Project Description

This project aims to build and verify a complete UVM-based testbench for an SPI Wrapper system composed of a SPI Slave and a Single-Port RAM. It is divided into three main parts:

1. SPI Slave– Develop a UVM environment to verify the SPI Slave protocol, including random stimulus generation, assertions, and coverage.
2. RAM– Create a UVM environment to validate RAM read/write operations and data integrity.
3. SPI Wrapper– Integrate both environments to verify full system functionality and demonstrate UVM reusability.

UVM-Based verification of SPI wrapper system

Team Name: Sensitivity_List.

Radwa Yasser

Sherif Ahmed

Rofayda Alaa

Special thank to Eng. Kareem waseem

UVM diagram:

SPI Verification:

In this part, create a full UVM environment to verify the SPI slave design.

Verification Plan:

Label	Design requirement description	Stimulus generation	Functional coverage	Functionality check
Reset sequence	Check reset	Directed	-	A checker in testbench to make sure the output is correct
Main sequence	Check other randomize cases	Randomize Under constraints: 1- The reset signal (rst_n) shall be deasserted most of the time. 2- The SS_n signal to be high for one cycle every 13 cycles for all cases except read data to be high for one cycle every 23 cycles.	1- Add coverpoints on rx_data[9:8] to take all possible values and all possible transitions. 2- Add coverpoints on SS_n to capture: • Check full transaction duration: 1	A checker in testbench to make sure the output is correct

		3- Declare a randomized array to drive bit by bit the MOSI and the first 3 bits sent serially to the MOSI after the SS_n falls and reach chk_cmd are valid combinations only (000, 001, 110, 111). 4- The tx_valid signal to be high in case of read data.	→ 0 [*13] → 1 for normal operations. • Check extended transaction: 1 → 0 [*23] → 1 for READ_DATA. 3- Add coverpoints on MOSI to validate correct transitions: • 000 (Write Address). • 001 (Write Data). • 110 (Read Address). • 111 (Read Data). 4- Cross coverage between SS_n and MOSI bins (Exclude irrelevant bins to focus on legal operation scenarios).	

Design with corrected bugs:

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

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

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

reg [3:0] counter;
reg       received_address;

reg [2:0] cs, ns;

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

always @(*) begin
    case (cs)
        IDLE : begin
```

```
            if (SS_n)
                ns = IDLE;
            else
                ns = CHK_CMD;
        end
        CHK_CMD : begin
            if (SS_n)
                ns = IDLE;
            else begin
                if (~MOSI)
                    ns = WRITE;
                else begin
                    if (!received_address)   //When received address = 0 then
READ ADDRESS
                        ns = READ_ADD;
                    else
                        ns = READ_DATA;
                end
            end
        end
        WRITE : begin
            if (SS_n)
                ns = IDLE;
            else
                ns = WRITE;
        end
        READ_ADD : begin
            if (SS_n)
                ns = IDLE;
            else
                ns = READ_ADD;
        end
        READ_DATA : begin
            if (SS_n)
                ns = IDLE;
            else
                ns = READ_DATA;
        end

    endcase

end

always @(posedge clk) begin
    if (~rst_n) begin
```

```
        rx_data <= 0;
        rx_valid <= 0;
        received_address <= 0;
        MISO <= 0;
        counter<=0;           //counter must be reseted
    end

    else begin
        case (cs)
            IDLE : begin
                rx_valid <= 0;
            end
            CHK_CMD : begin
                counter <= 10;
            end
            WRITE : begin
                if (counter > 0) begin
                    rx_data[counter-1] <= MOSI;
                    counter <= counter - 1;
                end
                else begin
                    rx_valid <= 1;
                end
            end
            READ_ADD : begin
                if (counter > 0) begin
                    rx_data[counter-1] <= MOSI;
                    counter <= counter - 1;
                end
                else begin
                    rx_valid <= 1;
                    received_address <= 1;
                end
            end

            READ_DATA : begin
                //cycle 13   13 14 15 16 17 18 19 20  21 22
                if (tx_valid) begin
                    rx_valid <= 0;
                    if (counter > 0) begin
                        MISO <= tx_data[counter-1];
                        counter <= counter - 1;
                    end
                    else begin
                        received_address <= 0;
```

```
                    end
                end

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

                    else begin
                        rx_valid <= 1;
                        counter <= 9;           //counter be 9 to take MSB of tx
data to MISO
                    end

                end
            end

        endcase
    end
end

`ifdef SIM
//1- An assertion ensures that whenever reset is asserted, the outputs (MISO,
rx_valid, and rx_data) are all low.

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

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

property chkcmd_ra;
    @(posedge clk)  (cs == CHK_CMD &&rst_n && !SS_n && MOSI && !received_address)
|=> (cs == READ_ADD);
endproperty
property chkcmd_rd;
    @(posedge clk) (cs == CHK_CMD &&rst_n && !SS_n && MOSI && received_address)
|=> (cs == READ_DATA);
endproperty
property write_idle;
    @(posedge clk) (cs == WRITE&&rst_n&&SS_n) |=> (cs == IDLE);
```

```
endproperty

property ra_idle;
    @(posedge clk) (cs == READ_ADD&&rst_n&&SS_n) |=> (cs == IDLE);
endproperty

    property rd_idle;
    @(posedge clk) (cs == READ_DATA&&rst_n&&SS_n) |=> (cs == IDLE);
endproperty



assert property (chkcmd_idle);
cover  property (chkcmd_idle);

assert property (chkcmd_write);
cover  property (chkcmd_write);

assert property (chkcmd_ra);
cover  property (chkcmd_ra);

assert property (chkcmd_rd);
cover  property (chkcmd_rd);

assert property (write_idle);
cover  property (write_idle);

assert property (ra_idle);
cover  property (ra_idle);

assert property (rd_idle);
cover  property (rd_idle);


`endif

endmodule
```

There, s three bugs:

At address received signal

Assert the counter when reset

Counter should be 9 to take tx data with its MSB to MISO

Golden module:

```
module spi_slave (
    input   clk,
    input   rst_n,
    input   MOSI,
    output reg MISO,
    input   SS_n,

    output reg [9:0] rx_data,

    output reg rx_valid,

    input  tx_valid,
    input [7:0] tx_data
);

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

    reg  [2:0] current_state, next_state;
   reg signed [5:0] bit_cnt_rx;
    reg addr_received;


    always @(posedge clk ) begin
        if (!rst_n)
            current_state <= IDLE;
        else
            current_state <= next_state;
    end

    always @(*) begin
        case (current_state)
            IDLE:
                if (!SS_n)
                    next_state = CHK_CMD;
                else
                    next_state = IDLE;

            CHK_CMD:
                if (!SS_n)
```

```
                    next_state = (MOSI == 1'b0) ? WRITE :(addr_received == 1'b0)?
READ_ADD:READ_DATA;
                else
                    next_state = IDLE;

            WRITE:
                if (SS_n)
                    next_state = IDLE;
                else
                    next_state=WRITE;

            READ_ADD:
                if (SS_n)
                    next_state = IDLE;
                else
                    next_state = READ_ADD;

            READ_DATA:
                if (SS_n)
                    next_state = IDLE;
                else
                    next_state = READ_DATA;

            default:
                    next_state = IDLE;
        endcase
    end
//output_logic
    always @(posedge clk) begin
        if (!rst_n) begin
            rx_data       <= 0;
            rx_valid      <= 0;
            MISO          <= 0;
            bit_cnt_rx       <= 0;
            addr_received <= 0;
        end
        else begin

            case (current_state)
                WRITE: begin
                  if (bit_cnt_rx <= 9) begin
                     rx_data <= {rx_data[8:0], MOSI};
                          bit_cnt_rx <= bit_cnt_rx + 1;
                        end
                    else if (bit_cnt_rx == 10) begin
```

```
                    rx_valid <= 1;
            end
        end
        READ_ADD: begin

                if (bit_cnt_rx <= 9) begin
            rx_data <= {rx_data[8:0], MOSI};
                    bit_cnt_rx <= bit_cnt_rx + 1;
                end
            else if  (bit_cnt_rx == 10) begin
                addr_received<=1;
                 rx_valid <= 1;
            end

        end

        READ_DATA:begin
            if (!tx_valid)begin
         rx_valid<=0;
                    if (bit_cnt_rx <= 9) begin
            rx_data <= {rx_data[8:0], MOSI};
                    bit_cnt_rx <= bit_cnt_rx + 1;
                end
            else if  (bit_cnt_rx == 10) begin
                rx_valid <= 1;
               bit_cnt_rx<=-1;

            end

            end
            else  begin
                    rx_valid<=0;
                if(bit_cnt_rx <= 7)begin
                MISO <= tx_data[7 - bit_cnt_rx];
                 bit_cnt_rx <= bit_cnt_rx + 1;
                end

               else if (bit_cnt_rx == 8) begin
                    // bit_cnt_tx <= 0;
                     addr_received <= 0;
                end
            end
        end
        default: begin
       bit_cnt_rx <= 0;
```

```
                rx_valid <= 0;
                end
            endcase
        end
 end
endmodule
```

SPI shared package:

```
package SPI_shared_pkg;
int error_count;
int correct_count;
int error_count_2;
int correct_count_2;
int error_count_r;
int correct_count_r;
int error_count_w;
int correct_count_w;
int error_count_s;
int correct_count_s;
int error_count_2_s;
int correct_count_2_s;
int Count_seq, i, address_recieved;
endpackage
```

SPI interface:

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

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

SPI assertions:

```
module SPI_sva (SPI_if.svaa SPIif);
sequence write_add;
    SPIif.SS_n == 1 ##2 SPIif.MOSI == 0 ##1 SPIif.MOSI == 0 ##1 SPIif.MOSI == 0;
endsequence
```

```
sequence read_add;
    SPIif.SS_n == 1 ##2 SPIif.MOSI == 1 ##1 SPIif.MOSI == 1 ##1 SPIif.MOSI == 0;
endsequence

sequence write_data;
    SPIif.SS_n == 1 ##2 SPIif.MOSI == 0 ##1 SPIif.MOSI == 0 ##1 SPIif.MOSI == 1;
endsequence
sequence read_data;
    SPIif.SS_n == 1 ##2 SPIif.MOSI == 1 ##1 SPIif.MOSI == 1 ##1 SPIif.MOSI == 1;
endsequence

sequence ss_n_seq;
    ##10 SPIif.rx_valid ##1 SPIif.SS_n[->1];
endsequence

property rst_is_ok;
    @(posedge SPIif.clk) (!SPIif.rst_n) |=> (SPIif.MISO==0 &&
SPIif.rx_valid==0  && (SPIif.rx_data ==0));
endproperty
property rx_valid_p;
    @(posedge SPIif.clk) disable iff (!SPIif.rst_n)
    (write_add or read_add or write_data or read_data) |-> ss_n_seq;
endproperty

assert property (rst_is_ok);
cover  property (rst_is_ok);

assert property (rx_valid_p);
cover  property (rx_valid_p);
endmodule
```

SPI config:

```
package SPI_config_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class SPI_config extends uvm_object;
 `uvm_object_utils(SPI_config);
virtual SPI_if SPI_config_vif;
function new (string name = "SPI_config");
    super.new(name);
endfunction

endclass
endpackage
```

SPI sequence item:

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

 class SPI_seq_item extends uvm_sequence_item;
    `uvm_object_utils(SPI_seq_item)


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

    typedef enum {IDLE, CHK_CMD, WRITE, READ_ADD, READ_DATA} states;
 typedef enum bit [2:0]{WR_ADDR=3'b000, WR_DATA=3'b001, RD_ADDR=3'b110,
RD_DATA=3'b111} statess;


    rand bit MOSI, rst_n, SS_n;
    rand bit [7:0] tx_data;
    bit [7:0] old_tx_data;
    rand bit tx_valid;
    rand bit [10:0] mosi_arr;


    bit  rx_valid, MISO;
    bit [9:0] rx_data;
  bit [9:0] rx_data_ref;
bit rx_valid_ref, MISO_ref;


    bit [10:0] old_mosi_arr;
    states cs;
    rand states ns;



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

    constraint ns_c {

      if (!rst_n || SS_n) { ns == IDLE; }
      else if (cs == IDLE && !SS_n) { ns == CHK_CMD; }
```

```
      else if (cs == CHK_CMD && !SS_n && !MOSI) { ns == WRITE; }
      else if (cs == CHK_CMD && !SS_n && MOSI && address_recieved == 0) { ns ==
READ_ADD; }
      else if (cs == CHK_CMD && !SS_n && MOSI && address_recieved == 1) { ns ==
READ_DATA; }
      else if ((cs == WRITE) && Count_seq != 13) { ns == cs; }
      else if ((cs == READ_ADD) && Count_seq != 13) { ns == cs; }
      else if ((cs == READ_DATA) && Count_seq != 23) { ns == cs; }
    }

        constraint SS_n_c {

      if (cs != READ_DATA) {
        (Count_seq == 13) -> SS_n == 1;
        (Count_seq != 13) -> SS_n == 0;
      }
      else if (cs == READ_DATA) {
        (Count_seq == 23) -> SS_n == 1;
        (Count_seq != 23) -> SS_n == 0;
      }
    }

constraint mosi_arr_c {
  if (!rst_n || Count_seq == 0) {
        if(old_mosi_arr[10:8] == WR_ADDR) //Write_address
                mosi_arr[10:8]inside {WR_ADDR, WR_DATA};         // write_address
or write_data
            else if (old_mosi_arr[10:8]  == WR_DATA) //write_data
                mosi_arr[10:8]dist {RD_ADDR:= 60, WR_ADDR:=40}; // read_address =
60%, write_address = 40%
            else if  (old_mosi_arr[10:8]  == RD_ADDR) // Read_address
                mosi_arr[10:8]== RD_DATA;
           else if  (old_mosi_arr[10:8]  == RD_DATA)
                mosi_arr[10:8]dist {WR_ADDR:=60, RD_ADDR:= 40}; // Write Address
= 60%, Read Address = 40%
  }
         else {
             mosi_arr == old_mosi_arr;}

}
    constraint MOSI_c { if (!SS_n && rst_n) MOSI == mosi_arr[10 - i]; }

    constraint tx_valid_c {
      if (Count_seq > 13 && Count_seq <= 23) { tx_valid == 1; }
      else { tx_valid == 0; }
```

```
  }


  constraint tx_data_c {
    if (Count_seq > 12 && Count_seq <= 23) {tx_data==old_tx_data};
  }

  function void post_randomize();
    old_mosi_arr = mosi_arr;
    old_tx_data=tx_data;
    if (!rst_n) begin
      cs = IDLE;
      Count_seq = 0;
      i = 0;
      address_recieved = 0;
    end
    else begin
      cs = ns;
      if (cs == READ_ADD) address_recieved = 1;
      else if (cs == READ_DATA) address_recieved = 0;
      if (cs != READ_DATA) begin
        if (Count_seq >= 13) begin
          Count_seq = 0;
        end
        else begin
          Count_seq++;
        end
      end
      else begin
        if (Count_seq >= 23) begin
          Count_seq = 0;
        end
        else begin
          Count_seq++;
        end
      end

      i++;
      if (i == 11 || Count_seq == 1)
        i = 0;
    end
  endfunction

  function string convert2string();
```

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

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

SPI reset sequence:

```
package SPI_reset_sequence_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import SPI_shared_pkg::* ;
class SPI_reset_sequence extends uvm_sequence #(SPI_seq_item);
`uvm_object_utils(SPI_reset_sequence)
 SPI_seq_item seq_item;

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

task body;
 seq_item = SPI_seq_item :: type_id :: create("seq_item");
  start_item(seq_item);
  seq_item.rst_n = 0;
  seq_item.rx_data = 0;
  seq_item.rx_valid = 0;
  seq_item.MOSI = 0;
  seq_item.MISO =0;
  seq_item.SS_n = 0;
  finish_item(seq_item);
endtask

endclass
endpackage
```

