ECEN 689 Formal Verification

Laboratory Exercise

MOESI Cache Coherence Verification using SV assertions and Jasper Gold

Prof. Jiang Hu

TA: Zili Fang

Name: Velmurugan Mohan Krishnapuram UIN: 630009348

Lab1: List of Assertions

```
module moesi_fsm_assert #(
                           parameter MOESI_WID = 3
                          )(
                            input                           read_miss,
                            input                           write_miss,
                            input                           write_hit,
                            input                           shared,
                            input                           exclusive,
                            input                           probe_write_hit,
                            input                           probe_read_hit,
                            input                           reset,
                            input                           clk,
                            input      [MOESI_WID - 1 : 0] current_moesi,
                            input      [MOESI_WID - 1 : 0] updated_moesi
                          );

  parameter INVALID   = 3'b000;
  parameter SHARED    = 3'b001;
  parameter EXCLUSIVE = 3'b010;
  parameter MODIFIED  = 3'b011;
  parameter OWNED     = 3'b100;

  //Assertions
  //TO DO: add assertions here
  //Assertions
  //TO DO: add assertions here


// CHECK RESET
  property assr_RANDOM_reset_INVALID;
      @(posedge clk) (reset==0) |=> (updated_moesi == INVALID);
  endproperty


// CHECK FROM SHARED TO OTHER STATES
  property assr_SHARED_probe_wr_hit_INVALID;
      @(posedge clk) (reset==1) && (current_moesi == SHARED) &&
(probe_write_hit == 1) |=> (updated_moesi == INVALID);
  endproperty
```

```
  property assr_SHARED_write_hit_MODIFIED;
      @(posedge clk)  (reset==1) &&  (current_moesi == SHARED) && (write_hit
== 1) |=> (updated_moesi == MODIFIED);
  endproperty



// CHECK FROM INVALID TO OTHER STATES
  property assr_INVALID_read_miss_and_exclusive_EXCLUSIVE;
      @(posedge clk)  (reset==1) && (current_moesi == INVALID) && (read_miss
== 1 && exclusive==1) |=> (updated_moesi == EXCLUSIVE);
  endproperty

  property assr_INVALID_read_miss_and_shared_SHARED;
      @(posedge clk)  (reset==1) && (current_moesi == INVALID) && (read_miss
== 1 && shared==1) |=> (updated_moesi == SHARED);
  endproperty

  property assr_INVALID_write_miss_and_shared_MODIFIED;
      @(posedge clk)  (reset==1) && (current_moesi == INVALID) &&
(write_miss==1) |=> (updated_moesi == MODIFIED);
  endproperty

// CHECK FROM EXCLUSIVE TO OTHER STATES
  property assr_EXCLUSIVE_probe_write_hit_INVALID;
      @(posedge clk)  (reset==1) && (current_moesi == EXCLUSIVE) &&
(probe_write_hit == 1) |=> (updated_moesi == INVALID);
  endproperty

  property assr_EXCLUSIVE_probe_read_hit_SHARED;
      @(posedge clk)  (reset==1) && (current_moesi == EXCLUSIVE) &&
(probe_read_hit == 1) |=> (updated_moesi == SHARED);
  endproperty

  property assr_EXCLUSIVE_write_hit_MODIFIED;
      @(posedge clk)  (reset==1) && (current_moesi == EXCLUSIVE) &&
(write_hit == 1) |=> (updated_moesi == MODIFIED);
  endproperty


// CHECK FROM MODIFIED TO OTHER STATES
  property assr_MODIFIED_write_hit_INVALID;
      @(posedge clk)  (reset==1) && (current_moesi == MODIFIED) &&
(probe_write_hit == 1) |=> (updated_moesi == INVALID);
  endproperty

  property assr_MODIFIED_write_hit_OWNED;
      @(posedge clk)  (reset==1) && (current_moesi == MODIFIED) &&
(probe_read_hit == 1) |=> (updated_moesi == OWNED);
  endproperty

// CHECK FROM OWNED TO OTHER STATES
  property assr_OWNED_write_hit_MODIFIED;
      @(posedge clk)  (reset==1) && (current_moesi == OWNED) && (write_hit ==
1) |=> (updated_moesi == MODIFIED);
  endproperty
```

```
  property assr_OWNED_write_hit_INVALID;
