Part (1): RAM :

(A) SEQUENCE ITEM USING THE CONSTRAINTS :

```
package squence_item_pkg; import uvm_pkg::*;
`include "uvm_macros.svh" class ram_seq_item
extends uvm_sequence_item;
`uvm_object_utils(ram_seq_item) rand bit [9:0] din; rand bit
rx_valid,rst_n; bit [7:0] dout; bit tx_valid; function  new(string name
="ram_seq_item"); super.new(name); endfunction     function string
convert2string();     return $sformatf("%s rst_n=0b%0b din=0b%0b rx_valid
=0b%0b dout =0b%0b  tx_valid=0b%0b
",super.convert2string(),rst_n,din,rx_valid,dout,tx_valid); endfunction
 function string convert2string_stimulus();     return
$sformatf("rst_n=0b%0b din=0b%0b rx_valid =0b%0b
",rst_n,din,rx_valid);
endfunction
    constraint write_only{
rx_valid dist {1:=90,0:=10};
rst_n dist {1:=98,0:=2};
din[9:8] inside {2'b00,2'b01};
    }         constraint
read_only{ rx_valid dist
{1:=90,0:=10}; rst_n dist
{1:=98,0:=2}; din[9:8] inside
{2'b10,2'b11};
    }             constraint
read_and_write{ rx_valid dist
{1:=90,0:=10}; rst_n dist
{1:=98,0:=2};
    }
endclass
endpackage
```

We have made 3 constraints in the sequence item and in each sequence of the sequences required I will enable the corresponding constraint and disable the others.

(B) cover groups and assertions and code coverage :

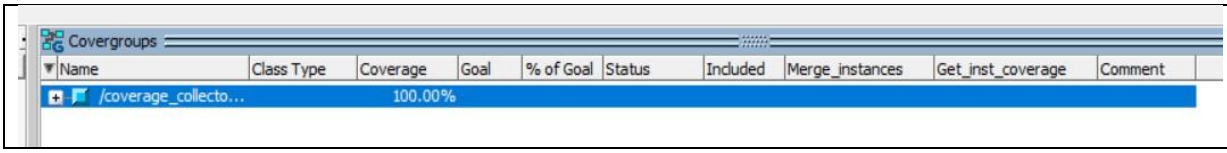
- cover groups:

(1)code :--

```
  package
coverage_collector; import
uvm_pkg::*; import
squence_item_pkg::*;
`include "uvm_macros.svh"
 class ram_coverage extends uvm_component;
`uvm_component_utils(ram_coverage)
uvm_analysis_export #(ram_seq_item) cov_export;
uvm_tlm_analysis_fifo #(ram_seq_item) cov_fifo;
ram_seq_item seq_item_cov;
     covergroup g;          din_cp:coverpoint
seq_item_cov.din ;          rx_valid_cp: coverpoint
seq_item_cov.rx_valid;          the_order_cp:
coverpoint seq_item_cov.din[9:8]{              bins
read_add={2'b10};             bins read_data={2'b11};
bins write_add={2'b00};             bins
write_data={2'b01};
        }          rx_valid_high_cp: coverpoint
seq_item_cov.rx_valid{              bins high ={1};
        }          rst_n_dasserted_cp:coverpoint
seq_item_cov.rst_n{              bins nonactive={1};
        }          the_address_cp: coverpoint
seq_item_cov.din[7:0];          rst_n_cp:coverpoint
seq_item_cov.rst_n;          the_order_with_rx_valid_cp: cross
the_order_cp,rx_valid_cp;          rx_valid_with_rst_n_cp: cross
rx_valid_cp,rst_n_cp;          the_order_with_rst_n_cp:cross
the_order_cp, rst_n_cp;

the_address_with_rx_valid_and_dasserted_rst_and_write_and_read_op:cross
rst_n_dasserted_cp,rx_valid_high_cp,the_address_cp,the_order_cp;
endgroup
     function  new(string name ="ram_coverage",uvm_component parent
=null);          super.new(name,parent);          g=new;      endfunction
         function void build_phase( uvm_phase
phase);          super.build_phase(phase);
cov_export=new("cov_export",this);
```

```
        cov_fifo=new("cov_fifo",this);
endfunction function void
connect_phase(uvm_phase phase);
super.connect_phase(phase);
cov_export.connect(cov_fifo.analysis_export);
endfunction
task run_phase(uvm_phase
phase); super.run_phase(phase);
forever begin
    cov_fifo.get(seq_item_cov);

g.sample();
end    endtask
endclass
endpackage
```