SPI main sequence:

```
package SPI_main_sequence_pkg;
    import SPI_shared_pkg::* ;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import SPI_seq_item_pkg::* ;

class SPI_main_sequence extends uvm_sequence #(SPI_seq_item);
 `uvm_object_utils(SPI_main_sequence)
 SPI_seq_item seq_item;

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

task body ;
         seq_item=SPI_seq_item::type_id::create("seq_item");
            repeat(10000) begin
                start_item(seq_item);
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask
endclass
endpackage
```

SPI sequencer:

```
package SPI_sequencer_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import SPI_shared_pkg::* ;
class SPI_sequencer extends uvm_sequencer #(SPI_seq_item);
 `uvm_component_utils(SPI_sequencer)

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

endclass
endpackage
```

SPI driver:

```
package SPI_driver_pkg;
import uvm_pkg::*;
import SPI_seq_item_pkg::* ;
`include "uvm_macros.svh"
class SPI_driver extends uvm_driver #(SPI_seq_item);
 `uvm_component_utils(SPI_driver)
virtual SPI_if SPI_vif;
SPI_seq_item stim_seq_item;

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

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

SPI environment:

```
package SPI_env_pkg;
import uvm_pkg::*;
import SPI_scoreboard_pkg::* ;
import SPI_agent_pkg::* ;
import SPI_cvr_collector_pkg::* ;
`include "uvm_macros.svh"

class SPI_env extends uvm_env;
  `uvm_component_utils(SPI_env)
SPI_agent agt;
SPI_scoreboard sb;
```

```
SPI_cvr_collector cvr;
//SPI_sequencer sqr;

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

 function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  agt = SPI_agent::type_id::create("agt", this);
  cvr = SPI_cvr_collector::type_id::create("cvr", this);
  sb = SPI_scoreboard::type_id::create("sb", this);
  endfunction: build_phase

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

SPI monitor:

```
package SPI_monitor_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import SPI_shared_pkg::* ;
class SPI_monitor extends uvm_monitor;
`uvm_component_utils(SPI_monitor)
virtual SPI_if SPI_vif;
SPI_seq_item rsp_seq_item;
uvm_analysis_port #(SPI_seq_item) mon_ap;

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

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

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

```
 forever begin
 rsp_seq_item = SPI_seq_item::type_id::create("rsp_seq_item");
 @(negedge SPI_vif.clk);
 rsp_seq_item.rst_n=SPI_vif.rst_n;
 rsp_seq_item.MOSI=SPI_vif.MOSI;
 rsp_seq_item.SS_n=SPI_vif.SS_n;
 rsp_seq_item.MISO=SPI_vif.MISO;
 rsp_seq_item.tx_data=SPI_vif.tx_data;
 rsp_seq_item.tx_valid=SPI_vif.tx_valid;
 rsp_seq_item.rx_data=SPI_vif.rx_data;
 rsp_seq_item.rx_valid=SPI_vif.rx_valid;
  rsp_seq_item.MISO_ref=SPI_vif.MISO_ref;
 rsp_seq_item.rx_data_ref=SPI_vif.rx_data_ref;
 rsp_seq_item.rx_valid_ref=SPI_vif.rx_valid_ref;
 mon_ap.write(rsp_seq_item);
  `uvm_info("run_phase", rsp_seq_item.convert2string(), UVM_HIGH)
 end
endtask

endclass
endpackage
```

SPI agent:

```
package SPI_agent_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import SPI_shared_pkg::* ;
import SPI_monitor_pkg::* ;
import SPI_driver_pkg::* ;
import SPI_sequencer_pkg::* ;
import SPI_config_pkg::* ;
class SPI_agent extends uvm_agent;
 `uvm_component_utils(SPI_agent)
SPI_sequencer sqr;
SPI_driver drv;
SPI_monitor mon;
SPI_config conf;
uvm_analysis_port #(SPI_seq_item) agt_ap;

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

function void build_phase(uvm_phase phase);
```

```
 super.build_phase(phase);
if(!uvm_config_db #(SPI_config)::get(this, "", "CFG", conf))
 `uvm_fatal("build_phase", "unable to get configuration object")

sqr = SPI_sequencer::type_id::create("sqr",this);
drv = SPI_driver::type_id::create("drv", this);
mon = SPI_monitor::type_id::create("mon", this);
agt_ap = new("agt_ap", this);

endfunction

function void connect_phase(uvm_phase phase);
 super.connect_phase(phase);
drv.SPI_vif = conf.SPI_config_vif;
drv.seq_item_port.connect(sqr.seq_item_export);
mon.SPI_vif = conf.SPI_config_vif;
mon.mon_ap.connect(agt_ap);

endfunction
endclass
endpackage
```

SPI scoreboard:

```
package SPI_scoreboard_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import SPI_shared_pkg::* ;
class SPI_scoreboard extends uvm_scoreboard;
 `uvm_component_utils(SPI_scoreboard)
 uvm_analysis_export #(SPI_seq_item) sb_export;
uvm_tlm_analysis_fifo #(SPI_seq_item) sb_fifo;
SPI_seq_item seq_item_sb;



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

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

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

task run_phase(uvm_phase phase);
 super.run_phase(phase);
forever begin
 sb_fifo.get(seq_item_sb);
  if (seq_item_sb.rx_valid ||seq_item_sb.rx_valid_ref ) begin
 if(seq_item_sb.rx_data != seq_item_sb.rx_data_ref) begin
  `uvm_error("run_phase", $sformatf ("Comparsion 1failed, transaction done by DUT
= %s , MISO_ref = 0b%0b , rx_data_ref = 0b%b, rx_valid_ref = 0b%b" ,
seq_item_sb.convert2string(), seq_item_sb.MISO_ref, seq_item_sb.rx_data_ref,
seq_item_sb.rx_valid_ref ));
  error_count++;
 end
 else begin
 `uvm_info("run_phase" , $sformatf("Correct_out1 : %s",
seq_item_sb.convert2string()), UVM_HIGH);
  correct_count++;
 end
 end

 if(seq_item_sb.MISO != seq_item_sb.MISO_ref  || seq_item_sb.rx_valid !=
seq_item_sb.rx_valid_ref ) begin
  `uvm_error("run_phase", $sformatf (" Comparsion 2failed, transaction done by
DUT = %s , MISO_ref = 0b%0b , rx_valid_ref = 0b%b" ,
seq_item_sb.convert2string(), seq_item_sb.MISO_ref,seq_item_sb.rx_valid_ref ));
  error_count_2++;
 end
 else begin
 `uvm_info("run_phase" , $sformatf("Correct_out2 : %s",
seq_item_sb.convert2string()), UVM_HIGH);
  correct_count_2++;
 end

end
endtask


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

```
`uvm_info("report_phase", $sformatf("Total successful transactions: %d",
correct_count), UVM_MEDIUM);
`uvm_info("report_phase", $sformatf("Total failed transactions: %d",
error_count), UVM_MEDIUM);
endfunction
endclass
endpackage
```

SPI coverage collector:

NOTE:Bins ignored as it is impossible changing 2 bits at one time as rx data take bit by bit from MOSI

```
package SPI_cvr_collector_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import SPI_shared_pkg::* ;
class SPI_cvr_collector extends uvm_component;
 `uvm_component_utils(SPI_cvr_collector)
uvm_analysis_export #(SPI_seq_item) cvr_export;
uvm_tlm_analysis_fifo #(SPI_seq_item) cvr_fifo;
SPI_seq_item seq_item_cvr;

covergroup SPIcg ;
cp1: coverpoint seq_item_cvr.rx_data[9:8] {
    bins all_possible_values [] = {2'b00, 2'b01, 2'b10, 2'b11};
    bins trans1 [] = (0=>1,2,3);
    ignore_bins ignored_trans1 = (0 => 3);
    bins trans2 [] = (1=>0,2,3);
        ignore_bins ignored_trans2 = (1 => 2);
    bins trans3 [] = (2=>0,1,3);
        ignore_bins ignored_trans3 = (2 => 1);
    bins trans4 [] = (3=>0,1,2);
}


cp2: coverpoint seq_item_cvr.SS_n {
    option.auto_bin_max = 0;
    bins trans_full = (1=>0[*13]=>1);
    bins trans_extended = (1=>0[*23]=>1);
}

cp3: coverpoint seq_item_cvr.MOSI {
    option.auto_bin_max = 0;
    bins trans_write_address = (0=>0=>0);
```

```
    bins trans_write_data = (0=>0=>1);
    bins trans_read_address = (1=>1=>0);
    bins trans_read_data = (1=>1=>1);
}

cross_MOSI_SS_n: cross cp2, cp3 {
    option.cross_auto_bin_max = 0;
   bins write_address_SS_n = binsof(cp3.trans_write_address) &&
binsof(cp2.trans_full);
   bins write_data_SS_n = binsof(cp3.trans_write_data) && binsof(cp2.trans_full);
   bins read_address_SS_n = binsof(cp3.trans_read_address) &&
binsof(cp2.trans_full);
   bins read_data_SS_n = binsof(cp3.trans_read_data) &&
binsof(cp2.trans_extended);
}

endgroup

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

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

function void connect_phase(uvm_phase phase);
 super.connect_phase(phase);
 cvr_export.connect(cvr_fifo.analysis_export);
endfunction

task run_phase(uvm_phase phase);
 super.run_phase(phase);
forever begin
 cvr_fifo.get(seq_item_cvr);
 SPIcg.sample();
end
endtask
endclass
endpackage
```

SPI test:

```
package SPI_test_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import SPI_env_pkg::* ;
import SPI_config_pkg::* ;
import SPI_reset_sequence_pkg::* ;
import SPI_main_sequence_pkg::* ;
import SPI_sequencer_pkg::* ;
class SPI_test extends uvm_test;
  `uvm_component_utils(SPI_test)

  // Build the enviornment in the build phase
  virtual SPI_if SPI_config_vif;
  SPI_env env;
  SPI_config SPI_cfg;
  SPI_reset_sequence reset_seq;
  SPI_main_sequence main_seq;

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

  function void build_phase(uvm_phase phase);
   super.build_phase(phase);
   env = SPI_env::type_id::create("env", this);
   SPI_cfg = SPI_config::type_id::create("SPI_cgf");
   main_seq = SPI_main_sequence::type_id::create("main_seq", this);
   reset_seq = SPI_reset_sequence::type_id::create("reset_seq", this);

    if(!uvm_config_db#(virtual SPI_if)::get(this, "","SPI_if",
SPI_cfg.SPI_config_vif))
     `uvm_fatal("build_phase", "Test - unable to get virsual interface");
   uvm_config_db #(SPI_config)::set(this, "*", "CFG", SPI_cfg);


  endfunction

  task run_phase(uvm_phase phase);
   super.run_phase(phase);
   phase.raise_objection(this);
   `uvm_info("run_phase","reset deasserted",UVM_LOW)
   reset_seq.start(env.agt.sqr);
   `uvm_info("run_phase","reset asserted",UVM_LOW)
   //main_seq
   `uvm_info("run_phase","main asserted",UVM_LOW)
```

```
   main_seq.start(env.agt.sqr);
      `uvm_info("run_phase","main asserted",UVM_LOW)
   phase.drop_objection(this);
  endtask: run_phase
endclass: SPI_test
endpackage
```

SPI top:

```
import uvm_pkg::*;
`include "uvm_macros.svh"
import SPI_test_pkg::*;
module SPI_top();
bit clk;
  initial begin
    forever #1 clk=~clk;
  end

SPI_if SPIif (clk);
SLAVE DUT (SPIif.MOSI, SPIif.MISO, SPIif.SS_n, SPIif.clk, SPIif.rst_n,
SPIif.rx_data, SPIif.rx_valid,  SPIif.tx_data, SPIif.tx_valid);
spi_slave golden(SPIif.clk,  SPIif.rst_n, SPIif.MOSI,
SPIif.MISO_ref,  SPIif.SS_n, SPIif.rx_data_ref, SPIif.rx_valid_ref,
SPIif.tx_valid, SPIif.tx_data);

  bind SLAVE SPI_sva ass_spi(SPIif.svaa);
initial begin
   uvm_config_db#(virtual SPI_if)::set(null, "uvm_test_top", "SPI_if", SPIif);
   run_test("SPI_test");
  end
endmodule
```

Assertions table:

Feature	Assertions
whenever reset is asserted, the outputs (MISO, rx_valid, and rx_data) are all low.	@(posedge SPIif.clk) (!SPIif.rst_n) |=> (SPIif.MISO==0 && SPIif.rx_valid==0 && (SPIif.rx_data ==0))

check that after any valid command sequence (write_add_seq (000), write_data_seq (001), read_add_seq (110), or read_data_seq (111)), the rx_valid signal must assert exactly after 10 cycles and the SS_n should eventually after the 10 cycles to close communication.	sequence write_add;
	  SPIif.SS_n == 1 ##2 SPIif.MOSI == 0 ##1 SPIif.MOSI == 0 ##1 SPIif.MOSI == 0;
	endsequence
	
	sequence read_add;
	  SPIif.SS_n == 1 ##2 SPIif.MOSI == 1 ##1 SPIif.MOSI == 1 ##1 SPIif.MOSI == 0;
	endsequence
	
	sequence write_data;

	SPIif.SS_n == 1 ##2 SPIif.MOSI == 0 ##1 SPIif.MOSI == 0 ##1 SPIif.MOSI == 1; Endsequence sequence read_data; SPIif.SS_n == 1 ##2 SPIif.MOSI == 1 ##1 SPIif.MOSI == 1 ##1 SPIif.MOSI == 1; endsequence sequence ss_n_seq; ##10 SPIif.rx_valid ##1 SPIif.SS_n[->1]; Endsequence @(posedge SPIif.clk) disable iff (!SPIif.rst_n) (write_add or read_add or write_data or read_data) |-> ss_n_seq
• IDLE → CHK_CMD • CHK_CMD → WRITE or READ_ADD or READ_DATA • WRITE → IDLE • READ_ADD → IDLE • READ_DATA → IDLE	@(posedge clk) (cs == IDLE&&rst_n&&!SS_n) |=> (cs == CHK_CMD) @(posedge clk) (cs == CHK_CMD &&rst_n && !SS_n && !MOSI) |=> (cs == WRITE) @(posedge clk) (cs == CHK_CMD &&rst_n && !SS_n && MOSI && !received_address) |=> (cs == READ_ADD) @(posedge clk) (cs == CHK_CMD &&rst_n && !SS_n && MOSI && received_address) |=> (cs == READ_DATA) @(posedge clk) (cs == WRITE&&rst_n&&SS_n) |=> (cs == IDLE) @(posedge clk) (cs == READ_ADD&&rst_n&&SS_n) |=> (cs == IDLE) @(posedge clk) (cs == READ_DATA&&rst_n&&SS_n) |=> (cs == IDLE)

Do file: NOTE: as discussed i made seed random with sv_seed value "1336091805" to make the waveform as in word photos

```
vlib work
vlog +define+SIM -f src_files.list +cover -covercells
vsim -voptargs=+acc work.SPI_top -classdebug -uvmcontrol=all -cover -sv_seed 1336091805 -l sim.log
add wave -position insertpoint sim:/SPI_top/SPIif/*
add wave -position insertpoint \
sim:/SPI_top/DUT/counter \
sim:/SPI_top/DUT/cs \
sim:/SPI_top/DUT/ns
add wave -position insertpoint \
sim:/SPI_shared_pkg::error_count \
sim:/SPI_shared_pkg::error_count_2
add wave -position insertpoint \
sim:/SPI_top/golden/bit_cnt_rx
add wave -position insertpoint \
sim:/SPI_shared_pkg::Count_seq
add wave -position insertpoint \
sim:/SPI_top/DUT/received_address
coverage save SPI_top.ucdb -onexit
```

run -all
#quit -sim

Src_files.list:

SPI_slave.sv
golden_SPI_slave.v
SPI_shared_pkg.sv
SPI_if.sv
spi_sva.sv
SPI_config.sv
SPI_seq_item.sv
SPI_reset_sequence.sv
SPI_main_sequence.sv
SPI_sequencer.sv
SPI_driver.sv
SPI_monitor.sv
SPI_agent.sv
SPI_scoreboard.sv
SPI_cvr_collector.sv
SPI_env.sv
SPI_test.sv
SPI_top.sv

