•RTL RAM VERIFICATION USING SYSTEMVERILOG AND UVM.

- **Under supervision of:** Eng. Sherif Hosny

Table of Contents

A class-based UVM Verification for synchronous RAM.

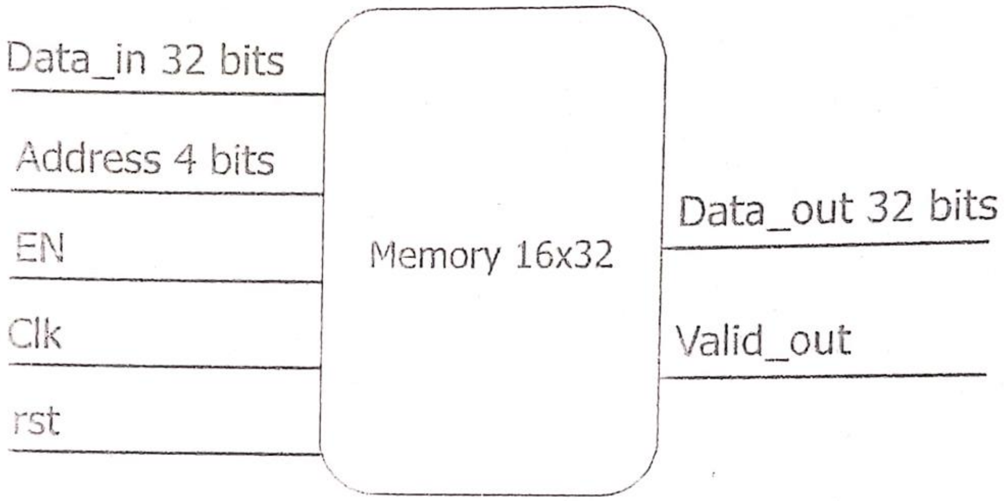
RAM design:

The design is memory-based circuit that stores data in writing case and out the data in the reading case. It is synchronized with a clock edge and has asynchronous reset signal. The design help management between different parts of a system.

RAM design specifications:

specification	Details
Inputs	- Data_in :32-bits input data signal. - Address :4-bits input address signal. - EN : 1-bit input enable signal. - CLK :1-bit clk signal. - RST :1-bit synchronize active high reset signal. - W_R :1-bit read /write signal (0 for write, 1 for read).
Outputs	-Data_out :32-bits output data signal. -valid_out :1-bit output refers to new data out.
Behavior	**-Reset**: it is asynchronize reset so on high level reset all locations on the memory should be equal zero. **-Read operation:** in this case memory should out data on output.in this case rst signal should be =0, EN signal should be =1 and W_R = 1 for work on read mode. **-write operation:** in this case memory should monitor the input data and save it on the input address.in this case rst signal should be =0, EN signal should be =1 and W_R = 0 for work on write mode.

RTL design Architecture.



Note: the following RTL code was made by AI copilot tool.

RTL design Modules:

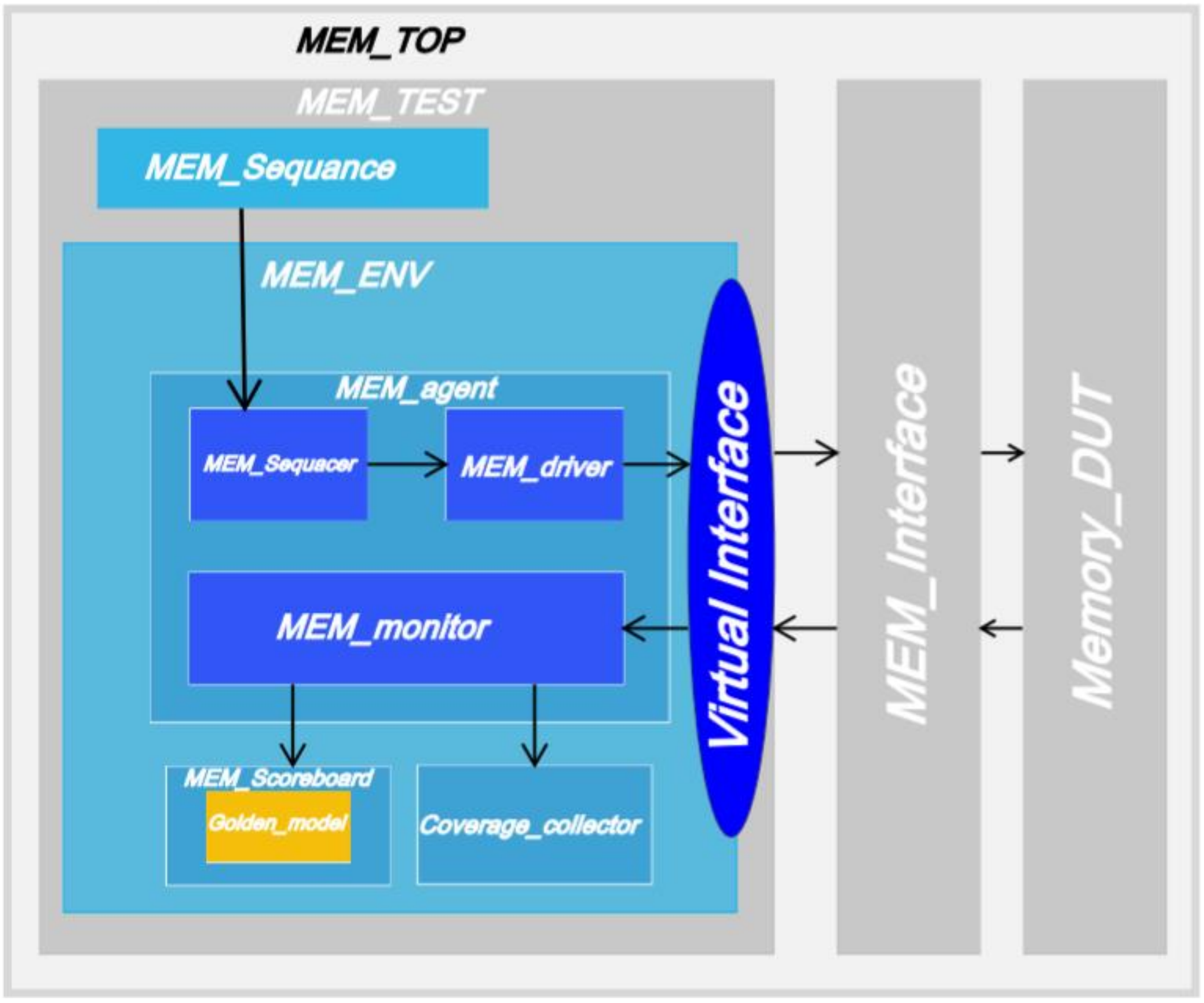
- Memory.v

```
module Memory (
    input [31:0] Data_in,
    input [3:0] Address,
    input EN,
    input CLK,
    input RST,
    input W_R, // 0 for write, 1 for read
    output reg [31:0] Data_out,
    output reg valid_out
);
    //declare index
    reg [4:0] i;
    // Declare memory array of size 16x32
    reg [31:0] memory [0:15];

    always @(posedge CLK or posedge RST) begin
        if (RST) begin
            // Reset memory and outputs
            Data_out <= 32'b0;
            valid_out <= 1'b0; //do that for can monitor the response of the memory
            for (i = 0; i < 16; i = i + 1) begin
                memory[i] <= 32'b0;
            end
        end else if (EN) begin
            if (W_R == 0) begin
                // Write to memory
                memory[Address] <= Data_in;
                valid_out <= 1'b0; // No valid output on write
            end
             else begin
                // Read from memory
                Data_out <= memory[Address];
                valid_out <= 1'b1; // Valid output on read
            end
        end else begin
            valid_out <= 1'b0;
        end
    end

endmodule
```

test plan:

