Test plan:

Function	How to test
Reset	Make assertions on values after reset
Write address	Assert if the din [9:8]==00 the next will be write data which is din[9:8]==01
Read address	Similar constrain input and cover point then make assertions
Write data	Similar constrain input and cover point then make assertions
Read data	Similar constrain input and cover point then make assertions

Run.do file :

```
vlib work
vlog project_ram.sv ram_if.sv  ram_monitor.sv ram_sva.sv ram_tb.sv ram_top.sv -covercells
vsim -voptargs=+acc work.ram_top -cover
add wave *
coverage save ram.ucdb -onexit
run -all
```

Constrains randomization to input:

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

constraint rx_valid_c {
    rx_valid dist {1:/90 , 0:/10};
}

constraint din_9_8_c {
    (old_value_9_8_din==2'b00) -> din[9:8] == 2'b01 ;
    (old_value_9_8_din==2'b10) -> din[9:8] == 2'b11 ;
    (old_value_9_8_din==2'b01||old_value_9_8_din==2'b11) -> din[9:8] dist {2'b00:/50 , 2'b10:/50} ;
}
```

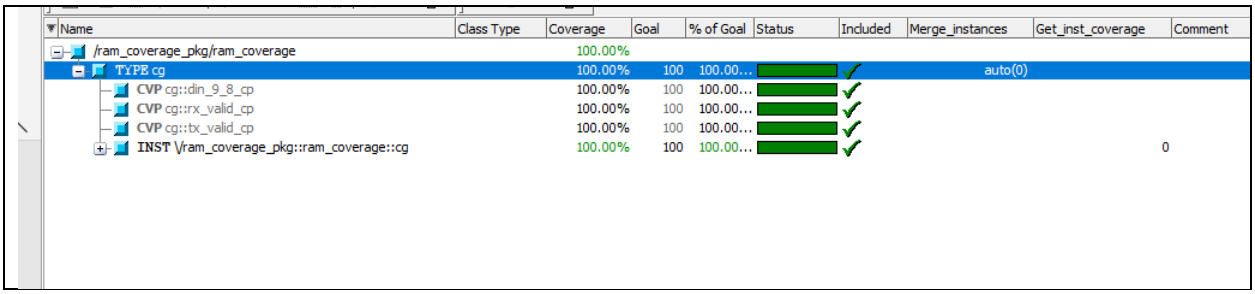
Constrain din[9:8] if the previous randomization is 00 next to be 10 .(write flow)

Constrain din[9:8] if the previous randomization is 10 next to be 11.(read flow)

Code coverage:

```
    Coverpoint rx_valid_cp                    100.00%        100         -    Covered
        covered/total bins:                         2          2         -
        missing/total bins:                         0          2         -
        % Hit:                                100.00%        100         -
        bin zero                                  109          1         -    Covered
        bin one                                   891          1         -    Covered
    Coverpoint tx_valid_cp                    100.00%        100         -    Covered
        covered/total bins:                         4          4         -
        missing/total bins:                         0          4         -
        % Hit:                                100.00%        100         -
        bin tx_0                                  783          1         -    Covered
        bin tx_1                                  217          1         -    Covered
        bin tx_trans_0_1                          217          1         -    Covered
        bin tx_trans_1_0                          216          1         -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

Cover groups form Questa :

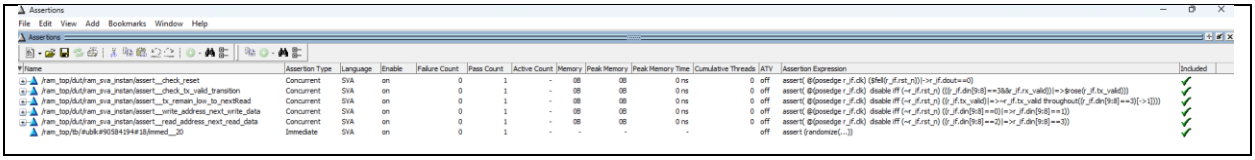


Assertions coverage:

```
Assertion Coverage:
    Assertions                       5         5         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/ram_top/dut/ram_sva_instan/assert__read_address_next_read_data
                     ram_sva.sv(46)                     0          1
/ram_top/dut/ram_sva_instan/assert__write_address_next_write_data
                     ram_sva.sv(37)                     0          1
/ram_top/dut/ram_sva_instan/assert__tx_remain_low_to_nextRead
                     ram_sva.sv(28)                     0          1
/ram_top/dut/ram_sva_instan/assert__check_tx_valid_transition
                     ram_sva.sv(19)                     0          1
/ram_top/dut/ram_sva_instan/assert__check_reset
                     ram_sva.sv(10)                     0          1
```

```
Directive Coverage:
    Directives                          5         5         0    100.00%
```