Questasim:

Wave form:

At 1ns checking reset is ok

At 19ns Mosi = 0 so transition occur from chk cmd to write

At 21ns and 23ns we get Mosi 0, 0 the first 2 bits that rx data will take

At 41ns now rx data take Mosi 10 bits which is 06a in hexadecimal and rx valid become =1

At 173 ns Mosi = 1 and address received =0 so transition will occur from chk cmd to read address

At 175ns and 177ns we get Mosi 1 , 0 this the first 2 bits in rx data received from Mosi

At 195ns rx valid =1 , rx data =25b 10 bits which has been taken from Mosi

At 201ns Mosi =1 and received address =1 so transaction will occur from chk cmd to read data

At 203 ns and 205 ns we get Mosi 1 ,1

At 227ns tx valid now =1 ,expecting that Miso start to take msb of tx data "msb =1"

Miso takes tx data which is 11010000

Assertions:

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Me
/uvm_pkg::uvm_re...	Immediate	SVA	on	0	0	-	-	-	
/uvm_pkg::uvm_re...	Immediate	SVA	on	0	0	-	-	-	
/SPI_main_sequen...	Immediate	SVA	on	0	1	-	-	-	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/asse...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/ass_...	Concurrent	SVA	on	0	1	-	0B	0B	
/SPI_top/DUT/ass_...	Concurrent	SVA	on	0	1	-	0B	0B	

Cover directives:

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	F
/SPI_top/DUT/cove...	SVA	✓	Off	95	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/cove...	SVA	✓	Off	129	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/cove...	SVA	✓	Off	367	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/cove...	SVA	✓	Off	119	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/cove...	SVA	✓	Off	143	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/cove...	SVA	✓	Off	409	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/cove...	SVA	✓	Off	676	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/ass_...	SVA	✓	Off	460	1	Unli...	1	100%		✓	0	
/SPI_top/DUT/ass_...	SVA	✓	Off	90	1	Unli...	1	100%		✓	0	

Cover groups:

Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/SPI_cvr_collector...		100.00%							
TYPE SPIcg		100.00%	100	100.00...			auto(0)		
CVP SPIcg:...		100.00%	100	100.00...					
CVP SPIcg:...		100.00%	100	100.00...					
CVP SPIcg:...		100.00%	100	100.00...					
CROSS SPI...		100.00%	100	100.00...					
INST \/SPI...		100.00%	100	100.00...			0		
CVP cp...		100.00%	100	100.00...					
(1) igno...		0	-	-					
(1) igno...		0	-	-					
(1) igno...		0	-	-					
(1) bin ...		3851	1	100.00...					
(1) bin ...		1764	1	100.00...					
(1) bin ...		1610	1	100.00...					
(1) bin ...		2776	1	100.00...					
(1) bin t...		32	1	100.00...					
(1) bin t...		134	1	100.00...					
(1) bin t...		66	1	100.00...					
(1) bin t...		125	1	100.00...					
(1) bin t...		133	1	100.00...					
(1) bin t...		14	1	100.00...					
(1) bin t...		115	1	100.00...					
(1) bin t...		57	1	100.00...					
(1) bin t...		26	1	100.00...					
CVP cp...		100.00%	100	100.00...					
(1) bin t...		429	1	100.00...					
(1) bin t...		100	1	100.00...					
CVP cp...		100.00%	100	100.00...					
(1) bin t...		1907	1	100.00...					
(1) bin t...		1180	1	100.00...					
(1) bin t...		1114	1	100.00...					
(1) bin t...		1467	1	100.00...					
CROSS...		100.00%	100	100.00...					
(1) bin ...		89	1	100.00...					

Cover report:

*code_cvrrrrrrr.txt - Notepad

File Edit Format View Help

```
================================Branch Details================================

Branch Coverage for instance /SPI_top/golden

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File golden_SPI_slave.v
------------------------------------IF Branch------------------------------------
    26                                      2692     Count coming in to IF
    26             1                          90             if (!rst_n)

    28             1                        2602             else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    33                                      7177     Count coming in to CASE
    34             1                        1318                 IDLE:

    40             1                         871                 CHK_CMD:

    46             1                        2642                 WRITE:

    52             1                        1127                 READ_ADD:

    58             1                        1218                 READ_DATA:

    64             1                           1                 default:

Branch totals: 6 hits of 6 branches = 100.00%

------------------------------------IF Branch------------------------------------
    35                                      1318     Count coming in to IF
    35             1                         723                     if (!SS_n)

    37             1                         595                     else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    41                                       871     Count coming in to IF
    41             1                         871                     if (!SS_n)
```

*code_cvrrrrrrr.txt - Notepad

File Edit Format View Help

```
 Coverage Report by instance with details

================================================================================
=== Instance: /SPI_top/SPIif
=== Design Unit: work.SPI_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         74        74         0   100.00%

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

Toggle Coverage for instance /SPI_top/SPIif --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                          MISO           1           1      100.00
                                      MISO_ref           1           1      100.00
                                          MOSI           1           1      100.00
                                          SS_n           1           1      100.00
                                           clk           1           1      100.00
                                         rst_n           1           1      100.00
                                   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     =        37
Toggled Node Count   =        37
Untoggled Node Count =         0

Toggle Coverage      =     100.00% (74 of 74 bins)

================================================================================
=== Instance: /SPI_top/DUT/ass_spi
=== Design Unit: work.SPI_sva
================================================================================

Assertion Coverage:
    Assertions                       2         2         0   100.00%
--------------------------------------------------------------------------------
Name                File(Line)                    Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------------------
/SPI_top/DUT/ass_spi/assert__rx_valid_p
                    spi_sva.sv(33)                     0            1
```

*code_cvrrrrrrr.txt - Notepad

File Edit Format View Help

```
 46          1          119        CHK_CMD -> READ_DATA
 44          2          143        CHK_CMD -> READ_ADD
 41          3          409        CHK_CMD -> WRITE
 38          4            5        CHK_CMD -> IDLE
 64          5          118        READ_DATA -> IDLE
 58          6          143        READ_ADD -> IDLE
 52          7          409        WRITE -> IDLE


   Summary                  Bins    Hits   Misses  Coverage
   -------                  ----    ----   ------  --------
      FSM States               5       5        0   100.00%
      FSM Transitions          8       8        0   100.00%
Statement Coverage:
   Enabled Coverage         Bins    Hits   Misses  Coverage
   ----------------         ----    ----   ------  --------
   Statements                 39      39        0   100.00%

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

Statement Coverage for instance /SPI_top/DUT --

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

   2

   3                                           localparam IDLE     = 3'b000;

   4                                           localparam WRITE    = 3'b001;

   5                                           localparam CHK_CMD  = 3'b010;

   6                                           localparam READ_ADD = 3'b011;

   7                                           localparam READ_DATA = 3'b100;

   8

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

*code_cvrrrrrrr.txt - Notepad

File Edit Format View Help

```
            Covergroup Bins         23        23         0   100.00%
-------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status
-------------------------------------------------------------------------------------------------
 TYPE /SPI_cvr_collector_pkg/SPI_cvr_collector/SPIcg
                                                    100.00%        100          -    Covered
    covered/total bins:                                  23         23          -
    missing/total bins:                                   0         23          -
    % Hit:                                          100.00%        100          -
    Coverpoint cp1                                  100.00%        100          -    Covered
        covered/total bins:                              13         13          -
        missing/total bins:                               0         13          -
        % Hit:                                      100.00%        100          -
    Coverpoint cp2                                  100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
    Coverpoint cp3                                  100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                      100.00%        100          -
    Cross cross_MOSI_SS_n                           100.00%        100          -    Covered
        covered/total bins:                               4          4          -
        missing/total bins:                               0          4          -
        % Hit:                                      100.00%        100          -
 Covergroup instance \/SPI_cvr_collector_pkg::SPI_cvr_collector::SPIcg
                                                    100.00%        100          -    Covered
    covered/total bins:                                  23         23          -
    missing/total bins:                                   0         23          -
    % Hit:                                          100.00%        100          -
    Coverpoint cp1                                  100.00%        100          -    Covered
        covered/total bins:                              13         13          -
        missing/total bins:                               0         13          -
        % Hit:                                      100.00%        100          -
        ignore_bin ignored_trans1                         0                     -    ZERO
        ignore_bin ignored_trans2                         0                     -    ZERO
        ignore_bin ignored_trans3                         0                     -    ZERO
        bin all_possible_values[0]                     3851          1          -    Covered
        bin all_possible_values[1]                     1764          1          -    Covered
        bin all_possible_values[2]                     1610          1          -    Covered
        bin all_possible_values[3]                     2776          1          -    Covered
        bin trans1[0->2]                                 32          1          -    Covered
        bin trans1[0->1]                                134          1          -    Covered
        bin trans2[1->3]                                 66          1          -    Covered
        bin trans2[1->0]                                125          1          -    Covered
        bin trans3[2->3]                                133          1          -    Covered
        bin trans3[2->0]                                 14          1          -    Covered
        bin trans4[3->2]                                115          1          -    Covered
        bin trans4[3->1]                                 57          1          -    Covered
        bin trans4[3->0]                                 26          1          -    Covered
    Coverpoint cp2                                  100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100
```

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

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

Toggle Coverage for instance /SPI_top/DUT --

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

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

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

```
Directive Coverage:
    Directives                    7         7         0   100.00%

DIRECTIVE COVERAGE:
-------------------------------------------------------------------------------------------
Name                                Design Design   Lang File(line)      Hits Status
                                    Unit   UnitType
-------------------------------------------------------------------------------------------
/SPI_top/DUT/cover__rd_idle         SLAVE  Verilog  SVA  SPI_slave.sv(195)
                                                                          95 Covered
/SPI_top/DUT/cover__ra_idle         SLAVE  Verilog  SVA  SPI_slave.sv(192)
                                                                         129 Covered
/SPI_top/DUT/cover__write_idle      SLAVE  Verilog  SVA  SPI_slave.sv(189)
                                                                         367 Covered
/SPI_top/DUT/cover__chkcmd_rd       SLAVE  Verilog  SVA  SPI_slave.sv(186)
                                                                         119 Covered
/SPI_top/DUT/cover__chkcmd_ra       SLAVE  Verilog  SVA  SPI_slave.sv(183)
                                                                         143 Covered
/SPI_top/DUT/cover__chkcmd_write    SLAVE  Verilog  SVA  SPI_slave.sv(180)
                                                                         409 Covered
/SPI_top/DUT/cover__chkcmd_idle     SLAVE  Verilog  SVA  SPI_slave.sv(177)
                                                                         676 Covered
FSM Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    FSM States                       5         5         0   100.00%
    FSM Transitions                  8         8         0   100.00%

=============================FSM Details=============================

FSM Coverage for instance /SPI_top/DUT --

FSM_ID: cs
    Current State Object : cs
```

```
*code_cvmmm.txt - Notepad
File  Edit  Format  View  Help
--------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
                                      Unit   UnitType
--------------------------------------------------------------------------------
/SPI_top/DUT/cover__rd_idle           SLAVE  Verilog  SVA  SPI_slave.sv(195)
                                                                          95 Covered
/SPI_top/DUT/cover__rx_idle           SLAVE  Verilog  SVA  SPI_slave.sv(192)
                                                                         129 Covered
/SPI_top/DUT/cover__write_idle        SLAVE  Verilog  SVA  SPI_slave.sv(189)
                                                                         367 Covered
/SPI_top/DUT/cover__chkcmd_rd         SLAVE  Verilog  SVA  SPI_slave.sv(186)
                                                                         119 Covered
/SPI_top/DUT/cover__chkcmd_rx         SLAVE  Verilog  SVA  SPI_slave.sv(183)
                                                                         143 Covered
/SPI_top/DUT/cover__chkcmd_write      SLAVE  Verilog  SVA  SPI_slave.sv(180)
                                                                         409 Covered
/SPI_top/DUT/cover__chkcmd_idle       SLAVE  Verilog  SVA  SPI_slave.sv(177)
                                                                         676 Covered
/SPI_top/DUT/ass_spi/cover__rx_valid_p  SPI_sva Verilog  SVA  spi_sva.sv(34)   460 Covered
/SPI_top/DUT/ass_spi/cover__rst_is_ok   SPI_sva Verilog  SVA  spi_sva.sv(31)    90 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 9

ASSERTION RESULTS:
--------------------------------------------------------------------------------
Name                  File(Line)                    Failure      Pass
                                                    Count        Count
--------------------------------------------------------------------------------
/SPI_top/DUT/assert__rd_idle
                      SPI_slave.sv(194)                  0          1
/SPI_top/DUT/assert__rx_idle
                      SPI_slave.sv(191)                  0          1
/SPI_top/DUT/assert__write_idle
                      SPI_slave.sv(188)                  0          1
/SPI_top/DUT/assert__chkcmd_rd
                      SPI_slave.sv(185)                  0          1
/SPI_top/DUT/assert__chkcmd_rx
                      SPI_slave.sv(182)                  0          1
/SPI_top/DUT/assert__chkcmd_write
                      SPI_slave.sv(179)                  0          1
/SPI_top/DUT/assert__chkcmd_idle
                      SPI_slave.sv(176)                  0          1
/SPI_top/DUT/ass_spi/assert__rx_valid_p
                      spi_sva.sv(33)                     0          1
/SPI_top/DUT/ass_spi/assert__rst_is_ok
                      spi_sva.sv(30)                     0          1
/SPI_main_sequence_pkg/SPI_main_sequence/body/#ublk#224836255#17/immed__19
                      SPI_main_sequence.sv(19)           0          1
```

```
================================================================================
=== Instance: /SPI_top/DUT/ass_spi
=== Design Unit: work.SPI_sva
================================================================================

Assertion Coverage:
    Assertions                    2       2       0  100.00%
--------------------------------------------------------------------------------
Name                File(Line)                Failure      Pass
                                              Count        Count
--------------------------------------------------------------------------------
/SPI_top/DUT/ass_spi/assert__rx_valid_p
                    spi_sva.sv(33)                  0          1
/SPI_top/DUT/ass_spi/assert__rst_is_ok
                    spi_sva.sv(30)                  0          1

Directive Coverage:
    Directives                    2       2       0  100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
                                      Unit   UnitType
--------------------------------------------------------------------------------
/SPI_top/DUT/ass_spi/cover__rx_valid_p  SPI_sva Verilog  SVA  spi_sva.sv(34)   460 Covered
/SPI_top/DUT/ass_spi/cover__rst_is_ok   SPI_sva Verilog  SVA  spi_sva.sv(31)    90 Covered

================================================================================
=== Instance: /SPI_top/DUT
```

RAM verification:

In this part, create a full UVM environment to verify the RAM design

Verification Plan:

Label	Design requirement description	Stimulus generation	Functional coverage	Functionality check
Reset sequence	Check reset	Directed	-	A checker in testbench to make sure the output is correct
-Write only sequence -Read only sequence -Write Read sequence	Check other randomize cases	Randomize Under constraints: 1- The reset signal (rst_n) shall be deasserted most of the time. 2- The rx_valid signal shall be asserted most of time. 3- For a write-only sequence, every Write Address operation shall always be followed by either Write Address or Write Data operation. 4- For a read-only sequence, every Read Address operation shall always be followed by Read Data. After a Read Data operation shall always be followed by Read Address. 5- For a randomized read/write sequence, the following ordering rules shall be enforced: • Every Write Address operation shall always be followed by either Write Address or Write Data operation. • After a Write Data, the next operation shall be chosen with the following probability distribution: 60% → Read Address & 40% → Write Address. • Every Read Address operation shall always be followed by Read Data operation.	1- Write Coverpoint to check transaction ordering for din[9:8]: • Check din[9:8] takes 4 possible values • Check write data after write address • Check read data after read address • Check write address => write data => read address => read data 2- Cross coverage: • Between all bins of din[9:8] and rx_valid signal when it is high • Between din[9:8] when it equals read data and tx_valid when it is high	A checker in testbench to make sure the output is correct

		• After a Read Data, the next operation shall be chosen with the following probability distribution: 60% → Write Address & 40% → Read Address.		

Design code with corrected bugs:

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

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

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

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

reg [7:0] Rd_Addr, Wr_Addr;

always @(posedge clk) begin
    if (~rst_n) begin
        dout <= 0;
        tx_valid <= 0;
        Rd_Addr <= 0;
        Wr_Addr <= 0;
    end
    else begin
        if (rx_valid) begin
            case (din[9:8])
                2'b00 : Wr_Addr <= din[7:0];
                2'b01 : MEM[Wr_Addr] <= din[7:0];
                2'b10 : Rd_Addr <= din[7:0];
                2'b11 : dout <= MEM[Rd_Addr]; //bug Rd_addr not Wr_Addr
                default : dout <= 0;
            endcase
                tx_valid <= (din[9] && din[8])? 1'b1 : 1'b0; //tx valid should be
inside if (rx_valid) to be high until next state comes
        end
    end
end
endmodule
```

