

SPI master slave with a dual-port memory

Project report

December,2024

Abanob Evram Shahata

You can check the files in [\[Github\]](#)

~ | ~

Table of Contents

Verification plan of RAM:	4
Verification plan of Slave:	5
UVM Structure:	7
UVM FLOW:	7
Designs and assertions: -	11
RAM Design:.....	11
SPI SLAVE Design:.....	12
Wrapper Design:.....	13
Slave Golden model:	14
Wrapper Golden model:	14
RAM Assertions:.....	18
SLAVE Assertions:	19
Wrapper Assertions:	19
RAM Shared_pkg:.....	20
SLAVE&Wrapper Shared_pkg:.....	21
RAM Interface:.....	21
SLAVE Interface:.....	21
Wrapper Interface:.....	21
Objects: -	22
RAM Configuration:.....	22
SLAVE Configuration:.....	22
Wrapper Configuration:.....	22
RAM Sequence item:.....	22
SLAVE Sequence item:.....	24
Wrapper Sequence item:.....	24
RAM Sequences:	25
SLAVE Sequences:	27
Wrapper Sequences:	28
RAM Environment: -	29
RAM Driver:	29
RAM Monitor:.....	29
RAM Sequencer:	29
RAM Agent:	30
RAM Scoreboard:.....	30
RAM Coverage Collector:.....	31
RAM Environment:	32

SLAVE Environment: -	33
SLAVE Driver:.....	33
SLAVE Monitor:.....	33
SLAVE Sequencer:.....	34
SLAVE Agent:	34
SLAVE Scoreboard:.....	34
SLAVE Coverage Collector:.....	34
SLAVE Environment:	36
Wrapper Environment: -	36
Wrapper Driver:.....	37
Wrapper Monitor:.....	37
Wrapper Sequencer:	37
Wrapper Agent:	38
Wrapper Scoreboard:.....	38
Wrapper Coverage Collector:.....	39
Wrapper Environment:	40
Test & Top: -	40
Test file:.....	40
Top file:	42
For running the test files: -	43
Do file to run the RAM:	43
Do file to run the SLAVE:.....	43
Do file to run the Wrapper:.....	43
List of RAM:	44
List of SLAVE:.....	44
List of Wrapper:.....	44
Reports: -	45
Questasim snippets:.....	45
Report snippets:	46
Sequential Domain Coverage report:.....	46
Coverage groups:.....	47
Codecoverage:	48

Verification plan of RAM:

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
rst_assert	When the reset is asserted, the output will be low	Directed at the start of the simulation	Cover possible values of rst	A checker in the testbench to make sure the output is correct. Immdiate assertion to check Reset(ASY) functionality.
write_only	This seq for write address and write data only. In this seq we will write 10000 iteration and will write in all locations	repeat block to test randomized value	Cover the number of wr_add and wr_data occurs	A checker in the testbench to make sure the output is correct. Conncurrent assertions to check the wr_addr&wr_data
read_only	This seq for read address and read data only. In this seq we will read 10000 iteration and will read in all locations	repeat block to test randomized value	Cover the number of rd_add and rd_data occurs	A checker in the testbench to make sure the output is correct. Conncurrent assertions to check the rd_addr&rd_data
write_read	Randomize the 4 operations 10000 iteration	repeat block to test randomized value	Cover all operation , din and rx_valid	A checker in the testbench to make sure the output is correct. Conncurrent assertions to check all output
rst_prob	When the reset is asserted, the output will be low	Randomize during simulation to be inactive most of the time	Cover possible values of rst	A checker in the testbench to make sure the output is correct. Immdiate assertion to check Reset(ASY) functionality.
rx_valid_prob	The signal rx_valid when be high the ram will be check the operation and do it	Randomize during simulation to be on 95 of the time	Cover the two values of rx_valid	A checker in the testbench to make sure the output is correct. Conncurrent assertions to check all output

din_prob	This signal is 10 bit the latest to bit detect the operation and the other 8 bits for data	Randomize under constraint which handle all operations and all cases and handel the read address befor read data	Cover all operation , din and rx_valid	A checker in the testbench to make sure the output is correct. Concurrent assertions to check all output
-----------------	--	--	--	--

Verification plan of Slave:

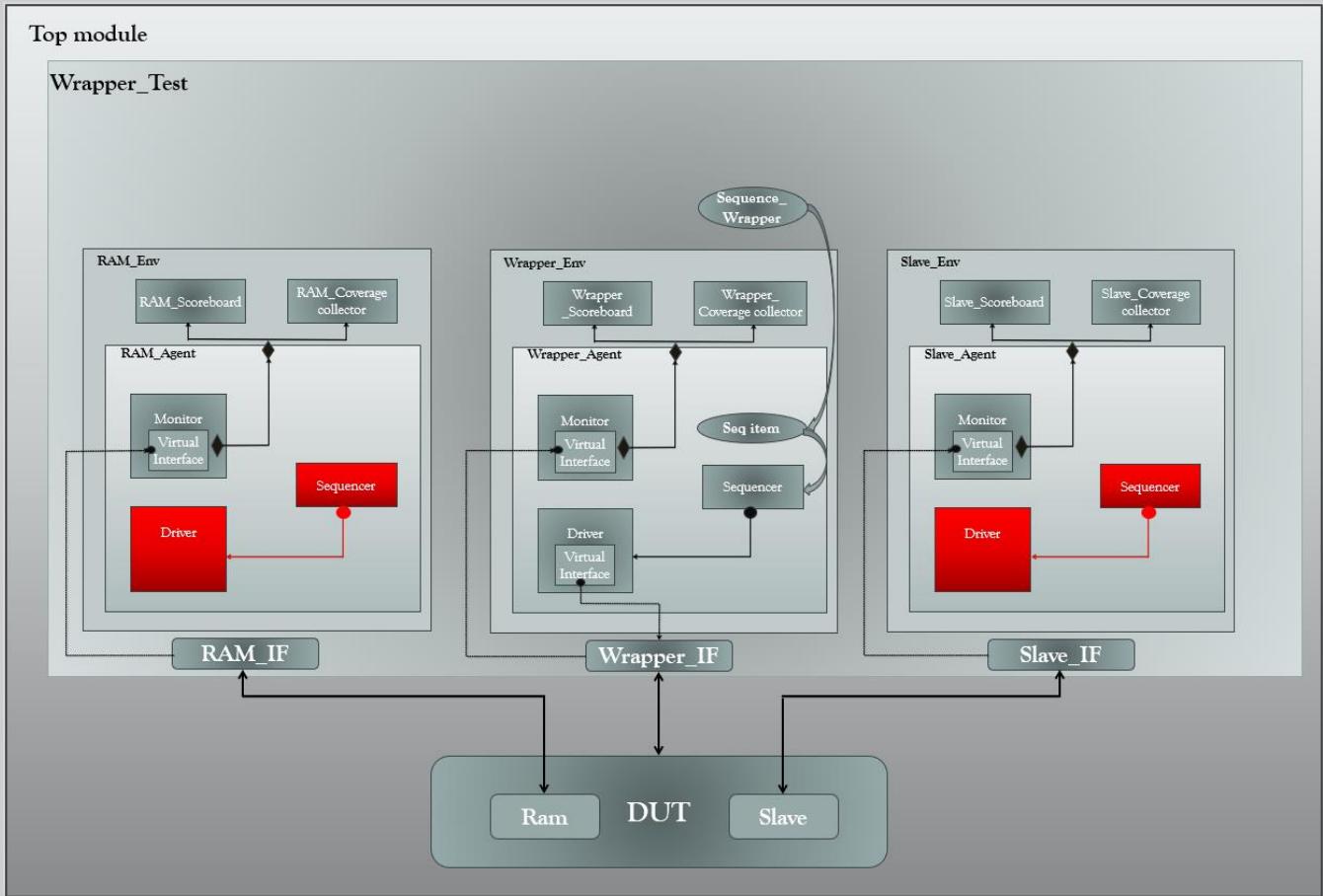
Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
rst_assert	When the reset is asserted, the output will be low	Directed at the start of the simulation	Cover possible values of rst	A checker in the testbench to make sure the output is correct. Immdiate assertion to check Reset(ASY) functionality.
main_seq	This seq will test all operation in FSM from start the communication when the SS_n active to the end of the operation. And will do this 20000 iteration	repeat block to test randomized value under some constrains	Cover the four current state 1.wr_add 2.wr_data 3.rd_add 4.rd_data	A checker in the testbench to make sure the output is correct. Concurrent assertions to check the Current state
rand_seq	This seq will randomize all signal with out any constrains to other cases and conditons	repeat block to test randomized value with ouut any constrains	Cover the four current state 1.wr_add 2.wr_data 3.rd_add 4.rd_data	A checker in the testbench to make sure the output is correct. Concurrent assertions to check the Current state
rst_prob	When the reset is asserted, the output will be low	Randomize during simulation to be inactive most of the time	Cover possible values of rst	A checker in the testbench to make sure the output is correct. Immdiate assertion to check Reset(ASY) functionality.
SS_N_prob	This constraint will active the SS_n at the start of communication and after end the communicationaccording to the operation	Randomize under specific functionality	Cover the four current state 1.wr_add 2.wr_data 3.rd_add 4.rd_data	A checker in the testbench to make sure the output is correct. Concurrent assertions to check the Current state

tx_valid_c	This signal tell the ram when the operation is read_data that the slave finish sending all 10 bits and the ram will send the data from memory	Randomize to be active after sending the dummy data and before the RAM finishes sending the data	-	A checker in the testbench to make sure the output is correct. Concurrent assertions to check the Current state
MOSI_arr_prop	This array will be used to randomize all MOSI bits, which will be used in the FSM after the end of every communication of operation.	Randomize under constraint which handle all operations and all cases and handel the read address befor read data	-	A checker in the testbench to make sure the output is correct. Concurrent assertions to check the Current state
MOSI_prop	This constraint will reflect the current bit of MOSI_arr	Randomize values from MOSI_Arr	-	A checker in the testbench to make sure the output is correct. Concurrent assertions to check the Current state

For wrapper:

It is the same plane as the slave, but the difference is that there is no rand_sequence, and we do not randomize the tx_valid signal

UVM Structure:



UVM FLOW:

❖ Initial Testing

- Started by creating standalone tests for the RAM and Slave components.

❖ Wrapper Flow Creation

1. Create Two Shared Packages:

- Developed two shared packages:
 - One for the RAM files.
 - Another for the Wrapper and Slave files.
- These packages include shared variables and parameters used across the files.

2. Create Three Interfaces:

- Defined interfaces for the three designs to connect the DUT with the testbench components.

3. Develop Assertion Files:

- Created three assertion files to verify the designs.
- Bind these assertion files to the top-level module.

4. Create Top File:

- Generated clocks and initialized memory with default values.
- Passed data from the wrapper interface to the DUT (Main Design).
- Routed data from the Slave and Wrapper interfaces to their respective golden models.
- Bind the three assertion files to the design.
- When the Slave and RAM were passive, they sent data from their interfaces to the DUT.
- Registered the interfaces in the configuration database.
- Ran tests on the Wrapper.

3. Configuration Classes

- Created configuration classes for the three designs to retrieve their virtual interfaces.

4. Sequence Item Classes

- Designed sequence item classes with the following features:
 - Included design signals and random input signals.
 - For passive designs, skipped randomization and directly transferred values from the monitor to the scoreboard and coverage collector.
 - Implemented the convert2string function to print transactions.
 - Added constraints for all three files to control randomization.

5. Wrapper Sequences

- Used two sequences to verify the Wrapper design:
 1. **Reset Sequence:** Directed reset sequence at the beginning.
 2. **Main Sequence:** Generated 50,000 constrained random iterations for testing.

6. Environment Components

I. Sequencer:

- Managed sequences and drive data.
- Not required for Slave and RAM, as they are passive designs.

2. Driver File:

- Established a virtual interface between the driver and the actual interface.
- Made the driver a producer, retrieving data and assigning it from the sequence item to interface inputs.
- Not used for the RAM and Slave.

3. Monitor:

- Set up virtual interfaces for the three designs and the real interfaces.
- Captured data from the virtual interface and passed it to the monitor object using the sequence item's inputs and outputs.
- Sent data to the agent via analysis_port.

4. Agent Files:

- Retrieved configuration objects from the database and passed them to the monitor and driver.
- Linked the monitor to the agent to send data to the scoreboard and coverage collector.
- Connected the sequencer and driver.

5. Scoreboard Files:

- Compared monitor inputs against reference models:
 - Used a task-based reference model for RAM.
 - Used golden models for the Wrapper and Slave.
- Logged mismatches and test failures.
- Generated a report for each design at the end of the simulation, detailing the correct iteration count and errors.

6. Coverage Collector Files:

- Monitored states and inputs for all files.
- Ensured thorough coverage of signals and transitions.
- Measured test completeness.

7. Environments

- Created three environments (one for each design):
 - Instantiated components like agents, scoreboards, and coverage collectors.
 - Connected the agent to the scoreboard and coverage collector.

8. Test File

- Retrieved virtual interfaces and passed them to the environment.
- Created objects for the sequences.
- Determined whether each design was active or passive.
- Signaled the start of the test using raise_objection.
- Executed the sequences during the run phase.
- Signaled test completion using drop_objection.

Designs and assertions: -

RAM Design:

```
1 //////////////////////////////////////////////////////////////////
2 // Author: Abanob Evram
3 // Description: RAM_Design
4 // Date: December, 2024
5 //////////////////////////////////////////////////////////////////
6 import RAM_shared_pkg::*;
7 module RAM(din,clk,rst_n,rx_valid,dout,tx_valid);
8   input [ADDR_SIZE+1:0] din;
9   input clk,rst_n,rx_valid;
10  output reg [ADDR_SIZE-1:0] dout;
11  output reg tx_valid;
12  reg [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];
13  reg [ADDR_SIZE-1:0] wr_addr,rd_addr;
14
15  always @ (posedge clk or negedge rst_n) begin
16    if (~rst_n) begin
17      tx_valid<=0;
18      dout<=0;
19      wr_addr<=0;
20      rd_addr<=0;
21    end
22
23    else begin
24      if(rx_valid) begin
25        tx_valid<=((din[9]==1)&(din[8]==1)&(rx_valid==1))?1:0;
26        case(din[9:8])
27          2'b00:wr_addr<=din[ADDR_SIZE-1:0];
28          2'b01:mem[wr_addr]<=din[ADDR_SIZE-1:0];
29          2'b10:rd_addr<=din[ADDR_SIZE-1:0];
30          2'b11:dout<=mem[rd_addr];
31        endcase
32      end
33    end
34  end
35 endmodule
```