(2) questa snippets :--



(3) coverage report :--

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

Covergroup Coverage:
    Covergroups                 1        na        na   100.00%
        Coverpoints/Crosses    11        na        na        na
            Covergroup Bins   414       414         0   100.00%
--------------------------------------------------------------------------------------------------------------
```

We have made coverage of the following:

- din.
- rx_valid.
- the L2SB of the din (to check all operations !) (din[9:8]).
- din[7:0]: all addresses of read or write.
- Checking that each operation have come when rx_valid is high.
- Checking that rx_valid have got high when the reset is asserted.
- Checking that each operation have come when reset is asserted.
- Checking that all addresses(din[7:0]) have come when the reset is deasserted and the rx_valid asserted.

- assertions :

(1)code :--

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

        property property1;
             @(posedge clk) (~rst_n) |-> ( (~tx_valid) && (~dout) ) ;
      endproperty
        property property2;
                @(posedge clk) disable iff(!rst_n)
((rx_valid)&&(din[9:8]==2'b11)) |=> (tx_valid==1)  ;
        endproperty

        property property3;
                @(posedge clk) disable iff(!rst_n)
((rx_valid)&&(din[9:8]!=2'b11)) |=> (tx_valid==0)  ;
        endproperty

                property property4;
                @(posedge clk) disable iff(!rst_n)
((rx_valid)&&(din[9:8]!=2'b11)) |=> $stable(dout)  ;
        endproperty

        property property5;
                @(posedge clk) disable iff(!rst_n)
((!rx_valid)&&(din[9:8]==2'b11)) |=> ($stable(tx_valid))
;     endproperty       property property6;
                @(posedge clk) disable iff(!rst_n)
((!rx_valid)&&(din[9:8]==2'b11)) |=> $stable(dout)  ;
        endproperty
lb1: assert property (property1); lb2:
assert property (property2); lb3: assert
property (property3); lb4: assert
property (property4); lb5: assert
property (property5); lb6: assert
property (property6);
  lb7: cover property (property1);
lb8: cover property (property2);
lb9: cover property (property3);
lb10: cover property (property4);
lb11: cover property (property5);
lb12: cover property (property6);

endmodule
```

- code coverage :

```
 Toggle Coverage:
    Enabled Coverage                Bins      Hits    Misses  Coverage
    ----------------                ----      ----    ------  --------
    Toggles                           44        44         0   100.00%

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

Toggle Coverage for instance /\top#dut /RAM_assertions_inst --

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

Total Node Count     =          22
Toggled Node Count   =          22
Untoggled Node Count =           0

Toggle Coverage      =     100.00% (44 of 44 bins)

================================================================================
=======
=== Instance: /\top#dut
=== Design Unit: work.project_ram
================================================================================
=======
Branch Coverage:
    Enabled Coverage                Bins      Hits    Misses  Coverage
    ----------------                ----      ----    ------  --------
    Branches                           7         7         0   100.00%

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

Branch Coverage for instance /\top#dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ram.v
------------------------------------IF Branch----------------------------
-------
```