There, s two bugs:

Read data should take data out from read address

Tx valid should be inside `if (rx_valid) to be high until next state comes`

Golden design:

```
module RAM_golden(din,clk,rst_n,rx_valid,dout,tx_valid);
parameter MEM_DEPTH = 256;
parameter ADDR_SIZE  = 8;
input clk,rst_n,rx_valid;
input [9:0] din;
output reg [7:0]dout;
output reg tx_valid;
reg [ADDR_SIZE-1:0] mem[MEM_DEPTH-1:0];
reg [ADDR_SIZE-1:0]addr_wr, addr_rd;
always @(posedge clk)begin
if(!rst_n)begin
    dout<=0;
    tx_valid<=0;
    addr_wr<=0;
    addr_rd<=0;
end
else if(rx_valid)begin
if(din[9:8]==0)begin
    addr_wr<=din[7:0];
    tx_valid<=0;
end
else if(din[9:8]==1)begin
    mem[addr_wr]<=din[7:0];
    tx_valid<=0;
end
else if(din[9:8]==2)begin
    addr_rd<=din[7:0];
    tx_valid<=0;
end

 else if(din[9:8]==3)begin
    dout<=mem[addr_rd];
     tx_valid<=1;
 end
end
end
endmodule
```

RAM interface:

```
interface RAM_interface(clk);
input clk;
```

```
logic    [9:0] din;
logic          clk, rst_n, rx_valid;

logic[7:0] dout; //out
logic  tx_valid; //out

logic[7:0] dout_gn; //out
logic  tx_valid_gn; //out
   modport sva (input din,clk,rst_n,rx_valid, output dout,tx_valid);
endinterface
```

RAM SVA:

```
module ram_sva (RAM_interface.sva ram_if);
 property reset_is_Ok;

        @(posedge ram_if.clk)  (!ram_if.rst_n) |=> (ram_if.dout == 0&&
ram_if.tx_valid == 0);
    endproperty


    property tx_deassert;
            @(posedge
ram_if.clk)  ((ram_if.din[9:8]==2'b00||ram_if.din[9:8]==2'b01||ram_if.din[9:8]==2
'b10)&&ram_if.rst_n&&ram_if.rx_valid) |=>  (ram_if.tx_valid==0);
    endproperty
//3- An assertion checks that after a read_data_seq occurs, the tx_valid signal
must rise to indicate
//valid output and after it rises by one clock cycle, it should eventually fall.
   property tx_assert_deassert;
            @(posedge ram_if.clk)  (ram_if.din[9:8]==2'b11&&ram_if.rst_n) |=>
($rose(ram_if.tx_valid) |=> $fell(ram_if.tx_valid)[->1]);
    endproperty


   property write_eventually;
        @(posedge ram_if.clk)  (ram_if.din[9:8]==2'b00&&ram_if.rst_n)
|=>  (ram_if.din[9:8]==2'b01 [->1]);
    endproperty

    property read_eventually;
            @(posedge ram_if.clk)  (ram_if.din[9:8]==2'b10&&ram_if.rst_n)
|=>  (ram_if.din[9:8]==2'b11 [->1]);
    endproperty
////////////
```

```
assert property (reset_is_Ok);
cover  property (reset_is_Ok);


assert property (tx_deassert);
cover  property (tx_deassert);

////////////
assert property (tx_assert_deassert);
cover  property (tx_assert_deassert);

assert property (write_eventually);
cover  property (write_eventually);

assert property (read_eventually);
cover  property (read_eventually);

endmodule
```

RAM config:

```
package ram_config_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ram_config extends uvm_object;
`uvm_object_utils(ram_config )
virtual RAM_interface ram_vif;
//uvm_active_passive_enum  is_active;
function new (string name ="ram_config");
super.new(name);
endfunction
endclass
endpackage
```

RAM sequence item:

```
package ram_sequence_item_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class  ram_sequence_item extends uvm_sequence_item;
`uvm_object_utils(ram_sequence_item)


 rand bit  [9:0] din;
 rand bit rst_n, rx_valid;
/////
```

```
logic[1:0] old_din; //out
logic[7:0] dout; //out
logic  tx_valid; //out

logic[7:0] dout_gn; //out
logic  tx_valid_gn; //out
function new (string name ="ram_sequence_item");
super.new(name);
 old_din = 2'b00;
endfunction

constraint RESET{rst_n dist{0:=10 ,1:=90};} //low 90/100
constraint RX{rx_valid dist{1:=90 ,0:=10};} //high 90/100
//3- For a write-only sequence, every Write Address operation shall always be
followed by eitherWrite Address or Write Data operation.
 constraint write_only { if (old_din == 2'b00) din[9:8]==2'b01;
 else if (old_din == 2'b01) din[9:8]==2'b00;
 else din[9:8]==2'b00; }


//4- For a read-only sequence, every Read Address operation shall always be
followed by Read
//Data. After a Read Data operation shall always be followed by Read Address.
 constraint read_only { if (old_din== 2'b10) din[9:8] ==2'b11 ;
  else if (old_din == 2'b11) din[9:8] ==2'b10 ;
  else din[9:8] ==2'b10;}

constraint write_read {
  if (old_din == 2'b00) din[9:8] dist {2'b00 := 50, 2'b01 := 50};
  else if (old_din == 2'b01) din[9:8] dist {2'b00 := 40, 2'b10 := 60};
  else if (old_din == 2'b10) din[9:8] dist {2'b11 := 100}; //• Every Read Address
operation shall always be followed by Read Data operation.
  else if (old_din == 2'b11) din[9:8] dist {2'b10 := 40, 2'b00 := 60};
}
function void post_randomize;
old_din=din[9:8];
endfunction

function string convert2string();
return $sformatf("%s
din=0b%0b,rst_n=0b%0b,rx_valid=0b%0b,dout=0b%0b,tx_valid=0b%0b",super.convert2str
ing(),din,rst_n,rx_valid,dout,tx_valid);
endfunction

function string convert2string_stimulus();
```

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

RAM reset sequence:

```
package ram_sequence_rst_pkg;
import ram_sequence_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class  ram_sequence_reset extends uvm_sequence#(ram_sequence_item);
`uvm_object_utils(ram_sequence_reset)
ram_sequence_item seq_item;
function new (string name ="ram_sequence_reset");
super.new(name);
endfunction
task body;
seq_item=ram_sequence_item::type_id::create("seq_item");
    start_item(seq_item);
seq_item.rst_n =0;
seq_item.din     =0;
seq_item.rx_valid=0;
    finish_item(seq_item);
endtask
endclass
endpackage
```

RAM read only sequence:

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

class  ram_sequence_read extends  uvm_sequence#(ram_sequence_item);
`uvm_object_utils(ram_sequence_read)
ram_sequence_item seq_item;
function new (string name ="ram_sequence_read");
super.new(name);
endfunction
task body;
   seq_item=ram_sequence_item::type_id::create("seq_item");
 seq_item.constraint_mode(0);  //turning all constrain off
```

```
  seq_item.RESET.constraint_mode(1);
  seq_item.RX.constraint_mode(1);
  seq_item.read_only.constraint_mode(1);
repeat(10000) begin
    start_item(seq_item);
    assert(seq_item.randomize());
    finish_item(seq_item);
end
endtask
endclass
endpackage
```

RAM write only sequence:

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

class  ram_sequence_write extends  uvm_sequence#(ram_sequence_item);
`uvm_object_utils(ram_sequence_write)
ram_sequence_item seq_item;
function new (string name ="ram_sequence_write");
super.new(name);
endfunction
task body;
   seq_item=ram_sequence_item::type_id::create("seq_item");
 seq_item.constraint_mode(0);  //turning all constrain off
  seq_item.RESET.constraint_mode(1);
  seq_item.RX.constraint_mode(1);
  seq_item.write_only.constraint_mode(1);
repeat(10000) begin
    start_item(seq_item);
    assert(seq_item.randomize());
    finish_item(seq_item);
end
endtask
endclass
endpackage
```

RAM read write sequence:

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

```
class  ram_sequence_write_read extends  uvm_sequence#(ram_sequence_item);
`uvm_object_utils(ram_sequence_write_read)
ram_sequence_item seq_item;
function new (string name ="ram_sequence_write_read");
super.new(name);
endfunction
task body;
   seq_item=ram_sequence_item::type_id::create("seq_item");
 seq_item.constraint_mode(0);  //turning all constrain off
  seq_item.RESET.constraint_mode(1);
  seq_item.RX.constraint_mode(1);
  seq_item.write_read.constraint_mode(1);
repeat(10000) begin
    start_item(seq_item);
    assert(seq_item.randomize());
    finish_item(seq_item);
end
endtask
endclass
endpackage
```

RAM sequencer:

```
package ram_sequencer_pkg;
import uvm_pkg::*;
import ram_sequence_item_pkg::*;
`include "uvm_macros.svh"
class  ram_sequencer extends  uvm_sequencer#(ram_sequence_item);
`uvm_component_utils(ram_sequencer)
function new (string name ="ram_sequencer", uvm_component parent =null);
super.new(name,parent);
endfunction
endclass
endpackage
```

RAM driver:

```
package ram_driver_pkg;
//import alsu_reg_config::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
import ram_sequence_item_pkg::*;

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

```
virtual RAM_interface ram_vif;
ram_sequence_item seq_item;
function new (string name ="ram_driver", uvm_component parent =null);
super.new(name,parent);
endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
  seq_item=ram_sequence_item::type_id::create("seq_item");
  seq_item_port.get_next_item(seq_item);
ram_vif.rst_n    = seq_item.rst_n      ;
ram_vif.din= seq_item.din;
ram_vif.rx_valid= seq_item.rx_valid;
@( negedge ram_vif.clk );
seq_item_port.item_done();
`uvm_info("runphase",seq_item.convert2string_stimulus(),UVM_HIGH)
end

endtask

endclass
endpackage
```

RAM monitor:

```
package ram_monitor_pkg;
import ram_sequence_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ram_monitor extends uvm_monitor;
`uvm_component_utils(ram_monitor )
virtual RAM_interface ram_vif;
ram_sequence_item seq_item;
uvm_analysis_port #(ram_sequence_item )mon_ap;

function new (string name ="ram_monitor", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
  mon_ap = new ("mon_ap",this);
  endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
      seq_item=ram_sequence_item::type_id::create("seq_item");
```

```
@(negedge ram_vif.clk );

seq_item.rst_n    = ram_vif.rst_n      ;
seq_item.din= ram_vif.din;
seq_item.rx_valid= ram_vif.rx_valid;
seq_item.dout= ram_vif.dout;
seq_item.tx_valid= ram_vif.tx_valid;
seq_item.dout_gn= ram_vif.dout_gn;
seq_item.tx_valid_gn= ram_vif.tx_valid_gn;
  mon_ap.write(seq_item);
`uvm_info("runphase",seq_item.convert2string_stimulus(),UVM_HIGH)
end
endtask

endclass
endpackage
```

RAM agent:

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

class ram_agent extends uvm_agent;
`uvm_component_utils(ram_agent)
ram_sequencer sqr;
ram_monitor mon;
ram_driver drv;
ram_config cfg;
uvm_analysis_port #(ram_sequence_item)agt_ap;
function new (string name ="ram_agent", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
   if (!uvm_config_db #(ram_config)::get(this,"","CFG",cfg ))
  `uvm_fatal("build_phase","failed");
sqr=ram_sequencer::type_id::create("sqr",this);
drv=ram_driver::type_id::create("drv",this);
mon=ram_monitor::type_id::create("mon",this);
agt_ap=new("agt_ap",this);
```

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

endclass
endpackage
```

RAM scoreboard:

```
package ram_scoreboard_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import ram_sequence_item_pkg::*;
class ram_scoreboard extends uvm_scoreboard;
`uvm_component_utils(ram_scoreboard)
uvm_analysis_export #(ram_sequence_item) sb_export;
uvm_tlm_analysis_fifo  #(ram_sequence_item) sb_fifo;
ram_sequence_item seq_item;
int correct_count=0;
int error_count=0;
function new (string name ="ram_scoreboard", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
sb_export=new("sb_export",this);
sb_fifo=new("sb_fifo",this);
endfunction
//connectphase
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
sb_export.connect(sb_fifo.analysis_export);   //
endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
sb_fifo.get(seq_item);
if(seq_item.dout!=seq_item.dout_gn||seq_item.tx_valid!=seq_item.tx_valid_gn)begin
    `uvm_error("run_phase",$sformatf("failed,recieved by dut:%s while golden out
:0b%0b andtx_valid_gn
:0b%0b",seq_item.convert2string(),seq_item.dout_gn,seq_item.tx_valid_gn));
```

```
error_count++;
end
else begin
    `uvm_info("run_phase",$sformatf(" by dut:%s"
,seq_item.convert2string()),UVM_HIGH)
correct_count++;
end
end
endtask
endclass
endpackage
```

RAM cover collector:

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

class ram_coverage extends uvm_component;
`uvm_component_utils(ram_coverage)
uvm_analysis_export #(ram_sequence_item) cov_export;
uvm_tlm_analysis_fifo  #(ram_sequence_item) cov_fifo;
ram_sequence_item seq_item;

 covergroup cvr_group;
    Din:coverpoint seq_item.din[9:8]{
bins write_data={2'b01};
bins write_add={2'b00};
bins read_data={2'b11};
bins read_add={2'b10};
bins wa_wd=(2'b00=>2'b01);
bins ra_rd=(2'b10=>2'b11);
bins full_transition=(2'b00=>2'b01=>2'b10=>2'b11);
}

   Rx:coverpoint seq_item.rx_valid{
       bins high_rx={1'b1};
}

   Tx:coverpoint seq_item.tx_valid{
       bins high_tx={1'b1};
}
 cross1 : cross Din , Rx ;
cross2: cross Din,Tx {
    option.cross_auto_bin_max=0;
```

```
    bins rd_tx =binsof (Din.read_data) && binsof (Tx.high_tx) ;
}

endgroup

function new (string name ="ram_coverage", uvm_component parent =null);
super.new(name,parent);
cvr_group=new();
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
cov_export=new("cov_export",this);
cov_fifo=new("cov_fifo",this);
endfunction
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
cov_export.connect(cov_fifo.analysis_export);
endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
    cov_fifo.get(seq_item);
    cvr_group.sample();
end
endtask
endclass
endpackage
```

RAM environment:

```
package ram_env_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import ram_scoreboard_pkg::*;
import ram_agent_pkg::*;
import ram_coverage_pkg::*;
class ram_env extends uvm_env;
`uvm_component_utils(ram_env)
ram_agent agt;
ram_scoreboard sb;
ram_coverage cov;
function new (string name ="alsu_reg_env", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
```

```
agt=ram_agent :: type_id:: create ("agt",this);
sb=ram_scoreboard :: type_id:: create ("sb",this);
cov=ram_coverage :: type_id:: create ("cov",this);
  endfunction
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
agt.agt_ap.connect(sb.sb_export);
agt.agt_ap.connect(cov.cov_export);
endfunction
endclass
endpackage
```