SPI SLAVE Design:

```
1 //////////////////////////////////////////////////////////////////
2 // Author: Abanob Evram
3 // Description: SLAVE_Design
4 // Date: December, 2024
5 //////////////////////////////////////////////////////////////////
6 import wrapper_shared_pkg::*;
7 module SLAVE(MOSI, SS_n, clk, rst_n, tx_valid,MISO,rx_valid,rx_data,tx_data);
8
9   input MOSI, SS_n, clk, rst_n, tx_valid;
10  input [ADDR_SIZE-1:0] tx_data;
11  output reg [ADDR_SIZE+1:0] rx_data;
12  output reg MISO;
13  output rx_valid;
14
15 (* fsm_encoding = "one_hot" *)
16
17 STATE_e cs,ns;
18 reg rd_add_signal;
19
20 integer count_rx,count_tx;
21
22 //MEM stage
23 always @(posedge clk or negedge rst_n) begin
24   if (~rst_n)
25     cs<=IDLE;
26   else
27     cs<=ns;
28 end
29
30 //Next stage logic
31 always @(*) begin
32   case(cs)
33     IDLE: begin
34       if(~SS_n)
35         ns=CHK_CMD;
36       else
37         ns=IDLE;
38     end
39     CHK_CMD: begin
40       if(SS_n==0&&MOSI==0)
41         ns = WRITE;
42       else if(SS_n==0&&MOSI==1&&rd_add_signal==0)
43         ns = READ_ADD;
44       else if(SS_n==0&&MOSI==1&&rd_add_signal==1)
45         ns = READ_DATA;
46       else
47         ns = IDLE;
48     end
49     WRITE: begin
50       if(~SS_n)
51         ns=WRITE;
52       else
53         ns=IDLE;
54     end
55     READ_ADD: begin
56       if(~SS_n)
57         ns=READ_ADD;
58       else
59         ns=IDLE;
60     end
61     READ_DATA: begin
62       if(~SS_n)
63         ns=READ_DATA;
64       else
65         ns=IDLE;
66     end
67   endcase
68 end
69
```

```

69 //output logic
70 always @(posedge clk or negedge rst_n) begin
71   if (~rst_n) begin
72     rd_add_signal<=0;
73     rx_data<=0;
74     count_tx<=0;
75     count_rx<=0;
76   end
77   else begin
78     case(cs)
79       IDLE: begin
80         MISO<=0;
81         count_tx<=0;
82         count_rx<=0;
83       end
84       CHK_CMD: begin
85         count_tx<=0;
86         count_rx<=0;
87       end
88       WRITE: begin
89         if(count_rx<=9) begin
90           rx_data[9-count_rx]<=MOSI;
91           count_rx<=count_rx+1;
92         end
93       end
94       READ_ADD: begin
95         if(count_rx<=9) begin
96           rx_data[9-count_rx]<= MOSI;
97           count_rx<=count_rx+1;
98           rd_add_signal<=1;
99         end
100      end
101    endcase
102  end
103  assign rx_valid =((cs==WRITE||cs==READ_ADD||cs==READ_DATA)&&count_rx>9)?1:0 ;
104 endmodule

```

Wrapper Design:

```

1 //////////////////////////////////////////////////////////////////
2 // Author: Abanob Evram
3 // Description: Wrapper_Design
4 // Date: December, 2024
5 //////////////////////////////////////////////////////////////////
6 module Maindesign(MOSI,SS_n,clk,rst_n,MISO);
7   input MOSI,SS_n,clk,rst_n;
8   output MISO;
9   wire [9:0] rx_data;
10  wire [7:0] tx_data;
11  wire rx_valid, tx_valid;
12
13  SLAVE  spi(MOSI, SS_n, clk, rst_n, tx_valid,MISO,rx_valid,rx_data,tx_data);
14
15  RAM    ram(rx_data,clk,rst_n,rx_valid,tx_data,tx_valid);
16
17 endmodule

```

Slave Golden model:

```
6 import wrapper_shared_pkg::*;
7 module SPI_SLAVE (MOSI,MISO,SS_n,rx_data,rx_valid,tx_data,tx_valid,clk,rst_n );
8   input MOSI,SS_n,tx_valid,clk,rst_n;
9   input [7:0] tx_data;
10  output rx_valid;
11  output reg MISO;
12  output reg [9:0] rx_data;
13  reg internal_signal;
14
15 STATE_e cs,ns;
16 reg [4:0]counter1,counter2;
17 wire done_count ;
18 assign done_count =(counter1>=10)? 1:0;
19
20 always @(posedge clk or negedge rst_n) begin
21   if (~rst_n)
22     cs <= IDLE;
23   else
24     cs <= ns;
25 end
26
27 always @(*) begin
28   case (cs)
29     IDLE: begin
30       if (SS_n )
31         ns = IDLE;
32       else
33         ns = CHK_CMD;
34     end
35     CHK_CMD: begin
36       if (SS_n == 0 && MOSI == 0)
37         ns = WRITE;
38       else if (SS_n == 0 && MOSI == 1 && internal_signal == 0)
39         ns = READ_ADD;
40       else if (SS_n == 0 && MOSI == 1 && internal_signal == 1)
41         ns = READ_DATA;
42       else
43         ns = IDLE;
44     end
45     WRITE: begin
46       if (SS_n )
47         ns = IDLE;
48       else
49         ns = WRITE;
50     end
51     READ_ADD: begin
52       if (SS_n )
53         ns = IDLE;
54       else
55         ns = READ_ADD;
56     end
57     end
58     READ_DATA: begin
59       if (SS_n )
60         ns = IDLE;
61       else
62         ns = READ_DATA;
63     end
64     default: ns = IDLE;
65   endcase
66 end
67 always @(posedge clk or negedge rst_n) begin
68   if (~rst_n) begin
69     counter1 <= 0;
70     rx_data<=0;
71     internal_signal <= 0;
72     counter2<=0;
73   end
```

```

73     else begin
74         case (cs)
75             IDLE: begin
76                 counter1 <= 0;
77                 counter2<=0;
78                 MISO <= 0;
79             end
80             CHK_CMD: begin
81                 counter1<=0;
82                 counter2<=0;
83             end
84             WRITE: begin
85                 if (counter1<=9) begin
86                     rx_data[9-counter1] <= MOSI;
87                     counter1 <= counter1 + 1;
88                 end
89             end
90             READ_ADD: begin
91                 if (counter1<=9) begin
92                     internal_signal <= 1;
93                     rx_data[9-counter1] <= MOSI;
94                     counter1 <= counter1 + 1;
95                 end
96             end
97             READ_DATA : begin
98                 if (counter1>9)begin
99                     if (tx_valid==1&&counter2<8) begin
100                         MISO <= tx_data[7-counter2];
101                         counter2<=counter2+1;
102                     end
103                     else if (tx_valid==1&&counter2==8)
104                         internal_signal <= 0;
105                 end
106                 else begin
107                     rx_data[9-counter1] <= MOSI;
108                     counter1<=counter1+1;
109                 end
110             end
111             default: begin
112                 counter1<=0;
113                 internal_signal<=0;
114             end
115         endcase
116     end
117     assign rx_valid =((cs==WRITE||cs==READ_ADD||cs==READ_DATA)&&done_count)?1:0;
118 endmodule
119

```

Wrapper Golden model:

```

6 import wrapper_shared_pkg::*;
7 module wrapper_golden(MOSI,SS_n,clk,rst_n,MISO);
8
9     input MOSI, SS_n, clk, rst_n;
10    reg tx_valid;
11    reg [ADDR_SIZE-1:0] tx_data;
12    reg [ADDR_SIZE+1:0] rx_data;
13    output reg MISO;
14    wire rx_valid;
15    reg [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];
16    reg [ADDR_SIZE-1:0] wr_addr,rd_addr;
17
18    STATE_e cs,ns;
19    reg rd_add_signal;
20
21    integer count_rx,count_tx;
22
23    //MEM stage
24    always @ (posedge clk or negedge rst_n) begin
25        if (~rst_n)
26            cs<=IDLE;
27        else
28            cs<=ns;
29    end
30

```

```

31 //Next stage logic
32 always @(*) begin
33   case(cs)
34     IDLE: begin
35       if(~SS_n)
36         ns=CHK_CMD;
37       else
38         ns=IDLE;
39     end
40     CHK_CMD: begin
41       if(SS_n==0&&MOSI==0)
42         ns = WRITE;
43       else if(SS_n==0&&MOSI==1&&rd_add_signal==0)
44         ns = READ_ADD;
45       else if(SS_n==0&&MOSI==1&&rd_add_signal==1)
46         ns = READ_DATA;
47       else
48         ns = IDLE;
49     end
50     WRITE: begin
51       if(~SS_n)
52         ns=WRITE;
53       else
54         ns=IDLE;
55     end
56     READ_ADD: begin
57       if(~SS_n)
58         ns=READ_ADD;
59       else
60         ns=IDLE;
61     end
62     READ_DATA: begin
63       if(~SS_n)
64         ns=READ_DATA;
65       else
66         ns=IDLE;
67     end
68   endcase
69 end
70
71 //output logic
72 always @(posedge clk or negedge rst_n) begin
73   if (~rst_n) begin
74     rd_add_signal<=0;
75     count_tx<=0;
76     count_rx<=0;
77   end
78   else begin
79     case(cs)
80       IDLE: begin
81         MISO<=0;
82         count_tx<=0;
83         count_rx<=0;
84       end
85     end

```

```

85      CHK_CMD: begin
86          count_tx<=0;
87          count_rx<=0;
88      end
89      WRITE: begin
90          if(count_rx<=9) begin
91              rx_data[9-count_rx]<=MOSI;
92              count_rx<=count_rx+1;
93          end
94      end
95      READ_ADD: begin
96          if(count_rx<=9) begin
97              rx_data[9-count_rx]<= MOSI;
98              count_rx<=count_rx+1;
99              rd_add_signal<=1;
100         end
101     end
102     READ_DATA: begin
103         if(count_rx>9) begin
104             if(tx_valid==1&&count_tx<8) begin
105                 MISO <= tx_data[7-count_tx];
106                 count_tx<=count_tx+1;
107             end
108             else if(tx_valid==0&&count_tx>=8)begin
109                 rd_add_signal <= 0;
110             end
111         end
112         else begin
113             rx_data[9-count_rx]<= MOSI;
114             count_rx<=count_rx+1;
115         end
116     end
117 endcase
118 end
119 assign rx_valid =((cs==WRITE)||cs==READ_ADD||cs==READ_DATA)&&count_rx>9)?1:0 ;
120 //*****
121 
```

```

122 //RAM
123 always @(posedge clk or negedge rst_n) begin
124     if (~rst_n) begin
125         tx_valid<=0;
126         wr_addr<=0;
127         rd_addr<=0;
128     end
129
130     else begin
131         tx_valid<=(rx_data[9]&rx_data[8]&rx_valid)?1:0;
132         if(rx_valid) begin
133             case(rx_data[9:8])
134                 2'b00:wr_addr<=rx_data[ADDR_SIZE-1:0];
135                 2'b01:mem[wr_addr]<=rx_data[ADDR_SIZE-1:0];
136                 2'b10:rd_addr<=rx_data[ADDR_SIZE-1:0];
137                 2'b11:tx_data<=mem[rd_addr];
138             endcase
139         end
140     end
141 end
142 endmodule

```

RAM Assertions:

```
6 import RAM_shared_pkg::*;
7 module RAM_sva (din,clk,rst_n,rx_valid,dout,tx_valid,wr_addr,rd_addr,mem);
8   input [ADDR_SIZE+1:0] din;
9   input clk,rst_n,rx_valid;
10  input [ADDR_SIZE-1:0] dout;
11  input tx_valid;
12  input [ADDR_SIZE-1:0] wr_addr,rd_addr;
13  input [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];
14
15 //Always comp to test asynch rst_n
16 always_comb begin
17   if(rst_n==ACTIVE_RESET) begin
18     rst_wr_addr_assert : assert final (wr_addr == INACTIVE);
19     rst_rd_addr_assert : assert final (rd_addr == INACTIVE);
20     rst_tx_valid_assert: assert final (tx_valid == INACTIVE);
21     rst_wr_addr_cover : cover final (wr_addr == INACTIVE);
22     rst_rd_addr_cover : cover final (rd_addr == INACTIVE);
23     rst_tx_valid_cover : cover final (tx_valid == INACTIVE);
24   end
25 end
26
27 property wr_addr_prop;
28   @ (posedge clk) disable iff (!rst_n) (rx_valid && din[ADDR_SIZE+1:ADDR_SIZE] == WRITE_ADD) |=> (wr_addr == $past(din[ADDR_SIZE-1:0]));
29 endproperty
30
31 property wr_data_prop;
32   @ (posedge clk) disable iff (!rst_n) (rx_valid && din[ADDR_SIZE+ 1:ADDR_SIZE] == WRITE_DATA) |=> (mem[wr_addr] == $past(mem[wr_addr]));
33 endproperty
34
35 property rd_addr_prop;
36   @ (posedge clk) disable iff (!rst_n) (rx_valid && din[ADDR_SIZE+ 1:ADDR_SIZE] == READ_ADD) |=> (rd_addr == $past(din[ADDR_SIZE-1:0]));
37 endproperty
38
39 property rd_data_prop;
40   @ (posedge clk) disable iff (!rst_n) (rx_valid && din[ADDR_SIZE+ 1:ADDR_SIZE] == READ_DATA) |=> (dout == $past(mem[rd_addr]));
41 endproperty
42
43 property tx_valid_prop;
44   @ (posedge clk) disable iff (!rst_n) (rx_valid && din[ADDR_SIZE+ 1:ADDR_SIZE] == READ_DATA) |=> (tx_valid == 1);
45 endproperty
46
47 /*************************************************************************/
48 wr_addr_assert :assert property(wr_addr_prop);
49 wr_data_assert :assert property(wr_data_prop);
50 rd_addr_assert :assert property(rd_addr_prop);
51 rd_data_assert :assert property(rd_data_prop);
52 tx_valid_assert:assert property(tx_valid_prop);
53 /*************************************************************************/
54 wr_addr_cover :cover property(wr_addr_prop);
55 wr_data_cover :cover property(wr_data_prop);
56 rd_addr_cover :cover property(rd_addr_prop);
57 rd_data_cover :cover property(rd_data_prop);
58 tx_valid_cover:cover property(tx_valid_prop);
59 endmodule
```

SLAVE Assertions:

```

6 import wrapper_shared_pkg::*;
7 module slave_sva (MOSI, SS_n, clk, rst_n, tx_valid,MISO,rx_valid,rx_data,tx_data,cs,rd_add_signal,count_rx,count_tx);
8
9     input MOSI, SS_n, clk, rst_n, tx_valid;
10    input [ADDR_SIZE+1:0] tx_data;
11    input [ADDR_SIZE+1:0] rx_data;
12    input MISO;
13    input rx_valid;
14    input STATE_e cs;
15    input rd_add_signal;
16    input integer count_rx,count_tx;
17
18    always_comb begin
19        if (rst_n==ACTIVE_RESET) begin
20            rst_count_rx_assert      : assert final (count_rx == 0);
21            rst_count_tx_assert      : assert final (count_tx == 0);
22            rst_rd_add_signal_assert : assert final (rd_add_signal == INACTIVE);
23            rst_cs_assert            : assert final (cs == IDLE);
24            rst_count_rx_cover       : cover final (count_rx == 0);
25            rst_count_tx_cover       : cover final (count_tx == 0);
26            rst_rd_add_signal_cover  : cover final (rd_add_signal == INACTIVE);
27            rst_cs_cover             : cover final (cs == IDLE);
28        end
29    end
30 /*This assertinos for the cs*/
31 property idel_prop;
32     @ (posedge clk) disable iff (!rst_n) SS_n |=> cs == IDLE;
33 endproperty
34
35 property check_prop;
36     @ (posedge clk) disable iff (!rst_n) (!SS_n&&cs==IDLE) |=> cs == CHK_CMD;
37 endproperty
38
39 property write_prop;
40     @ (posedge clk) disable iff (!rst_n) (!SS_n&&cs==CHK_CMD&&!MOSI) |=> cs == WRITE;
41 endproperty
42
43 property read_add_prop;
44     @ (posedge clk) disable iff (!rst_n) (!SS_n&&cs==CHK_CMD&&MOSI&&!rd_add_signal) |=> cs == READ_ADD;
45 endproperty
46
47 property read_data_prop;
48     @ (posedge clk) disable iff (!rst_n) (!SS_n&&cs==CHK_CMD&&MOSI&&rd_add_signal) |=> cs == READ_DATA;
49 endproperty
50 /*This assertions for the rd_add_signal*/
51 property rd_flag_ra_pr;
52     @ (posedge clk) disable iff (!rst_n) ((cs == READ_ADD) && (count_rx == 10)) |=> (rd_add_signal);
53 endproperty
54
55 property rd_flag_rd_pr;
56     @ (posedge clk) disable iff (!rst_n) ((cs == READ_DATA) && (count_tx == 8)) |=> (!rd_add_signal);
57 endproperty
58
59 //*****
60 //*****
61 idel_assert      :assert property(idel_prop);
62 check_assert     :assert property(check_prop);
63 write_assert     :assert property(write_prop);
64 read_add_assert :assert property(read_add_prop);
65 read_data_assert :assert property(read_data_prop);
66 rd_flag_ra_assert :assert property(rd_flag_ra_pr);
67 rd_flag_rd_assert :assert property(rd_flag_rd_pr);
68
69 idel_cover      :cover property(idel_prop);
70 check_cover     :cover property(check_prop);
71 write_cover     :cover property(write_prop);
72 read_add_cover  :cover property(read_add_prop);
73 read_data_cover :cover property(read_data_prop);
74 rd_flag_ra_cover :cover property(rd_flag_ra_pr);
75 rd_flag_rd_cover :cover property(rd_flag_rd_pr);
76
77 endmodule