      @(posedge clk)  (reset==1) && (current_moesi == OWNED) &&
(probe_write_hit == 1) |=> (updated_moesi == INVALID);
  endproperty



//SELF LOOP PROPERTY SHARED
   property assr_SHARED_probe_read_hit_SHARED;
      @(posedge clk) (reset==1) &&  (current_moesi == SHARED) &&
(probe_read_hit == 1) |=> (updated_moesi == SHARED);
  endproperty

//SELF LOOP PROPERTY OWNED
  property assr_OWNED_probe_read_hit_OWNED;
      @(posedge clk)  (reset==1) && (current_moesi == OWNED) &&
(probe_read_hit == 1) |=> (updated_moesi == OWNED);
  endproperty


  // assertions// CHECK FROM RANDOM TO INVALID
  assert property(assr_RANDOM_reset_INVALID)
$display("assr_RANDOM_reset_INVALID SUCCESS");
      else $error("assr_RANDOM_reset_INVALID FAIL");

  // assertions// CHECK FROM SHARED TO OTHER STATES
  assert property(assr_SHARED_probe_wr_hit_INVALID)
$display("assr_SHARED_probe_wr_hit_INVALID SUCCESS");
      else $error("assr_SHARED_probe_wr_hit_INVALID FAIL");

  assert property(assr_SHARED_write_hit_MODIFIED)
$display("assr_SHARED_write_hit_MODIFIED SUCCESS");
      else $error("assr_SHARED_write_hit_MODIFIED FAIL");



// CHECK FROM INVALID TO OTHER STATES
  assert property(assr_INVALID_read_miss_and_exclusive_EXCLUSIVE)
$display("assr_INVALID_read_miss_and_exclusive_EXCLUSIVE SUCCESS");
      else $error("assr_INVALID_read_miss_and_exclusive_EXCLUSIVE FAIL");

  assert property(assr_INVALID_read_miss_and_shared_SHARED)
$display("assr_INVALID_read_miss_and_shared_SHARED SUCCESS");
      else $error("assr_INVALID_read_miss_and_shared_SHARED FAIL");

  assert property(assr_INVALID_write_miss_and_shared_MODIFIED)
$display("assr_INVALID_write_miss_and_shared_MODIFIED SUCCESS");
      else $error("assr_INVALID_write_miss_and_shared_MODIFIED FAIL");

// CHECK FROM EXCLUSIVE TO OTHER STATES
  assert property(assr_EXCLUSIVE_probe_write_hit_INVALID)
$display("assr_EXCLUSIVE_probe_write_hit_INVALID SUCCESS");
      else $error("assr_EXCLUSIVE_probe_write_hit_INVALID FAIL");
```

```
  assert property(assr_EXCLUSIVE_probe_read_hit_SHARED)
$display("assr_EXCLUSIVE_probe_read_hit_SHARED SUCCESS");
      else $error("assr_EXCLUSIVE_probe_read_hit_SHARED FAIL");


  assert property(assr_EXCLUSIVE_write_hit_MODIFIED)
$display("assr_EXCLUSIVE_write_hit_MODIFIED SUCCESS");
      else $error("assr_EXCLUSIVE_write_hit_MODIFIED FAIL");

// CHECK FROM MODIFIED TO OTHER STATES
  assert property(assr_MODIFIED_write_hit_INVALID)
$display("assr_MODIFIED_write_hit_INVALID SUCCESS");
      else $error("assr_MODIFIED_write_hit_INVALID FAIL");

  assert property(assr_MODIFIED_write_hit_OWNED)
$display("assr_MODIFIED_write_hit_OWNED SUCCESS");
      else $error("assr_MODIFIED_write_hit_OWNED FAIL");

// CHECK FROM OWNED TO OTHER STATES
  assert property(assr_OWNED_write_hit_MODIFIED)
$display("assr_OWNED_write_hit_MODIFIED SUCCESS");
      else $error("assr_OWNED_write_hit_MODIFIED FAIL");

  assert property(assr_OWNED_write_hit_INVALID)
$display("assr_OWNED_write_hit_INVALID SUCCESS");
      else $error("assr_OWNED_write_hit_INVALID FAIL");

  assert property(assr_OWNED_probe_read_hit_OWNED)
$display("assr_OWNED_probe_read_hit_OWNED SUCCESS");
      else $error("assr_OWNED_probe_read_hit_OWNED FAIL");

// SELF LOOP SHARED
  assert property(assr_SHARED_probe_read_hit_SHARED)
$display("assr_SHARED_probe_read_hit_SHARED SUCCESS");
      else $error("assr_SHARED_probe_read_hit_SHARED FAIL");

// SELF LOOP OWNED
  assert property(assr_OWNED_probe_read_hit_OWNED)
$display("assr_OWNED_probe_read_hit_OWNED SUCCESS");
      else $error("assr_OWNED_probe_read_hit_OWNED FAIL");

endmodule
```

Simulation Result: (Total of 15 assertions properties)

13 checked in TB – 2 self-loop properties of vacuous success since precondition is not checked in TB hence not shown in output.