RAM test:

```
package ram_test_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
import ram_sequence_rst_pkg::*;
import ram_sequence_write_pkg::*;
import ram_sequence_read_pkg::*;
import ram_sequence_write_read_pkg::*;
import ram_env_pkg::*;
import  ram_config_pkg ::*;

class ram_test extends uvm_test;
`uvm_component_utils(ram_test)

ram_env env;
ram_config ram_cfg;
 ram_sequence_reset seq1;
ram_sequence_write seq2;
 ram_sequence_read seq3;
ram_sequence_write_read seq4;
function new (string name ="ram_test", uvm_component parent =null);
super.new(name,parent);
endfunction
  function void build_phase (uvm_phase phase );
  super.build_phase (phase);
  env =ram_env :: type_id:: create ("env",this);
 ram_cfg = ram_config :: type_id:: create ("ram_config");
  seq1= ram_sequence_reset :: type_id:: create ("seq1");
  seq2= ram_sequence_write :: type_id:: create ("seq2");
  seq3= ram_sequence_read :: type_id:: create ("seq3");
  seq4= ram_sequence_write_read :: type_id:: create ("seq4");
```

```
  if (!uvm_config_db #(virtual
RAM_interface)::get(this,"","key_ram",ram_cfg.ram_vif ))
  `uvm_fatal("build_phase","failed");

  uvm_config_db#(ram_config)::set(this,"*","CFG",ram_cfg);
  endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
phase.raise_objection(this);
`uvm_info("run_phase","reset asserted",UVM_LOW)
seq1.start(env.agt.sqr); //reset sequence
`uvm_info("run_phase","reset deasserted",UVM_LOW)
//write
`uvm_info("run_phase","write asserted",UVM_LOW)
seq2.start(env.agt.sqr); //write sequence
`uvm_info("run_phase"," write deasserted",UVM_LOW)
//read
`uvm_info("run_phase","read asserted",UVM_LOW)
seq3.start(env.agt.sqr); //read sequence
`uvm_info("run_phase","read deasserted",UVM_LOW)
//write_read
`uvm_info("run_phase","write_read_started",UVM_LOW)
seq4.start(env.agt.sqr); //write_read_ sequence
`uvm_info("run_phase","write_read_finished",UVM_LOW)
phase.drop_objection(this);
endtask
endclass: ram_test
endpackage
```

RAM top:

```
import ram_test_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
module ram_top();
bit clk;

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

  // Instantiate the interface and DUT and golden
RAM_interface ram_if(clk);
```

```
RAM DUT
(  ram_if.din,ram_if.clk,ram_if.rst_n,ram_if.rx_valid,ram_if.dout,ram_if.tx_valid
);
RAM_golden DUT_golden (
ram_if.din,ram_if.clk,ram_if.rst_n,ram_if.rx_valid,ram_if.dout_gn,ram_if.tx_valid
_gn);
//bind_assertions
  bind RAM ram_sva my_ass(ram_if.sva);
initial
begin
  // run test
uvm_config_db #(virtual RAM_interface) :: set (null , "uvm_test_top" ,"key_ram"
,ram_if );

  run_test("ram_test");

end
endmodule
```

Assertions table:

Feature	Assertions
whenever reset is asserted, the output signals (tx_valid and dout) are low	@(posedge ram_if.clk) (!ram_if.rst_n) |=> (ram_if.dout == 0&& ram_if.tx_valid == 0)
checks that during address or data input phases(write_add_seq,write_data_seq,read_add_seq), the tx_valid signal must remain deasserted.	@(posedge ram_if.clk) ((ram_if.din[9:8]==2'b00||ram_if.din[9:8]==2'b01||ram_if.din[9:8]==2'b10)&& ram_if.rst_n&&ram_if.rx_valid) |=> (ram_if.tx_valid==0)
checks that after a read_data_seq occurs, the tx_valid signal must rise to indicate valid output and after it rises by one clock cycle, it should eventually fall.	@(posedge ram_if.clk) (ram_if.din[9:8]==2'b11&&ram_if.rst_n) |=> ($rose(ram_if.tx_valid) |=> $fell(ram_if.tx_valid)[->1])
checks that every Write Address operation must be eventually followed by a Write Data operation.	@(posedge ram_if.clk) (ram_if.din[9:8]==2'b00&&ram_if.rst_n) |=> (ram_if.din[9:8]==2'b01 [->1])
checks that every Read Address operation must be eventually followed by a Read Data operation.	@(posedge ram_if.clk) (ram_if.din[9:8]==2'b10&&ram_if.rst_n) |=> (ram_if.din[9:8]==2'b11 [->1])

Do file:

NOTE: as discussed i made seed random with sv_seed value “811737313“ to make the waveform as in word photos

```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.ram_top -classdebug -uvmcontrol=all -cover -sv_seed 811737313 -l sim.log
add wave /ram_top/ram_if/*
add wave -position insertpoint  \
sim:/ram_top/DUT/MEM \
```

```
sim:/ram_top/DUT/Rd_Addr \
sim:/ram_top/DUT/Wr_Addr
add wave -position insertpoint  \
sim:/ram_top/DUT_golden/addr_rd \
sim:/ram_top/DUT_golden/addr_wr \
sim:/ram_top/DUT_golden/mem
coverage save ram_top.ucdb -onexit
run -all
#quit -sim
```

Src_files.list:

RAM.v
golden_ram.v
interface_ram.sv
ram_sva.sv
ram_config.sv
ram_seq_item.sv
reset_sequence.sv
read_only_seq.sv
write_only_sequence.sv
write_read_seq.sv
ram_sequencer.sv
ram_driver.sv
ram_monitor.sv
ram_agent.sv
ram_scoreboard.sv
ram_cvr.sv
ram_env.sv
ram_test.sv
top_ram.sv

Questasim:

Wave form:

At 1ns checking reset is ok

At 9ns din [9:8] 2'b00 so it write address
Write address = din[7:0] = b5 in hexadecimal , 181 in decimal

At 11ns din[9:8] 2'b01 so it write data in previous address din[7:0] = de in hexadecimal in mem[181]
dout = 0 during writing operation no change in it

True writing at mem [181] de in hexadecimal

at 20007 ns read sequence has started din[9:8]=2'b10 so it read address din[7:0] = 34 in hexadecimal, 52 in decimal

at 20009 ns din[9:8] 2'b11 so it read data so dout = mem[52] = d4

Assertions:

/ram_sequence_write_read_pkg::ram_sequence_write_read::body/#ublk#156563271#18/immed__...	Immediate	SVA	on	0	1	-	-	-	-	off	assert (randomize(...))	✓
/ram_sequence_read_pkg::ram_sequence_read::body/#ublk#39469479#18/immed__20	Immediate	SVA	on	0	1	-	-	-	-	off	assert (randomize(...))	✓
/ram_sequence_write_pkg::ram_sequence_write::body/#ublk#44633143#18/immed__20	Immediate	SVA	on	0	1	-	-	-	-	off	assert (randomize(...))	✓
/ram_top/DUT/my_ass/assert__reset_is_Ok	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge ram_if.clk) (~ra...	✓
/ram_top/DUT/my_ass/assert__tx_deassert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge ram_if.clk) ((((r...	✓
/ram_top/DUT/my_ass/assert__tx_assert_deassert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge ram_if.clk) ((ra...	✓
/ram_top/DUT/my_ass/assert__write_eventually	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge ram_if.clk) ((ra...	✓
/ram_top/DUT/my_ass/assert__read_eventually	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert(@(posedge ram_if.clk) ((ra...	✓

Cover directive:

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/ram_top/DUT/my_ass/cover__read_eventually	SVA	✓	Off	6287	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/my_ass/cover__write_eventually	SVA	✓	Off	8032	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/my_ass/cover__tx_assert_deassert	SVA	✓	Off	5314	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/my_ass/cover__tx_deassert	SVA	✓	Off	18585	1	Unli...	1	100%		✓	0	0	0 ns	0
/ram_top/DUT/my_ass/cover__reset_is_Ok	SVA	✓	Off	3062	1	Unli...	1	100%		✓	0	0	0 ns	0

Cover group:

Covergroups

Class Type	Coverage	Name	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
	100.00%	/ram_coverage_pkg/ram_coverage							
	100.00%	TYPE cvr_group	100	100.00...		✓	auto(0)		
	100.00%	CVP cvr_group::Din	100	100.00...		✓			
	100.00%	CVP cvr_group::Rx	100	100.00...		✓			
	100.00%	CVP cvr_group::Tx	100	100.00...		✓			
	100.00%	CROSS cvr_group::cross1	100	100.00...		✓			
	100.00%	CROSS cvr_group::cross2	100	100.00...		✓			
	100.00%	INST \/ram_coverage_pkg::ram_coverage::cv...	100	100.00...		✓		0	
	100.00%	CVP Din	100	100.00...		✓			
	100.00%	CVP Rx	100	100.00...		✓			
	100.00%	CVP Tx	100	100.00...		✓			
	100.00%	CROSS cross1	100	100.00...		✓			
	100.00%	CROSS cross2	100	100.00...		✓			

Transcript:

```
Transcript
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test ram_test...
# UVM_INFO ram_test.sv(42) @ 0: uvm_test_top [run_phase] reset asserted
# ************************************************************************************
# * Questa UVM Transaction Recording Turned ON.                                      *
# * recording_detail has been set.                                                   *
# *  To turn off, set 'recording_detail' to off:                                     *
# * uvm_config_db#(int)          ::set(null, "", "recording_detail", 0); *
# * uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0); *
# ************************************************************************************
# UVM_INFO ram_test.sv(44) @ 2: uvm_test_top [run_phase] reset deasserted
# UVM_INFO ram_test.sv(46) @ 2: uvm_test_top [run_phase] write asserted
# UVM_INFO ram_test.sv(48) @ 20002: uvm_test_top [run_phase]  write deasserted
# UVM_INFO ram_test.sv(50) @ 20002: uvm_test_top [run_phase] read asserted
# UVM_INFO ram_test.sv(52) @ 40002: uvm_test_top [run_phase] read deasserted
# UVM_INFO ram_test.sv(54) @ 40002: uvm_test_top [run_phase] write_read_started
# UVM_INFO ram_test.sv(56) @ 60002: uvm_test_top [run_phase] write_read_finished
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 60002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   12
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [run_phase]     8
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 60002 ns  Iteration: 61  Instance: /ram_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

VSIM 10>
```

Cover report:

NOTE : in Code coverage default has been excluded

```
code_cvr.txt - Notepad
File  Edit  Format  View  Help
            bin <full_transition,high_rx>          1072        1        -    Covered
            bin <ra_rd,high_rx>                    6273        1        -    Covered
            bin <wa_wd,high_rx>                    6326        1        -    Covered
            bin <read_add,high_rx>                 6254        1        -    Covered
            bin <read_data,high_rx>                6273        1        -    Covered
            bin <write_add,high_rx>                8132        1        -    Covered
            bin <write_data,high_rx>               6326        1        -    Covered
    Cross cross2                                 100.00      100        -    Covered
        covered/total bins:                           1        1        -
        missing/total bins:                           0        1        -
        % Hit:                                   100.00      100        -
        Auto, Default and User Defined Bins:
            bin rd_tx                              5630        1        -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
                                      Unit   UnitType
--------------------------------------------------------------------------------
/ram_top/DUT/my_ass/cover__read_eventually
                                      ram_sva Verilog  SVA  ram_sva.sv(58)  6248 Covered
/ram_top/DUT/my_ass/cover__write_eventually
                                      ram_sva Verilog  SVA  ram_sva.sv(55)  8082 Covered
/ram_top/DUT/my_ass/cover__tx_assert_deassert
                                      ram_sva Verilog  SVA  ram_sva.sv(49)  5254 Covered
/ram_top/DUT/my_ass/cover__tx_deassert ram_sva Verilog  SVA  ram_sva.sv(43)  10501 Covered
/ram_top/DUT/my_ass/cover__reset_is_Ok ram_sva Verilog  SVA  ram_sva.sv(39)  3140 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 5

ASSERTION RESULTS:
--------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------
/ram_top/DUT/my_ass/assert__read_eventually
                     ram_sva.sv(57)                   0            1
/ram_top/DUT/my_ass/assert__write_eventually
                     ram_sva.sv(54)                   0            1
/ram_top/DUT/my_ass/assert__tx_assert_deassert
                     ram_sva.sv(48)                   0            1
/ram_top/DUT/my_ass/assert__tx_deassert
                     ram_sva.sv(42)                   0            1
/ram_top/DUT/my_ass/assert__reset_is_Ok
                     ram_sva.sv(38)                   0            1
/ram_sequence_write_read_pkg/ram_sequence_write_read/body/#ublk#156563271#18/immed__20
                     write_read_seq.sv(20)            0            1
/ram_sequence_read_pkg/ram_sequence_read/body/#ublk#304469479#18/immed__20
                     read_only_seq.sv(20)             0            1
/ram_sequence_write_pkg/ram_sequence_write/body/#ublk#44633143#18/immed__20
                     write_only_sequence.sv(20)
                                                      0            1
```

```
code_cvr.txt - Notepad
File  Edit  Format  View  Help
=== Design Unit: work.ram_coverage_pkg
----------------------------------------------------------------------------------

Covergroup Coverage:
    Covergroups                    1        na        na   100.00%
        Coverpoints/Crosses        5        na        na        na
            Covergroup Bins       17        17         0   100.00%
--------------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /ram_coverage_pkg/ram_coverage/cvr_group      100.00%        100          -    Covered
    covered/total bins:                                  17         17          -
    missing/total bins:                                   0         17          -
    % Hit:                                          100.00%        100          -
    Coverpoint Din                                  100.00%        100          -    Covered
        covered/total bins:                               7          7          -
        missing/total bins:                               0          7          -
        % Hit:                                      100.00%        100          -
    Coverpoint Rx                                   100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
    Coverpoint Tx                                   100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
    Cross cross1                                    100.00%        100          -    Covered
        covered/total bins:                               7          7          -
        missing/total bins:                               0          7          -
        % Hit:                                      100.00%        100          -
    Cross cross2                                    100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
 Covergroup instance \/ram_coverage_pkg::ram_coverage::cvr_group
                                                    100.00%        100          -    Covered
    covered/total bins:                                  17         17          -
    missing/total bins:                                   0         17          -
    % Hit:                                          100.00%        100          -
    Coverpoint Din                                  100.00%        100          -    Covered
        covered/total bins:                               7          7          -
        missing/total bins:                               0          7          -
        % Hit:                                      100.00%        100          -
        bin write_data                                 7033          1          -    Covered
        bin write_add                                  9056          1          -    Covered
        bin read_data                                  6956          1          -    Covered
        bin read_add                                   6956          1          -    Covered
        bin wa_wd                                      7033          1          -    Covered
        bin ra_rd                                      6956          1          -    Covered
        bin full_transition                            1189          1          -    Covered
    Coverpoint Rx                                   100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100
```

```
code_cvr.txt - Notepad
File  Edit  Format  View  Help

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

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

Expression Coverage for instance /ram_top/DUT --

  File RAM.v
----------------Focused Expression View-----------------
Line       29 Item    1  (din[9] && din[8])
Expression totals: 2 of 2 input terms covered = 100.00%

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

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


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

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

Statement Coverage for instance /ram_top/DUT --

    Line         Item                        Count     Source
    ----         ----                        -----     ------
  File RAM.v
    1                                                  module RAM (din,clk,rst_n,rx_valid,dout,tx_valid);

    2

    3                                                  input      [9:0] din;

    4                                                  input            clk, rst_n, rx_valid;

    5

    6                                                  output reg [7:0] dout;

    7                                                  output reg       tx_valid;
```

```
code_cvr.txt - Notepad
File  Edit  Format  View  Help
/ram_top/DUT/my_ass/cover__reset_is_Ok   ram_sva Verilog  SVA  ram_sva.sv(39)  3140 Covered

--------------------------------------------------------------------------------
=== Instance: /ram_top/DUT
=== Design Unit: work.RAM
--------------------------------------------------------------------------------
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         8         0   100.00%

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

Branch Coverage for instance /ram_top/DUT

    Line         Item                        Count     Source
    ----         ----                        -----     ------
  File RAM.v
------------------------------------IF Branch------------------------------------
    14                                       30001     Count coming in to IF
    14               1                        3140         if (~rst_n) begin

    20               1                       26861         else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    21                                       26861     Count coming in to IF
    21               1                       24144             if (rx_valid) begin

                                              2717     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    22                                       24144     Count coming in to CASE
    23               1                        7254                   2'b00 : Wr_Addr <= din[7:0];

    24               1                        5693                   2'b01 : MEM[Wr_Addr] <= din[7:0];

    25               1                        5615                   2'b10 : Rd_Addr <= din[7:0];

    26               1                        5582                   2'b11 : dout <= MEM[Rd_Addr]; //bug Rd_addr not Wr_Addr

Branch totals: 4 hits of 4 branches = 100.00%


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

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

Expression Coverage for instance /ram_top/DUT --
```

```
code_cvr.txt - Notepad
File  Edit  Format  View  Help
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         62        62         0   100.00%

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

Toggle Coverage for instance /ram_top/ram_i# --

                                          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_gn[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_gn          1           1      100.00

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

Toggle Coverage      =    100.00% (62 of 62 bins)

--------------------------------------------------------------------------------
=== Instance: /ram_top/DUT/my_ass
=== Design Unit: work.ram_sva
--------------------------------------------------------------------------------

Assertion Coverage:
    Assertions                       5         5         0   100.00%
--------------------------------------------------------------------------------
Name                File(line)                    Failure      Pass
                                                  Count       Count
--------------------------------------------------------------------------------
/ram_top/DUT/my_ass/assert__read_eventually
                          ram_sva.sv(57)              0           1
/ram_top/DUT/my_ass/assert__write_eventually
                          ram_sva.sv(54)              0           1
/ram_top/DUT/my_ass/assert__tx_assert_deassert
                          ram_sva.sv(68)              0           1
/ram_top/DUT/my_ass/assert__tx_deassert
                          ram_sva.sv(42)              0           1
/ram_top/DUT/my_ass/assert__reset_is_Ok
                          ram_sva.sv(38)              0           1

Directive Coverage:
    Directives                       5         5         0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
```