```

Wrapper Assertions:

```
6  import wrapper_shared_pkg::*;
7  module Wrapper_sva (MOSI,SS_n,rst_n,MISO,rx_data,tx_data,rx_valid,tx_valid,clk);
8    input MOSI,SS_n,rst_n,clk;
9    input MISO;
10   input [9:0] rx_data;
11   input [7:0] tx_data;
12   input rx_valid, tx_valid;
13
14  //Always comp to test asynch rst_n
15  always_comb begin
16    if(rst_n==ACTIVE_RESET) begin
17      rst_tx_valid_assert : assert final (tx_valid == INACTIVE);
18      rst_rst_rx_valid_assert : assert final (rx_valid == INACTIVE);
19      rst_rst_tx_data_assert : assert final (tx_data == INACTIVE);
20      rst_rst_rx_data_assert : assert final (rx_data == INACTIVE);
21      rst_tx_valid_cover : cover final (tx_valid == INACTIVE);
22      rst_rst_rx_valid_cover : cover final (rx_valid == INACTIVE);
23      rst_rst_tx_data_cover : cover final (tx_data == INACTIVE);
24      rst_rst_rx_data_cover : cover final (rx_data == INACTIVE);
25    end
26  end
27
28
29
30
31  property rx_valid_prop;
32    @ (posedge clk) disable iff (!rst_n) (SS_n ##1 (!SS_n) [*12] |=> (rx_valid));
33  endproperty
34
35  property tx_valid_prop;
36    @ (posedge clk) disable iff (!rst_n) (SS_n ##1 (!SS_n) [*12] ##1 ((!SS_n) &&
37    ($past(MOSI, 10), $past(MOSI, 9)) == 2'b11)) |=> (tx_valid);
38  endproperty
39
40  property rx_data_prop;
41    @ (posedge clk) disable iff (!rst_n) (SS_n ##1 (!SS_n) [*12] |=> (rx_data ==
42    {$past(MOSI, 10), $past(MOSI, 9), $past(MOSI, 8), $past(MOSI, 7),
43    $past(MOSI, 6), $past(MOSI, 5), $past(MOSI, 4), $past(MOSI, 3),
44    $past(MOSI, 2), $past(MOSI)}));
45  endproperty
46
47  /*************************************************************************/
48  rx_valid_assert : assert property(rx_valid_prop);
49  tx_valid_assert : assert property(tx_valid_prop);
50  rx_data_assert : assert property(rx_data_prop);
51  /*************************************************************************/
52  rx_valid_cover : cover property(rx_valid_prop);
53  tx_valid_cover : cover property(tx_valid_prop);
54  rx_data_cover : cover property(rx_data_prop);
55 endmodule
```

RAM Shared_pkg:

```
6  package RAM_shared_pkg;
7    parameter MEM_DEPTH = 256 ;
8    parameter ADDR_SIZE = $cLog2(MEM_DEPTH) ;
9
10   parameter WRITE_ADD=0;
11   parameter WRITE_DATA=1;
12   parameter READ_ADD=2;
13   parameter READ_DATA=3;
14
15   parameter RX_VALID_ON_DIST=90;
16   parameter RESET_ON_DIST=5;
17
18   parameter ACTIVE=1;
19   parameter INACTIVE=0;
20
21   parameter ACTIVE_RESET=0;
22   parameter INACTIVE_RESET=1;
23 endpackage : RAM_shared_pkg
```

SLAVE&Wrapper Shared_pkg:

```
6  package wrapper_shared_pkg;
7    localparam MEM_DEPTH = 256;
8    localparam ADDR_SIZE = $clog2(MEM_DEPTH);
9
10   /*****
11   //For Slave & Wrapper
12   typedef enum bit [2:0] {IDLE,CHK_CMD,WRITE,READ_ADD,READ_DATA} STATE_e;
13
14   typedef enum bit [2 : 0] {IDLE_COV, WRITE_ADD_COV, WRITE_DATA_COV, READ_ADD_COV, READ_DATA_COV} STATE_COV_e;
15   STATE_COV_e cs_cov;
16
17   parameter RX_VALID_ON_DIST=90;
18   parameter RESET_ON_DIST=1;
19
20   parameter ACTIVE=1;
21   parameter INACTIVE=0;
22
23   parameter ACTIVE_RESET=0;
24   parameter INACTIVE_RESET=1;
25 endpackage
```

RAM Interface:

```
6  import RAM_shared_pkg::*;
7  interface RAM_if (clk);
8    logic [ADDR_SIZE+1:0] din;
9    logic rst_n,rx_valid;
10   logic [ADDR_SIZE-1:0] dout;
11   logic tx_valid;
12   input clk;
13 endinterface : RAM_if
```

SLAVE Interface:

```
6  import wrapper_shared_pkg::*;
7  interface slave_if(clk);
8    //Input signals
9    logic MOSI, SS_n, rst_n, tx_valid;
10   logic [ADDR_SIZE-1:0] tx_data;
11   input clk;
12   //Output signals
13   logic [ADDR_SIZE+1:0] rx_data,rx_data_ref;
14   logic MISO,MISO_ref;
15   logic rx_valid,rx_valid_ref;
16
17 endinterface : slave_if
```

Wrapper Interface:

```
6  import wrapper_shared_pkg::*;
7  interface wrapper_if (clk);
8    logic MOSI,SS_n,rst_n;
9    logic MISO,MISO_ref;
10   input clk;
11
12 endinterface : wrapper_if
```

Objects: -

RAM Configuration:

```

6   package RAM_config_pkg;
7     import uvm_pkg::*;
8     `include "uvm_macros.svh"
9     class RAM_config extends uvm_object;
10    `uvm_object_utils(RAM_config)
11
12    virtual RAM_if RAM_vif;
13    uvm_active_passive_enum is_active;
14
15    function new(string name = "RAM_config");
16      super.new(name);
17    endfunction : new
18
19    endclass : RAM_config
20  endpackage : RAM_config_pkg

```

SLAVE Configuration:

```

6   package slave_config_pkg;
7     import uvm_pkg::*;
8     `include "uvm_macros.svh"
9     class slave_config extends uvm_object;
10    `uvm_object_utils(slave_config)
11
12    virtual slave_if slave_vif;
13    uvm_active_passive_enum is_active;
14
15    function new(string name = "slave_config");
16      super.new(name);
17    endfunction : new
18
19    endclass : slave_config
20  endpackage : slave_config_pkg

```

Wrapper Configuration:

```

6   package wrapper_config_pkg;
7     import uvm_pkg::*;
8     `include "uvm_macros.svh"
9     class wrapper_config extends uvm_object;
10    `uvm_object_utils(wrapper_config)
11
12    virtual wrapper_if wrapper_vif;
13    uvm_active_passive_enum is_active;
14
15    function new(string name = "wrapper_config");
16      super.new(name);
17    endfunction : new
18
19    endclass : wrapper_config
20  endpackage : wrapper_config_pkg

```

RAM Sequence item:

```

6   package RAM_seq_item_pkg;
7     import RAM_shared_pkg::*;
8     import uvm_pkg::*;
9     `include "uvm_macros.svh"
10
11    class RAM_seq_item extends uvm_sequence_item;
12      `uvm_object_utils(RAM_seq_item)
13
14      //Randomized properties to generate stimulus
15      rand bit rst_n,rx_valid;
16      rand bit [ADDR_SIZE+1:0] din;
17      //output signals
18      bit [ADDR_SIZE-1:0] dout;
19      bit tx_valid;
20      bit [ADDR_SIZE-1:0] mem [MEM_DEPTH-1:0];

```

```

22      /*Write and Read address array to handle all possible addresses based on ADDR_SIZE*/
23      bit [ADDR_SIZE-1:0] wa_arr [MEM_DEPTH];
24      bit [ADDR_SIZE-1:0] ra_arr [MEM_DEPTH];
25      //Write & Read index for the array
26      bit [ADDR_SIZE-1:0] wa_idx,ra_idx;
27      //Flags for check the read and write addresses occurred
28      bit wr_addr_occur, rd_addr_occur;
29
30
31      function new(string name ="RAM_seq_item");
32          super.new(name);
33          //loop to save all possible addresses in the array
34          for (int i = 0; i < MEM_DEPTH; i++) begin
35              wa_arr [i] = i;
36              ra_arr [i] = i;
37          end
38          wa_arr.shuffle();
39          ra_arr.shuffle();
40      endfunction : new
41
42      function string convert2string();
43          return $sformatf("%s rst_n = %0b, rx_valid = %0b, din = %0h, tx_valid = %0b, dout = %0h",
44          super.convert2string(), rst_n, rx_valid, din, tx_valid, dout);
45      endfunction
46
47      /*****************************************************************************Constraints*****/
48
49      //Constraint to randomize rst_n with a higher probability of being active
50      constraint rst_prob {rst_n dist {ACTIVE_RESET:/RESET_ON_DIST,INACTIVE_RESET:/100 RESET_ON_DIST};}
51
52      // Constraint to randomize rx_valid with a high probability of being valid
53      constraint rx_valid_prob {rx_valid dist {ACTIVE:/RX_VALID_ON_DIST,INACTIVE:/100 RX_VALID_ON_DIST};}
54
55      // Constraints on din to ensure valid address mapping
56      constraint din_prob {
57          din [ADDR_SIZE + 1] dist {ACTIVE:/40,INACTIVE:/ 60}; // Randomize high bit for read/write operation
58          din [ADDR_SIZE]      dist {ACTIVE:/40,INACTIVE:/ 60}; // Randomize low bit for operation validity
59
60          //Ensure proper handling of write and read operations based on flags
61          if (din[ADDR_SIZE + 1] == 0) {
62              if (!wr_addr_occur) //if the write address not occur the bit 8 will be 0
63                  din[ADDR_SIZE] == 0;
64          }
65          else {
66              if (!rd_addr_occur) //if the read address not occur the bit 8 will be 0
67                  din[ADDR_SIZE] == 0;
68          }
69
70          //if the operation is write or read address we will send the address from the arrays
71          if (din[ADDR_SIZE+1:ADDR_SIZE]==WRITE_ADD)
72              din[ADDR_SIZE-1:0] == wa_arr[wa_idx];
73          else if (din[ADDR_SIZE+1:ADDR_SIZE]==READ_ADD)
74              din[ADDR_SIZE-1:0] == ra_arr[ra_idx];
75      }
76
77
78      //Function to handle updates and shuffling post-randomization
79      function void post_randomize();
80          if(wa_idx==MEM_DEPTH-1) //Check if all write addresses have been use
81              wa_arr.shuffle();
82          else if(din[ADDR_SIZE+1:ADDR_SIZE]==WRITE_ADD)
83              wa_idx++; //Increment if the operation is write address
84
85          if(ra_idx==MEM_DEPTH-1) //Check if all read addresses have been use
86              ra_arr.shuffle();
87          else if(din[ADDR_SIZE+1:ADDR_SIZE]==READ_ADD)
88              ra_idx++; //Increment if the operation is read address
89
90          //update the flags
91          if (din[ADDR_SIZE + 1 : ADDR_SIZE] == WRITE_ADD)
92              wr_addr_occur = 1;
93          else if (din[ADDR_SIZE + 1 : ADDR_SIZE] == READ_ADD)
94              rd_addr_occur = 1;
95      endfunction : post_randomize
96
97
98
99
100     endclass : RAM_seq_item
101    endpackage : RAM_seq_item_pkg

```

SLAVE Sequence item:

```

6   package slave_seq_item_pkg;
7     import wrapper_shared_pkg::*;
8     import uvm_pkg::*;
9     `include "uvm_macros.svh"
10
11    class slave_seq_item extends uvm_sequence_item;
12      `uvm_object_utils(slave_seq_item)
13
14      //Randomized properties to generate stimulus
15      rand bit MOSI, SS_n, clk, rst_n, tx_valid;
16      rand bit [ADDR_SIZE-1:0] tx_data;
17      //output signals
18      logic [ADDR_SIZE+1:0] rx_data,rx_data_ref;
19      logic MISO,rx_valid,MISO_ref,rx_valid_ref;
20
21      //Array to hold MOSI data ,The size is 21 bit for the longest way (READ_DATA) [2+2+8 dummy+8]
22      rand bit MOSI_arr [20:0];
23      //counter for the mosi_arr and the size is 5
24      bit [4:0] count;
25      //this is Read/Write flag (1 = read, 0 = write)
26      bit rd_flag;
27
28      function new(string name ="slave_seq_item");
29        super.new(name);
30      endfunction : new
31
32      function string convert2string();
33        return $format("%s rst_n = %0b, SS_n = %0b, MOSI = %0b, tx_valid = %0b, tx_data = %0h, rx_valid = %0b, rx_data = %0h, MISO = %0b",
34          super.convert2string(), rst_n, SS_n, MOSI, tx_valid, tx_data, rx_valid, rx_data, MISO);
35      endfunction
36
37  /*********************************************************************
38  ****Constraints*****
39
40  //Constraint to randomize rst_n with a higher probability of being active
41  constraint rst_prob {rst_n_distr {ACTIVE_RESET:/RESET_ON_DIST,INACTIVE_RESET:/100 RESET_ON_DIST};}
42
43  constraint SS_N_prob {
44    if(MOSI_arr[2]&&MOSI_arr[3]){
45      if(count==0) //when the state is read data and we finish reading the SS_n will equal 1 to end the communication
46        SS_n=1;
47      else
48        SS_n=0;
49    }
50    else {
51      if(count==12) //when we finish states the SS_n will equal 1 to end the communication
52        SS_n=1;
53      else
54        SS_n=0;
55    }
56  }
57
58  constraint tx_valid_c {
59    //if the state is read data and we finish send the dummy bits the tx_valid should be active
60    if((MOSI_arr[2]&&MOSI_arr[3])&&(count>11))
61      tx_valid=1;
62    else
63      tx_valid=0;
64  }
65
66  constraint MOSI_arr_prop {
67    MOSI_arr[1] dist {0:/50,1:/50};
68    MOSI_arr[2] == MOSI_arr[1];
69    if(MOSI_arr[2]){//if the bit 9 equal 1 [Read operation]
70      if(rd_flag)//if we read address
71        MOSI_arr[3]=1;//We will Read data
72      else
73        MOSI_arr[3]=0;//We will Read address
74    }
75    else//[Write operation]
76      MOSI_arr[3] dist {0:/60,1:/40};
77  }
78
79  constraint MOSI_prop {
80    MOSI=MOSI_arr[count];//MOSI reflects the current bit of MOSI_arr based on count
81  }
82
83  function void pre_randomize();
84    if(count==0)//Enable the randomization of the array at the idle state
85      MOSI_arr.rand_mode(1);
86    else
87      MOSI_arr.rand_mode(0);
88
89    if((MOSI_arr[2]&&MOSI_arr[3])&&(count==20))//Enable the randomization of the tx_data at the end of the read data
90      tx_data.rand_mode(1);
91    else
92      tx_data.rand_mode(0);
93
94    /*before the randomization we shold know the state of the rd_flag*/
95    if(rst_n==ACTIVE_RESET)
96      rd_flag=0;
97    else if(count==0) begin//At the beginning when the count equal zero we will check the rd flag
98      if(MOSI_arr[2]&&MOSI_arr[3])//READ_DATA
99        rd_flag=0;
100     else if ((MOSI_arr[2]&&!MOSI_arr[3]))//READ_ADD
101       rd_flag=1;
102    end
103
104  endfunction : pre_randomize
105