Feature	Check list	Stimulus
Reset behavior	-All memory locations should be equal zero. -The memory does not have response about all inputs in case rst signal high. -in case rst signal is low we can test other features.	**Rst :0->1->0**
Write all memory location	-in this check we should write all memory locations.	**Rst =0** **EN=1** **W_R=0** **-other inputs randomize them.**
Read all memory location	-in this check we should read all memory locations that we write it from the previous scenario. -check the values are right as we write in the previous scenario.	**Rst =0** **EN=1** **W_R=1** **-other inputs randomize them.**
Randomize all inputs and check the correction of the outputs (in this case we check all feature with random order)	-in this case we will check No error has been found From check the output data	**All inputs randomize**

environment:



Parts of environment code:

- Top Module.

```
`timescale 1ns/1ps
`include"MEM_INTERFACE.sv"
import uvm_pkg::*;
import MEM_Package::*;

module  MEM_TOP;
parameter clk_period =10;
bit clk;
//interface instantiation
MEM_INTERFACE intf(clk);
```

```
//dut instantiation
Memory  mem_dut(.CLK(clk),
                .Data_in(intf.Data_in),
                .Data_out(intf.Data_out),
                .Address(intf.Address),
                .EN(intf.EN),
                .RST(intf.RST),
                .W_R(intf.W_R),
                .valid_out(intf.valid_out)
                );

//clk triggering
initial begin
    clk=0;
    forever
    begin
        #(clk_period/2) clk=~clk;
    end
end

initial begin
    uvm_config_db#(virtual
MEM_INTERFACE)::set(null,"uvm_test_top","top2tast",intf);
    run_test("MEM_TEST");
end
endmodule
```

- Pack file

```
package MEM_Package;
import uvm_pkg::*;
parameter size_of_memory_location=16;
 `include"uvm_macros.svh"
 `include "MEM_seq_item.sv"
 `include "MEM_Sequancer.sv"
 `include "MEM_Driver.sv"
 `include "MEM_Monitor.sv"
 `include "MEM_Agent.sv"
 `include "MEM_Scoreboard.sv"
 `include "MEM_coverage_collector.sv"
 `include "MEM_Sequance.sv"
 `include "MEM_ENV.sv"
 `include "MEM_TEST.sv"
endpackage
```

- AES_TEST

```
class MEM_TEST extends uvm_test;
`uvm_component_utils(MEM_TEST)
// Constructor
function new(string name = "MEM_TEST", uvm_component parent = null);
super.new(name, parent);
endfunction: new

//declare the environment and sequence
MEM_env my_env;
MEM_Sequence my_seq;

//declare the interface
virtual MEM_INTERFACE vif;

//declare the build function
function void build_phase(uvm_phase phase);
super.build_phase(phase);
my_env = MEM_env::type_id::create("my_env", this);
my_seq = MEM_Sequence::type_id::create("my_seq", this);

if (
    !uvm_config_db #(virtual MEM_INTERFACE)::get(this,"", "top2tast", vif)
    ) begin
        `uvm_fatal(get_full_name(), "[MEM_TEST] vif not get");
end
    uvm_config_db #(virtual MEM_INTERFACE)::set(this, "my_env", "test2env",
vif);
endfunction: build_phase

//declare the connect function
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);

endfunction: connect_phase

// Task: run_phase
task run_phase(uvm_phase phase);
super.run_phase(phase);
phase.raise_objection(this);
$display("Starting the test");
my_seq.start(my_env.my_agent.m_sequencer);
phase.drop_objection(this);
$display("Ending the test");
```

```
endtask: run_phase

endclass: MEM_TEST
```