Wrapper verification:

In this part, build a complete UVM environment to verify the end to end functionality of the SPI wrapper design. The environment should integrate both the Single-Port RAM and the SPI Slave environments developed previously to reuse them but with passive agents

Verification Plan:

Label	Design requirement description	Stimulus generation	Functional coverage	Functionality check
Reset sequence	Check reset	Directed	-	A checker in testbench to make sure the output is correct
-Write only sequence -Read only sequence -Write Read sequence	Check other randomize cases	Randomize Under constraints: 1- The reset signal (rst_n) shall be deasserted most of the time. 2- The SS_n signal to be high for one cycle every 13 cycles for all cases except read data to be high for one cycle every 23 cycles. 3- Declare a randomized array to drive bit by bit the MOSI and the first 3 bits sent serially to the MOSI after the SS_n falls and reach chk_cmd are valid combinations only (000, 001, 110, 111)	-	A checker in testbench to make sure the output is correct

		4- For a write-only sequence, every Write Address operation shall always be followed by either Write Address or Write Data operation. 5- For a read-only sequence, every Read Address operation shall always be followed by Read Data. After a Read Data operation shall always be followed by Read Address. 6- For a randomized read/write sequence, the following ordering rules shall be enforced: • Every Write Address operation shall always be followed by either Write Address or Write Data operation. • After a Write Data, the next operation shall be chosen with the following probability distribution: 60% → Read Address & 40% → Write Address. • Every Read Address operation shall always be followed by Read Data operation. • After a Read Data, the next operation shall be chosen with the following probability distribution: 60% → Write Address & 40% → Read Address.		

Design:

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

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

wire [9:0] rx_data_din;
wire       rx_valid;
wire       tx_valid;
wire [7:0] tx_data_dout;
RAM   RAM_instance   (rx_data_din,clk,rst_n,rx_valid,tx_data_dout,tx_valid);
SLAVE SLAVE_instance
(MOSI,MISO,SS_n,clk,rst_n,rx_data_din,rx_valid,tx_data_dout,tx_valid);

endmodule
```

Golden module:

```
module spi_wrapper(clk,rst_n, SS_n,MOSI,MISO);
input  clk,rst_n, SS_n,MOSI;
output MISO;
wire [9:0] rx_data;
wire rx_valid;
wire tx_valid;
wire [7:0] tx_data;
 spi_slave  m1 (clk,rst_n,MOSI,MISO,SS_n,rx_data,rx_valid,tx_valid,tx_data);
 RAM_golden  m2 (rx_data,clk,rst_n,rx_valid,tx_data,tx_valid);
 endmodule
```

Wrapper shared package:

```
package wrapper_shared_pkg;
int error_count;
int correct_count;
int error_count_2;
int correct_count_2;
int error_count_r;
int correct_count_r;
int error_count_w;
int correct_count_w;
int error_count_s;
int correct_count_s;
int error_count_2_s;
int correct_count_2_s;
int Count_seq, i, address_recieved;
int count;
bit start_read;
endpackage
```

Wrapper interface:

```
import wrapper_shared_pkg::* ;
interface wrapper_if(clk);
input bit clk;
logic  MOSI, rst_n, SS_n;
logic MISO;
logic  MISO_ref;
modport SVA ( input clk,rst_n,SS_n,MOSI,output MISO);
endinterface
```

Wrapper assertions:

```
module wrapper_sva(wrapper_if.SVA wrapperif);

property assert_reset;
 @(posedge wrapperif.clk)  (!wrapperif.rst_n) |=> (wrapperif.MISO== 0);
endproperty

//2- An assertion to make sure that the MISO remains with a stable value
eventually as long as it is not a read data operation
property MISO_IS_OK;
  @(posedge wrapperif.clk) disable iff (!wrapperif.rst_n) (!(wrapperif.MOSI&&
$past(wrapperif.MOSI) && $past(wrapperif.MOSI,2))&& $past(wrapperif.SS_n,4))|=>
($stable(wrapperif.MISO) throughout (wrapperif.SS_n == 0));
endproperty
```

```
assert property (assert_reset);
cover  property (assert_reset);

assert property (MISO_IS_OK);
cover  property (MISO_IS_OK);

endmodule
```

wrapper config:

```
package wrapper_config_pkg;
import uvm_pkg::*;
`include "uvm_macros.svh"
class wrapper_config extends uvm_object;
`uvm_object_utils(wrapper_config)
virtual wrapper_if wrapper_vif;
virtual SPI_if spi_vif;
virtual RAM_interface ram_vif;
uvm_active_passive_enum   is_active;

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

Wrapper 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)


        typedef enum bit [1:0] { WR_ADDR_INS = 2'b00, WR_DATA_INS = 2'b01,
RD_ADDR_INS = 2'b10, RD_DATA_INS = 2'b11 } state_e;

        rand bit MOSI, SS_n, rst_n;
        bit MISO;
        bit MISO_ref;

        rand state_e state = WR_ADDR_INS;
```

```
        state_e state_old = WR_ADDR_INS;

        rand bit [10:0] mosi_arr;

        bit rand_signal, reset_flag;
        int count = 0;
        bit SS_n_signal = 1;


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

        function void pre_randomize();
            if (count == 0 || !rst_n) begin
                mosi_arr.rand_mode(1);
                state.rand_mode(1);
                rand_signal = 1;
            end
            else begin
                mosi_arr.rand_mode(0);
                state.rand_mode(0);
                rand_signal = 0;
            end
        endfunction


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

        constraint mosi_arr_c { mosi_arr[10:8] == {state[1],state}; }

        constraint MOSI_c {
            if (count > 0 && count < 12)
                MOSI == mosi_arr[11 - count];
        }

        constraint SS_n_c {
            SS_n == SS_n_signal;
        }

        constraint write_only_const {
            state inside {WR_ADDR_INS, WR_DATA_INS};
        }
```

```
        constraint read_only_const {
            if (start_read || reset_flag || (state_old == RD_DATA_INS &&
rand_signal)) {
                state == RD_ADDR_INS;
            }
            else if (state_old == RD_ADDR_INS && rand_signal) {
                state == RD_DATA_INS;
            }
        }

        constraint write_read_const {
            if (start_read || reset_flag || (state_old == WR_ADDR_INS &&
rand_signal)) {
                state inside {WR_ADDR_INS, WR_DATA_INS};
            }
            else if (state_old == WR_DATA_INS && rand_signal) {
                state dist {RD_ADDR_INS := 60, WR_ADDR_INS := 40};
            }
            else if (state_old == RD_DATA_INS && rand_signal) {
                state dist {WR_ADDR_INS := 60, RD_ADDR_INS := 40};
            }
            else if (state_old == RD_ADDR_INS && rand_signal) {
                state inside {RD_DATA_INS};
            }
        }

        function void post_randomize();
            if (!rst_n || SS_n) begin
                count = 0;
                SS_n_signal = 0;
            end
            else count++;

            if (count > 12 && count < 23) begin
                if (state != RD_DATA_INS)
                    SS_n_signal = 1;
                else
                    SS_n_signal = 0;
            end
            else if (count == 23) begin
                if (state == RD_DATA_INS)
                    SS_n_signal = 1;
                else
                    SS_n_signal = 0;
            end
```

```
            else SS_n_signal = 0;

            state_old = state;
            if (!rst_n) reset_flag = 1;
            else reset_flag = 0;
        endfunction


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

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

    endclass
endpackage
```

Wrapper reset sequence:

```
package wrapper_sequence_rst_pkg;
import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
 class WRAPPER_reset_seq extends uvm_sequence #(wrapper_seq_item);
        `uvm_object_utils(WRAPPER_reset_seq)

        wrapper_seq_item seq_item;

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

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

Wrapper read only sequence:

```
package wrapper_sequence_read_pkg;
import uvm_pkg::*;
import wrapper_shared_pkg::*;
`include "uvm_macros.svh"
import wrapper_seq_item_pkg::*;
class WRAPPER_read_only_seq extends uvm_sequence #(wrapper_seq_item);
        `uvm_object_utils(WRAPPER_read_only_seq)

        wrapper_seq_item seq_item;

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

    task body();
                seq_item = wrapper_seq_item::type_id::create("seq_item");
                seq_item.write_only_const.constraint_mode(0);
                seq_item.write_read_const.constraint_mode(0);
            for(int i = 0; i<10000; i++) begin
                start_item(seq_item);
                if(i == 1)
                start_read = 1;
                else
                start_read = 0;
                assert(seq_item.randomize);
                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

Wrapper write only sequence:

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


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

```
        wrapper_seq_item seq_item;

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

  task body();
                seq_item = wrapper_seq_item::type_id::create("seq_item");
                seq_item.read_only_const.constraint_mode(0);
                seq_item.write_read_const.constraint_mode(0);
            repeat (10000) begin
                start_item(seq_item);
                assert(seq_item.randomize);
                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

Wrapper write read sequence:

```
package wrapper_sequence_write_read_pkg;
import uvm_pkg::*;
import wrapper_shared_pkg::*;
`include "uvm_macros.svh"
import wrapper_seq_item_pkg::*;
class WRAPPER_write_read_seq extends uvm_sequence #(wrapper_seq_item);
        `uvm_object_utils(WRAPPER_write_read_seq)

        wrapper_seq_item seq_item;

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

      task body();
                seq_item = wrapper_seq_item::type_id::create("seq_item");
                seq_item.write_only_const.constraint_mode(0);
                seq_item.read_only_const.constraint_mode(0);

            for(int i = 0; i<10000; i++) begin
                start_item(seq_item);
                if(i == 1)
                start_read = 1;
                else
```

```
                start_read = 0;
                assert(seq_item.randomize);
                finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

Wrapper sequencer:

```
package wrapper_sequencer_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import wrapper_seq_item_pkg::* ;
import wrapper_shared_pkg::* ;
class wrapper_sequencer extends uvm_sequencer #(wrapper_seq_item);
 `uvm_component_utils(wrapper_sequencer)

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

endclass
endpackage
```

Wrapper driver:

```
package wrapper_driver_pkg;

import wrapper_seq_item_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class wrapper_driver extends uvm_driver#(wrapper_seq_item);
`uvm_component_utils(wrapper_driver)
virtual wrapper_if wrapper_vif;
wrapper_seq_item seq_item;

function new (string name ="wrapper_driver", uvm_component parent =null);
super.new(name,parent);
endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
  seq_item=wrapper_seq_item::type_id::create("seq_item");
  seq_item_port.get_next_item(seq_item);
```

```
wrapper_vif.rst_n= seq_item.rst_n;
wrapper_vif.SS_n= seq_item.SS_n;
wrapper_vif.MOSI     = seq_item.MOSI ;
 @(negedge wrapper_vif.clk);
seq_item_port.item_done();
`uvm_info("runphase",seq_item.convert2string_stimulus(),UVM_HIGH)
end

endtask

endclass
endpackage
```

Wrapper monitor:

```
package wrapper_monitor_pkg;
import wrapper_seq_item_pkg::*;
import wrapper_shared_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class wrapper_monitor extends uvm_monitor;
`uvm_component_utils(wrapper_monitor)
virtual wrapper_if wrapper_vif;
wrapper_seq_item seq_item;
uvm_analysis_port #(wrapper_seq_item)mon_ap;

function new (string name ="wrapper_monitor", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
  mon_ap = new ("mon_ap",this);
  endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
      seq_item=wrapper_seq_item::type_id::create("seq_item");
seq_item.rst_n=wrapper_vif.rst_n;
seq_item.SS_n= wrapper_vif.SS_n;
seq_item.MOSI =wrapper_vif.MOSI ;
seq_item.MISO =wrapper_vif.MISO ;
seq_item.MISO_ref=wrapper_vif.MISO ;
 @(negedge wrapper_vif.clk);
  mon_ap.write(seq_item);
`uvm_info("runphase",seq_item.convert2string_stimulus(),UVM_HIGH)
end
```

```
endtask

endclass
endpackage
```

Wrapper agent:

```
package wrapper_agent_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import wrapper_seq_item_pkg::* ;
import wrapper_shared_pkg::* ;
import wrapper_monitor_pkg::* ;
import wrapper_driver_pkg::* ;
import wrapper_sequencer_pkg::* ;
import wrapper_config_pkg::* ;
class wrapper_agent extends uvm_agent;
`uvm_component_utils(wrapper_agent)
wrapper_sequencer sqr;
wrapper_driver drv;
wrapper_monitor mon;
wrapper_config conf;
uvm_analysis_port #(wrapper_seq_item) agt_ap;
function new(string name = "wrapper_agent", uvm_component parent = null);
 super.new(name,parent);
endfunction

function void build_phase(uvm_phase phase);
 super.build_phase(phase);
if(!uvm_config_db #(wrapper_config)::get(this, "", "CFG", conf))
 `uvm_fatal("build_phase", "unable to get configuration object")
mon = wrapper_monitor::type_id::create("mon", this);
agt_ap = new("agt_ap", this);
if (conf.is_active==UVM_ACTIVE)begin
sqr=wrapper_sequencer::type_id::create("sqr",this);
drv=wrapper_driver::type_id::create("drv",this);
end
endfunction

function void connect_phase(uvm_phase phase);
 super.connect_phase(phase);
 if (conf.is_active==UVM_ACTIVE)begin
drv.wrapper_vif = conf.wrapper_vif;
drv.seq_item_port.connect(sqr.seq_item_export);
end
mon.wrapper_vif = conf.wrapper_vif;
```

```
mon.mon_ap.connect(agt_ap);
endfunction
endclass
endpackage
```

SPI agent:

```
package SPI_agent_pkg;
import uvm_pkg::* ;
`include "uvm_macros.svh"
import SPI_seq_item_pkg::* ;
import wrapper_shared_pkg::*;
import SPI_monitor_pkg::* ;
import SPI_driver_pkg::* ;
import SPI_sequencer_pkg::* ;
import wrapper_config_pkg::* ;
class SPI_agent extends uvm_agent;
 `uvm_component_utils(SPI_agent)
SPI_sequencer sqr;
SPI_driver drv;
SPI_monitor mon;
wrapper_config conf;
uvm_analysis_port #(SPI_seq_item) agt_ap;