Test Bench Changes:

- Added a extra @ (negedge CLK) at the end of the Test Bench to get the last assertion result – since simulation ended before that.

```
ncsim> source /opt/coe/cadence/INCISIVE152/tools/dfca/ft
ncsim> set assert_output_stop_level {};
ncsim> run;
Testing Reset
Testing INVALID to EXCLUSIVE
assr_RANDOM_reset_INVALID SUCCESS
Testing INVALID to SHARED
assr_INVALID_read_miss_and_exclusive_EXCLUSIVE SUCCESS
Testing INVALID to MODIFIED
assr_INVALID_read_miss_and_shared_SHARED SUCCESS
Testing SHARED to INVALID
assr_INVALID_write_miss_and_shared_MODIFIED SUCCESS
Testing SHARED to MODIFIED
assr_SHARED_probe_wr_hit_INVALID SUCCESS
Testing EXCLUSIVE to INVALID
assr_SHARED_write_hit_MODIFIED SUCCESS
Testing EXCLUSIVE to MODIFIED
assr_EXCLUSIVE_probe_write_hit_INVALID SUCCESS
Testing EXCLUSIVE to SHARED
assr_EXCLUSIVE_write_hit_MODIFIED SUCCESS
Testing MODIFIED to OWNED
assr_EXCLUSIVE_probe_read_hit_SHARED SUCCESS
Testing MODIFIED to INVALID
assr_MODIFIED_write_hit_OWNED SUCCESS
Testing OWNED to INVALID
assr_MODIFIED_write_hit_INVALID SUCCESS
Testing OWNED to MODIFIED
assr_OWNED_write_hit_INVALID SUCCESS
assr_OWNED_write_hit_MODIFIED SUCCESS
Test Finished!
Simulation stopped via $stop(1) at time 150 NS + 0
../tb/moesi_fsm_tb.sv:113       $stop;
ncsim>
```

Lab2:

Iteration 1: First Run: 10 properties failed (assert1 precondition)

Type	Name	Engine	Bound	Time	Task
Cover (related)	top.dut.u_assert_moesi_fsm._assert_1:precondition1	PRE	Infinite	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_3	N	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_5	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_6	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_9	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_13	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp	2	0.0	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp	2	0.0	<embedded>

Debug Assert_1:

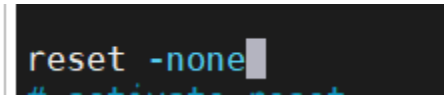
Filename ../tb/moesi_fsm_assert.sv

```
101   assert property(assr_RANDOM_reset_INVALID) $display("assr_RANDOM_reset_INVALID SUCCESS");
102         else $error("assr_RANDOM_reset_INVALID FAIL");
```

Observation: The reset !reset condition causes the issue at time t=0 of simulation.

Reason: The reset !reset condition resets the design initially at t=0 instant but then thereafter the first clock pulse reset=1 always, Hence, JG cannot trace the reset==0 precondition and the cover fails.

Solution: Add “reset -none” instead of “reset !reset” to make reset as an input signal of the assertion property.