```

```

105     function void post_randomize();
106         //Update the counter after every randomization
107         if(ss_n||rst_n==ACTIVE_RESET)
108             count=0;
109         else
110             count++;
111
112
113
114     endfunction : post_randomize
115
116     endclass : slave_seq_item
117 endpackage : slave_seq_item_pkg

```

Wrapper Sequence item:

```

6   package wrapper_seq_item_pkg;
7       import wrapper_shared_pkg::*;
8       import uvm_pkg::*;
9       `include "uvm_macros.svh"
10
11      class wrapper_seq_item extends uvm_sequence_item;
12          `uvm_object_utils(wrapper_seq_item)
13
14          //Randomized properties to generate stimulus
15          rand bit MOSI, SS_n, rst_n;
16          //output signals
17          logic MISO,MISO_ref;
18
19          //Array to hold MOSI data ,The size is 21 bit for the longest way (READ_DATA) |
20          rand bit [2 * ADDR_SIZE] + 6];
21          //counter for the mosi arr and the size is 5
22          bit [ ${clog2((2 * ADDR_SIZE) + 5)} - 1 : 0] count;
23          //this is Read/Write flag (1 = read, 0 = write)
24          bit rd_flag;
25
26          function new(string name = "wrapper_seq_item");
27              super.new(name);
28          endfunction : new
29
30          function string convert2string();
31              return $sformatf("%s rst_n = %0b, SS_n = %0b, MOSI = %0b, MISO = %0b",
32                  super.convert2string(), rst_n, SS_n, MOSI, MISO);
33          endfunction
34
35
36  /*****************************************************************************Constraints*****/
37
38  //Constraint to randomize rst_n with a higher probability of being active
39  constraint rst_prob {rst_n dist {ACTIVE_RESET:RESET_ON_DIST,INACTIVE_RESET:/100-RESET_ON_DIST};}
40
41  constraint SS_N_prob {
42      if(MOSI_arr[2]&&MOSI_arr[3]){
43          if(count==(2 * ADDR_SIZE) + 5) //when the state is read data and we finish reading the SS_n will equal 1 to end the communication
44              SS_n=1;
45          else
46              SS_n=0;
47      }
48      else {
49          if(count==12) //when we finish states the SS_n will equal 1 to end the communication
50              SS_n=1;
51          else
52              SS_n=0;
53      }
54  }
55
56  constraint MOSI_arr_prop {
57      MOSI_arr[1] dist {0:/50,1:/50};
58      MOSI_arr[2] == MOSI_arr[1];
59      if(MOSI_arr[2]){//if the bit 9 equal 1 [Read operation]
60          if(rd_flag)//if we read address
61              MOSI_arr[3]==1;//We will Read date
62          else
63              MOSI_arr[3]==0;//We will Read address
64      }
65      else// [Write operation]
66          MOSI_arr[3] dist {0:/60,1:/40};
67  }
68
69  constraint MOSI_prop {
70      MOSI=MOSI_arr[count];//MOSI reflects the current bit of MOSI_arr based on count
71  }

```

```

72     function void pre_randomize();
73         if(count==0)//Enable the randomization of the array at the idle state
74             MOSI_arr.rand_mode(1);
75         else
76             MOSI_arr.rand_mode(0);
77
78
79
80         /*before the randomization we should know the state of the rd_flag*/
81         if(rst_n==ACTIVE_RESET)
82             rd_flag=0;
83         else if(count==0) begin//At the beginning when the count equal zero we will check the rd flag
84             if(MOSI_arr[2]&&MOSI_arr[3])//READ_DATA
85                 rd_flag=0;
86             else if ((MOSI_arr[2]&&!MOSI_arr[3]))//READ_ADD
87                 rd_flag=1;
88         end
89
90     endfunction : pre_randomize
91
92     function void post_randomize();
93         //Update the counter after every randomization
94         if(SS_n||rst_n==ACTIVE_RESET)
95             count=0;
96         else
97             count++;
98
99         //Update the variable cs_cov for the coverage collector after the state comes true
100        if(SS_n||rst_n==ACTIVE_RESET)
101            cs_cov=IDLE_COV;
102        else if (count==12)begin//for the three cases that need 12 clk cycle
103            if((MOSI_arr[2]&&!MOSI_arr[3])
104                cs_cov=WRITE_ADD_COV;
105            else if(!MOSI_arr[2]&&MOSI_arr[3])
106                cs_cov=WRITE_DATA_COV;
107            else if(MOSI_arr[2]&&MOSI_arr[3])
108                cs_cov=READ_ADD_COV;
109            end
110        else if(count==20)begin
111            if(MOSI_arr[2]&&MOSI_arr[3])
112                cs_cov=READ_DATA_COV;
113            end
114        end
115
116    endfunction : post_randomize
117
118    endclass : wrapper_seq_item
119  endpackage : wrapper_seq_item_pkg

```

RAM Sequences:

```

6  package RAM_seq_pkg;
7      import RAM_shared_pkg::*;
8      import RAM_seq_item_pkg::*;
9      import uvm_pkg::*;
10     `include "uvm_macros.svh"
11  /*********************************************************************
12  class RAM_rst_assert extends uvm_sequence #(RAM_seq_item);
13      `uvm_object_utils(RAM_rst_assert)
14
15      RAM_seq_item seq_item;
16
17      function new(string name = "RAM_rst_assert");
18          super.new(name);
19      endfunction : new
20
21      task pre_body;
22          seq_item = RAM_seq_item::type_id::create("seq_item");
23      endtask
24
25      task body();
26          start_item(seq_item);
27          seq_item.rst_n=ACTIVE_RESET;
28          seq_item.rx_valid=INACTIVE;
29          seq_item.din=INACTIVE;
30          finish_item(seq_item);
31      endtask : body
32  endclass : RAM_rst_assert
33  /*********************************************************************
34  class RAM_write_only extends uvm_sequence #(RAM_seq_item);
35      `uvm_object_utils(RAM_write_only)
36
37      RAM_seq_item seq_item;
38
39      function new(string name = "RAM_write_only");
40          super.new(name);
41      endfunction : new
42
43      task pre_body;
44          seq_item = RAM_seq_item::type_id::create("seq_item");
45      endtask
46
47      task body();
48          repeat(10000) begin
49              start_item(seq_item);
50              assert(seq_item.randomize() with {seq_item.din[ADDR_SIZE+1:ADDR_SIZE] inside {2'b00, 2'b01};});
51              finish_item(seq_item);
52          end
53      endtask : body
54  endclass : RAM_write_only
55  /*********************************************************************

```

```

56  /*****************************************************************/
57  class RAM_read_only extends uvm_sequence #(RAM_seq_item);
58    `uvm_object_utils(RAM_read_only)
59
60    RAM_seq_item seq_item;
61
62    function new(string name = "RAM_read_only");
63      super.new(name);
64    endfunction : new
65
66    task pre_body();
67      seq_item = RAM_seq_item::type_id::create("seq_item");
68    endtask
69
70    task body();
71      repeat(10000) begin
72        start_item(seq_item);
73        assert(seq_item.randomize() with {seq_item.din[ADDR_SIZE+1:ADDR_SIZE] inside {2'b10, 2'b11}});
74        finish_item(seq_item);
75      end
76    endtask : body
77  endclass : RAM_read_only
78 /*****************************************************************/
79 class RAM_write_read extends uvm_sequence #(RAM_seq_item);
80   `uvm_object_utils(RAM_write_read)
81
82   RAM_seq_item seq_item;
83
84   function new(string name = "RAM_write_read");
85     super.new(name);
86   endfunction : new
87
88   task pre_body();
89     seq_item = RAM_seq_item::type_id::create("seq_item");
90   endtask
91
92   task body();
93     repeat(10000) begin
94       //seq_item=RAM_seq_item::type_id::create("seq_item");
95       start_item(seq_item);
96       assert(seq_item.randomize());
97       finish_item(seq_item);
98     end
99   endtask : body
100 endclass : RAM_write_read
101 /*****************************************************************/
102 endpackage : RAM_seq_pkg

```

SLAVE Sequences:

```

6 package slave_seq_pkg;
7   import slave_seq_item_pkg::*;
8   import uvm_pkg::*;
9   `include "uvm_macros.svh"
10  /*****************************************************************/
11  class slave_rst_assert extends uvm_sequence #(slave_seq_item);
12    `uvm_object_utils(slave_rst_assert)
13
14    slave_seq_item seq_item;
15
16    function new(string name = "slave_rst_assert");
17      super.new(name);
18    endfunction : new
19
20    task pre_body();
21      seq_item = slave_seq_item::type_id::create("seq_item");
22    endtask : pre_body
23
24    task body();
25      start_item(seq_item);
26      seq_item.rst_n  = 0;
27      seq_item.SS_n   = 0;
28      seq_item.MOSI   = 0;
29      seq_item.tx_valid = 0;
30      seq_item.tx_data = 0;
31      finish_item(seq_item);
32
33    endtask : body
34  endclass : slave_rst_assert
35  /*****************************************************************/
36  class slave_main_seq extends uvm_sequence #(slave_seq_item);
37    `uvm_object_utils(slave_main_seq)
38
39    slave_seq_item seq_item;
40
41    function new(string name = "slave_main_seq");
42      super.new(name);
43    endfunction : new
44
45    task pre_body();
46      seq_item = slave_seq_item::type_id::create("seq_item");
47    endtask : pre_body
48
49    task body();
50      repeat(20000) begin
51        start_item(seq_item);
52        assert(seq_item.randomize());
53        finish_item(seq_item);
54      end
55    endtask : body
56  endclass : slave_main_seq
57  /*****************************************************************/

```

```

57  /*************************************************************************/
58  class slave_rand_seq extends uvm_sequence #(slave_seq_item);
59    `uvm_object_utils(slave_rand_seq)
60
61    slave_seq_item seq_item;
62
63    function new(string name = "slave_rand_seq");
64      super.new(name);
65    endfunction : new
66
67    task pre_body;
68      seq_item = slave_seq_item::type_id::create("seq_item");
69    endtask : pre_body
70
71    task body();
72
73      repeat(10000) begin
74        start_item(seq_item);
75        seq_item.constraint_mode(0);
76        assert(seq_item.randomize());
77        finish_item(seq_item);
78      end
79    endtask : body
80  endclass : slave_rand_seq
81
82 endpackage : slave_seq_pkg
83

```

Wrapper Sequences:

```

6 package wrapper_sequence_pkg;
7   import wrapper_seq_item_pkg::*;
8   import uvm_pkg::*;
9   include "uvm_macros.svh"
10  /*************************************************************************/
11  class wrapper_rst_assert extends uvm_sequence #(wrapper_seq_item);
12    `uvm_object_utils(wrapper_rst_assert);
13
14    wrapper_seq_item seq_item;
15
16    function new(string name = "wrapper_rst_assert");
17      super.new(name);
18    endfunction : new
19
20    task pre_body;
21      seq_item = wrapper_seq_item::type_id::create("seq_item");
22    endtask : pre_body
23
24    task body;
25      start_item(seq_item);
26      seq_item.rst_n = 0;
27      seq_item.SS_n = 0;
28      seq_item.MOSI = 0;
29      finish_item(seq_item);
30
31    endtask : body
32  endclass : wrapper_rst_assert
33
34  /*************************************************************************/
35  class wrapper_main_seq extends uvm_sequence #(wrapper_seq_item);
36    `uvm_object_utils(wrapper_main_seq)
37
38    wrapper_seq_item seq_item;
39
40    function new(string name = "wrapper_main_seq");
41      super.new(name);
42    endfunction : new
43
44    task pre_body;
45      seq_item = wrapper_seq_item::type_id::create("seq_item");
46    endtask : pre_body
47
48    task body();
49      repeat(50000) begin
50        start_item(seq_item);
51        assert(seq_item.randomize());
52        finish_item(seq_item);
53      end
54    endtask : body
55  endclass : wrapper_main_seq
56

```

RAM Environment: -

RAM Driver:

```

6 package RAM_driver_pkg;
7 import RAM_seq_item_pkg::*;
8 import uvm_pkg::*;
9 `include "uvm_macros.svh"
10
11 class RAM_driver extends uvm_driver #(RAM_seq_item);
12   `uvm_component_utils(RAM_driver);
13
14   RAM_seq_item stim_seq_item;
15   virtual RAM_if RAM_vif;
16
17   function new(string name = "RAM_driver", uvm_component parent = null);
18     super.new(name,parent);
19   endfunction : new
20
21   task run_phase(uvm_phase phase);
22     super.run_phase(phase);
23     forever begin
24       stim_seq_item = RAM_seq_item::type_id::create("stim_seq_item");
25       seq_item_port.get_next_item(stim_seq_item);
26
27       RAM_vif.rst_n = stim_seq_item.rst_n;
28       RAM_vif.rx_valid = stim_seq_item.rx_valid;
29       RAM_vif.din = stim_seq_item.din;
30       @(posedge RAM_vif.clk);
31
32       seq_item_port.item_done();
33       `uvm_info("run_phase", stim_seq_item.convert2string(), UVM_HIGH)
34
35     end
36   endtask : run_phase
37
38 endclass : RAM_driver
39
endpackage : RAM_driver_pkg

```

RAM Monitor:

```

6 package RAM_monitor_pkg;
7 import RAM_seq_item_pkg::*;
8 import uvm_pkg::*;
9 `include "uvm_macros.svh"
10 class RAM_monitor extends uvm_monitor;
11   `uvm_component_utils(RAM_monitor);
12
13   RAM_seq_item seq_item;
14   virtual RAM_if RAM_vif;
15   uvm_analysis_port #(RAM_seq_item) mon_ap;
16
17   function new(string name = "RAM_monitor", uvm_component parent = null);
18     super.new(name,parent);
19   endfunction : new
20
21   function void build_phase(uvm_phase phase);
22     super.build_phase(phase);
23     mon_ap=new("mon_ap",this);
24   endfunction : build_phase
25
26   task run_phase(uvm_phase phase);
27     super.run_phase(phase);
28     forever begin
29       seq_item=RAM_seq_item::type_id::create("seq_item");
30       @(posedge RAM_vif.clk);
31       //input
32       seq_item.rst_n=RAM_vif.rst_n;
33       seq_item.din=RAM_vif.din;
34       seq_item.rx_valid=RAM_vif.rx_valid;
35       //output
36       seq_item.tx_valid=RAM_vif.tx_valid;
37       seq_item.dout=RAM_vif.dout;
38       //seq item.dout.ref = RAM_vif.dout.ref;
39       //seq item.tx_valid.ref=RAM_vif.tx_valid.ref;
40       mon_ap.write(seq_item);
41       `uvm_info("run_phase",seq_item.convert2string(),UVM_HIGH);
42     end
43   endtask : run_phase
44
45 endclass : RAM_monitor
46
endpackage : RAM_monitor_pkg

```

RAM Sequencer:

```

6 package RAM_sequencer_pkg;
7 import RAM_seq_item_pkg::*;
8 import uvm_pkg::*;
9 `include "uvm_macros.svh"
10 class RAM_sequencer extends uvm_sequencer #(RAM_seq_item);
11   `uvm_component_utils(RAM_sequencer)
12
13   function new(string name = "RAM_sequencer", uvm_component parent = null);
14     super.new(name, parent);
15   endfunction : new
16
17 endclass : RAM_sequencer
18
endpackage : RAM_sequencer_pkg

