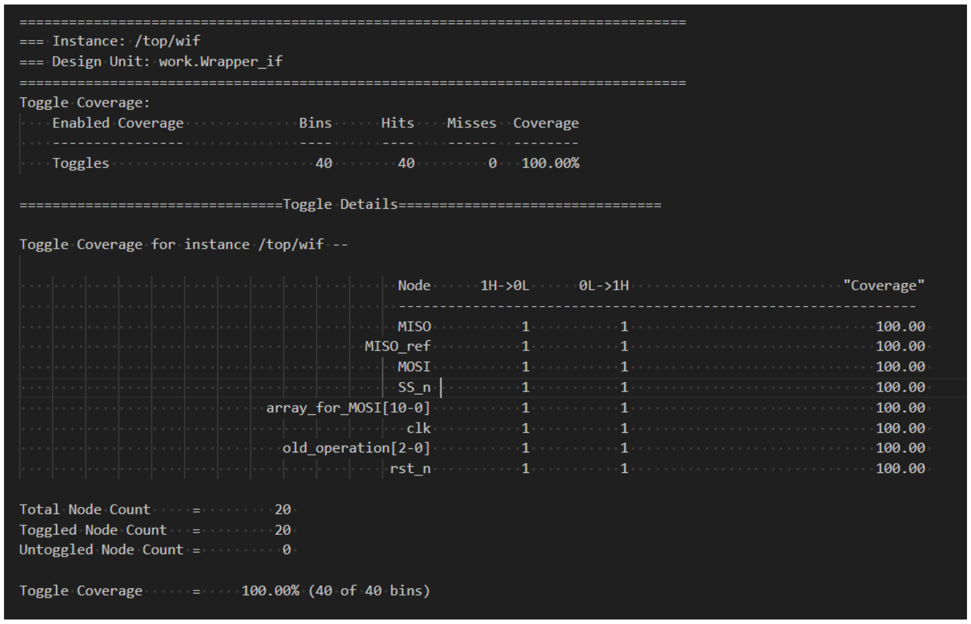
```
===============================================================================
=== Instance: /top/sif
=== Design Unit: work.SPI_slave_if
===============================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         96        96         0   100.00%

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

Toggle Coverage for instance /top/sif --

                                              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
                              array_for_MOSI[10-0]           1           1                              100.00
                                               clk           1           1                              100.00
                                             rst_n           1           1                              100.00
                                      rx_data[9-0]           1           1                              100.00
                                  rx_data_ref[9-0]           1           1                              100.00
                                          rx_valid           1           1                              100.00
                                      rx_valid_ref           1           1                              100.00
                                      tx_data[7-0]           1           1                              100.00
                                          tx_valid           1           1                              100.00

Total Node Count     =         48
Toggled Node Count   =         48
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (96 of 96 bins)
```

    - Ram interface



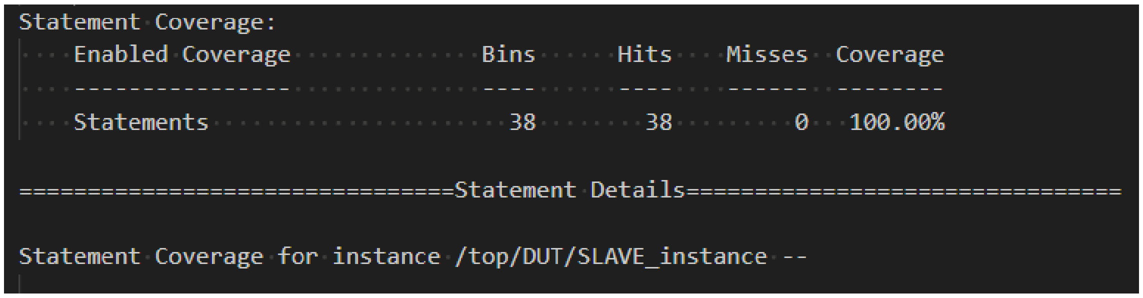
```
===============================================================================
=== Instance: /top/rif
=== Design Unit: work.Ram_if
===============================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage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

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

Toggle Coverage for instance /top/rif --

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

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

Toggle Coverage      =     100.00% (62 of 62 bins)
```