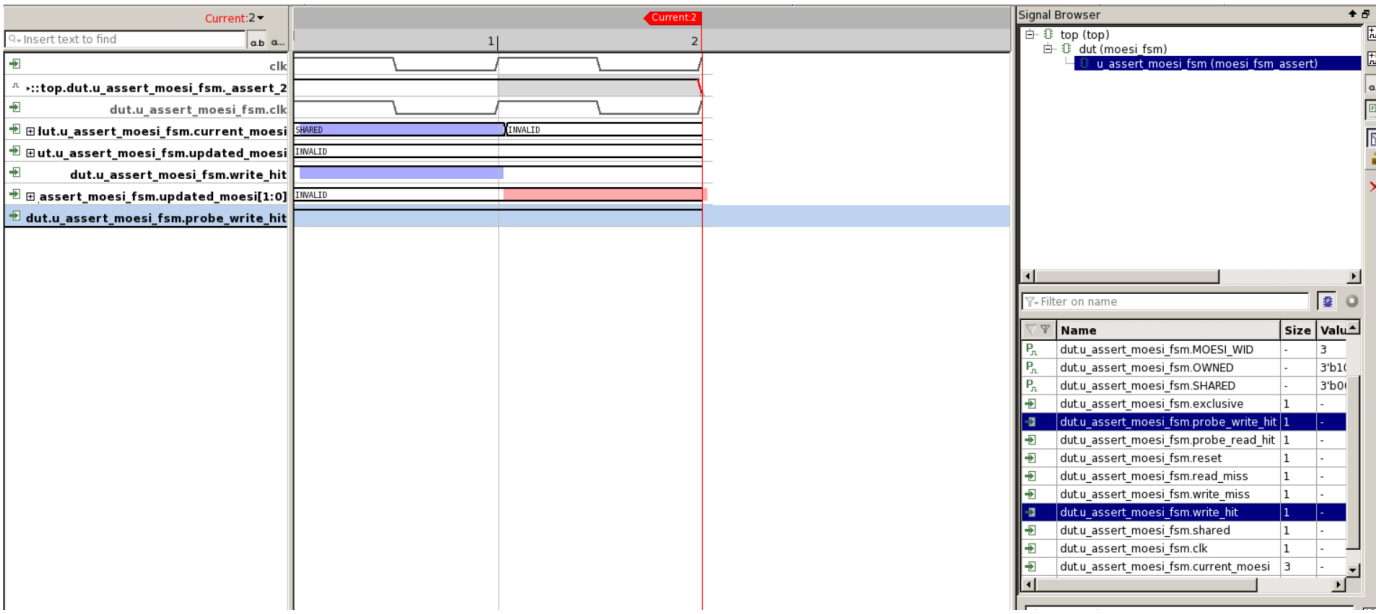
Iteration 2: 9 properties failed

Type	Name	Engine	Bound	Time	Task
Assert	top.dut.u_assert_moesi_fsm._assert_3	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_5	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_6	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_9	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_13	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp	2	0.1	<embedded>
Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp	2	0.1	<embedded>

Debug Assert_3:

```
ename   ../tb/moesi_fsm_assert.sv
96    assert property(assr_SHARED_write_hit_MODIFIED) $display("assr_SHARED_write_hit_MODIFIED SUCCESS");
97          else $error("assr_SHARED_write_hit_MODIFIED FAIL");
```

CEX:



Observation: update_moesi reaches INVALID state when MODIFIED is expected.

Reason: Since both probe_write_hit and write_hit are 1 at the same time – It takes the first possible output in the IF_ELSE branch – Ambiguous Inputs

```
            updated_moesi <= INVALID;
        end
        SHARED : begin
          if (probe_write_hit)
            updated_moesi <= INVALID;
          else if (write_hit)
            updated_moesi <= MODIFIED;
          else
            updated_moesi <= SHARED;
        end
```

Solution: Add following assumption to correct this assertion **(i.e., both probe_write_hit and write_hit is NOT 1 at the same time)**

```
4)}
# TODO add assumptions here
assume -name restrict_SHARED_state_input {!(probe_write_hit == 1'b1 && write_hit == 1'b1)}
```

Iteration 3: 6 properties failed

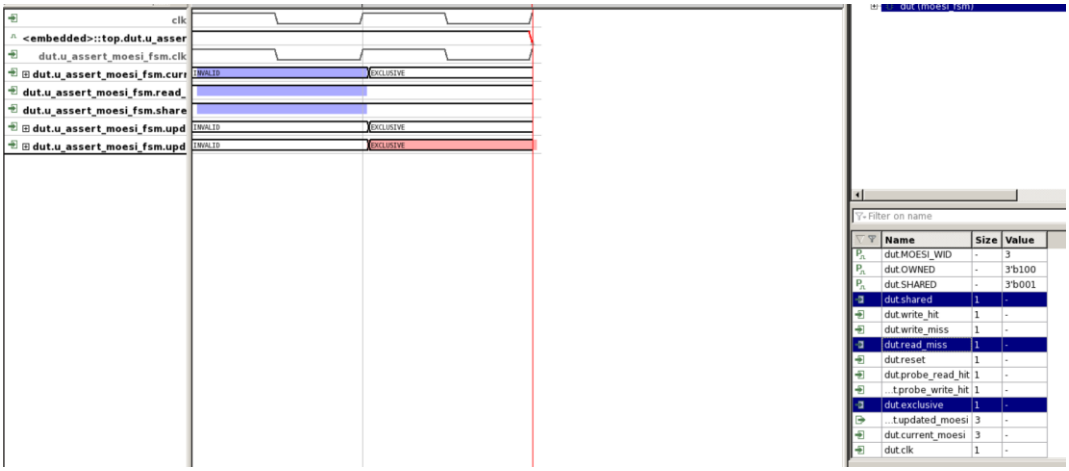
	Type	Name	Engine	Bound	Time	Task
✖	Assert	top.dut.u_assert_moesi_fsm._assert_5	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_6	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp	2	0.1	<embedded>