```

RAM Agent:

```
6 ▼ package RAM_agent_pkg;
7   import RAM_seq_item_pkg::*;
8   import RAM_driver_pkg::*;
9   import RAM_monitor_pkg::*;
10  import RAM_sequencer_pkg::*;
11  import RAM_config_pkg::*;
12  import uvm_pkg::*;
13  `include "uvm_macros.svh"
14 ▼ class RAM_agent extends uvm_agent;
15   `uvm_component_utils(RAM_agent)
16
17   RAM_driver driver;
18   RAM_monitor monitor;
19   RAM_sequencer sqr;
20   RAM_config RAM_cfg;
21   uvm_analysis_port #(RAM_seq_item) agt_ap;
22
23 ▼ function new(string name = "RAM_agent", uvm_component parent = null);
24   super.new(name,parent);
25 endfunction : new
26
27 ▼ function void build_phase(uvm_phase phase);
28   super.build_phase(phase);
29
30   if(!uvm_config_db#(RAM_config)::get(this,"","CFG",RAM_cfg))
31   `uvm_fatal("build_phase","Agent - unable to get configuration object");
32
33 ▼ if(RAM_cfg.is_active==UVM_ACTIVE) begin
34   driver =RAM_driver::type_id::create("driver",this);
35   sqr =RAM_sequencer::type_id::create("sqr",this);
36 end
37 monitor =RAM_monitor::type_id::create("monitor",this);
38 agt_ap =new("agt_ap",this);
39
40 endfunction : build_phase
41
42 ▼ function void connect_phase(uvm_phase phase);
43   super.connect_phase(phase);
44   if(RAM_cfg.is_active==UVM_ACTIVE) begin
45     driver.RAM_vif=RAM_cfg.RAM_vif;
46     driver.seq_item_port.connect(sqr.seq_item_export);
47   end
48   monitor.RAM_vif=RAM_cfg.RAM_vif;
49   monitor.mon_ap.connect(agt_ap);
50 endfunction : connect_phase
51
52 endclass : RAM_agent
53 endpackage : RAM_agent_pkg
```

RAM Scoreboard:

```
6 package RAM_scoreboard_pkg;
7   import RAM_shared_pkg::*;
8   import RAM_seq_item_pkg::*;
9   import uvm_pkg::*;
10  `include "uvm_macros.svh"
11  class RAM_scoreboard extends uvm_scoreboard;
12   `uvm_component_utils(RAM_scoreboard)
13
14   uvm_analysis_export #(RAM_seq_item) sb_export;
15   uvm_tlm_analysis_fifo #(RAM_seq_item) sb_fifo;
16   RAM_seq_item seq_item_sb;
17
18   bit tx_valid_ref;
19   bit [ADDR_SIZE - 1 : 0] dout_ref, wr_addr, rd_addr;
20   bit [ADDR_SIZE - 1 : 0] mem_fest [MEM_DEPTH];
21   // We use rx valid_reg and din_reg because the slave sends the values, and in the next clock cycle, the RAM takes these values
22   bit rx_valid_reg;
23   bit [ADDR_SIZE + 1 : 0] din_reg;
24   //counters
25   int error_count = 0;
26   int correct_count = 0;
27
28   function new(string name = "RAM_scoreboard", uvm_component parent = null);
29     super.new(name, parent);
30   endfunction : new
31
32   function void build_phase(uvm_phase phase);
33     super.build_phase(phase);
34     sb_export = new("sb_export", this);
35     sb_fifo = new("sb_fifo", this);
36   endfunction : build_phase
37
38   function void connect_phase(uvm_phase phase);
39     super.connect_phase(phase);
40     sb_export.connect(sb_fifo.analysis_export);
41   endfunction : connect_phase
42
43   task run_phase(uvm_phase phase);
44     super.run_phase(phase);
45     $readmemh("RAM_data.dat", mem_test, 0, 255);
46     forever begin
47       sb_fifo.get(seq_item_sb);
48       ref_model(seq_item_sb);
49       if ((seq_item_sb.dout !== dout_ref) || (seq_item_sb.tx_valid !== tx_valid_ref)) begin
50         `uvm_error("run_phase", $sformatf("Comparison failed, Transaction received by the DUT:%s While the reference dout: dout_ref = %0h, tx_valid_ref = %0b",
51         seq_item_sb.convert2string(), dout_ref, tx_valid_ref))
52         error_count++;
53       end
54       else begin
55         `uvm_info("run_phase", $sformatf("Correct RAM Transaction: %s", seq_item_sb.convert2string()), UVM_HIGH)
56         correct_count++;
57       end
58     end
59   endtask : run_phase
```

```

61      task ref_model(RAM_seq_item seq_item_chk);
62        if (!seq_item_chk.rst_n) begin
63          wr_addr = 0;
64          rd_addr = 0;
65          dout_ref = 0;
66          din_reg = 0;
67          rx_valid_reg = 0;
68        end
69        else begin
70          din_reg <= seq_item_sb.din;
71          rx_valid_reg <= seq_item_sb.rx_valid;
72        end
73        if (rx_valid_reg && seq_item_chk.rst_n) begin
74          case (din_reg [ADDR_SIZE + 1 : ADDR_SIZE])
75            2'b00: wr_addr = din_reg [ADDR_SIZE - 1 : 0];
76            2'b01: mem_test [wr_addr] = din_reg [ADDR_SIZE - 1 : 0];
77            2'b10: rd_addr = din_reg [ADDR_SIZE - 1 : 0];
78            2'b11: dout_ref = mem_test [rd_addr];
79        endcase
80      end
81
82      if (!seq_item_chk.rst_n)
83        tx_valid_ref = 0;
84      else if (rx_valid_reg)
85        if (din_reg [ADDR_SIZE + 1 : ADDR_SIZE] == 2'b11)
86          tx_valid_ref = 1;
87        else
88          tx_valid_ref = 0;
89    endtask : ref_model
90
91    function void report_phase (uvm_phase phase);
92      super.report_phase(phase);
93      `uvm_info("report_phase", $format("Total successful transactions: %d", correct_count), UVM_MEDIUM)
94      `uvm_info("report_phase", $format("Total failed transactions: %d", error_count), UVM_MEDIUM)
95    endfunction : report_phase
96
97  endclass : RAM_scoreboard
98 endpackage : RAM_scoreboard_pkg

```

RAM Coverage Collector:

```

6  package RAM_coverage_pkg;
7    import RAM_seq_item_pkg::*;
8    import RAM_shared_pkg::*;
9    import uvm_pkg::*;
10   `include "uvm_macros.svh"
11   class RAM_coverage extends uvm_component;
12     `uvm_component_utils(RAM_coverage)
13
14     uvm_analysis_export #(RAM_seq_item) cov_export;
15     uvm_tlm_analysis_fifo #(RAM_seq_item) cov_fifo;
16     RAM_seq_item seq_item_cov;
17
18     covergroup covgrp;
19       rst_cp:coverpoint seq_item_cov.rst_n {
20         bins rst_active = {ACTIVE_RESET};
21         bins rst_inactive= {INACTIVE_RESET};
22       }
23       rx_valid_cp:coverpoint seq_item_cov.rx_valid{
24         bins rx_active = {ACTIVE};
25         bins rx_inactive= {INACTIVE};
26       }
27       din_addr_cp:coverpoint seq_item_cov.din[ADDR_SIZE-1:0];
28       din_option_cp:coverpoint seq_item_cov.din[ADDR_SIZE+1:ADDR_SIZE]{
29         bins Write_addr = {WRITE_ADD};
30         bins Write_data = {WRITE_DATA};
31         bins Read_addr = {READ_ADD};
32         bins Read_data = {READ_DATA};
33       }
34
35       wr_addr_cr: cross rx_valid_cp, din_option_cp iff (seq_item_cov.rst_n==INACTIVE_RESET) {
36         option.cross_auto_bin_max = 0;
37         bins wr_addr_occur = binsof(rx_valid_cp) intersect {ACTIVE} && binsof(din_option_cp) intersect {WRITE_ADD};
38       }
39       wr_data_cr: cross rx_valid_cp, din_option_cp iff (seq_item_cov.rst_n==INACTIVE_RESET) {
40         option.cross_auto_bin_max = 0;
41         bins wr_data_occur = binsof(rx_valid_cp) intersect {ACTIVE} && binsof(din_option_cp) intersect {WRITE_DATA};
42       }
43       rd_addr_cr: cross rx_valid_cp, din_option_cp iff (seq_item_cov.rst_n==INACTIVE_RESET) {
44         option.cross_auto_bin_max = 0;
45         bins rd_addr_occur = binsof(rx_valid_cp) intersect {ACTIVE} && binsof(din_option_cp) intersect {READ_ADD};
46       }
47       rd_data_cr: cross rx_valid_cp, din_option_cp iff (seq_item_cov.rst_n==INACTIVE_RESET) {
48         option.cross_auto_bin_max = 0;
49         bins rd_data_occur = binsof(rx_valid_cp) intersect {ACTIVE} && binsof(din_option_cp) intersect {READ_DATA};
50       }
51     endgroup : covgrp

```

```

53     function new(string name = "RAM_coverage", uvm_component parent = null);
54         super.new(name, parent);
55         covgrp = new;
56     endfunction : new
57
58     function void build_phase(uvm_phase phase);
59         super.build_phase(phase);
60         cov_export = new("cov_export", this);
61         cov_fifo = new("cov_fifo", this);
62     endfunction : build_phase
63
64     function void connect_phase(uvm_phase phase);
65         super.connect_phase(phase);
66         cov_export.connect(cov_fifo.analysis_export);
67     endfunction : connect_phase
68
69     task run_phase(uvm_phase phase);
70         super.run_phase(phase);
71         forever begin
72             cov_fifo.get(seq_item_cov);
73             covgrp.sample();
74         end
75     endtask : run_phase
76
77 endclass : RAM_coverage
78
endpackage : RAM_coverage_pkg

```

RAM Environment:

```

6 package RAM_env_pkg;
7     import RAM_scoreboard_pkg::*;
8     import RAM_coverage_pkg::*;
9     import RAM_agent_pkg::*;
10    import uvm_pkg::*;
11    `include "uvm_macros.svh"
12    class RAM_env extends uvm_env;
13        uvm_component_utils(RAM_env)
14
15        RAM_agent agt;
16        RAM_scoreboard sb;
17        RAM_coverage cov;
18
19        function new(string name = "RAM_env" , uvm_component parent = null);
20            super.new(name,parent);
21        endfunction : new
22
23        function void build_phase(uvm_phase phase);
24            super.build_phase(phase);
25            agt=RAM_agent::type_id::create("agt",this);
26            sb=RAM_scoreboard::type_id::create("sb",this);
27            cov=RAM_coverage::type_id::create("cov",this);
28        endfunction : build_phase
29
30        function void connect_phase(uvm_phase phase);
31            super.connect_phase(phase);
32            agt.agt_ap.connect(sb.sb_export);
33            agt.agt_ap.connect(cov.cov_export);
34        endfunction : connect_phase
35
36    endclass : RAM_env
37
38 endpackage : RAM_env_pkg

```

SLAVE Environment: -

SLAVE Driver:

```

6   package slave_driver_pkg;
7     import slave_seq_item_pkg::*;
8     import uvm_pkg::*;
9     `include "uvm_macros.svh"
10
11    class slave_driver extends uvm_driver #(slave_seq_item);
12      `uvm_component_utils(slave_driver);
13
14      slave_seq_item stim_seq_item;
15      virtual slave_if slave_vif;
16
17      function new(string name = "slave_driver", uvm_component parent = null);
18        super.new(name, parent);
19      endfunction : new
20
21      task run_phase(uvm_phase phase);
22        super.run_phase(phase);
23        forever begin
24          stim_seq_item = slave_seq_item::type_id::create("stim_seq_item");
25          seq_item_port.get_next_item(stim_seq_item);
26
27          slave_vif.rst_n = stim_seq_item.rst_n;
28          slave_vif.SS_n = stim_seq_item.SS_n;
29          slave_vif.tx_valid = stim_seq_item.tx_valid;
30          slave_vif.tx_data = stim_seq_item.tx_data;
31          slave_vif.MOSI = stim_seq_item.MOSI;
32          @(negedge slave_vif.clk);
33          seq_item_port.item_done();
34          `uvm_info("run_phase", stim_seq_item.convert2string(), UVM_HIGH);
35        end
36      endtask : run_phase
37
38    endclass : slave_driver
39  endpackage : slave_driver_pkg

```

SLAVE Monitor:

```

6   package slave_monitor_pkg;
7     import slave_seq_item_pkg::*;
8     import uvm_pkg::*;
9     `include "uvm_macros.svh"
10    class slave_monitor extends uvm_monitor;
11      `uvm_component_utils(slave_monitor)
12
13      slave_seq_item seq_item;
14      virtual slave_if slave_vif;
15      uvm_analysis_port #(slave_seq_item) mon_ap;
16
17      function new(string name = "slave_monitor", uvm_component parent = null);
18        super.new(name, parent);
19      endfunction : new
20
21      function void build_phase(uvm_phase phase);
22        super.build_phase(phase);
23        mon_ap = new("mon_ap", this);
24      endfunction : build_phase
25
26      task run_phase(uvm_phase phase);
27        super.run_phase(phase);
28        forever begin
29          seq_item = slave_seq_item::type_id::create("seq_item");
30          @(negedge slave_vif.clk);
31          //Input
32          seq_item.rst_n = slave_vif.rst_n;
33          seq_item.SS_n = slave_vif.SS_n;
34          seq_item.MOSI = slave_vif.MOSI;
35          seq_item.tx_valid = slave_vif.tx_valid;
36          seq_item.tx_data = slave_vif.tx_data;
37          //Output
38          seq_item.rx_valid = slave_vif.rx_valid;
39          seq_item.rx_data = slave_vif.rx_data;
40          seq_item.MISO = slave_vif.MISO;
41          seq_item.rx_valid_ref = slave_vif.rx.valid_ref;
42          seq_item.rx_data_ref = slave_vif.rx.data_ref;
43          seq_item.MISO_ref = slave_vif.MISO_ref;
44          mon_ap.write(seq_item);
45          `uvm_info("run_phase", seq_item.convert2string(), UVM_HIGH)
46        end
47      endtask : run_phase
48
49    endclass : slave_monitor
50  endpackage : slave_monitor_pkg

```

SLAVE Sequencer:

```
6 package slave_sequencer_pkg;
7   import slave_seq_item_pkg::*;
8   import uvm_pkg::*;
9   `include "uvm_macros.svh"
10  class slave_sequencer extends uvm_sequencer #(slave_seq_item);
11    `uvm_component_utils(slave_sequencer)
12
13  function new(string name = "slave_sequencer", uvm_component parent = null);
14    super.new(name, parent);
15  endfunction : new
16
17 endclass : slave_sequencer
18 endpackage : slave_sequencer_pkg
```

SLAVE Agent:

```
6 package slave_agent_pkg;
7   import slave_seq_item_pkg::*;
8   import slave_driver_pkg::*;
9   import slave_monitor_pkg::*;
10  import slave_sequencer_pkg::*;
11  import slave_config_pkg::*;
12  import uvm_pkg::*;
13  `include "uvm_macros.svh"
14  class slave_agent extends uvm_agent;
15    `uvm_component_utils(slave_agent)
16
17    slave_driver driver;
18    slave_monitor monitor;
19    slave_sequencer sqr;
20    slave_config slave_cfg;
21    uvm_analysis_port #(slave_seq_item) agt_ap;
22
23  function new(string name = "slave_agent" , uvm_component parent = null);
24    super.new(name,parent);
25  endfunction : new
26
27  function void build_phase(uvm_phase phase);
28    super.build_phase(phase);
29
30    if(!uvm_config_db#(slave_config)::get(this,"","CFG",slave_cfg))
31      `uvm_fatal("build_phase","Agent - unable to get configuration object");
32
33    if(slave_cfg.is_active==UVM_ACTIVE) begin
34      driver = slave_driver::type_id::create("driver",this);
35      sqr = slave_sequencer::type_id::create("sqr",this);
36    end
37    monitor = slave_monitor::type_id::create("monitor",this);
38    agt_ap = new("agt_ap",this);
39
40  endfunction : build_phase
41
42  function void connect_phase(uvm_phase phase);
43    super.connect_phase(phase);
44    if(slave_cfg.is_active==UVM_ACTIVE) begin
45      driver.slave_vif=slave_cfg.slave_vif;
46      driver.seq_item_port.connect(sqr.seq_item_export);
47    end
48    monitor.slave_vif=slave_cfg.slave_vif;
49    monitor.mon_ap.connect(agt_ap);
50  endfunction : connect_phase
51
52 endclass : slave_agent
53 endpackage : slave_agent_pkg
```