- Sequence class

```
class MEM_Sequence extends uvm_sequence;
`uvm_object_utils(MEM_Sequence)

//*************************************//
///**declare my_trans object**///
MEM_Transaction my_trans;

//*************************************//
///**Constructor**///
extern function new(string name = "MEM_Sequence");

//*************************************//
/////////****body task****/////////
extern task body();

//*************************************//
///**Task for Test RST at first time**///
extern task RST();

//**********************************************//
////****read_n_number****////
extern task read_n_number(input int num_to_read);

//**********************************************//
////****write_n_number****////
extern task write_n_number(input int num_to_write);

//**********************************************//
////****rand_task****////
extern task rand_task(input int num_to_rand);
endclass: MEM_Sequence


//////////////////////////////
/////**** Constructor****/////
function MEM_Sequence::new(string name = "MEM_Sequence");
    super.new(name);
    endfunction: new
```

```
///////////////////////////////
/////////****body****/////////
 task MEM_Sequence::body();
 int num_2_rand;

///**test rst at first time **///
    #10;
    RST();
///**test read all memory sequencially**///
read_n_number(size_of_memory_location);
//test write all memory sequencially
write_n_number(size_of_memory_location);
//test read all memory sequencially
read_n_number(size_of_memory_location);
//rand task to test the randomization of the transaction
num_2_rand =$urandom_range(100000,50000);
rand_task(num_2_rand);
endtask: body


//*************************************//
///**Task for Test RST at first time**///
task MEM_Sequence::RST();
my_trans=MEM_Transaction::type_id::create("my_trans");
start_item(my_trans);
my_trans.RST.rand_mode(0);     //I put it as without it rst will be randomize on the next
step
my_trans.RST = 1;              //set RST to 1 and will not affect by the randomize
assert(my_trans.randomize());  //randomize the my_trans
$display("at time(%0t): [SQEUANCER] the randomized my_trans RST :%p",$realtime,my_trans);
finish_item(my_trans);

///////////////////////////////////////////////////////////////
///****this iteration to fall the RST with the same task****///
my_trans=MEM_Transaction::type_id::create("my_trans");
start_item(my_trans);
//my_trans.RST.rand_mode(0);     //I put it as without it rst will be randomize on the
next step
my_trans.RST = 0;              //set RST to 1 and will not affect by the randomize
my_trans.EN  = 0;
assert(my_trans.randomize(Address,Data_in,W_R));  //randomize the my_trans
$display("at time(%0t): [SQEUANCER] the randomized my_trans RST :%p",$realtime,my_trans);
finish_item(my_trans);
my_trans.RST.rand_mode(1);
 $display("at time(%0t):[SQEUANCER] Driver Done on RST check",$realtime);
```

```
endtask:RST


//*******************************************//
//*Task to write with certain number to task*//
task MEM_Sequence::write_n_number(input int num_to_write);
    int i =0;
    my_trans=MEM_Transaction::type_id::create("my_trans");
    repeat(num_to_write) begin
    start_item(my_trans);
    /*initial values for do the functionality of the task without the randomization*/
    my_trans.EN  = 1;
    my_trans.W_R = 0;
    my_trans.RST = 0;
    assert(my_trans.randomize(Address,Data_in));
    $display("[SQEUANCER] the randomized transaction on write_n_number task
:%p",my_trans);
    finish_item(my_trans);
    $display("[SQEUANCER] send to driver number : %0d on write_n_number task ",i);
    i++;
    end
    /*change the inputs after finish the required function from task*/
    my_trans=MEM_Transaction::type_id::create("my_trans");
    start_item(my_trans);
    my_trans.EN = 0;
    my_trans.W_R = 0;
    my_trans.RST=0;
    assert(my_trans.randomize(Address,Data_in));
    finish_item(my_trans);
endtask:write_n_number

//*******************************************//
//**Task to read with certain number to task**//
task MEM_Sequence::read_n_number(input int num_to_read);
int i =0;
        repeat(num_to_read) begin
        my_trans=MEM_Transaction::type_id::create("my_trans");
        start_item(my_trans);
        /*initial values for do the functionality of the task without the randomization*/
        my_trans.EN = 1;
        my_trans.W_R = 1;
        my_trans.RST=0;
        assert(my_trans.randomize(Address,Data_in));
        $display("at time(%0t): [SQEUANCER] send to driver no to read on read_n_number
task: %0d",$time,i);
```

```
       // if(my_trans.W_R)
        //$stop;
        finish_item(my_trans);
        $display("at time(%0t): [SQEUANCER] Driver Done****",$time);
        i++;
    end
    /****this iteration to fall the valid_out from the inputs****/
    my_trans=MEM_Transaction::type_id::create("my_trans");
    start_item(my_trans);
    my_trans.EN = 0;
    my_trans.W_R = 0;
    my_trans.RST=0;
    assert(my_trans.randomize(Address,Data_in));
    finish_item(my_trans);
endtask:read_n_number

 task MEM_Sequence::rand_task(input int num_to_rand);

 for(int i=0;i<num_to_rand;i++) begin
    my_trans=MEM_Transaction::type_id::create("my_trans");
    start_item(my_trans);
    assert(my_trans.randomize());
    finish_item(my_trans);
 end

 /****this iteration to fall the valid_out from the inputs****/
    my_trans=MEM_Transaction::type_id::create("my_trans");
    start_item(my_trans);
    my_trans.EN = 0;
    my_trans.W_R = 0;
    my_trans.RST=0;
    assert(my_trans.randomize(Address,Data_in));
    finish_item(my_trans);
 endtask:rand_task
```

- <u>Env class</u>

```
class MEM_env extends uvm_env;
`uvm_component_utils(MEM_env)

// Constructor
function new(string name = "MEM_env", uvm_component parent = null);
super.new(name, parent);
endfunction: new
```

```
//declare the conponents
MEM_Agent my_agent;
MEM_Scoreboard my_scoreboard;
MEM_coverage_collector my_collector;

//declare the virtual interface
virtual MEM_INTERFACE my_vif;

//declare the build function
function void build_phase(uvm_phase phase);
super.build_phase(phase);
my_agent = MEM_Agent::type_id::create("my_agent", this);
my_scoreboard = MEM_Scoreboard::type_id::create("my_scoreboard", this);
my_collector = MEM_coverage_collector::type_id::create("my_collector", this);

if(
    !uvm_config_db #(virtual MEM_INTERFACE)::get(this, "", "test2env", my_vif)
  )
`uvm_fatal(get_full_name(), "[MEM_env] vif not get")
uvm_config_db #(virtual MEM_INTERFACE)::set(this, "my_agent", "env2agent", my_vif);
endfunction: build_phase

//declare the connect function
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
my_agent.m_monitor.port.connect(my_collector.analysis_export);
my_agent.m_monitor.port.connect(my_scoreboard.imp);
endfunction: connect_phase

// Task: run_phase
task run_phase(uvm_phase phase);
super.run_phase(phase);
endtask: run_phase
endclass: MEM_env
```

- <u>Agent class</u>

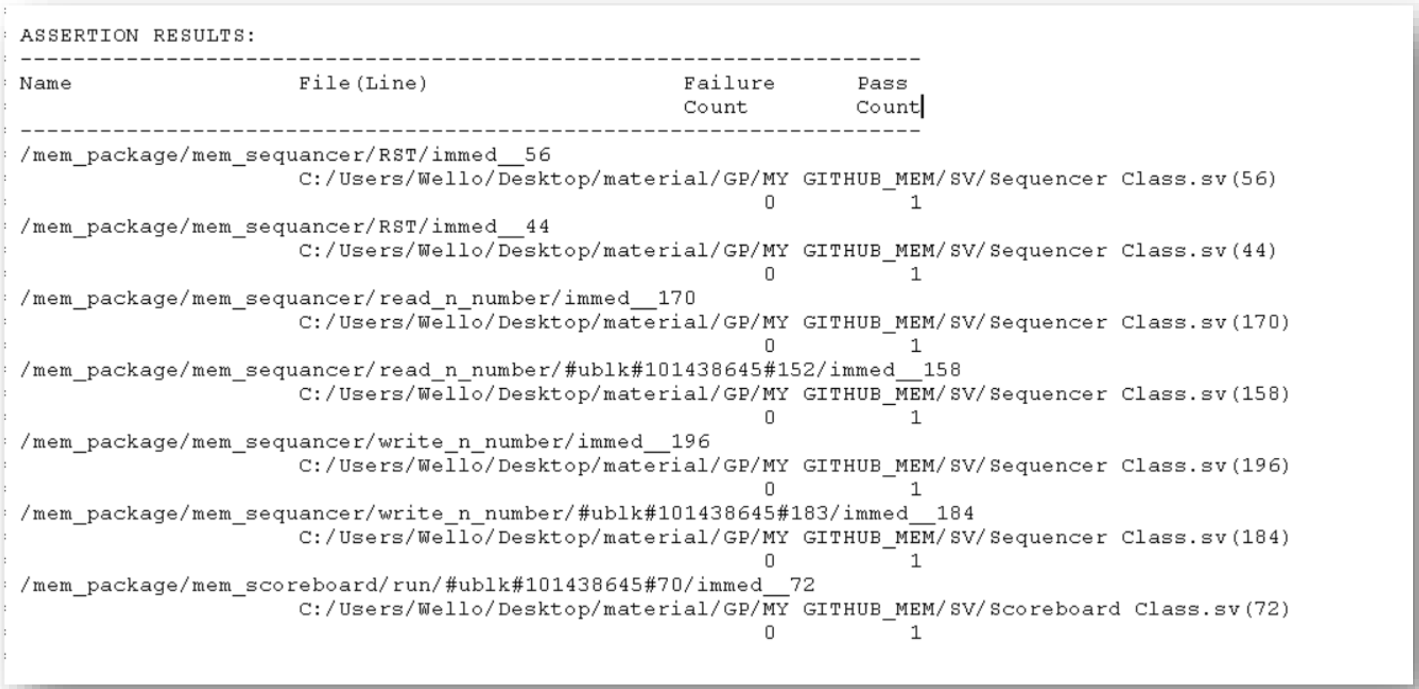
```
class MEM_Agent extends uvm_agent;
  `uvm_component_utils(MEM_Agent)
  // Components
  MEM_Driver m_driver;
  MEM_Sequencer m_sequencer;
  MEM_Monitor m_monitor;

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

  //interface
  virtual MEM_INTERFACE m_vif;

  // Build Phase
    function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    m_driver    = MEM_Driver::type_id::create("m_driver", this);
    m_sequencer = MEM_Sequencer::type_id::create("m_sequencer", this);
    m_monitor   = MEM_Monitor::type_id::create("m_monitor", this);


    if(
      !uvm_config_db #(virtual MEM_INTERFACE)::get(this, "", "env2agent", 
m_vif)
      )
    `uvm_fatal(get_full_name(), "[MEM_Agent] vif not get")
    uvm_config_db #(virtual MEM_INTERFACE)::set(this, "m_driver", 
"driver2agent", m_vif);
    uvm_config_db #(virtual MEM_INTERFACE)::set(this, "m_monitor", 
"monitor2agent", m_vif);
  endfunction: build_phase

  // Connect Phase
    function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    m_driver.seq_item_port.connect(m_sequencer.seq_item_export);
    endfunction: connect_phase

  // Task: run_phase
    task run_phase(uvm_phase phase);
    super.run_phase(phase);
    endtask: run_phase
endclass
```