Debug_Assert_5:

ame ../tb/moesi_fsm_assert.sv

```
3    assert property(assr_INVALID_read_miss_and_shared_SHARED) $display("assr_INVALID_read_miss_and_shared_SHARED SUCCESS");
4          else $error("assr_INVALID_read_miss_and_shared_SHARED FAIL");
```

CEX:



Observation: update_moesi reaches EXCLUSIVE state when SHARED is expected.

Reason: Since all three signals, Shared, Exclusive and Read_miss are 1 at the same time – It takes the first possible output in the IF_ELSE branch. – Ambiguous Inputs

```
case (current_moesi)
  INVALID : begin
    if (read_miss && exclusive)
      updated_moesi <= EXCLUSIVE;
    else if (read_miss && shared)
      updated_moesi <= SHARED;
    else
      updated_moesi <= INVALID;
```

Solution: Add following assumption to correct this assertion

(Exclusive and Shared should not be equal to 1 at the same time)

```
assume -name restrict_INVALID_state_input {!(exclusive == 1'b1 && shared == 1'b1)}
```

Iteration 4: 5 properties failed

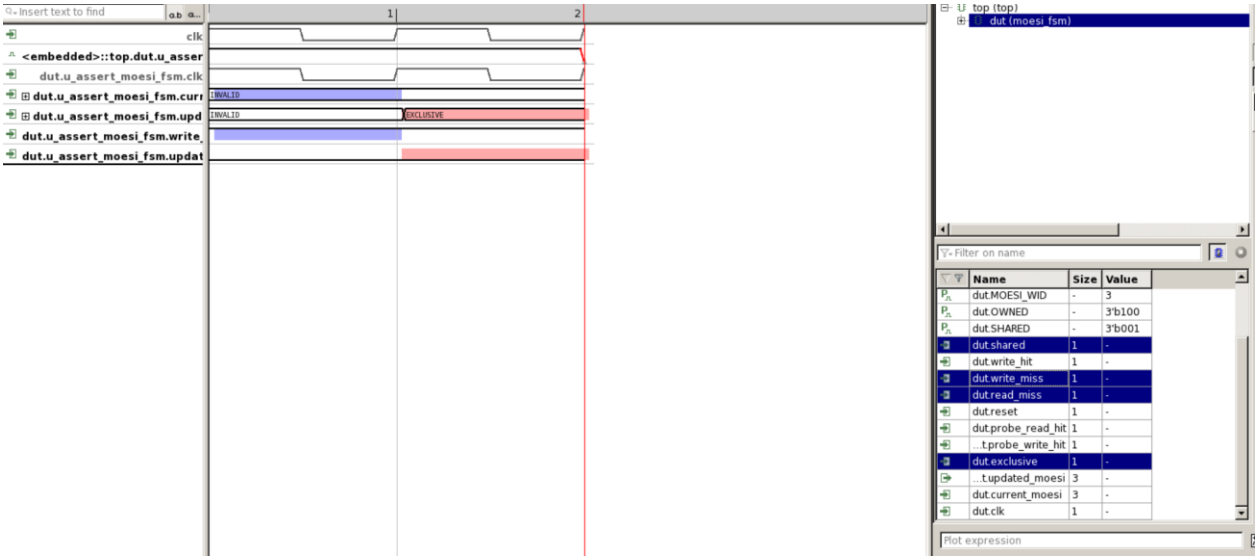
	Type	Name	Engine	Bound	Time	Task
✖	Assert	top.dut.u_assert_moesi_fsm._assert_6	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp	2	0.1	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp	2	0.1	<embedded>

Debug Assert_6

Filename ../tb/moesi_fsm_assert.sv

```
106    assert property(assr_INVALID_write_miss_and_shared_MODIFIED) $display("assr_INVALID_write_miss_and_shared_MC
107          else $error("assr_INVALID_write_miss_and_shared_MODIFIED FAIL");
```

CEX:



Observation: update_moesi reaches EXCLUSIVE state when MODIFIED is expected.