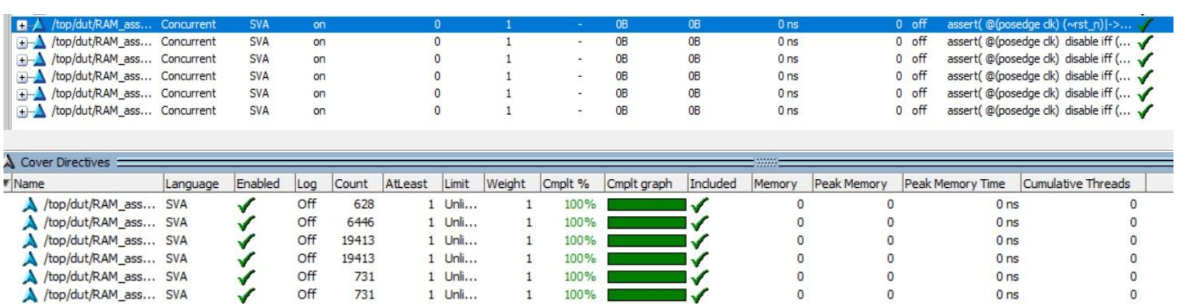
```
    14                                    30608     Count coming in to IF
    14              1                      1239      if (~rst_n) begin
    26              1                     26408      else if (rx_valid)
begin
                                           2961     All False Count
Branch totals: 3 hits of 3 branches = 100.00%

-----------------------------------IF Branch----------------------------
-------
    28                                    26408     Count coming in to IF
    28              1                      6653              if (din[9:8]
== 2'b00) begin
    32              1                      6588              else if
(din[9:8] == 2'b01) begin
    38              1                      6583              else if
(din[9:8] == 2'b10) begin
    42              1                      6584              else begin
Branch totals: 4 hits of 4 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Conditions                       3          3         0   100.00%

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

Condition Coverage for instance /\top#dut  --

  File ram.v
----------------Focused Condition View-------------------
Line       28 Item    1  (din[9:8] == 0)
Condition totals: 1 of 1 input term covered = 100.00%

       Input Term   Covered  Reason for no coverage   Hint
      -----------  --------  -----------------------  --------------
  (din[9:8] == 0)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (din[9:8] == 0)_0     -
  Row   2:          1  (din[9:8] == 0)_1     -

----------------Focused Condition View-------------------
Line       32 Item    1  (din[9:8] == 1)
Condition totals: 1 of 1 input term covered = 100.00%

       Input Term   Covered  Reason for no coverage   Hint
      -----------  --------  -----------------------  --------------
  (din[9:8] == 1)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
```

```
  Row   1:           1  (din[9:8] == 1)_0     -
  Row   2:           1  (din[9:8] == 1)_1     -

----------------Focused Condition View-------------------
Line       38 Item    1  (din[9:8] == 2)
Condition totals: 1 of 1 input term covered = 100.00%

       Input Term   Covered  Reason for no coverage   Hint
      -----------  --------  ----------------------  --------------
  (din[9:8] == 2)         Y

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


Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      19        19         0   100.00%

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

Statement Coverage for instance /\top#dut  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ram.v
    1                                                module
project_ram(din, rx_valid, dout, tx_valid, clk, rst_n);
    2                                                parameter MEM_DEPTH =
256;
    3                                                parameter ADDR_SIZE =
8;
    4                                                input rx_valid, clk,
rst_n;
    5                                                input [9:0] din;
    6                                                output reg tx_valid;
    7                                                output reg [7:0]
dout;
    8                                                reg [ADDR_SIZE-1:0]
addr_rd, addr_wr;
    9                                                reg [7:0] mem
[MEM_DEPTH-1:0];
    10                                               reg [8:0] i ;
    11                                               /*
    12                                               bugs  if rst_n
activated the internal register of read/write adderesses is not cleared*/
    13               1                       30608   always @(posedge clk
or negedge rst_n) begin
    14                                                if (~rst_n) begin
    15               1                        1239            dout <= 8'b0;
```

```
    16                1                        1239
        tx_valid <= 1'b0;
    17                1                        1239
        addr_rd<=0;
    18                1                        1239
        addr_wr<=0;
    19                1                        1239                    i=0;
    20                1                        1239             for (i = 0; i
< MEM_DEPTH; i=i+1) begin
    20                2                      317184
    21                1                      317184                    mem
[i] <= 1'b0;
    22                                                          end
    23
    24                                                  end
    25
    26                                                  else if (rx_valid)
begin
    27                1                       26408
        i=8'b1111_1111;
    28                                                          if (din[9:8]
== 2'b00) begin
    29                1                        6653
        addr_wr <= din[7:0];
    30                1                        6653
        tx_valid <= 0;
    31                                                          end
    32                                                          else if
(din[9:8] == 2'b01) begin
    33                1                        6588                    mem
[addr_wr] <= din[7:0];
    34                1                        6588
        tx_valid <= 0;
    35
    36                1                        6588             i=0;
    37                                                          end
    38                                                          else if
(din[9:8] == 2'b10) begin
    39                1                        6583
        addr_rd <= din[7:0];
    40                1                        6583
        tx_valid <= 0;
    41                                                          end
    42                                                          else begin
    43                1                        6584                    dout
<= mem[addr_rd];
    44                1                        6584
        tx_valid <= 1;

Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         94        94         0   100.00%
```

```
==================================Toggle
Details==================================

Toggle Coverage for instance /\top#dut  --

                                              Node      1H->0L      0L->1H
"Coverage"
                                              ----------------------------
-----------
                                      addr_rd[7-0]           1           1
100.00
                                      addr_wr[7-0]           1           1
100.00
                                               clk           1           1
100.00
                                          din[0-9]           1           1
100.00
                                         dout[7-0]           1           1
100.00
                                            i[8-0]           1           1
100.00
                                             rst_n           1           1
100.00
                                          rx_valid           1           1
100.00
                                          tx_valid           1           1
100.00

Total Node Count     =         47
Toggled Node Count   =         47
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (94 of 94 bins)
```