- Coverage collector

```
class MEM_coverage_collector extends uvm_subscriber #(MEM_Transaction);
`uvm_component_utils(MEM_coverage_collector)

//transaction object
MEM_Transaction tran_mon2sub;

//coverage group
covergroup cov;
ENABLE: coverpoint tran_mon2sub.EN { bins H_EN={1'd1};
                                     bins L_EN={1'd0};
                                    }
RESET: coverpoint tran_mon2sub.RST{ bins H_RST={1'd1};
                                    bins L_RST={1'd0};
                                  }
TRANS_RST:coverpoint tran_mon2sub.RST{bins trans_H2L=(1'd1 => 1'd0);
                                  }
W_R: coverpoint tran_mon2sub.W_R{ bins H_W_R={1'd1};
                                  bins L_W_R={1'd0};
                                }
TRANS_W_R:coverpoint tran_mon2sub.W_R{
                                    bins trans_L2H=(1'd0 => 1'd1);
                                    }
ADDRESS: coverpoint tran_mon2sub.Address{bins low_add ={4'd0};
                                         bins med_add[]={[4'd1:4'd14]};
                                         bins high_add={4'd15};
                                        }
DATA_IN: coverpoint tran_mon2sub.Data_in{bins low_data  ={32'h0};
                                         bins med_data  ={[32'h1:32'hfffffffe]};
                                         bins high_data ={32'hffffffff};
                               }

check_check_W_R_all_addresses :cross ENABLE, W_R ,DATA_IN , ADDRESS ; //check read and
write on all addresses
check_rst :cross RESET, ENABLE, W_R, ADDRESS ,DATA_IN ;  //check rst is domenant on
all cases on the other values for variables
endgroup:cov

  // Constructor
  function new(string name = "MEM_coverage_collector", uvm_component parent = null);
  super.new(name, parent);
  cov  = new();
  endfunction: new
  //declare the build function
  function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  endfunction: build_phase

  //declare the connect function
  function void connect_phase(uvm_phase phase);
  super.connect_phase(phase);
  endfunction: connect_phase

  // Task: run_phase
  task run_phase(uvm_phase phase);
  super.run_phase(phase);
  endtask: run_phase
  //write function
  function void write(MEM_Transaction t);
  tran_mon2sub =t;
  if(tran_mon2sub.inp_trans ==1)
  cov.sample();
  endfunction: write

endclass
```

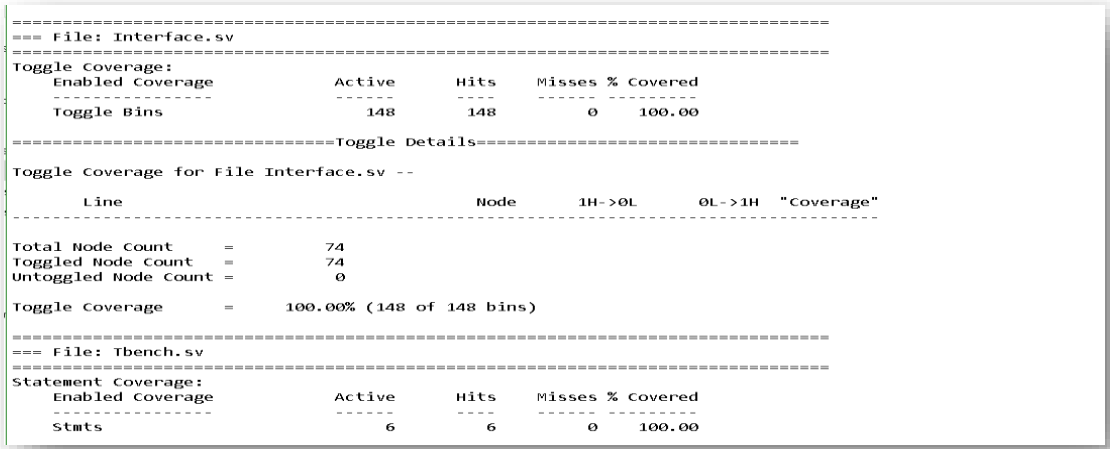
Results:

- Assertion coverage:

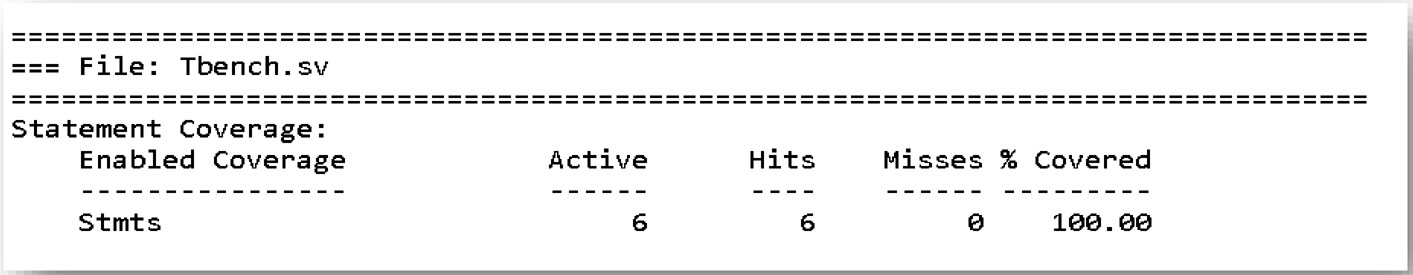


```
ASSERTION RESULTS:
--------------------------------------------------------------------
Name                    File(Line)                      Failure      Pass
                                                        Count        Count
--------------------------------------------------------------------
/mem_package/mem_sequancer/RST/immed__56
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(56)
                                                              0           1
/mem_package/mem_sequancer/RST/immed__44
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(44)
                                                              0           1
/mem_package/mem_sequancer/read_n_number/immed__170
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(170)
                                                              0           1
/mem_package/mem_sequancer/read_n_number/#ublk#101438645#152/immed__158
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(158)
                                                              0           1
/mem_package/mem_sequancer/write_n_number/immed__196
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(196)
                                                              0           1
/mem_package/mem_sequancer/write_n_number/#ublk#101438645#183/immed__184
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Sequencer Class.sv(184)
                                                              0           1
/mem_package/mem_scoreboard/run/#ublk#101438645#70/immed__72
                        C:/Users/Wello/Desktop/material/GP/MY GITHUB_MEM/SV/Scoreboard Class.sv(72)
                                                              0           1
```

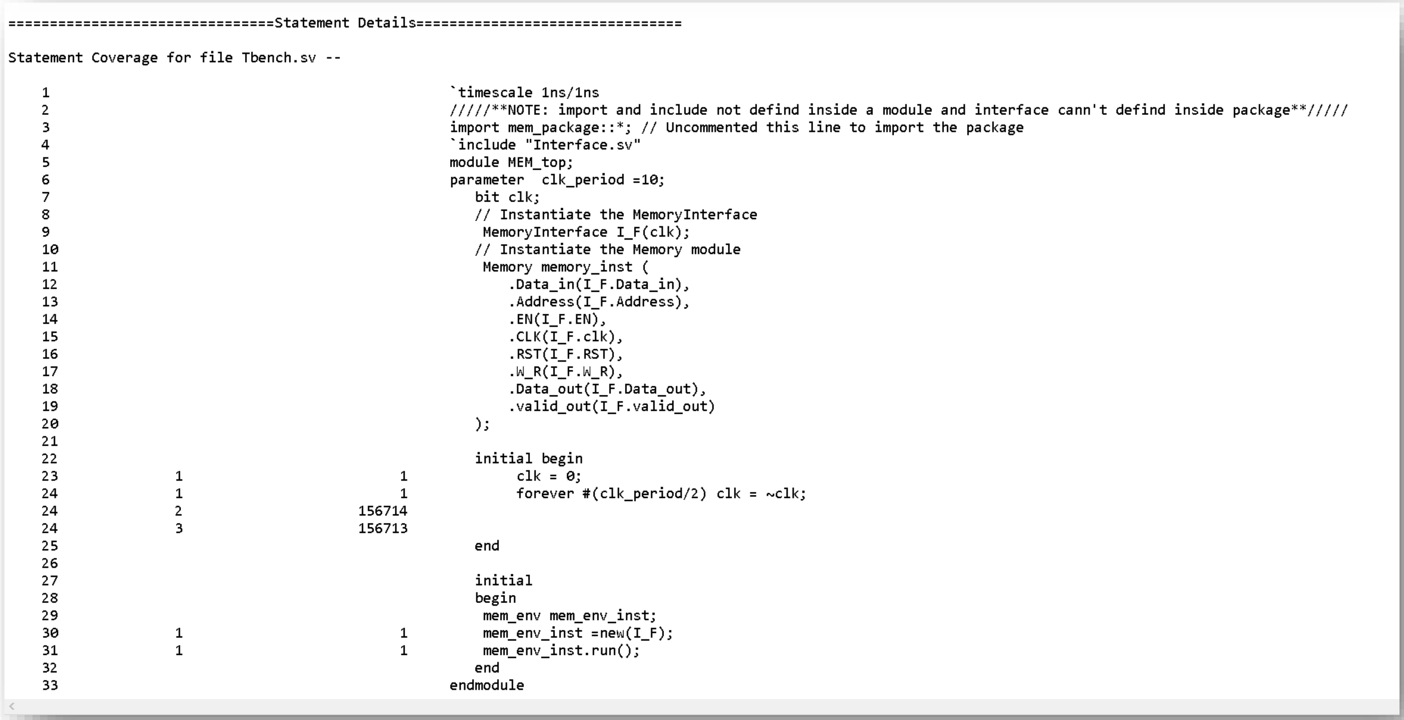
- Code coverage:



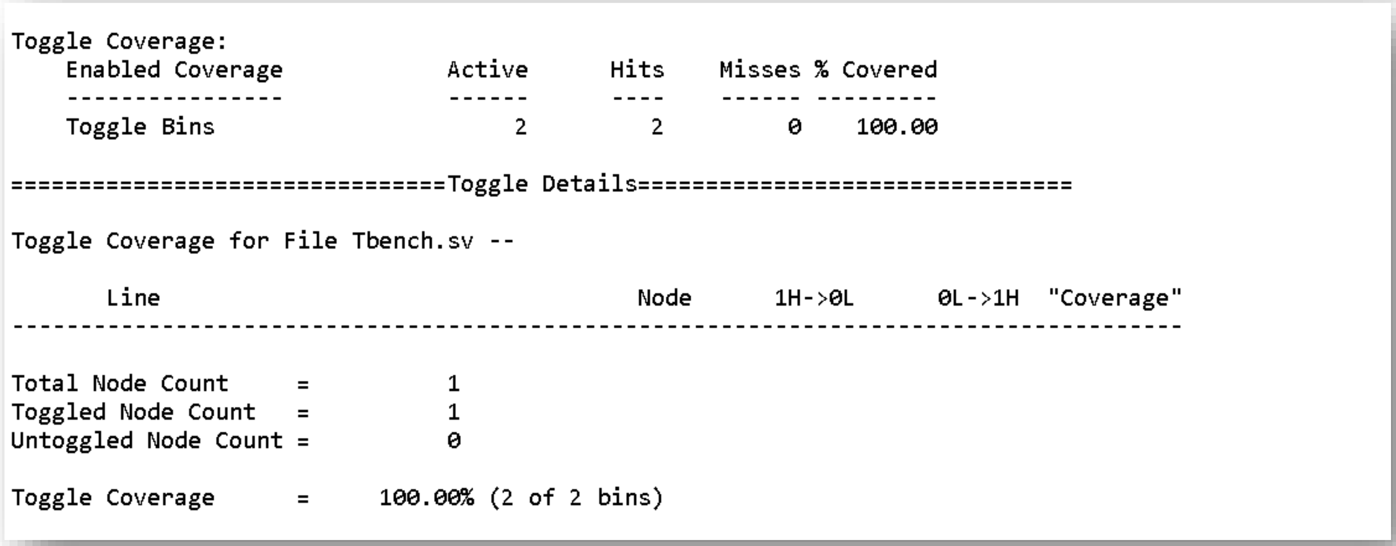
```
=====================================================================
=== File: Interface.sv
=====================================================================
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                      148       148         0    100.00

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

Toggle Coverage for File Interface.sv --

       Line                                   Node       1H->0L       0L->1H  "Coverage"
---------------------------------------------------------------------------------------

Total Node Count     =          74
Toggled Node Count   =          74
Untoggled Node Count =           0

Toggle Coverage      =     100.00% (148 of 148 bins)

=====================================================================
=== File: Tbench.sv
=====================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                              6         6         0    100.00
```



```
=================================================================================
=== File: Tbench.sv
=================================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                              6         6         0    100.00
```