Reason: RTL Bug

- MODIFIED state is no-where found in the IF-ELSE branch of the Design.s

```
end else begin
  case (current_moesi)
    INVALID : begin
      if (read_miss && exclusive)
        updated_moesi <= EXCLUSIVE;
      else if (read_miss && shared)
        updated_moesi <= SHARED;
      else
        updated_moesi <= INVALID;
    end
```

Additional Bug: Since all 4 signals, Shared, Exclusive, Read_miss and Write_miss are 1 at the same time – It takes the first possible output in the IF_ELSE branch. – Ambiguous Inputs

Solution_1: Add the MODIFIED state to the FSM

```
    end else begin
      case (current_moesi)
        INVALID : begin
          if (read_miss && exclusive)
            updated_moesi <= EXCLUSIVE;
          else if (read_miss && shared)
            updated_moesi <= SHARED;
          else if( write_miss)
            updated_moesi <= MODIFIED;
          else
            updated_moesi <= INVALID;
        end
```

Solution_2: For Ambigious Inputs, modify the condition in previous iteration to include read_miss and write_miss. Should not reach the below state – Will create ambiguous decision at IF-ELSE branch.

Add the below assumption to prevent such combinations.

NOT{"read miss and write_miss both are 1 simulataneously" AND

"{exclusive,shared }={1,0} OR {exclusive,shared }={0,1} " }

```
assume -name restrict_INVALID_state_input {!((read_miss == 1'b1) && (write_miss == 1'b1)) && ((exclusive == 1'b0 && shared == 1'b1)||
(exclusive == 1'b1 && shared == 1'b0)) }
```

Iteration 5: 4 properties failed

	Type	Name	Engine	Bound	Time	Task
✖	Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp	2	0.0	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp	2	0.0	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp	2	0.0	<embedded>
✖	Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp	2	0.0	<embedded>

Debug Assert_8

Filename ../tb/moesi_fsm_assert.sv

```
114    assert property(assr_EXCLUSIVE_probe_read_hit_SHARED) $display("assr_EXCLUSIVE_probe_read_hit_SHARED SUCCESS");
115          else $error("assr_EXCLUSIVE_probe_read_hit_SHARED FAIL");
```

CEX:

Observation: update_moesi reaches INVALID state when SHARED is expected.

Reason: Since all 3 write_hit,probe_read_hit and probe_write_hit are 1 at the same time.

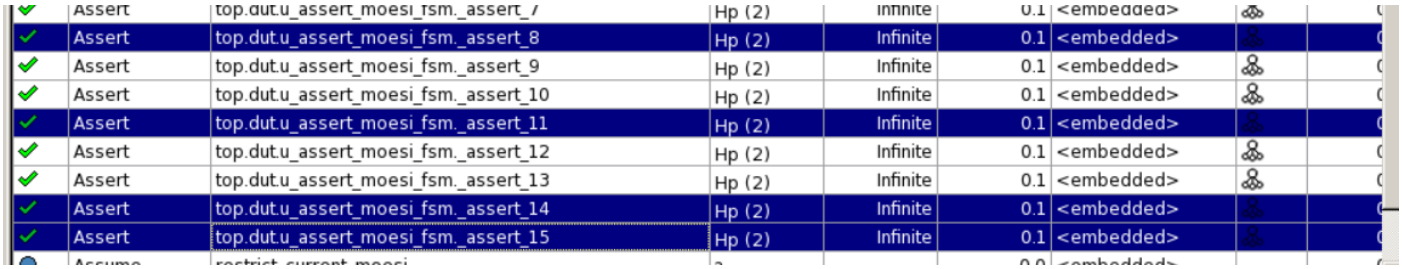
– It takes the first possible output in the IF_ELSE branch. – Ambiguous Inputs

```
end
EXCLUSIVE : begin
  if (probe_write_hit)
    updated_moesi <= INVALID;
  else if (write_hit)
    updated_moesi <= MODIFIED;
  else if (probe_read_hit)
    updated_moesi <= SHARED;
  else
    updated_moesi <= EXCLUSIVE;
end
MODIFIED : begin
  if (probe_write_hit)
    updated_moesi <= INVALID;
  else if (probe_read_hit)
    updated_moesi <= OWNED;
  else
    updated_moesi <= MODIFIED;
end
```