- Wrapper interface



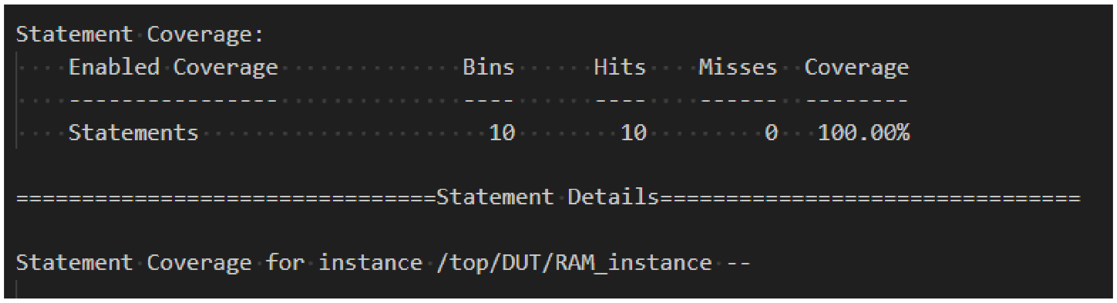
```
=================================================================================
=== Instance: /top/wif
=== Design Unit: work.Wrapper_if
=================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         40        40         0   100.00%

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

Toggle Coverage for instance /top/wif --

                                              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
                                array_for_MOSI[10-0]           1           1                              100.00
                                               clk           1           1                              100.00
                                  old_operation[2-0]           1           1                              100.00
                                             rst_n           1           1                              100.00

Total Node Count     =         20
Toggled Node Count   =         20
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (40 of 40 bins)
```

- Statement Coverage
    - Slave module



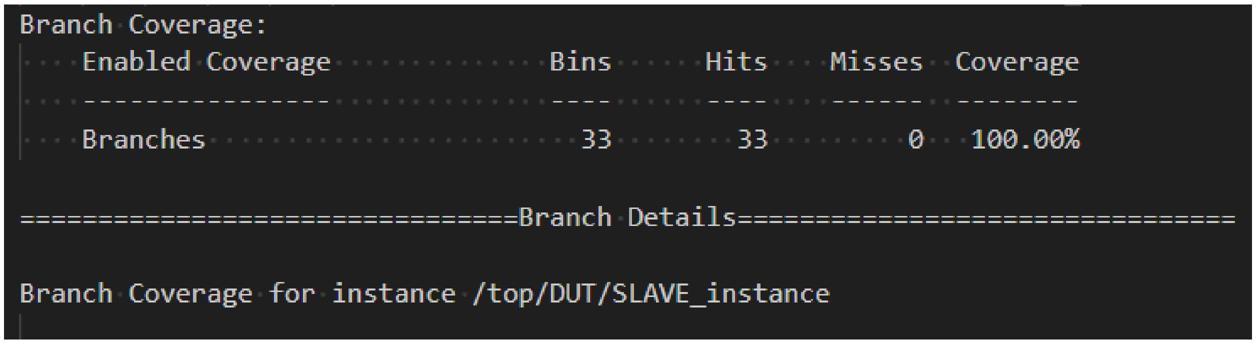
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      38        38         0   100.00%

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

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

- Ram module



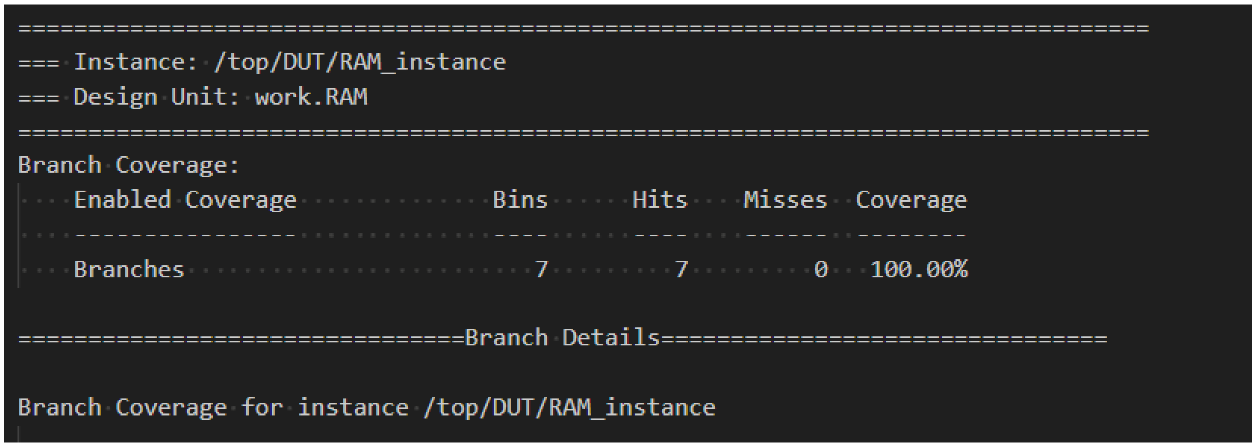
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      10        10         0   100.00%

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

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

- Branch coverage
  - Slave module



```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        33        33         0   100.00%

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

Branch Coverage for instance /top/DUT/SLAVE_instance
```

  - Ram module



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

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

Branch Coverage for instance /top/DUT/RAM_instance
```