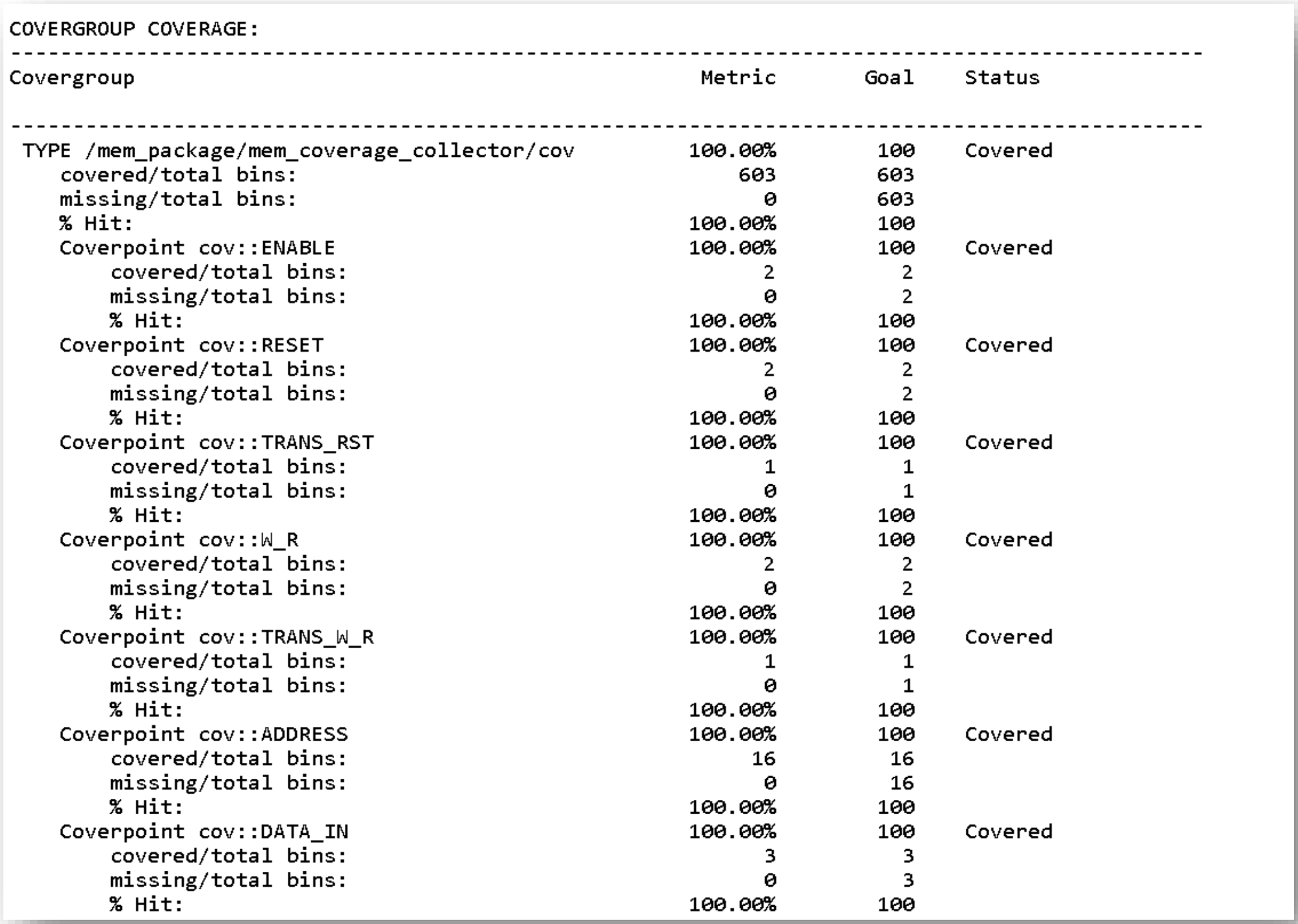


```
===============================Statement Details===============================

Statement Coverage for file Tbench.sv --

    1                                                 `timescale 1ns/1ns
    2                                                 /////**NOTE: import and include not defind inside a module and interface cann't defind inside package**/////
    3                                                 import mem_package::*; // Uncommented this line to import the package
    4                                                 `include "Interface.sv"
    5                                                 module MEM_top;
    6                                                 parameter  clk_period =10;
    7                                                    bit clk;
    8                                                    // Instantiate the MemoryInterface
    9                                                     MemoryInterface I_F(clk);
    10                                                   // Instantiate the Memory module
    11                                                    Memory memory_inst (
    12                                                        .Data_in(I_F.Data_in),
    13                                                        .Address(I_F.Address),
    14                                                        .EN(I_F.EN),
    15                                                        .CLK(I_F.clk),
    16                                                        .RST(I_F.RST),
    17                                                        .W_R(I_F.W_R),
    18                                                        .Data_out(I_F.Data_out),
    19                                                        .valid_out(I_F.valid_out)
    20                                                   );
    21
    22                                                   initial begin
    23               1                          1             clk = 0;
    24               1                          1             forever #(clk_period/2) clk = ~clk;
    24               2                     156714
    24               3                     156713
    25                                                   end
    26
    27                                                   initial
    28                                                   begin
    29                                                    mem_env mem_env_inst;
    30               1                          1         mem_env_inst =new(I_F);
    31               1                          1         mem_env_inst.run();
    32                                                   end
    33                                                endmodule
```