SLAVE Scoreboard:

```
6 package slave_scoreboard_pkg;
7   import slave_seq_item_pkg::*;
8   import uvm_pkg::*;
9   `include "uvm_macros.svh"
10  class slave_scoreboard extends uvm_scoreboard;
11    `uvm_component_utils(slave_scoreboard)
12
13    uvm_analysis_export #(slave_seq_item) sb_export;
14    uvm_tlm_analysis_fifo #(slave_seq_item) sb_fifo;
15    slave_seq_item seq_item_sb;
16
17    //Counters for check the result
18    int rx_error_count = 0;
19    int rx_correct_count = 0;
20    int MISO_error_count = 0;
21    int MISO_correct_count = 0;
```

```

22
23     function new(string name = "slave_scoreboard", uvm_component parent = null);
24         super.new(name, parent);
25     endfunction : new
26
27     function void build_phase(uvm_phase phase);
28         super.build_phase(phase);
29         sb_export = new("sb_export", this);
30         sb_fifo = new("sb_fifo", this);
31     endfunction : build_phase
32
33     function void connect_phase(uvm_phase phase);
34         super.connect_phase(phase);
35         sb_export.connect(sb_fifo.analysis_export);
36     endfunction : connect_phase
37
38     task run_phase(uvm_phase phase);
39         super.run_phase(phase);
40         forever begin
41             sb_fifo.get(seq_item_sb);
42             if ((seq_item_sb.rx_data_ref == seq_item_sb.rx_data) || (seq_item_sb.rx_valid_ref == seq_item_sb.rx_valid)) begin
43                 `uvm_error("run_phase", $sformatf("comparision RX failed, DUT:%s While :rx_data_ref = %0h, rx_valid_ref = %0b",
44                 seq_item_sb.convert2string(), seq_item_sb.rx_data_ref, seq_item_sb.rx_valid_ref))
45                 rx_error_count++;
46             end
47             else begin
48                 `uvm_info("run_phase", $sformatf("Correct RX: %s", seq_item_sb.convert2string()), UVM_HIGH)
49                 rx_correct_count++;
50             end
51             if (seq_item_sb.MISO != seq_item_sb.MISO_ref) begin
52                 `uvm_error("run_phase", $sformatf("comparision MISO failed, DUT:%s While :MISO_ref = %0b",
53                 seq_item_sb.convert2string(), seq_item_sb.MISO_ref))
54                 MISO_error_count++;
55             end
56             else begin
57                 `uvm_info("run_phase", $sformatf("Correct RX: %s", seq_item_sb.convert2string()), UVM_HIGH)
58                 MISO_correct_count++;
59             end
60         end
61     endtask : run_phase
62
63     function void report_phase (uvm_phase phase);
64         super.report_phase(phase);
65         `uvm_info("report_phase", $sformatf("Total successful[RX]: %0d", rx_correct_count), UVM_LOW)
66         `uvm_info("report_phase", $sformatf("Total failed[RX]: %0d", rx_error_count), UVM_LOW)
67         `uvm_info("report_phase", $sformatf("Total successful[MISO]: %0d", MISO_correct_count), UVM_LOW)
68         `uvm_info("report_phase", $sformatf("Total failed[MISO]: %0d", MISO_error_count), UVM_LOW)
69     endfunction : report_phase
70
71 endclass : slave_scoreboard
72
73 endpackage : slave_scoreboard_pkg

```

SLAVE Coverage Collector:

```
6  package slave_coverage_pkg;
7      import slave_seq_item_pkg::*;
8      import wrapper_shared_pkg::*;
9      import uvm_pkg::*;
10     `include "uvm_macros.svh"
11     class slave_coverage extends uvm_scoreboard;
12         `uvm_component_utils(slave_coverage)
13
14     uvm_analysis_export #(slave_seq_item) cov_export;
15     uvm_tlm_analysis_fifo #(slave_seq_item) cov_fifo;
16     slave_seq_item seq_item_cov;
17
18     covergroup covgrp;
19         rst_cp:coverpoint seq_item_cov.rst_n{
20             bins active = {0};
21             bins inactive = {1};
22         }
23
24         cs_cov_cp: coverpoint cs_cov {
25             bins wr_addr = {WRITE_ADD_COV};
26             bins wr_data = {WRITE_DATA_COV};
27             bins rd_addr = {READ_ADD_COV};
28             bins rd_data = {READ_DATA_COV};
29         }
30     endgroup : covgrp
31
32     function new(string name = "slave_coverage", uvm_component parent = null);
33         super.new(name, parent);
34         covgrp = new;
35     endfunction : new
36
37     function void build_phase(uvm_phase phase);
38         super.build_phase(phase);
39         cov_export = new("cov_export", this);
40         cov_fifo = new("cov_fifo", this);
41     endfunction : build_phase
42
43     function void connect_phase(uvm_phase phase);
44         super.connect_phase(phase);
45         cov_export.connect(cov_fifo.analysis_export);
46     endfunction : connect_phase
47
48     task run_phase(uvm_phase phase);
49         super.run_phase(phase);
50         forever begin
51             cov_fifo.get(seq_item_cov);
52             covgrp.sample();
53         end
54     endtask : run_phase
55
56
57     endclass : slave_coverage
58 endpackage : slave_coverage_pkg
```

SLAVE Environment:

```
6  package slave_env_pkg;
7      import slave_scoreboard_pkg::*;
8      import slave_coverage_pkg::*;
9      import slave_agent_pkg::*;
10     import uvm_pkg::*;
11     `include "uvm_macros.svh"
12     class slave_env extends uvm_env;
13         `uvm_component_utils(slave_env)
14
15         slave_agent agt;
16         slave_scoreboard sb;
17         slave_coverage cov;
18
19         function new(string name = "slave_env" , uvm_component parent = null);
20             super.new(name, parent);
21         endfunction : new
22
23         function void build_phase(uvm_phase phase);
24             super.build_phase(phase);
25             agt=slave_agent::type_id::create("agt",this);
26             sb=slave_scoreboard::type_id::create("sb",this);
27             cov=slave_coverage::type_id::create("cov",this);
28         endfunction : build_phase
29
30         function void connect_phase(uvm_phase phase);
31             super.connect_phase(phase);
32             agt.agt_ap.connect(sb.sb_export);
33             agt.agt_ap.connect(cov.cov_export);
34         endfunction : connect_phase
35
36     endclass : slave_env
37
38 endpackage : slave_env_pkg
```

Wrapper Environment: -

Wrapper Driver:

```

6  package wrapper_driver_pkg;
7    import wrapper_seq_item_pkg::*;
8    import uvm_pkg::*;
9    `include "uvm_macros.svh"
10
11  class wrapper_driver extends uvm_driver #(wrapper_seq_item);
12    `uvm_component_utils(wrapper_driver);
13
14    wrapper_seq_item stim_seq_item;
15    virtual wrapper_if wrapper_vif;
16
17    function new(string name = "wrapper_driver", uvm_component parent = null);
18      super.new(name, parent);
19    endfunction : new
20
21    task run_phase(uvm_phase phase);
22      super.run_phase(phase);
23      forever begin
24        stim_seq_item = wrapper_seq_item::type_id::create("stim_seq_item");
25        seq_item_port.get_next_item(stim_seq_item);
26        wrapper_vif.rst_n = stim_seq_item.rst_n;
27        wrapper_vif.MOSI = stim_seq_item.MOSI;
28        wrapper_vif.SS_n = stim_seq_item.SS_n;
29        @(negedge wrapper_vif.clk);
30        seq_item_port.item_done();
31        `uvm_info("run_phase", stim_seq_item.convert2string(), UVM_HIGH);
32      end
33    endtask : run_phase
34
35  endclass : wrapper_driver
36 endpackage : wrapper_driver_pkg

```

Wrapper Monitor:

```

6  package wrapper_monitor_pkg;
7    import wrapper_seq_item_pkg::*;
8    import uvm_pkg::*;
9    `include "uvm_macros.svh"
10   class wrapper_monitor extends uvm_monitor;
11     `uvm_component_utils(wrapper_monitor)
12
13     wrapper_seq_item seq_item;
14     virtual wrapper_if wrapper_vif;
15     uvm_analysis_port #(wrapper_seq_item) mon_ap;
16
17     function new(string name = "wrapper_monitor", uvm_component parent = null);
18       super.new(name, parent);
19     endfunction : new
20
21     function void build_phase(uvm_phase phase);
22       super.build_phase(phase);
23       mon_ap = new("mon_ap", this);
24     endfunction : build_phase
25
26     task run_phase(uvm_phase phase);
27       super.run_phase(phase);
28       forever begin
29         seq_item = wrapper_seq_item::type_id::create("seq_item");
30         @(negedge wrapper_vif.clk);
31         //Input
32         seq_item.rst_n = wrapper_vif.rst_n;
33         seq_item.SS_n = wrapper_vif.SS_n;
34         seq_item.MOSI = wrapper_vif.MOSI;
35         //Output
36         seq_item.MISO = wrapper_vif.MISO;
37         seq_item.MISO_ref = wrapper_vif.MISO_ref;
38         mon_ap.write(seq_item);
39         `uvm_info("run_phase", seq_item.convert2string(), UVM_HIGH)
40       end
41     endtask : run_phase
42
43   endclass : wrapper_monitor
44 endpackage : wrapper_monitor_pkg

```

Wrapper Sequencer:

```

6  package wrapper_sequencer_pkg;
7    import wrapper_seq_item_pkg::*;
8    import uvm_pkg::*;
9    `include "uvm_macros.svh"
10   class wrapper_sequencer extends uvm_sequencer #(wrapper_seq_item);
11     `uvm_component_utils(wrapper_sequencer)
12
13     function new(string name = "wrapper_sequencer", uvm_component parent = null);
14       super.new(name, parent);
15     endfunction : new
16
17   endclass : wrapper_sequencer
18 endpackage : wrapper_sequencer_pkg

```

Wrapper Agent:

```
6 package wrapper_agent_pkg;
7     import wrapper_seq_item_pkg::*;
8     import wrapper_driver_pkg::*;
9     import wrapper_monitor_pkg::*;
10    import wrapper_sequencer_pkg::*;
11    import wrapper_config_pkg::*;
12    import uvm_pkg::*;
13    `include "uvm_macros.svh"
14    class wrapper_agent extends uvm_agent;
15        `uvm_component_utils(wrapper_agent)
16
17        wrapper_driver driver;
18        wrapper_monitor monitor;
19        wrapper_sequencer sqr;
20        wrapper_config wrapper_cfg;
21        uvm_analysis_port #(wrapper_seq_item) agt_ap;
22
23        function new(string name = "wrapper_agent", uvm_component parent = null);
24            super.new(name, parent);
25        endfunction : new
26
27        function void build_phase(uvm_phase phase);
28            super.build_phase(phase);
29
30            if(!uvm_config_db#(wrapper_config)::get(this,"","CFG",wrapper_cfg))
31                `uvm_fatal("build phase","Agent - unable to get configuration object");
32
33            if(wrapper_cfg.is_active==UVM_ACTIVE) begin
34                driver = wrapper_driver::type_id::create("driver",this);
35                sqr = wrapper_sequencer::type_id::create("sqr",this);
36            end
37            monitor = wrapper_monitor::type_id::create("monitor",this);
38            agt_ap = new("agt_ap",this);
39
40        endfunction : build_phase
41
42        function void connect_phase(uvm_phase phase);
43            super.connect_phase(phase);
44            if(wrapper_cfg.is_active==UVM_ACTIVE) begin
45                driver.wrapper_vif=wrapper_cfg.wrapper_vif;
46                driver.seq_item_port.connect(sqr.seq_item_export);
47            end
48            monitor.wrapper_vif=wrapper_cfg.wrapper_vif;
49            monitor.mon_ap.connect(agt_ap);
50        endfunction : connect_phase
51
52    endclass : wrapper_agent
53 endpackage : wrapper_agent_pkg
```

Wrapper Scoreboard:

```
6 package wrapper_scoreboard_pkg;
7     import wrapper_seq_item_pkg::*;
8     import uvm_pkg::*;
9     `include "uvm_macros.svh"
10    class wrapper_scoreboard extends uvm_scoreboard;
11        `uvm_component_utils(wrapper_scoreboard)
12
13        uvm_analysis_export #(wrapper_seq_item) sb_export;
14        uvm_tlm_analysis_fifo #(wrapper_seq_item) sb_fifo;
15        wrapper_seq_item seq_item_sb;
16
17        //Counters for check the result
18        int MISO_error_count = 0;
19        int MISO_correct_count = 0;
20
21        function new(string name = "wrapper_scoreboard", uvm_component parent = null);
22            super.new(name, parent);
23        endfunction : new
24
25        function void build_phase(uvm_phase phase);
26            super.build_phase(phase);
27            sb_export = new("sb_export", this);
28            sb_fifo = new("sb_fifo", this);
29        endfunction : build_phase
30
31        function void connect_phase(uvm_phase phase);
32            super.connect_phase(phase);
33            sb_export.connect(sb_fifo.analysis_export);
34        endfunction : connect_phase
35
36        task run_phase(uvm_phase phase);
37            super.run_phase(phase);
38            //$readmemh("RAM_data.dat", mem_ref, 0, 255);
39            forever begin
40                sb_fifo.get(seq_item_sb);
41                if (seq_item_sb.MISO==seq_item_sb.MISO_ref) begin
42                    `uvm_error("run_phase", $sformatf("Comparison MISO failed, DUT:%s While :MISO_ref = %0b",
43                        seq_item_sb.convert2string(), seq_item_sb.MISO_ref))
44                    MISO_error_count++;
45                end
46                else begin
47                    `uvm_info("run_phase", $sformatf("Correct RX: %s", seq_item_sb.convert2string()), UVM_HIGH)
48                    MISO_correct_count++;
49                end
50            end
51        endtask : run_phase
```

```

53     function void report_phase (uvm_phase phase);
54         super.report_phase(phase);
55         `uvm_info("report_phase", $sformatf("Total successful[MOSO]: %0d", MISO_correct_count), UVM_LOW)
56         `uvm_info("report_phase", $sformatf("Total failed[MOSO]: %0d", MISO_error_count), UVM_LOW)
57     endfunction : report_phase
58
59 endclass : wrapper_scoreboard
60 endpackage : wrapper_scoreboard_pkg