Solution: Assume a condition where **ONLY one of {probe_write_hit, probe_read_hit, write_hit} is 1 - i.e., {100} OR {010} OR {001}** and avoid all other combinations.

```
assume -name restrict_EXCLUSIVE_state_input {(probe_write_hit==1'b1 && probe_read_hit==1'b0 && write_hit==1'b0)||
(probe_write_hit==1'b0 && probe_read_hit==1'b1 && write_hit==1'b0)||
(probe_write_hit==1'b0 && probe_read_hit==1'b0 && write_hit==1'b1)}
```

Iteration 6: All Satisfied



✓	Assert	top.dut.u_assert_moesi_fsm._assert_7	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_9	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_10	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_12	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_13	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp (2)	Infinite	0.1	<embedded>	
✓	Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp (2)	Infinite	0.1	<embedded>	
	Assume	restrict_current_moesi			0.0	<embedded>	

- **Note from Above:** Change applied for assertion 8 automatically corrects the error in assertions 11,14, and 15 as shown below – since now only one of probe_read_hit OR probe_write_hit will be 1

```
assume -name restrict_SHARED_state_input {!(probe_write_hit == 1'b1 && write_hit == 1'b1)}

assume -name restrict_INVALID_state_input {!((read_miss == 1'b1) && (write_miss == 1'b1)) && ((exclusive == 1'b0 && shared == 1'b1)||
(exclusive == 1'b1 && shared == 1'b0)) }

assume -name restrict_EXCLUSIVE_state_input {(probe_write_hit==1'b1 && probe_read_hit==1'b0 && write_hit==1'b0)||
(probe_write_hit==1'b0 && probe_read_hit==1'b1 && write_hit==1'b0)||
(probe_write_hit==1'b0 && probe_read_hit==1'b0 && write_hit==1'b1)}
```

- Also, note that the assumption made in **"restrict_EXCLUSIVE_state_input"** already covers the combinations of the assumption **"restrict_SHARED_state_input" (shown above)**

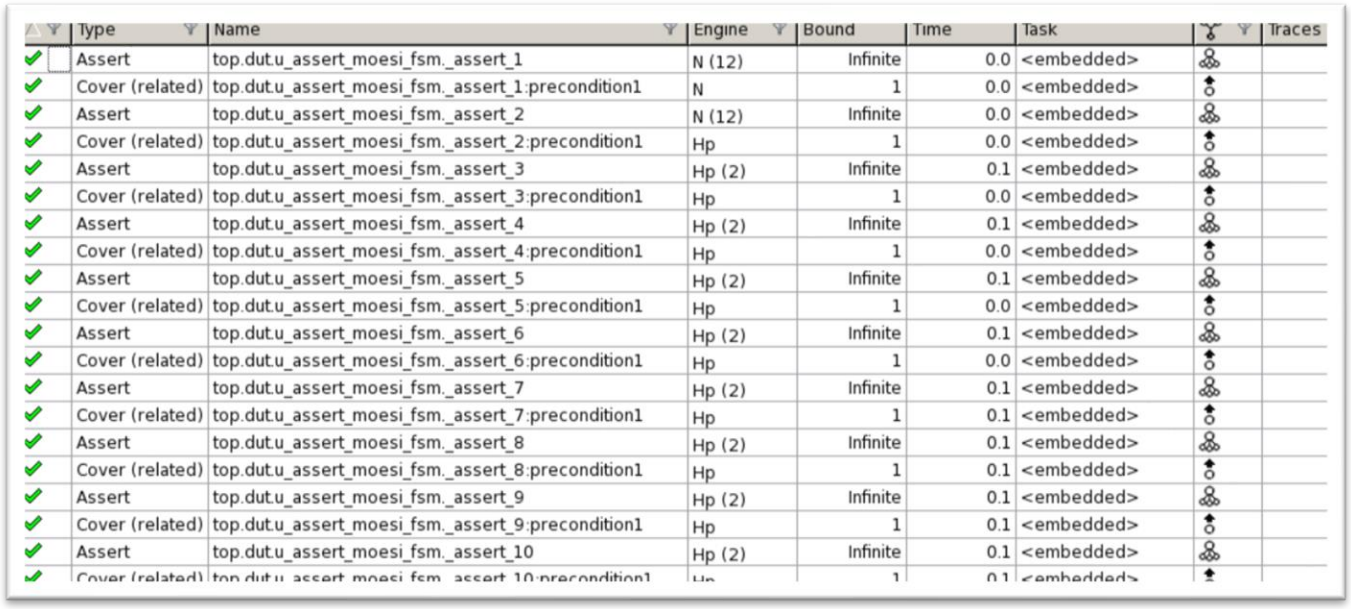
- Hence, we can REMOVE the redundant **"restrict_SHARED_state_input"** assumption to get minimized assumptions as shown below:

Final Minimized Set of Assumptions:

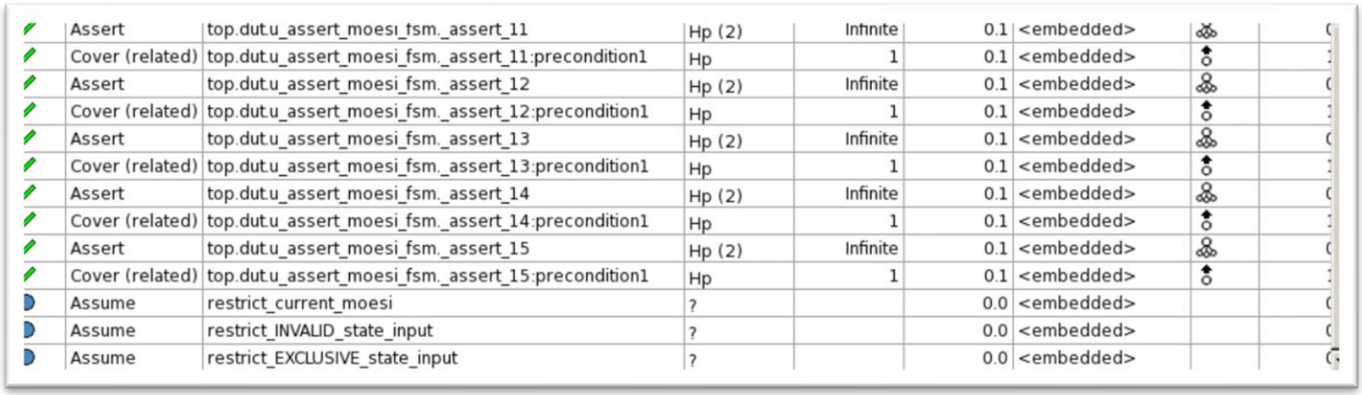
```
reset -none
# activate reset
# assumptions
assume -name restrict_current_moesi {(current_moesi == 3'd0)||(current_moesi == 3'd1)||(current_moesi == 3'd2)||(current_moesi == 3'd3)||(current_moesi == 3'd
4)}

assume -name restrict_INVALID_state_input {!((read_miss == 1'b1) && (write_miss == 1'b1)) && ((exclusive == 1'b0 && shared == 1'b1)||
(exclusive == 1'b1 && shared == 1'b0)) }

assume -name restrict_EXCLUSIVE_state_input {(probe_write_hit==1'b1 && probe_read_hit==1'b0 && write_hit==1'b0)||
(probe_write_hit==1'b0 && probe_read_hit==1'b1 && write_hit==1'b0)||
(probe_write_hit==1'b0 && probe_read_hit==1'b0 && write_hit==1'b1)}
```

Final Check Run: All properties are satisfied as shown below

-No CX generated



Type	Name	Engine	Bound	Time	Task	Traces
Assert	top.dut.u_assert_moesi_fsm._assert_1	N (12)	Infinite	0.0	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_1:precondition1	N	1	0.0	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_2	N (12)	Infinite	0.0	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_2:precondition1	Hp	1	0.0	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_3	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_3:precondition1	Hp	1	0.0	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_4	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_4:precondition1	Hp	1	0.0	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_5	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_5:precondition1	Hp	1	0.0	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_6	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_6:precondition1	Hp	1	0.0	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_7	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_7:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_8	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_8:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_9	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_9:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_10	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_10:precondition1	Hp	1	0.1	<embedded>	



Assert	top.dut.u_assert_moesi_fsm._assert_11	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_11:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_12	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_12:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_13	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_13:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_14	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_14:precondition1	Hp	1	0.1	<embedded>	
Assert	top.dut.u_assert_moesi_fsm._assert_15	Hp (2)	Infinite	0.1	<embedded>	
Cover (related)	top.dut.u_assert_moesi_fsm._assert_15:precondition1	Hp	1	0.1	<embedded>	
Assume	restrict_current_moesi	?		0.0	<embedded>	
Assume	restrict_INVALID_state_input	?		0.0	<embedded>	
Assume	restrict_EXCLUSIVE_state_input	?		0.0	<embedded>	