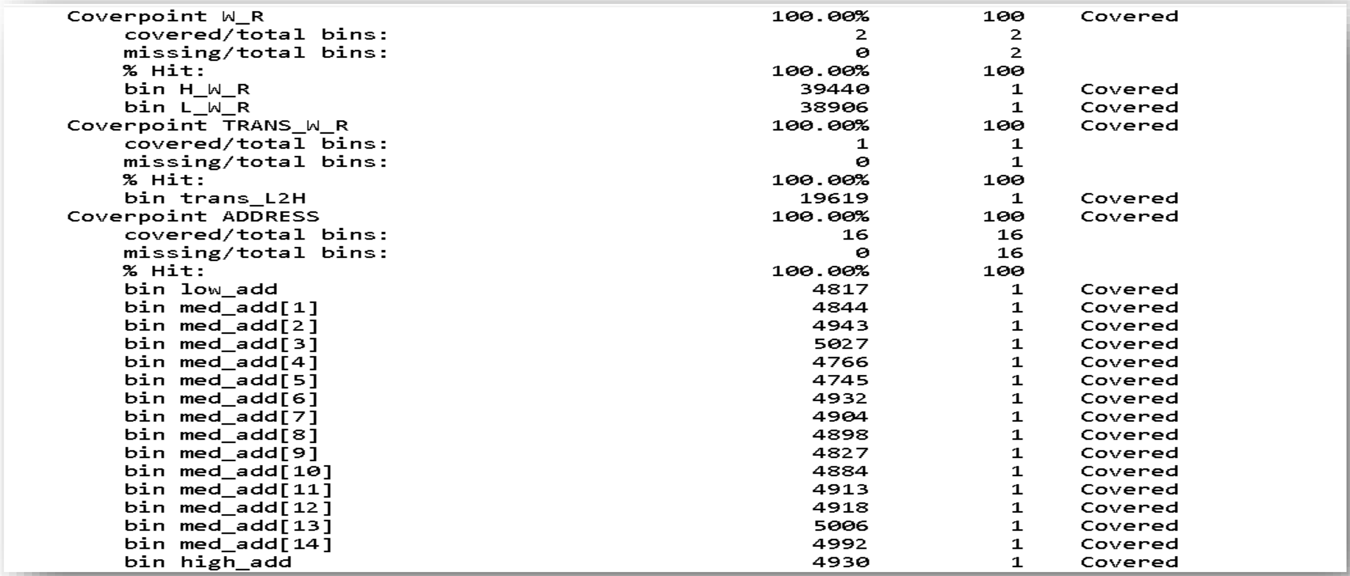


```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                        2         2         0    100.00

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

Toggle Coverage for File Tbench.sv --

       Line                                   Node      1H->0L      0L->1H  "Coverage"
---------------------------------------------------------------------------------------

Total Node Count     =          1
Toggled Node Count   =          1
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (2 of 2 bins)
```