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

function void build_phase(uvm_phase phase);
 super.build_phase(phase);
if(!uvm_config_db #(wrapper_config)::get(this, "", "CFGs", conf))
 `uvm_fatal("build_phase", "unable to get configuration object")
if (conf.is_active==UVM_ACTIVE)begin
sqr=SPI_sequencer::type_id::create("sqr",this);
drv=SPI_driver::type_id::create("drv",this);
end
mon = SPI_monitor::type_id::create("mon", this);
agt_ap = new("agt_ap", this);

endfunction

function void connect_phase(uvm_phase phase);
 super.connect_phase(phase);
 if (conf.is_active==UVM_ACTIVE)begin
drv.SPI_vif = conf.spi_vif;
drv.seq_item_port.connect(sqr.seq_item_export);
```

```
end
mon.SPI_vif = conf.spi_vif;
mon.mon_ap.connect(agt_ap);

endfunction
endclass
endpackage
```

RAM agent:

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

class ram_agent extends uvm_agent;
`uvm_component_utils(ram_agent)
ram_sequencer sqr;
ram_monitor mon;
ram_driver drv;
wrapper_config cfg;
uvm_analysis_port #(ram_sequence_item)agt_ap;
function new (string name ="ram_agent", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
   if (!uvm_config_db #(wrapper_config)::get(this,"","CFGr",cfg ))
  `uvm_fatal("build_phase","failed");
  if (cfg.is_active==UVM_ACTIVE)begin
sqr=ram_sequencer::type_id::create("sqr",this);
drv=ram_driver::type_id::create("drv",this);
end

mon=ram_monitor::type_id::create("mon",this);
agt_ap=new("agt_ap",this);
endfunction
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
 if (cfg.is_active==UVM_ACTIVE)begin
drv.ram_vif = cfg.ram_vif;
drv.seq_item_port.connect(sqr.seq_item_export);
```

```
end
mon.ram_vif=cfg.ram_vif;
mon.mon_ap.connect(agt_ap);                        //monitor/analysis port agent
endfunction

endclass
endpackage
```

Ram coverage

NOTE:now din is not randomized and take its value from rx_data so we will ignore un necessary transition(2 bits will never change at once)

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

class ram_coverage extends uvm_component;
`uvm_component_utils(ram_coverage)
uvm_analysis_export #(ram_sequence_item) cov_export;
uvm_tlm_analysis_fifo  #(ram_sequence_item) cov_fifo;
ram_sequence_item seq_item;

 covergroup cvr_group;
    Din:coverpoint seq_item.din[9:8]{
bins write_data={2'b01};
bins write_add={2'b00};
bins read_data={2'b11};
bins read_add={2'b10};
bins wa_wd=(2'b00=>2'b01);
bins ra_rd=(2'b10=>2'b11);
bins full_transition=(2'b00=>2'b01=>2'b10=>2'b11);
    ignore_bins ignored_trans_ram1 = (0 => 3);
        ignore_bins ignored_trans_ram2 = (1 => 2);
        ignore_bins ignored_trans_ram3 = (2 => 1);
}

   Rx:coverpoint seq_item.rx_valid{
       bins high_rx={1'b1};
}

   Tx:coverpoint seq_item.tx_valid{
       bins high_tx={1'b1};
}
 cross1 : cross Din , Rx{
```

```
    option.cross_auto_bin_max=0;
    bins wd_rx =binsof (Din.write_data) && binsof (Rx.high_rx) ;
        bins wa_rx =binsof (Din.write_add) && binsof (Rx.high_rx) ;
        bins rd_rx =binsof (Din.read_data) && binsof (Rx.high_rx) ;
            bins ra_rx =binsof (Din.read_add) && binsof (Rx.high_rx) ;
}

cross2: cross Din,Tx {
    option.cross_auto_bin_max=0;
    bins rd_tx =binsof (Din.read_data) && binsof (Tx.high_tx) ;
}

endgroup

function new (string name ="ram_coverage", uvm_component parent =null);
super.new(name,parent);
cvr_group=new();
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
cov_export=new("cov_export",this);
cov_fifo=new("cov_fifo",this);
endfunction
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
cov_export.connect(cov_fifo.analysis_export);
endfunction
task run_phase(uvm_phase phase);
super.run_phase(phase);
forever begin
    cov_fifo.get(seq_item);
    cvr_group.sample();
end
endtask
endclass
endpackage
```

Wrapper scoreboard:

```
package wrapper_scoreboard_pkg;
import uvm_pkg::* ;
```

```
`include "uvm_macros.svh"
import wrapper_seq_item_pkg::* ;
import wrapper_shared_pkg::* ;
class wrapper_scoreboard extends uvm_scoreboard;
 `uvm_component_utils(wrapper_scoreboard)
 uvm_analysis_export #(wrapper_seq_item) sb_export;
uvm_tlm_analysis_fifo #(wrapper_seq_item) sb_fifo;
wrapper_seq_item seq_item_sb;

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

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

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

task run_phase(uvm_phase phase);
 super.run_phase(phase);
forever begin
 sb_fifo.get(seq_item_sb);


 if(seq_item_sb.MISO != seq_item_sb.MISO_ref) begin
  `uvm_error("run_phase", $sformatf (" Comparsion 3failed, transaction done by
DUT = %s , MISO_ref = 0b%0b" , seq_item_sb.convert2string(),
seq_item_sb.MISO_ref));
  error_count_w++;
 end
 else begin
 `uvm_info("run_phase" , $sformatf("Correct_out3 : %s",
seq_item_sb.convert2string()), UVM_HIGH);
  correct_count_w++;
 end

end
endtask
```

```
function void report_phase(uvm_phase phase);
 super.report_phase(phase);
`uvm_info("report_phase", $sformatf("Total successful transactions: %d",
correct_count_w), UVM_MEDIUM);
`uvm_info("report_phase", $sformatf("Total failed transactions: %d",
error_count_w), UVM_MEDIUM);
endfunction


endclass
endpackage
```

Wrapper environment:

```
package wrapper_env_pkg;
`include "uvm_macros.svh"
import wrapper_scoreboard_pkg::*;
import wrapper_agent_pkg::*;
import uvm_pkg::*;
class wrapper_env extends uvm_env;
`uvm_component_utils(wrapper_env)
wrapper_agent agt;
wrapper_scoreboard sb;
wrapper_coverage cvr;
function new (string name ="wrapper_env", uvm_component parent =null);
super.new(name,parent);
endfunction
function void build_phase (uvm_phase phase );
  super.build_phase (phase);
agt=wrapper_agent :: type_id:: create ("agt",this);
sb=wrapper_scoreboard :: type_id:: create ("sb",this);
  endfunction
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
agt.agt_ap.connect(sb.sb_export);
endfunction
endclass
endpackage
```

Wrapper test:

```
package wrapper_test_pkg;
import wrapper_sequence_write_pkg::*;
import wrapper_sequence_write_read_pkg::*;
import wrapper_sequence_read_pkg::*;
```

```
import wrapper_sequence_rst_pkg::*;
import wrapper_env_pkg::*;
import SPI_env_pkg::*;
import ram_env_pkg::*;
import wrapper_config_pkg::*;
import uvm_pkg::*;
import wrapper_shared_pkg::*;
`include "uvm_macros.svh"
class wrapper_test extends uvm_test;
`uvm_component_utils(wrapper_test)
wrapper_env env_w;
SPI_env env_s;
ram_env env_r;
wrapper_config cfg_w;
wrapper_config cfg_s;
wrapper_config cfg_r;
WRAPPER_reset_seq seq1;
WRAPPER_write_only_seq seq2;
WRAPPER_read_only_seq seq3;
WRAPPER_write_read_seq seq4;
function new (string name ="wrapper_test", uvm_component parent =null);
super.new(name,parent);
endfunction
  // Build the enviornment in the build phase
  function void build_phase (uvm_phase phase );
  super.build_phase (phase);
  env_w =wrapper_env :: type_id:: create ("env_w",this);
  env_s =SPI_env :: type_id:: create ("env_r",this);
  env_r =ram_env :: type_id:: create ("env_s",this);
cfg_w = wrapper_config :: type_id:: create ("wrapper_cfg");
cfg_s = wrapper_config :: type_id:: create ("spi_cfg");
cfg_r = wrapper_config :: type_id:: create ("ram_cfg");

  seq1= WRAPPER_reset_seq :: type_id:: create ("seq1");
  seq2= WRAPPER_write_only_seq :: type_id:: create ("seq2");
  seq3= WRAPPER_read_only_seq :: type_id:: create ("seq3");
  seq4= WRAPPER_write_read_seq :: type_id:: create ("seq4");
cfg_w.is_active  = UVM_ACTIVE;
cfg_s.is_active = UVM_PASSIVE;
cfg_r.is_active = UVM_PASSIVE;
  if (!uvm_config_db #(virtual
wrapper_if)::get(this,"","wrapper_if",cfg_w.wrapper_vif ))
  `uvm_fatal("build_phase","failed");

  uvm_config_db#(wrapper_config)::set(this,"*","CFG",cfg_w);
```

```
   if (!uvm_config_db #(virtual SPI_if)::get(this,"","spi_if",cfg_s.spi_vif ))
  `uvm_fatal("build_phase","failed");

  uvm_config_db#(wrapper_config)::set(this,"*","CFGs",cfg_s);

   if (!uvm_config_db #(virtual
RAM_interface)::get(this,"","ram_if",cfg_r.ram_vif ))
  `uvm_fatal("build_phase","failed");

  uvm_config_db#(wrapper_config)::set(this,"*","CFGr",cfg_r);
  endfunction


task run_phase(uvm_phase phase);
super.run_phase(phase);
phase.raise_objection(this);
`uvm_info("run_phase","reset asserted",UVM_LOW)
seq1.start(env_w.agt.sqr);
`uvm_info("run_phase","reset deasserted",UVM_LOW)

//mainsequence
`uvm_info("run_phase","write started",UVM_LOW)
seq2.start(env_w.agt.sqr); //write sequence
`uvm_info("run_phase","write finished",UVM_LOW)

`uvm_info("run_phase","reset asserted",UVM_LOW)
seq1.start(env_w.agt.sqr);
`uvm_info("run_phase","reset deasserted",UVM_LOW)

`uvm_info("run_phase","read started",UVM_LOW)
seq3.start(env_w.agt.sqr); //read sequence
`uvm_info("run_phase","read finished",UVM_LOW)

`uvm_info("run_phase","reset asserted",UVM_LOW)
seq1.start(env_w.agt.sqr);
`uvm_info("run_phase","reset deasserted",UVM_LOW)

`uvm_info("run_phase","write_read started",UVM_LOW)
seq4.start(env_w.agt.sqr); //write_read sequence
`uvm_info("run_phase","write_read finished",UVM_LOW)

  `uvm_info("REPORT_SUMMARY",
            $sformatf("Total  recorded in shared_pkg.error_count = %0d",
error_count_w),UVM_LOW)
```

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

Wrapper top:

```
import uvm_pkg::*;
`include "uvm_macros.svh"
import wrapper_test_pkg::*;
module wrapper_top();
bit clk;

initial begin
           $readmemb("mem.dat", DUT_WRAPPER.RAM_instance.MEM);
      $readmemb("mem1.dat", golden_wrapper.m2.mem);
    forever begin
      #1 clk =~clk;
    end

  end


RAM_interface ramif(clk);
SPI_if spiif(clk);
wrapper_if wrapperif(clk);
WRAPPER DUT_WRAPPER
(wrapperif.MOSI,wrapperif.MISO,wrapperif.SS_n,wrapperif.clk,wrapperif.rst_n);
spi_wrapper golden_wrapper (wrapperif.clk,wrapperif.rst_n,
wrapperif.SS_n,wrapperif.MOSI,wrapperif.MISO_ref);




  SLAVE SLAVE_DUT
(spiif.MOSI,spiif.MISO,spiif.SS_n,spiif.clk,spiif.rst_n,spiif.rx_data,spiif.rx_va
lid,spiif.tx_data,spiif.tx_valid);


  spi_slave SLAVE_GOLDEN(spiif.clk,
spiif.rst_n,spiif.MOSI,spiif.MISO_ref,spiif.SS_n,
spiif.rx_data_ref,spiif.rx_valid_ref,spiif.tx_valid,spiif.tx_data);

    assign spiif.rst_n = DUT_WRAPPER.rst_n;

    assign spiif.SS_n = DUT_WRAPPER.SS_n;
```

```
    assign spiif.tx_valid = DUT_WRAPPER.tx_valid;
    assign spiif.tx_data = DUT_WRAPPER.tx_data_dout;
    assign spiif.MOSI = DUT_WRAPPER.MOSI;


  RAM RAM_DUT(ramif.din, ramif.clk, ramif.rst_n, ramif.rx_valid, ramif.dout,
ramif.tx_valid);


  RAM_golden  RAM_GOLDEN(ramif.din,ramif.clk, ramif.rst_n,  ramif.rx_valid,
ramif.dout_gn, ramif.tx_valid_gn);

    assign ramif.rst_n = DUT_WRAPPER.rst_n;
    assign ramif.rx_valid = DUT_WRAPPER.rx_valid;
    assign ramif.din = DUT_WRAPPER.rx_data_din;



  bind SLAVE SPI_sva ass_spi(spiif.svaa);
    bind RAM ram_sva my_ass(ramif.sva);
       bind WRAPPER wrapper_sva wrap_ass(wrapperif.SVA);
initial begin
   uvm_config_db#(virtual wrapper_if)::set(null, "uvm_test_top","wrapper_if",
wrapperif);
   uvm_config_db#(virtual SPI_if)::set(null, "uvm_test_top", "spi_if", spiif);
   uvm_config_db#(virtual RAM_interface)::set(null, "uvm_test_top", "ram_if",
ramif);
   run_test("wrapper_test");
  end
endmodule
```

Assertions table:

Feature	Assertions
whenever reset is asserted, the output (MISO) is inactive.	@(posedge ram_if.clk) (!ram_if.rst_n) |=> (ram_if.dout == 0&& ram_if.tx_valid == 0)
checks that during address or data input phases(write_add_seq,write_data_seq,read_add_seq), the tx_valid signal must remain deasserted.	@(posedge ram_if.clk) ((ram_if.din[9:8]==2'b00||ram_if.din[9:8]==2'b01||ram_if.din[9:8]==2'b10)&& ram_if.rst_n&&ram_if.rx_valid) |=> (ram_if.tx_valid==0)
checks that after a read_data_seq occurs, the tx_valid signal must rise to indicate valid output and after it rises by one clock cycle, it should eventually fall.	@(posedge ram_if.clk) (ram_if.din[9:8]==2'b11&&ram_if.rst_n) |=> ($rose(ram_if.tx_valid) |=> $fell(ram_if.tx_valid)[->1])

Do file:

NOTE: as discussed i made seed random with sv_seed value “829610917 “ to make the waveform as in word photos