(2) questa snippets :--



/top/dut/RAM_ass...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) (~rst_n)|->...
/top/dut/RAM_ass...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...
/top/dut/RAM_ass...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...
/top/dut/RAM_ass...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...
/top/dut/RAM_ass...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...
/top/dut/RAM_ass...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge clk) disable iff (...

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/dut/RAM_ass...	SVA	✓	Off	628	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/dut/RAM_ass...	SVA	✓	Off	6446	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/dut/RAM_ass...	SVA	✓	Off	19413	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/dut/RAM_ass...	SVA	✓	Off	19413	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/dut/RAM_ass...	SVA	✓	Off	731	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/dut/RAM_ass...	SVA	✓	Off	731	1	Unli...	1	100%		✓	0	0	0 ns	0

(3) coverage report :--

```
Directive Coverage:
    Directives                  6        6        0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                Design Design   Lang File(Line)      Hits Status
                                    Unit   UnitType
--------------------------------------------------------------------------------------------
/top/dut/RAM_assertions_inst/1b7    RAM_assertions Verilog  SVA  C:/Users/TECH SHOP/Desktop/a5oyaaa_phillloooo/RAM_assertions.sv(45)
                                                                    Enabled  Off  628       1 Unlimited      1 Covered
/top/dut/RAM_assertions_inst/1b8    RAM_assertions Verilog  SVA  C:/Users/TECH SHOP/Desktop/a5oyaaa_phillloooo/RAM_assertions.sv(46)
                                                                    Enabled  Off 6446       1 Unlimited      1 Covered
/top/dut/RAM_assertions_inst/1b9    RAM_assertions Verilog  SVA  C:/Users/TECH SHOP/Desktop/a5oyaaa_phillloooo/RAM_assertions.sv(47)
                                                                    Enabled  Off 19413      1 Unlimited      1 Covered
/top/dut/RAM_assertions_inst/1b10   RAM_assertions Verilog  SVA  C:/Users/TECH SHOP/Desktop/a5oyaaa_phillloooo/RAM_assertions.sv(48)
                                                                    Enabled  Off 19413      1 Unlimited      1 Covered
/top/dut/RAM_assertions_inst/1b11   RAM_assertions Verilog  SVA  C:/Users/TECH SHOP/Desktop/a5oyaaa_phillloooo/RAM_assertions.sv(49)
                                                                    Enabled  Off  731       1 Unlimited      1 Covered
/top/dut/RAM_assertions_inst/1b12   RAM_assertions Verilog  SVA  C:/Users/TECH SHOP/Desktop/a5oyaaa_phillloooo/RAM_assertions.sv(50)
                                                                    Enabled  Off  731       1 Unlimited      1 Covered
```

We have made assertions to check the following the following:

Assertion	Explain

`property property1;`
`@(posedge clk)`
`(~rst_n) |-> ( (~tx_valid) &&`
`(~dout) ) ;`
`endproperty`
	when rst_n is activated dout and tx_valid will be 0.