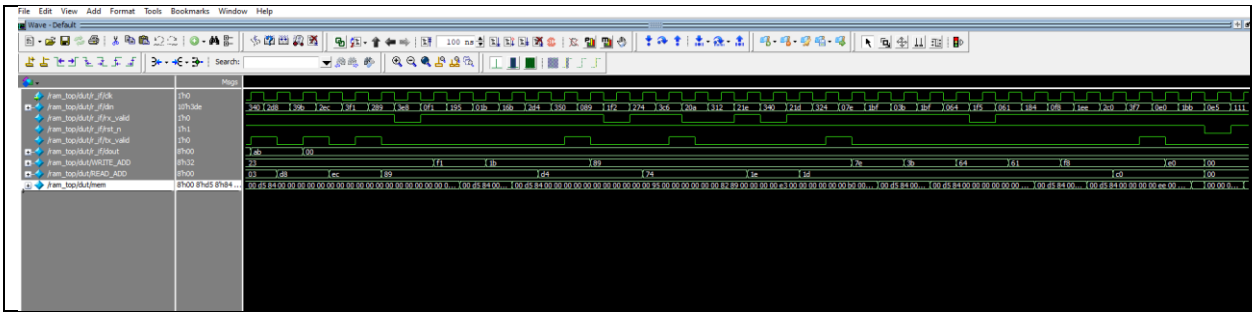
Assertions from Questa sim:



Assertions in code :

```
/*****************************************************************************/
property check_reset ;
    @(posedge r_if.clk) $fell(r_if.rst_n) |-> (r_if.dout==0) ;
endproperty

assert property (check_reset) ;
cover property (check_reset) ;
/*****************************************************************************/

/*****************************************************************************/
property check_tx_valid_transition ;
@(posedge r_if.clk) disable iff(!r_if.rst_n) (r_if.din[9:8]==2'b11)&&(r_if.rx_valid) |=> $rose(r_if.tx_valid) ;
endproperty

assert property (check_tx_valid_transition) ;
cover property (check_tx_valid_transition) ;
/*****************************************************************************/

/*****************************************************************************/
property tx_remain_low_to_nextRead ;
@(posedge r_if.clk) disable iff(!r_if.rst_n) (r_if.tx_valid) |=> ((!r_if.tx_valid) throughout (r_if.din[9:8]==2'b11)[->1]) ;
endproperty

assert property (tx_remain_low_to_nextRead) ;
cover property (tx_remain_low_to_nextRead) ;
/*****************************************************************************/

/*****************************************************************************/
property write_address_next_write_data ;
@(posedge r_if.clk) disable iff(!r_if.rst_n) (r_if.din[9:8]==2'b00) |=> (r_if.din[9:8]==2'b01) ;
endproperty

assert property (write_address_next_write_data) ;
cover property (write_address_next_write_data) ;
/*****************************************************************************/

/*****************************************************************************/
property read_address_next_read_data ;
@(posedge r_if.clk) disable iff(!r_if.rst_n) (r_if.din[9:8]==2'b10) |=> (r_if.din[9:8]==2'b11) ;
endproperty

assert property (read_address_next_read_data) ;
cover property (read_address_next_read_data) ;
/*****************************************************************************/

endmodule
```

Results on wave form:



Bug report:

```
reg [7:0] WRITE_ADD,READ_ADD;
reg[7:0] mem[255:0];
integer i;
always@(posedge clk or negedge rst_n) begin
if(rst_n==0) begin
dout<=0;
end
else if(rx_valid==1) begin
 if(din[9:8]==2'b00)
 WRITE_ADD<=din[7:0];
 else if(din[9:8]==2'b01)
      mem[WRITE_ADD]<=din[7:0];
 else if(din[9:8]==2'b10)
      READ_ADD<=din[7:0];
 else if(din[9:8]==2'b11) begin
      dout<=mem[READ_ADD];
      tx_valid<=1;
 end
end
end


endmodule
```

1- **tx_valid must be low for all cases except read data .**
2- **on reset WRITE_ADD<=0 ; READ_ADD <=0 ; should be zero .**
3- **also we should handle the case if rst_n = 1 and rx_valid not equal =1 ;**

code after correctness :

```
always@(posedge r_if.clk or negedge r_if.rst_n) begin
if(!r_if.rst_n) begin
     r_if.tx_valid<=0 ;
     for(i=0 ; i<= 255 ; i=i+1)
          mem[i] <= 0 ;
     r_if.dout<=0;
     WRITE_ADD<=0 ;
     READ_ADD <=0 ;
end
// ALSO we need to make tx_valid=0 in the cases that we are not read from the mem.
else if(r_if.rx_valid) begin
      if(r_if.din[9:8]==2'b00)
      begin
          WRITE_ADD<=r_if.din[7:0];
          r_if.tx_valid<=0;
      end

      else if(r_if.din[9:8]==2'b01) begin
           mem[WRITE_ADD]<=r_if.din[7:0];
            r_if.tx_valid<=0;
      end

      else if(r_if.din[9:8]==2'b10) begin
           READ_ADD<=r_if.din[7:0];
            r_if.tx_valid<=0;
      end

      else if(r_if.din[9:8]==2'b11) begin
           r_if.dout <= mem[READ_ADD];
           r_if.tx_valid<=1;
      end
end

else begin
r_if.dout <=0 ;
r_if.tx_valid <= 0 ;
end

end
```