```
vlib work
vlog -f src_files.list +cover -covercells
vsim -voptargs=+acc work.wrapper_top -classdebug -uvmcontrol=all -cover -sv_seed 829610917 -l sim.log
add wave -position insertpoint sim:/wrapper_top/wrapperif/*
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/RAM_instance/MEM \
sim:/wrapper_top/DUT_WRAPPER/RAM_instance/Rd_Addr \
sim:/wrapper_top/DUT_WRAPPER/RAM_instance/Wr_Addr
add wave -position insertpoint  \
sim:/wrapper_shared_pkg::error_count_w \
sim:/wrapper_shared_pkg::error_count_2_s
add wave -position insertpoint  \
sim:/wrapper_top/golden_wrapper/tx_valid \
sim:/wrapper_top/golden_wrapper/tx_data
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/tx_valid \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/counter \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/cs \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/ns
add wave -position insertpoint  \
sim:/wrapper_top/golden_wrapper/m1/tx_valid \
sim:/wrapper_top/golden_wrapper/m1/tx_data \
sim:/wrapper_top/golden_wrapper/m1/bit_cnt_rx
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/tx_data
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/rx_valid
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/RAM_instance/rx_valid
add wave -position insertpoint  \
sim:/wrapper_top/golden_wrapper/m1/rx_valid
add wave -position insertpoint  \
sim:/wrapper_top/golden_wrapper/m2/rx_valid
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/SLAVE_instance/rx_data
add wave -position insertpoint  \
sim:/wrapper_top/DUT_WRAPPER/RAM_instance/din
add wave -position insertpoint  \
sim:/wrapper_top/RAM_GOLDEN/mem
coverage save top.ucdb -onexit
run -all
```

Src_files.list:

wrapper_shared.sv
wrapper_golden.sv
SPI_wrapper.v
wrapper_interface.sv

wrapper_config.sv
wrapper_seq_item.sv
ram_seq_item.sv
SPI_seq_item.sv
SPI_slave.sv
golden_SPI_slave.v
SPI_if.sv

RAM.v
golden_ram.v
interface_ram.sv

wrapper_rst.sv
wrapper_write_seq.sv
wrapper_read_seq.sv
wrapper_read_write_seq.sv

wrapper_sequencer.sv
SPI_sequencer.sv
ram_sequencer.sv

ram_driver.sv
SPI_driver.sv
wrapper_driver.sv
ram_monitor.sv
SPI_monitor.sv
wrapper_monitor.sv

ram_agent.sv
SPI_agent.sv
wrapper_agent.sv

ram_scoreboard.sv
SPI_scoreboard.sv
wrapper_sc.sv

ram_cvr.sv
SPI_cvr_collector.sv

wrapper_coverage.sv
spi_sva.sv
ram_sva.sv
ram_env.sv
SPI_env.sv
wrapper_sva.sv
wrapper_env.sv
wrapper_test.sv
wrapper_top.sv

Questasim:

Waveform:

at 1ns checking reset

At 7ns received first Mosi=0 so transition from chk cmd to write

At 9ns and 11ns received Mosi =0 , 1 so it write data

Received 8 Mosi bits after that then rx valid =1

Then next cycle expecting in mem[0] contain data written which is 0110 0101 in binary = 65 in hexadecimal (at 31 ns)

At 39ns received first 3 bits from Mosi which 000 so it is write address

At 57 ns received the 8 bits from Mosi (00111010 in binary which is 3a in hexadecimal) then after that rx valid =1

At 59ns write address contain 3a

At 20363ns: first bit of Mosi received =1 and address received =0 so it is transition from chk cmd to read address & bit no 9 and 8 =2'b10

At 20383 ns received 8 bits from mosi which is 8'b11011110 = de in hexa then after clk cycle rx valid =1 then after clk cycle RD_addr= de

At 20391 address received =1 and and first received bit from Mosi =1 and then 2 bits received from Mosi =2'b11 so it read data

At 20415 tx data =de = 1101 1110 so starting from next cycle miso 11011110

Memory of read address de (222 in decimal) contain de that is in tx data

At 20417ns starting from it miso will take values

Assertions:

Cover directive:

Cover group:

Cover report:

*code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
                                        ram_sva Verilog  SVA  ram_sva.sv(43)  2882 Covered
/wrapper_top/RAM_DUT/my_ass/cover__reset_is_Ok
                                        ram_sva Verilog  SVA  ram_sva.sv(39)   298 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 16

ASSERTION RESULTS:
--------------------------------------------------------------------
Name                File(Line)                    Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/assert__read_eventually
                    ram_sva.sv(57)                0            1
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/assert__write_eventually
                    ram_sva.sv(54)                0            1
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/assert__tx_assert_deassert
                    ram_sva.sv(48)                0            1
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/assert__tx_deassert
                    ram_sva.sv(42)                0            1
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/assert__reset_is_Ok
                    ram_sva.sv(38)                0            1
/wrapper_top/DUT_WRAPPER/SLAVE_instance/ass_spi/assert__rx_valid_p
                    spi_sva.sv(33)                0            1
/wrapper_top/DUT_WRAPPER/SLAVE_instance/ass_spi/assert__rst_is_ok
                    spi_sva.sv(30)                0            1
/wrapper_top/DUT_WRAPPER/wrap_ass/assert__MISO_IS_OK
                    wrapper_sva.sv(19)            0            1
/wrapper_top/DUT_WRAPPER/wrap_ass/assert__assert_reset
                    wrapper_sva.sv(15)            0            1
/wrapper_top/SLAVE_DUT/ass_spi/assert__rx_valid_p
                    spi_sva.sv(33)                0            1
/wrapper_top/SLAVE_DUT/ass_spi/assert__rst_is_ok
                    spi_sva.sv(30)                0            1
/wrapper_top/RAM_DUT/my_ass/assert__read_eventually
                    ram_sva.sv(57)                0            1
/wrapper_top/RAM_DUT/my_ass/assert__write_eventually
                    ram_sva.sv(54)                0            1
/wrapper_top/RAM_DUT/my_ass/assert__tx_assert_deassert
                    ram_sva.sv(48)                0            1
/wrapper_top/RAM_DUT/my_ass/assert__tx_deassert
                    ram_sva.sv(42)                0            1
/wrapper_top/RAM_DUT/my_ass/assert__reset_is_Ok
                    ram_sva.sv(38)                0            1
/wrapper_sequence_read_pkg/WRAPPER_read_only_seq/body/#anonblk#164777879#19#4#/#ublk#164777879#19/immed__25
                    wrapper_read_seq.sv(25)       0            1
/wrapper_sequence_write_read_pkg/WRAPPER_write_read_seq/body/#anonblk#14481783#20#4#/#ublk#14481783#20/immed__26
                    wrapper_read_write_seq.sv(26)
                                                  0            1
/wrapper_sequence_write_pkg/WRAPPER_write_only_seq/body/#ublk#170518951#21/immed__23
```

*code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 2

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
                                      Unit   UnitType
--------------------------------------------------------------------------------
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/cover__read_eventually
                                        ram_sva Verilog  SVA  ram_sva.sv(58)  7883 Covered
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/cover__write_eventually
                                        ram_sva Verilog  SVA  ram_sva.sv(55)  9497 Covered
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/cover__tx_assert_deassert
                                        ram_sva Verilog  SVA  ram_sva.sv(49)   330 Covered
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/cover__tx_deassert
                                        ram_sva Verilog  SVA  ram_sva.sv(43)  2882 Covered
/wrapper_top/DUT_WRAPPER/RAM_instance/my_ass/cover__reset_is_Ok
                                        ram_sva Verilog  SVA  ram_sva.sv(39)   298 Covered
/wrapper_top/DUT_WRAPPER/SLAVE_instance/ass_spi/cover__rx_valid_p
                                        SPI_sva Verilog  SVA  spi_sva.sv(34)  1320 Covered
/wrapper_top/DUT_WRAPPER/SLAVE_instance/ass_spi/cover__rst_is_ok
                                        SPI_sva Verilog  SVA  spi_sva.sv(31)   298 Covered
/wrapper_top/DUT_WRAPPER/wrap_ass/cover__MISO_IS_OK
                                        wrapper_sva Verilog  SVA  wrapper_sva.sv(20)
                                                                                1362 Covered
/wrapper_top/DUT_WRAPPER/wrap_ass/cover__assert_reset
                                        wrapper_sva Verilog  SVA  wrapper_sva.sv(16)
                                                                                 298 Covered
/wrapper_top/SLAVE_DUT/ass_spi/cover__rx_valid_p
                                        SPI_sva Verilog  SVA  spi_sva.sv(34)  1320 Covered
/wrapper_top/SLAVE_DUT/ass_spi/cover__rst_is_ok
                                        SPI_sva Verilog  SVA  spi_sva.sv(31)   298 Covered
/wrapper_top/RAM_DUT/my_ass/cover__read_eventually
                                        ram_sva Verilog  SVA  ram_sva.sv(58)  7883 Covered
/wrapper_top/RAM_DUT/my_ass/cover__write_eventually
                                        ram_sva Verilog  SVA  ram_sva.sv(55)  9497 Covered
/wrapper_top/RAM_DUT/my_ass/cover__tx_assert_deassert
                                        ram_sva Verilog  SVA  ram_sva.sv(49)   330 Covered
/wrapper_top/RAM_DUT/my_ass/cover__tx_deassert
                                        ram_sva Verilog  SVA  ram_sva.sv(43)  2882 Covered
/wrapper_top/RAM_DUT/my_ass/cover__reset_is_Ok
                                        ram_sva Verilog  SVA  ram_sva.sv(39)   298 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 16

ASSERTION RESULTS:
--------------------------------------------------------------------
Name                File(Line)                    Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
```

*code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
    Summary                   Bins      Hits    Misses  Coverage
    -------                   ----      ----    ------  --------
        FSM States               5         5         0  100.00%
        FSM Transitions          8         8         0  100.00%
Statement Coverage:
    Enabled Coverage          Bins      Hits    Misses  Coverage
    ----------------          ----      ----    ------  --------
    Statements                  39        39         0  100.00%

--------------------------------Statement Details--------------------------------

Statement Coverage for Instance /wrapper_top/SLAVE_DUT --

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

    2

    3                                             localparam IDLE      = 3'b000;

    4                                             localparam WRITE     = 3'b001;

    5                                             localparam CHK_CMD   = 3'b010;

    6                                             localparam READ_ADD  = 3'b011;

    7                                             localparam READ_DATA = 3'b100;

    8

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

    10                                            input      [7:0] tx_data;

    11                                            output reg [9:0] rx_data;

    12                                            output reg       rx_valid, MISO;

    13

    14                                            reg [3:0] counter;

    15                                            reg       received_address;

    16

    17                                            reg [2:0] cs, ns;

    18
```

*code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
==============================================================================
=== Instance: /SPI_cvr_collector_pkg
=== Design Unit: work.SPI_cvr_collector_pkg
==============================================================================

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

--------------------------------------------------------------------------------
 TYPE /SPI_cvr_collector_pkg/SPI_cvr_collector/SPIcg
                                                   100.00%        100          -    Covered
    covered/total bins:                                 23         23          -
    missing/total bins:                                  0         23          -
    % Hit:                                         100.00%        100          -
    Coverpoint cp1                                 100.00%        100          -    Covered
        covered/total bins:                             13         13          -
        missing/total bins:                              0         13          -
        % Hit:                                     100.00%        100          -
    Coverpoint cp2                                 100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint cp3                                 100.00%        100          -    Covered
        covered/total bins:                              4          4          -
        missing/total bins:                              0          4          -
        % Hit:                                     100.00%        100          -
    Cross cross_MOSI_SS_n                          100.00%        100          -    Covered
        covered/total bins:                              4          4          -
        missing/total bins:                              0          4          -
        % Hit:                                     100.00%        100          -
  Covergroup instance \/SPI_cvr_collector_pkg::SPI_cvr_collector::SPIcg
                                                   100.00%        100          -    Covered
    covered/total bins:                                 23         23          -
    missing/total bins:                                  0         23          -
    % Hit:                                         100.00%        100          -
    Coverpoint cp1                                 100.00%        100          -    Covered
        covered/total bins:                             13         13          -
        missing/total bins:                              0         13          -
```

*code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
================================================================================
=== Instance: /ram_coverage_pkg
=== Design Unit: work.ram_coverage_pkg
================================================================================

Covergroup Coverage:
    Covergroups                    1        na        na   100.00%
        Coverpoints/Crosses        5        na        na        na
            Covergroup Bins       13        13         0   100.00%
--------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status
                                                                
--------------------------------------------------------------------------------
 TYPE /ram_coverage_pkg/ram_coverage/cvr_group      100.00%        100          -    Covered
    covered/total bins:                                    13         13          -
    missing/total bins:                                     0         13          -
    % Hit:                                             100.00%        100          -
    Coverpoint Din                                     100.00%        100          -    Covered
        covered/total bins:                                  6          6          -
        missing/total bins:                                  0          6          -
        % Hit:                                         100.00%        100          -
    Coverpoint Rx                                      100.00%        100          -    Covered
        covered/total bins:                                  1          1          -
        missing/total bins:                                  0          1          -
        % Hit:                                         100.00%        100          -
    Coverpoint Tx                                      100.00%        100          -    Covered
        covered/total bins:                                  1          1          -
        missing/total bins:                                  0          1          -
        % Hit:                                         100.00%        100          -
    Cross cross1                                       100.00%        100          -    Covered
        covered/total bins:                                  4          4          -
        missing/total bins:                                  0          4          -
        % Hit:                                         100.00%        100          -
    Cross cross2                                       100.00%        100          -    Covered
        covered/total bins:                                  1          1          -
        missing/total bins:                                  0          1          -
        % Hit:                                         100.00%        100          -
 Covergroup instance \/ram_coverage_pkg::ram_coverage::cvr_group
                                                       100.00%        100          -    Covered
    covered/total bins:                                    13         13          -
```

code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

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

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

Toggle Coverage for instance /wrapper_top/DUT_WRAPPER/SLAVE_instance --

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

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

Toggle Coverage      =     100.00% (72 of 72 bins)

================================================================================
=== Instance: /wrapper_top/DUT_WRAPPER/wrap_ass
=== Design Unit: work.wrapper_sva
================================================================================

Assertion Coverage:
    Assertions                       2         2         0   100.00%
--------------------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------------------
/wrapper_top/DUT_WRAPPER/wrap_ass/assert__MISO_IS_OK
                     wrapper_sva.sv(19)                  0            1
/wrapper_top/DUT_WRAPPER/wrap_ass/assert__assert_reset
                     wrapper_sva.sv(15)                  0            1
```

Type here to search

code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
FSM_ID: cs
    Current State Object : cs
    -------------------------
    State Value MapInfo :
    -------------------------
Line          State Name                Value
----          ----------                -----
  30               IDLE                    0
  36            CHK_CMD                    2
  62          READ_DATA                    4
  56           READ_ADD                    3
  50              WRITE                    1
    Covered States :
    ----------------
                   State            Hit_count
                   -----            ---------
                    IDLE                 2049
                 CHK_CMD                 2023
               READ_DATA                  752
                READ_ADD                  863
                   WRITE                 2363
    Covered Transitions :
    -------------------------
Line            Trans_ID           Hit_count          Transition
----            --------           ---------          ----------
  34                   0                2023          IDLE -> CHK_CMD
  46                   1                 380          CHK_CMD -> READ_DATA
  44                   2                 434          CHK_CMD -> READ_ADD
  41                   3                1188          CHK_CMD -> WRITE
  38                   4                  21          CHK_CMD -> IDLE
  64                   5                 379          READ_DATA -> IDLE
  58                   6                 434          READ_ADD -> IDLE
  52                   7                1188          WRITE -> IDLE


    Summary                         Bins      Hits    Misses  Coverage
    -------                         ----      ----    ------  --------
        FSM States                     5         5         0   100.00%
        FSM Transitions                8         8         0   100.00%
Statement Coverage:
    Enabled Coverage                Bins      Hits    Misses  Coverage
    ----------------                ----      ----    ------  --------
    Statements                        39        39         0   100.00%

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

Statement Coverage for instance /wrapper_top/DUT_WRAPPER/SLAVE_instance --
```

code_cvrrrrrrrr.txt - Notepad

File Edit Format View Help

```
================================================================================
== Instance: /wrapper_top/DUT_WRAPPER/RAM_instance
== Design Unit: work.RAM
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         8         8         0   100.00%

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

Branch Coverage for instance /wrapper_top/DUT_WRAPPER/RAM_instance

    Line         Item                    Count    Source
    ----         ----                    -----    ------
  File RAM.v
------------------------------------IF Branch------------------------------------
    14                                   14470    Count coming in to IF
    14             1                       298        if (~rst_n) begin

    20             1                     14172          else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    21                                   14172    Count coming in to IF
    21             1                      3211            if (rx_valid) begin

                                          10961    All False Count
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------CASE Branch----------------------------------
    22                                    3211    Count coming in to CASE
    23             1                      1166                 2'b00 : Wr_Addr <= din[7:0];

    24             1                       948                 2'b01 : MEM[Wr_Addr] <= din[7:0];

    25             1                       767                 2'b10 : Rd_Addr <= din[7:0];

    26             1                       330                 2'b11 : dout <= MEM[Rd_Addr]; //bug Rd_addr not Wr_Addr

Branch totals: 4 hits of 4 branches = 100.00%


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

Type here to search