```

Wrapper Coverage Collector:

```

6 package wrapper_coverage_pkg;
7     import wrapper_seq_item_pkg::*;
8     import wrapper_shared_pkg::*;
9     import uvm_pkg::*;
10    `include "uvm_macros.svh"
11 class wrapper_coverage extends uvm_scoreboard;
12     `uvm_component_utils(wrapper_coverage)
13
14     uvm_analysis_export #(wrapper_seq_item) cov_export;
15     uvm_tlm_analysis_fifo #(wrapper_seq_item) cov_fifo;
16     wrapper_seq_item seq_item_cov;
17
18     covergroup covgrp;
19         rst_cp:coverpoint seq_item_cov.rst_n{
20             bins active = {0};
21             bins inactive = {1};
22         }
23
24         cs_cov_cp: coverpoint cs_cov {
25             bins wr_addr = {WRITE_ADD_COV};
26             bins wr_data = {WRITE_DATA_COV};
27             bins rd_addr = {READ_ADD_COV};
28             bins rd_data = {READ_DATA_COV};
29         }
30     endgroup : covgrp
31
32     function new(string name = "wrapper_coverage", uvm_component parent = null);
33         super.new(name, parent);
34         covgrp = new;
35     endfunction : new
36
37     function void build_phase(uvm_phase phase);
38         super.build_phase(phase);
39         cov_export = new("cov_export", this);
40         cov_fifo = new("cov_fifo", this);
41     endfunction : build_phase
42
43     function void connect_phase(uvm_phase phase);
44         super.connect_phase(phase);
45         cov_export.connect(cov_fifo.analysis_export);
46     endfunction : connect_phase
47
48     task run_phase(uvm_phase phase);
49         super.run_phase(phase);
50         forever begin
51             cov_fifo.get(seq_item_cov);
52             covgrp.sample();
53         end
54     endtask : run_phase
55
56
57 endclass : wrapper_coverage
58 endpackage : wrapper_coverage_pkg

```

Wrapper Environment:

```
6  package wrapper_env_pkg;
7      import wrapper_scoreboard_pkg::*;
8      import wrapper_coverage_pkg::*;
9      import wrapper_agent_pkg::*;
10     import uvm_pkg::*;
11     `include "uvm_macros.svh"
12     class wrapper_env extends uvm_env;
13         `uvm_component_utils(wrapper_env)
14
15         wrapper_agent agt;
16         wrapper_scoreboard sb;
17         wrapper_coverage cov;
18
19         function new(string name = "wrapper_env", uvm_component parent = null);
20             super.new(name, parent);
21         endfunction : new
22
23         function void build_phase(uvm_phase phase);
24             super.build_phase(phase);
25             agt=wrapper_agent::type_id::create("agt",this);
26             sb=wrapper_scoreboard::type_id::create("sb",this);
27             cov=wrapper_coverage::type_id::create("cov",this);
28         endfunction : build_phase
29
30         function void connect_phase(uvm_phase phase);
31             super.connect_phase(phase);
32             agt.agt_ap.connect(sb.sb_export);
33             agt.agt_ap.connect(cov.cov_export);
34         endfunction : connect_phase
35
36     endclass : wrapper_env
37
38 endpackage : wrapper_env_pkg
```

Test & Top: -

Test file:

```
6  package wrapper_test_pkg;
7      import RAM_env_pkg::*;
8      import slave_env_pkg::*;
9      import wrapper_env_pkg::*;
10     import RAM_config_pkg::*;
11     import slave_config_pkg::*;
12     import wrapper_config_pkg::*;
13     import wrapper_sequence_pkg::*;
14     import uvm_pkg::*;
15     `include "uvm_macros.svh"
16
17     class wrapper_test extends uvm_test;
18         `uvm_component_utils(wrapper_test)
19
20         RAM_env env_RAM;
21         slave_env env_slave;
22         wrapper_env env_wrapper;
23         RAM_config RAM_cfg;
24         slave_config slave_cfg;
25         wrapper_config wrapper_cfg;
26         wrapper_rst_assert rst_seq;
27         wrapper_main_seq main_seq;
28
29         function new(string name = "wrapper_test", uvm_component parent = null);
30             super.new(name, parent);
31         endfunction : new
32
```

```

32         function void build_phase (uvm_phase phase);
33             super.build_phase(phase);
34             env_RAM      = RAM_env::type_id::create("env_RAM", this);
35             env_slave    = slave_env::type_id::create("env_slave", this);
36             env_wrapper  = wrapper_env::type_id::create("env_wrapper", this);
37             RAM_cfg      = RAM_config::type_id::create("RAM_cfg");
38             slave_cfg   = slave_config::type_id::create("slave_cfg");
39             wrapper_cfg = wrapper_config::type_id::create("wrapper_cfg");
40             rst_seq     = wrapper_rst_assert::type_id::create("rst_seq");
41             main_seq    = wrapper_main_seq::type_id::create("main_seq");
42
43             if (!uvm_config_db #(virtual RAM_if)::get(this, "", "RAM_IF", RAM_cfg.RAM_vif))
44                 `uvm_fatal("build_phase", "Test-unable to get the virtual interface of the RAM from the uvm_config_db")
45             if (!uvm_config_db #(virtual slave_if)::get(this, "", "slave_IF", slave_cfg.slave_vif))
46                 `uvm_fatal("build_phase", "Test-unable to get the virtual interface of the SLAVE from the uvm_config_db")
47             if (!uvm_config_db #(virtual wrapper_if)::get(this, "", "wrapper_IF", wrapper_cfg.wrapper_vif))
48                 `uvm_fatal("build_phase", "Test-unable to get the virtual interface of the wrapper from the uvm_config_db")
49
50             RAM_cfg.is_active  = UVM_PASSIVE;
51             slave_cfg.is_active = UVM_PASSIVE;
52             wrapper_cfg.is_active=UVM_ACTIVE;
53
54             uvm_config_db #(RAM_config) ::set(this, "*", "CFG", RAM_cfg);
55             uvm_config_db #(slave_config) ::set(this, "*", "CFG", slave_cfg);
56             uvm_config_db #(wrapper_config) ::set(this, "*", "CFG", wrapper_cfg);
57         endfunction : build_phase
58
59         task run_phase (uvm_phase phase);
60             super.run_phase(phase);
61             phase.raise_objection(this);
62                 `uvm_info("run_phase", "Reset Asserted", UVM_LOW);
63                 rst_seq.start(env_wrapper.agt.sqr);
64                 `uvm_info("run_phase", "Reset Deasserted", UVM_LOW);
65
66                 `uvm_info("run_phase", "Main Seq Started", UVM_LOW);
67                 main_seq.start(env_wrapper.agt.sqr);
68                 `uvm_info("run_phase", "Main Seq Ended", UVM_LOW);
69
70                 `uvm_info("run_phase", "Finish Test", UVM_LOW);
71             phase.drop_objection(this);
72         endtask : run_phase
73
74     endclass : wrapper_test
75 endpackage : wrapper_test_pkg

```

Top file:

```

6   import wrapper_test_pkg::*;
7   import uvm_pkg::*;
8   `include "uvm_macros.svh"
9   module top ();
10    bit clk;
11    initial begin
12      $readmemh("RAM_data.dat", DUT_WRAPPER.ram.mem, 0, 255);
13      $readmemh("RAM_data.dat", GOLDEN_WRAPPER.mem, 0, 255);
14      clk=0;
15      forever begin
16        #1 clk=~clk;
17      end
18    end
19  //*****
20  //Interfaces
21  slave_if s_if(clk);
22  RAM_if r_if(clk);
23  wrapper_if w_if(clk);
24  //*****
25  Maindesign DUT_WRAPPER(
26    .clk (clk),
27    .rst_n(w_if.rst_n),
28    .MISO (w_if.MISO),
29    .MOSI (w_if.MOSI),
30    .SS_n (w_if.SS_n)
31  );
32  //*****
33  SPI_SLAVE GOLDEN_SLAVE(
34    .clk (clk),
35    .MISO (s_if.MISO_ref),
36    .MOSI (s_if.MOSI),
37    .tx_valid(s_if.tx_valid),
38    .tx_data (s_if.tx_data),
39    .rx_valid(s_if.rx_valid_ref),
40    .SS_n (s_if.SS_n),
41    .rx_data (s_if.rx_data_ref),
42    .rst_n (s_if.rst_n)
43  );
44  wrapper_golden GOLDEN_WRAPPER(
45    .clk (clk),
46    .rst_n(w_if.rst_n),
47    .MISO (w_if.MISO_ref),
48    .MOSI (w_if.MOSI),
49    .SS_n (w_if.SS_n)
50  );
51  //*****
52  //Assertions
53  bind RAM RAM_sva RAM_SVA(
54    .din(r_if.din),
55    .rx_valid(r_if.rx_valid),
56    .rst_n(r_if.rst_n),
57    .dout(r_if.dout),
58    .tx_valid(r_if.tx_valid),
59    .wr_addr(DUT_WRAPPER.ram.wr_addr),
60    .rd_addr(DUT_WRAPPER.ram.rd_addr),
61    .mem(DUT_WRAPPER.ram.mem),
62    .clk(clk)
63  );
64
65  bind SLAVE slave_sva SLAVE_SVA(
66    .clk (clk),
67    .MISO (s_if.MISO),
68    .MOSI (s_if.MOSI),
69    .tx_valid (s_if.tx_valid),
70    .tx_data (s_if.tx_data),
71    .SS_n (s_if.SS_n),
72    .rx_valid (s_if.rx_valid),
73    .rst_n (s_if.rst_n),
74    .rx_data (s_if.rx_data),
75    .cs_ (DUT_WRAPPER.spi.cs),
76    .rd_add_signal(DUT_WRAPPER.spi.rd_add_signal),
77    .count_rx (DUT_WRAPPER.spi.count_rx),
78    .count_tx (DUT_WRAPPER.spi.count_tx)
79  );
80
81  bind Maindesign wrapper_sva WRAPPER_SVA(
82    .rst_n (w_if.rst_n),
83    .MOSI (w_if.MOSI),
84    .SS_n (w_if.SS_n),
85    .MISO (w_if.MISO),
86    .tx_valid(DUT_WRAPPER.tx_valid),
87    .rx_valid(DUT_WRAPPER.rx_valid),
88    .tx_data (DUT_WRAPPER.tx_data),
89    .rx_data (DUT_WRAPPER.rx_data),
90    .clk (clk)
91  );
92  //*****

```

```

92  *****/
93  assign r_if.din    = DUT_WRAPPER.rx_data;
94  assign r_if.rst_n  = w_if.rst_n;
95  assign r_if.rx_valid = DUT_WRAPPER.rx_valid;
96  assign r_if.dout   = DUT_WRAPPER.tx_data;
97  assign r_if.tx_valid = DUT_WRAPPER.tx_valid;
98
99  assign s_if.MOSI   = w_if.MOSI;
100 assign s_if.SS_n   = w_if.SS_n;
101 assign s_if.rst_n  = w_if.rst_n;
102 assign s_if.tx_valid = DUT_WRAPPER.tx_valid;
103 assign s_if.tx_data = DUT_WRAPPER.tx_data;
104 assign s_if.rx_data = DUT_WRAPPER.rx_data;
105 assign s_if.MISO   = w_if.MISO;
106 assign s_if.rx_valid = DUT_WRAPPER.rx_valid;
107 *****/
108 initial begin
109     uvm_config_db#(virtual RAM_if)::set(null,"uvm_test_top","RAM_IF",r_if );
110     uvm_config_db#(virtual slave_if)::set(null,"uvm_test_top","slave_IF",s_if );
111     uvm_config_db#(virtual wrapper_if)::set(null,"uvm_test_top","wrapper_IF",w_if );
112
113     run_test("wrapper_test");
114 end
115 endmodule : top

```

For running the test files: -

Do file to run the RAM:

```

1 vlib work
2 vlog -f list_ram.list +cover -covercells
3 vsim -voptargs+=acc work.top -cover
4 add wave -position insertpoint sim:/top/r_if/*
5 add wave -position insertpoint \
6 sim:/top/DUT/mem
7 run -all

```

Do file to run the SLAVE:

```

1 vlib work
2 vlog -f list_slave.list +cover -covercells
3 vsim -voptargs+=acc work.top -cover
4 add wave -position insertpoint sim:/top/s_if/*
5 coverage exclude -src ../slave/Design/SPI.sv -line 99 -code b
6 coverage exclude -src ../slave/Design/SPI.sv -line 27 -code b
7 coverage exclude -src ../slave/Design/SPI.sv -line 73 -code b
8 coverage exclude -src ../slave/Design/SPI.sv -line 98 -code b
9 coverage exclude -src ../slave/Design/SPI.sv -line 37 -code c
10 coverage exclude -src ../slave/Design/SPI.sv -line 39 -code c
11 coverage exclude -src ../slave/Design/SPI.sv -line 98 -code c
12 coverage exclude -src ../slave/Design/SPI.sv -line 102 -code c
13 run -all

```

Do file to run the Wrapper:

```

1 vlib work
2 vlog -f list_wrapper.list +cover -covercells
3 vsim -voptargs+=acc work.top -cover
4 add wave -position insertpoint sim:/top/w_if/*
5 run -all

```

List of RAM:

```
1  ./RAM/Shared_pkg/RAM_shared_pkg.sv
2  ./RAM/Design/RAM.sv
3  ./RAM/Interface/RAM_if.sv
4  ./RAM/Assertions/RAM_sva.sv
5
6  ./RAM/Objects/Configrstion/RAM_config_pkg.sv
7  ./RAM/Objects/Sequence_Item/RAM_seq_item_pkg.sv
8  ./RAM/Objects/Sequences/RAM_seq_pkg.sv
9
10 ./RAM/Top/Test/Enviroment/Agent/Driver/RAM_driver_pkg.sv
11 ./RAM/Top/Test/Enviroment/Agent/Monitor/RAM_monitor_pkg.sv
12 ./RAM/Top/Test/Enviroment/Agent/Sequencer/RAM_sequencer_pkg.sv
13 ./RAM/Top/Test/Enviroment/Agent/RAM_agent_pkg.sv
14
15 ./RAM/Top/Test/Coverage_Collector/RAM_coverage_pkg.sv
16 ./RAM/Top/Test/Enviroment/Scoreboard/RAM_scoreboard_pkg.sv
17 ./RAM/Top/Test/Enviroment/RAM_env_pkg.sv
18
19 ./RAM/Top/Test/RAM_test_pkg.sv
20 ./RAM/Top/RAM_top.sv
```

List of SLAVE:

```
1  ./slave/Shared_pkg/slave_shared_pkg.sv
2  ./slave/Design/SPI.sv
3  ./slave/Golden/Slave.sv
4  ./slave/Assertions/slave_sva.sv
5  ./slave/Interface/slave_if.sv
6
7
8  ./slave/Objects/Configrstion/slave_config_pkg.sv
9  ./slave/Objects/Sequence_Item/slave_seq_item_pkg.sv
10 ./slave/Objects/Sequences/slave_seq_pkg.sv
11
12 ./slave/Top/Test/Enviroment/Agent/Driver/slave_driver_pkg.sv
13 ./slave/Top/Test/Enviroment/Agent/Monitor/slave_monitor_pkg.sv
14 ./slave/Top/Test/Enviroment/Agent/Sequencer/slave_sequencer_pkg.sv
15 ./slave/Top/Test/Enviroment/Agent/slave_agent_pkg.sv
16
17 ./slave/Top/Test/Coverage_Collector/slave_coverage_pkg.sv
18 ./slave/Top/Test/Enviroment/Scoreboard/slave_scoreboard_pkg.sv
19 ./slave/Top/Test/Enviroment/slave_env_pkg.sv
20
21 ./slave/Top/Test/slave_test_pkg.sv
22 ./slave/Top/slave_top.sv
```

List of Wrapper:

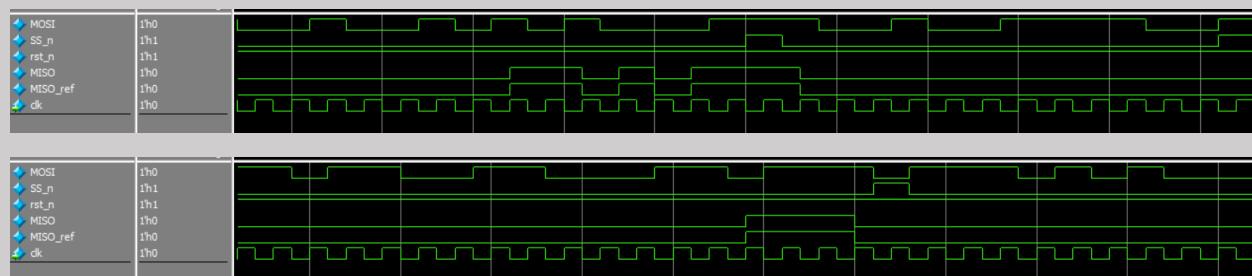
```

1  ./wrapper/Shared_pkg/RAM_shared_pkg.sv
2  ./wrapper/Shared_pkg/wrapper_shared_pkg.sv
3
4  ./wrapper/Design/RAM.sv
5  ./wrapper/Design/SPI.sv
6  ./wrapper/Design/Wrapper.sv
7
8  ./wrapper/Golden/Slave.sv
9  ./wrapper/Golden/wrapper_golden.sv
10
11 ./wrapper/Assertions/RAM_sva.sv
12 ./wrapper/Assertions/slave_sva.sv
13 ./wrapper/Assertions/Wrapper_sva.sv
14
15 ./wrapper/Interface/RAM_if.sv
16 ./wrapper/Interface/slave_if.sv
17 ./wrapper/Interface/wrapper_if.sv
18
19
20 ./wrapper/Objects/Configuration/RAM_config_pkg.sv
21 ./wrapper/Objects/Configuration/slave_config_pkg.sv
22 ./wrapper/Objects/Configuration/wrapper_config_pkg.sv
23
24 ./wrapper/Objects/Sequence_Item/RAM_seq_item_pkg.sv
25 ./wrapper/Objects/Sequence_Item/slave_seq_item_pkg.sv
26 ./wrapper/Objects/Sequence_Item/wrapper_seq_item.sv
27
28 ./wrapper/Objects/Sequences/RAM_seq_pkg.sv
29 ./wrapper/Objects/Sequences/slave_seq_pkg.sv
30 ./wrapper/Objects/Sequences/wrapper_sequence_pkg.sv
31
32
33 ./wrapper/Top/Test/Enviroment_RAM/Agent/Driver/RAM_driver_pkg.sv
34 ./wrapper/Top/Test/Enviroment_SLAVE/Agent/Driver/slave_driver_pkg.sv
35 ./wrapper/Top/Test/Enviroment_Wrapper/Agent/Driver/wrapper_driver_pkg.sv
36
37 ./wrapper/Top/Test/Enviroment_RAM/Agent/Monitor/RAM_monitor_pkg.sv
38 ./wrapper/Top/Test/Enviroment_SLAVE/Agent/Monitor/slave_monitor_pkg.sv
39 ./wrapper/Top/Test/Enviroment_Wrapper/Agent/Monitor/wrapper_monitor_pkg.sv
40
41 ./wrapper/Top/Test/Enviroment_RAM/Agent/Sequencer/RAM_sequencer_pkg.sv
42 ./wrapper/Top/Test/Enviroment_SLAVE/Agent/Sequencer/slave_sequencer_pkg.sv
43 ./wrapper/Top/Test/Enviroment_Wrapper/Agent/Sequencer/wrapper_sequencer_pkg.sv
44
45 ./wrapper/Top/Test/Enviroment_RAM/Agent/RAM_agent_pkg.sv
46 ./wrapper/Top/Test/Enviroment_SLAVE/Agent/slave_agent_pkg.sv
47 ./wrapper/Top/Test/Enviroment_Wrapper/Agent/wrapper_agent_pkg.sv
48
49 ./wrapper/Top/Test/Enviroment_RAM/Coverage_Collector/RAM_coverage_pkg.sv
50 ./wrapper/Top/Test/Enviroment_SLAVE/Coverage_Collector/slave_coverage_pkg.sv
51 ./wrapper/Top/Test/Enviroment_Wrapper/Coverage_Collector/wrapper_coverage_pkg.sv
52
53 ./wrapper/Top/Test/Enviroment_RAM/Scoreboard/RAM_scoreboard_pkg.sv
54 ./wrapper/Top/Test/Enviroment_SLAVE/Scoreboard/slave_scoreboard_pkg.sv
55 ./wrapper/Top/Test/Enviroment_Wrapper/Scoreboard/wrapper_scoreboard_pkg.sv
56
57
58 ./wrapper/Top/Test/Enviroment_RAM/RAM_env_pkg.sv
59 ./wrapper/Top/Test/Enviroment_SLAVE/slave_env_pkg.sv
60
61 ./wrapper/Top/Test/wrapper_test_pkg.sv
62 ./wrapper/Top(wrapper_top.sv
63

```

Reports: -

Questasim snippets:



Report snippets:

```

# UVM_INFO @ 0: reporter [RNTST] Running test wrapper_test...
# UVM_INFO .../wrapper/Top/Test/wrapper_test_pkg.sv(63) @ 0: uvm_test_top [run_phase] Reset Asserted
# UVM_INFO .../wrapper/Top/Test/wrapper_test_pkg.sv(65) @ 2: uvm_test_top [run_phase] Reset Deasserted
# UVM_INFO .../wrapper/Top/Test/wrapper_test_pkg.sv(67) @ 2: uvm_test_top [run_phase] Main Seq Started
# UVM_INFO .../wrapper/Top/Test/wrapper_test_pkg.sv(69) @ 100002: uvm_test_top [run_phase] Main Seq Ended
# UVM_INFO .../wrapper/Top/Test/wrapper_test_pkg.sv(71) @ 100002: uvm_test_top [run_phase] Finish Test
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(126) @ 100002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO .../wrapper/Top/Test/Enviroment_RAM/Scoreboard/RAM_scoreboard_pkg.sv(93) @ 100002: uvm_test_top.env.RAM_sb [report_phase] Total successful transactions: 50001
# UVM_INFO .../wrapper/Top/Test/Enviroment_RAM/Scoreboard/RAM_scoreboard_pkg.sv(94) @ 100002: uvm_test_top.env.RAM_sb [report_phase] Total failed transactions: 0
# UVM_INFO .../wrapper/Top/Test/Enviroment_SLAVE/Scoreboard/slave_scoreboard_pkg.sv(66) @ 100002: uvm_test_top.env.slave_sb [report_phase] Total successful[RX]: 50001
# UVM_INFO .../wrapper/Top/Test/Enviroment_SLAVE/Scoreboard/slave_scoreboard_pkg.sv(67) @ 100002: uvm_test_top.env.slave_sb [report_phase] Total failed[RX]: 0
# UVM_INFO .../wrapper/Top/Test/Enviroment_SLAVE/Scoreboard/slave_scoreboard_pkg.sv(68) @ 100002: uvm_test_top.env.slave_sb [report_phase] Total successful[MOSO]: 50001
# UVM_INFO .../wrapper/Top/Test/Enviroment_SLAVE/Scoreboard/slave_scoreboard_pkg.sv(69) @ 100002: uvm_test_top.env.slave_sb [report_phase] Total failed[MOSO]: 0
# UVM_INFO .../wrapper/Top/Test/Enviroment_Wrapper/Scoreboard/wrapper_scoreboard_pkg.sv(55) @ 100002: uvm_test_top.env.wrapper_sb [report_phase] Total successful[MOSO]: 50001
# UVM_INFO .../wrapper/Top/Test/Enviroment_Wrapper/Scoreboard/wrapper_scoreboard_pkg.sv(56) @ 100002: uvm_test_top.env.wrapper_sb [report_phase] Total failed[MOSO]: 0
#
# --- UVM Report Summary ---
#
# *** Report counts by severity
# UVM_INFO : 17
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# *** Report counts by id
# [Questas UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [report_phase] 8
# [run_phase] 5
# *** Note: $finish      : D:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 100000 ns Iteration: 61 Instance: /top

```

Sequential Domain Coverage report:

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	As
△ /uvm_pkg:uvm_reg_map::do_write/ublk#(215181159#1731)immed_1735	Immediate	SVA	on	0	0	-	-	-	-	off	as	
△ /uvm_pkg:uvm_reg_map::do_read/ublk#(215181159#1771)immed_1775	Immediate	SVA	on	0	0	-	-	-	-	off	as	
△ /wrapper_sequence_pseq::wrapp_main_seq::body/ublk#(176326455#48)imme... + top/DUT_WRAPPER/sp/SLAVE_SVA/rst_count_rx_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/rst_count_tx_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/rst_rd_addr_signal_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/rst_cs_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/idle_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/check_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/write_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/read_addr_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/read_data_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/rd_flag_ra_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/sp/SLAVE_SVA/rd_flag_rf_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/rst_yr_addr_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/rst_rd_addr_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/rst_tx_valid_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/wr_addr_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/wr_data_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/wr_d_addr_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/wr_data_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/ram/RAM_SVA/wr_tx_valid_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/rst_be_valid_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/rst_rx_valid_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/rst_tx_valid_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/rst_rx_data_assert	Immediate	SVA	on	0	1	-	-	-	-	off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/rx_valid_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/tx_valid_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	
+ top/DUT_WRAPPER/WRAPPER_SVA/rx_data_assert	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	as	

Name	Language	Enabled	Log	Count	AtLeastLimit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory	Time	Cumulative Thr
top/DUT_WRAPPER/spl/SLAVE_SVA/rst_count_rx_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/rst_count_tx_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/rst_rd_addr_signal_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/rst_cs_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/rdel_cover	SVA	✓	Off	3062	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/check_cover	SVA	✓	Off	3513	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/write_cover	SVA	✓	Off	1763	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/read_addr_cover	SVA	✓	Off	960	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/read_dat_cover	SVA	✓	Off	762	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/rd_flag_a_cover	SVA	✓	Off	860	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/spl/SLAVE_SVA/rd_flag_rd_cover	SVA	✓	Off	620	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_wdr_addr_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_rd_addr_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_tx_valid_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_rx_valid_cover	SVA	✓	Off	922	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/wr_data_cover	SVA	✓	Off	660	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rd_data_cover	SVA	✓	Off	860	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_rd_data_cover	SVA	✓	Off	6455	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_tx_valid_cover	SVA	✓	Off	6455	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_rx_valid_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_tx_rx_val Cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_rx_tx_data_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rst_rst_rx_data_cover	SVA	✓	Off	479	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rx_valid_cover	SVA	✓	Off	2745	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/bx_valid_cover	SVA	✓	Off	678	1 Unlimited	1	100%		✓	0	0	0	0 ns	
top/DUT_WRAPPER/ram/RAM_SVA/rx_data_cover	SVA	✓	Off	2745	1 Unlimited	1	100%		✓	0	0	0	0 ns	

Coverage groups:

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment	
RAM_coverage_pkg/RAM_coverage		100.00%								
TxPE covgrp		100.00%	100	100.00...		✓				auto(1)
CVP covgrp::rst_cp		100.00%	485	1 100.00...		✓				
[B] bin rst_active			49516	1 100.00...		✓				
[B] bin rst_inactive			9093	1 100.00...		✓				
CVP covgrp::rx_valid_cp		100.00%	40908	1 100.00...		✓				
[B] bin rx_active			14254	1 100.00...		✓				
[B] bin rx_inactive			8931	1 100.00...		✓				
CVP covgrp::din_addr_cp		100.00%	11873	1 Read...		✓				
CVP covgrp::din_option_cp		100.00%	14943	1 100.00...		✓				
[B] bin Write_add			947	1 100.00...		✓				
[B] bin Write_data			678	1 100.00...		✓				
[B] bin Read_add			881	1 100.00...		✓				
[B] bin Read_data			1264	1 100.00...		✓				
CROSS covgrp::wr_addr_cr		100.00%	947	1 100.00...		✓				
[B] bin wr_addr_occur			678	1 100.00...		✓				
CROSS covgrp::wr_data_cr		100.00%	881	1 100.00...		✓				
[B] bin wr_data_occur			1264	1 100.00...		✓				
CROSS covgrp::rd_addr_cr		100.00%	947	1 100.00...		✓				
[B] bin rd_addr_occur			678	1 100.00...		✓				
CROSS covgrp::rd_data_cr		100.00%	881	1 100.00...		✓				
[B] bin rd_data_occur			1264	1 100.00...		✓				
/RAM_coverage_pkg/RAM_coverage		100.00%								
/slave_coverage_pkg/slave_coverage		100.00%								
TxPE covgrp		100.00%	485	1 100.00...		✓				auto(1)
CVP covgrp::rst_cp		100.00%	49516	1 100.00...		✓				
[B] bin active			947	1 100.00...		✓				
[B] bin inactive			678	1 100.00...		✓				
CVP covgrp::cs_cov_cp		100.00%	881	1 100.00...		✓				
[B] bin wr_add			1264	1 100.00...		✓				
[B] bin wr_data			947	1 100.00...		✓				
[B] bin rd_add			678	1 100.00...		✓				
[B] bin rd_data			1264	1 100.00...		✓				
/wrapper_coverage_pkg/wrapper_coverage		100.00%								
TxPE covgrp		100.00%	485	1 100.00...		✓				auto(1)
CVP covgrp::rst_cp		100.00%	49516	1 100.00...		✓				
[B] bin active			947	1 100.00...		✓				
[B] bin inactive			678	1 100.00...		✓				
CVP covgrp::cs_cov_cp		100.00%	881	1 100.00...		✓				
[B] bin wr_add			1264	1 100.00...		✓				
[B] bin wr_data			947	1 100.00...		✓				
[B] bin rd_add			678	1 100.00...		✓				
[B] bin rd_data			1264	1 100.00...		✓				

Codecoverage:

```
=====
== Instance: /\top#DUT_WRAPPER /spi
== Design Unit: work.SLAVE
=====

Branch Coverage:
  Enabled Coverage      Bins    Hits    Misses  Coverage
  -----      -----  -----  -----
  Branches          26      26      0  100.00%


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

Condition Coverage:
  Enabled Coverage      Bins   Covered    Misses  Coverage
  -----      -----  -----  -----
  Conditions         9        9      0  100.00%


=====Condition Details=====

Condition Coverage for instance /\top#DUT_WRAPPER /spi --


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 /\top#DUT_WRAPPER /spi --


  FSM Transitions      8        8      0  100.00%
Statement Coverage:
  Enabled Coverage      Bins    Hits    Misses  Coverage
  -----      -----  -----  -----
  Statements          36      36      0  100.00%


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

Statement Coverage for instance /\top#DUT_WRAPPER /spi --


  120      1      44102
Toggle Coverage:
  Enabled Coverage      Bins    Hits    Misses  Coverage
  -----      -----  -----  -----
  Toggles          52      52      0  100.00%


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

Toggle Coverage for instance /\top#DUT_WRAPPER /spi --
```

```

Branch Coverage:
  Enabled Coverage      Bins    Hits    Misses   Coverage
  -----      -----  -----  -----  -----
  Branches          6       6       0     100.00%
=====
Branch Coverage for instance /\top#DUT_WRAPPER /ram
.

Statement Coverage:
  Enabled Coverage      Bins    Hits    Misses   Coverage
  -----      -----  -----  -----  -----
  Statements        10      10       0     100.00%
=====
Statement Coverage for instance /\top#DUT_WRAPPER /ram --
.

Toggle Coverage:
  Enabled Coverage      Bins    Hits    Misses   Coverage
  -----      -----  -----  -----  -----
  Toggles          76      76       0     100.00%
=====
Toggle Coverage for instance /\top#DUT_WRAPPER /ram --
.

=====
Node      14  \01      01  \14
=====
Toggle Coverage:
  Enabled Coverage      Bins    Hits    Misses   Coverage
  -----      -----  -----  -----  -----
  Toggles          50      50       0     100.00%
=====
Toggle Coverage for instance /\top#DUT_WRAPPER  --
.

  ...,\upper,lower,upper_low,upper_low,\...
  0           1
Total Coverage By Instance (filtered view): 100.00% ←

```

Note: I excluded the statements and conditions that will not be implemented.