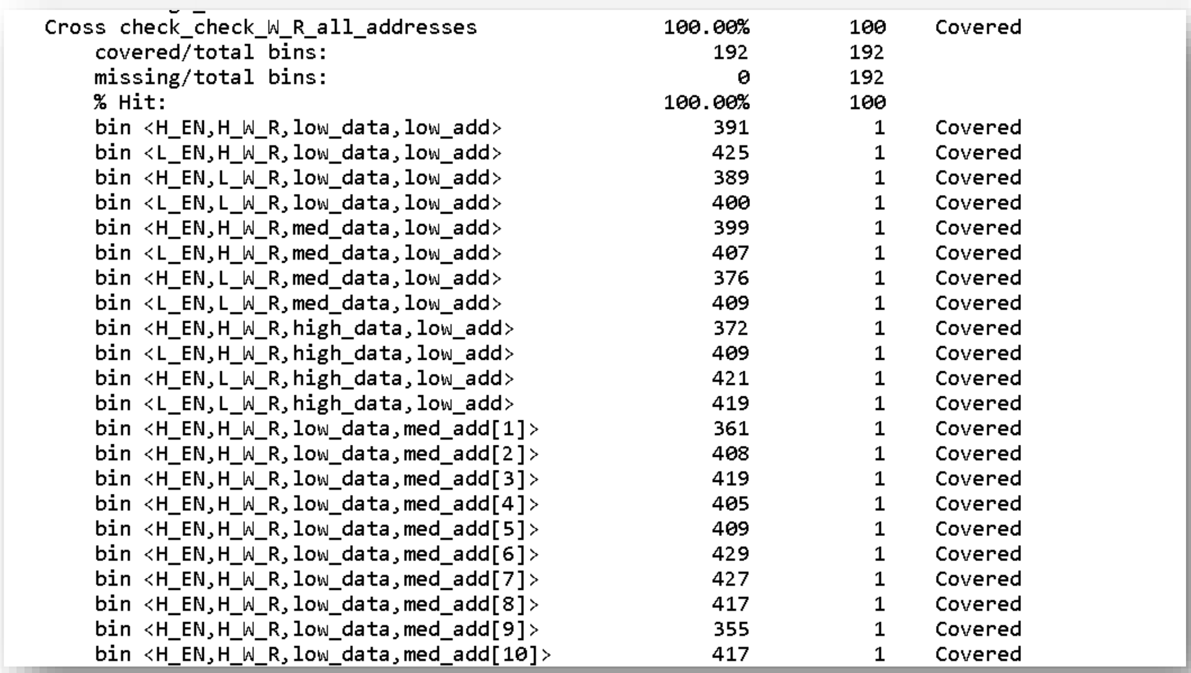
- <u>Function coverage:</u>



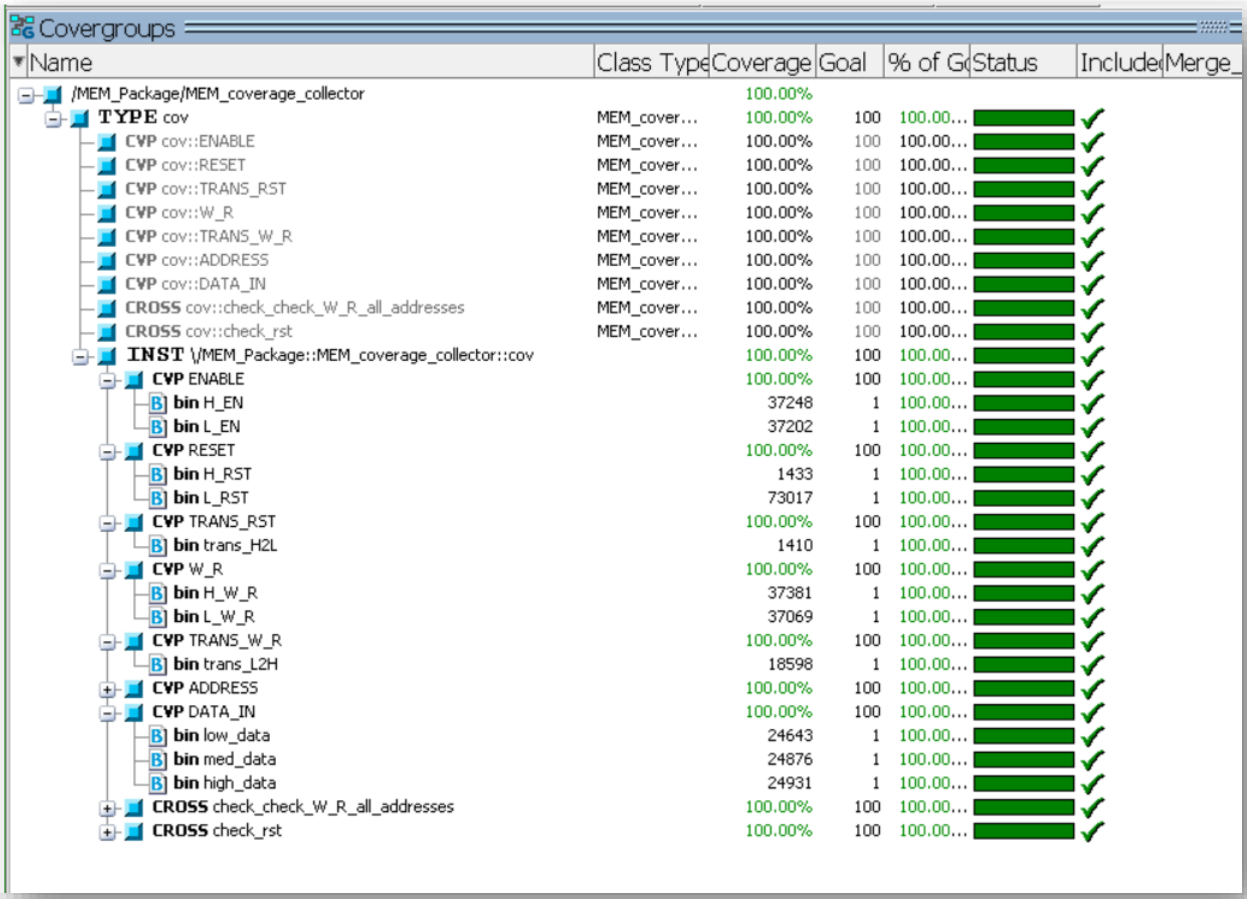
```
COVERGROUP COVERAGE:
-----------------------------------------------------------------------------------------------
Covergroup                                             Metric       Goal    Status

-----------------------------------------------------------------------------------------------
 TYPE /mem_package/mem_coverage_collector/cov         100.00%        100    Covered
    covered/total bins:                                   603        603
    missing/total bins:                                     0        603
    % Hit:                                            100.00%        100
    Coverpoint cov::ENABLE                            100.00%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                        100.00%        100
    Coverpoint cov::RESET                             100.00%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                        100.00%        100
    Coverpoint cov::TRANS_RST                         100.00%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                        100.00%        100
    Coverpoint cov::W_R                               100.00%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                        100.00%        100
    Coverpoint cov::TRANS_W_R                         100.00%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                        100.00%        100
    Coverpoint cov::ADDRESS                           100.00%        100    Covered
        covered/total bins:                                16         16
        missing/total bins:                                 0         16
        % Hit:                                        100.00%        100
    Coverpoint cov::DATA_IN                           100.00%        100    Covered
        covered/total bins:                                 3          3
        missing/total bins:                                 0          3
        % Hit:                                        100.00%        100
```



```
    Coverpoint W_R                                    100.00%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                        100.00%        100
        bin H_W_R                                       39440          1    Covered
        bin L_W_R                                       38906          1    Covered
    Coverpoint TRANS_W_R                              100.00%        100    Covered
        covered/total bins:                                 1          1
        missing/total bins:                                 0          1
        % Hit:                                        100.00%        100
        bin trans_L2H                                   19619          1    Covered
    Coverpoint ADDRESS                                100.00%        100    Covered
        covered/total bins:                                16         16
        missing/total bins:                                 0         16
        % Hit:                                        100.00%        100
        bin low_add                                      4817          1    Covered
        bin med_add[1]                                   4844          1    Covered
        bin med_add[2]                                   4943          1    Covered
        bin med_add[3]                                   5027          1    Covered
        bin med_add[4]                                   4766          1    Covered
        bin med_add[5]                                   4745          1    Covered
        bin med_add[6]                                   4932          1    Covered
        bin med_add[7]                                   4904          1    Covered
        bin med_add[8]                                   4898          1    Covered
        bin med_add[9]                                   4827          1    Covered
        bin med_add[10]                                  4884          1    Covered
        bin med_add[11]                                  4913          1    Covered
        bin med_add[12]                                  4918          1    Covered
        bin med_add[13]                                  5006          1    Covered
        bin med_add[14]                                  4992          1    Covered
        bin high_add                                     4930          1    Covered
```



```
Cross check_check_W_R_all_addresses            100.00%      100    Covered
    covered/total bins:                            192      192
    missing/total bins:                              0      192
    % Hit:                                     100.00%      100
    bin <H_EN,H_W_R,low_data,low_add>              391        1    Covered
    bin <L_EN,H_W_R,low_data,low_add>              425        1    Covered
    bin <H_EN,L_W_R,low_data,low_add>              389        1    Covered
    bin <L_EN,L_W_R,low_data,low_add>              400        1    Covered
    bin <H_EN,H_W_R,med_data,low_add>              399        1    Covered
    bin <L_EN,H_W_R,med_data,low_add>              407        1    Covered
    bin <H_EN,L_W_R,med_data,low_add>              376        1    Covered
    bin <L_EN,L_W_R,med_data,low_add>              409        1    Covered
    bin <H_EN,H_W_R,high_data,low_add>             372        1    Covered
    bin <L_EN,H_W_R,high_data,low_add>             409        1    Covered
    bin <H_EN,L_W_R,high_data,low_add>             421        1    Covered
    bin <L_EN,L_W_R,high_data,low_add>             419        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[1]>           361        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[2]>           408        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[3]>           419        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[4]>           405        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[5]>           409        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[6]>           429        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[7]>           427        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[8]>           417        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[9]>           355        1    Covered
    bin <H_EN,H_W_R,low_data,med_add[10]>          417        1    Covered
```



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_
/MEM_Package/MEM_coverage_collector		100.00%					
TYPE cov	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::ENABLE	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::RESET	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::TRANS_RST	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::W_R	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::TRANS_W_R	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::ADDRESS	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CVP cov::DATA_IN	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CROSS cov::check_check_W_R_all_addresses	MEM_cover...	100.00%	100	100.00...	🟩	✓	
CROSS cov::check_rst	MEM_cover...	100.00%	100	100.00...	🟩	✓	
INST \/MEM_Package::MEM_coverage_collector::cov		100.00%	100	100.00...	🟩	✓	
CVP ENABLE		100.00%	100	100.00...	🟩	✓	
bin H_EN		37248	1	100.00...	🟩	✓	
bin L_EN		37202	1	100.00...	🟩	✓	
CVP RESET		100.00%	100	100.00...	🟩	✓	
bin H_RST		1433	1	100.00...	🟩	✓	
bin L_RST		73017	1	100.00...	🟩	✓	
CVP TRANS_RST		100.00%	100	100.00...	🟩	✓	
bin trans_H2L		1410	1	100.00...	🟩	✓	
CVP W_R		100.00%	100	100.00...	🟩	✓	
bin H_W_R		37381	1	100.00...	🟩	✓	
bin L_W_R		37069	1	100.00...	🟩	✓	
CVP TRANS_W_R		100.00%	100	100.00...	🟩	✓	
bin trans_L2H		18598	1	100.00...	🟩	✓	
CVP ADDRESS		100.00%	100	100.00...	🟩	✓	
CVP DATA_IN		100.00%	100	100.00...	🟩	✓	
bin low_data		24643	1	100.00...	🟩	✓	
bin med_data		24876	1	100.00...	🟩	✓	
bin high_data		24931	1	100.00...	🟩	✓	
CROSS check_check_W_R_all_addresses		100.00%	100	100.00...	🟩	✓	
CROSS check_rst		100.00%	100	100.00...	🟩	✓	