`property property2;`
`@(posedge clk)`
`disable iff(!rst_n)`
`((rx_valid)&&(din[9:8]==2'b11)) |=>`
`(tx_valid==1) ;`
`endproperty`
	When rx_valid gets high and din[9:8] is ‘b 11 then at the next clock cycle tx_valid will be high.

`property property3;`
`@(posedge clk)`
`disable iff(!rst_n)`
`((rx_valid)&&(din[9:8]!=2'b11)) |=>`
`(tx_valid==0) ;`
`endproperty`
	When rx_valid gets high and din[9:8] is not equal to ‘b 11 then at the next clock cycle tx_valid will be low.

`property property4;`
`@(posedge clk)`
`disable iff(!rst_n)`
	When rx_valid gets high and din[9:8] is not equal to ‘b 11 then dout will not change.

((rx_valid)&&(din[9:8]!=2'b11)) |=> $stable(dout) ; endproperty	
property property5; @(posedge clk) disable iff(!rst_n) ((!rx_valid)&&(din[9:8]==2'b11)) |=> ($stable(tx_valid)) ; endproperty	When rx_valid gets low tx_valid will not change .
property property6; @(posedge clk) disable iff(!rst_n) ((!rx_valid)&&(din[9:8]==2'b11)) |=> $stable(dout) ; endproperty	When rx_valid gets low dout will not change .

(C) SEQUENCE of the RAM : (spilit to 4 sequences) 1) The reset sequence :

```
 package squence_pkg;
import uvm_pkg::*; import
squence_item_pkg::*;
`include "uvm_macros.svh"
//////////////////////////////////////////////////////////
class ram_reset_squence extends uvm_sequence;
`uvm_object_utils(ram_reset_squence)      function new(string
name ="ram_reset_squence");      super.new(name);
endfunction //new() virtual task body(); ram_seq_item item;
item=ram_seq_item::type_id::create("item");
start_item(item); item.din=0; item.rst_n=0;
item.rx_valid=0; finish_item(item); endtask
endclass
```

2) The write sequence:

```
 class ram_write_squence extends uvm_sequence;
`uvm_object_utils(ram_write_squence)     function new(string name
="ram_read_write_squence");     super.new(name);
endfunction //new() virtual task body(); repeat(10000) begin
ram_seq_item item; item=ram_seq_item::type_id::create("item");
start_item(item); item.constraint_mode(0);
item.write_only.constraint_mode(1); assert(item.randomize());
finish_item(item); end endtask endclass
```

3) The read sequence:

```
 class ram_read_squence extends uvm_sequence;
`uvm_object_utils(ram_read_squence)     function new(string name
="ram_read_write_squence");     super.new(name);
endfunction //new() virtual task body(); repeat(10000) begin
ram_seq_item item; item=ram_seq_item::type_id::create("item");
start_item(item); item.constraint_mode(0);
item.read_only.constraint_mode(1); assert(item.randomize());
finish_item(item); end endtask
endclass
```

4) The read sequence:

```
  class ram_read_and_write_squence extends uvm_sequence;
`uvm_object_utils(ram_read_and_write_squence)
function new(string name ="ram_read_write_squence");
super.new(name);          endfunction //new() virtual
task body(); repeat(10000) begin ram_seq_item item;
item=ram_seq_item::type_id::create("item");
start_item(item); item.constraint_mode(0);
item.read_and_write.constraint_mode(1);
assert(item.randomize()); finish_item(item); end endtask
endclass  endpackage
```

(D) TESTPLAN and uvm structure :

1) **Send the reset sequence.**
2) **Send the write sequence**
3) **Send the read sequence**
4) **Send the read write sequence**

For each sequence I will check in the scoreboard

- UVM Structure: