# CSCE 714: Advanced Hardware Design Functional Verification

**Texas A&M University**

# BUG REPORT

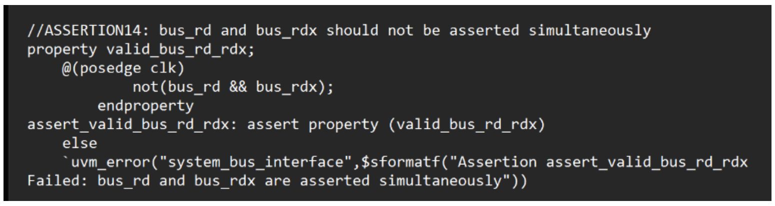
Multicore MESI Based Cache Design HAS

**Team 16**

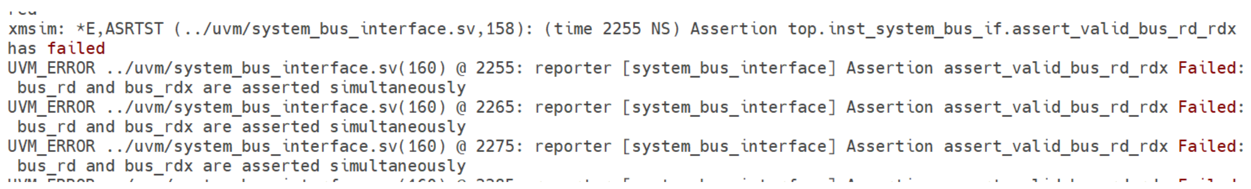
**Team Members:**

| **Team Member** | **Name** | **UIN** |
|---|---|---|
| 1 | Vinay Bayaneni | 131005670 |
| 2 | Jyothi Swaroopa Myneedi | 232003853 |
| 3 | Michelle Madubuike | 526003184 |

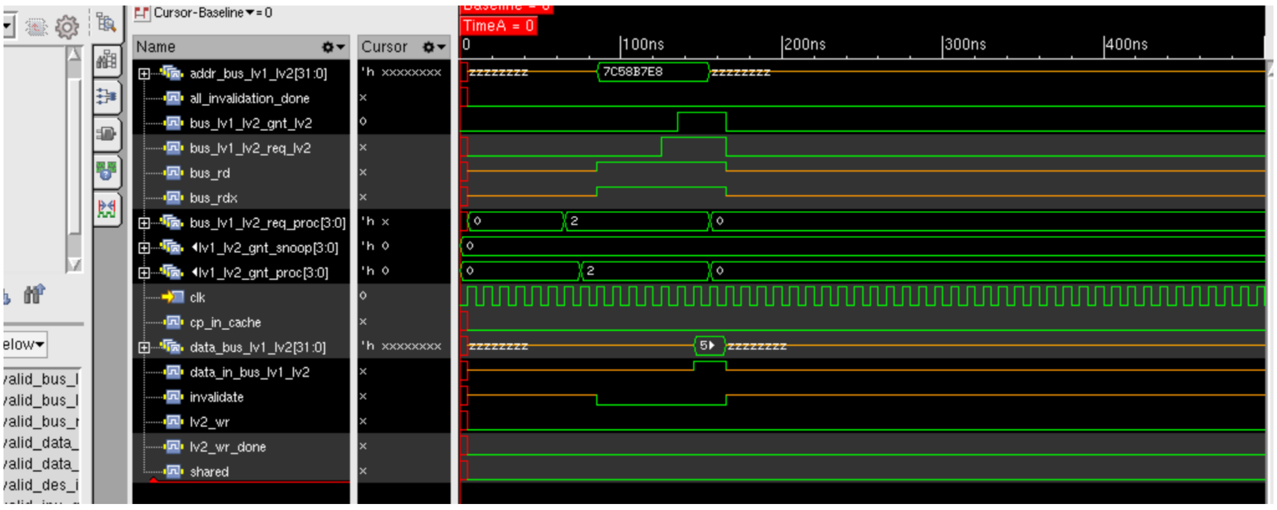
| Team Number | #16 |
|---|---|
| **Bug Number** | **#1** |
| **Bug Location** | line 107 in main_fun_lv1_dl.sv |
| **Bug Type** | Functional Bug |
| **SV Test Run to uncover the Bug** | wr_wr_rd : choosing processors randomly , we write first on one address followed write on the same address, followed by reading the same address |
| **Checker/Assertion Failed** | is in system_bus_interface: assert_valid_bus_rd_rdx |
| **Checker Description** | It is bus_rd and bus_rdx should not be asserted simulataneously |
| **Bug Description/ Debug Process** | In this case the second processor is writing to the same address so it asserts bus_rdx signal with intention to modify the data, but as seen from the waveform bus_rd is also getting asserted and is following similar pattern to bus_rdx throughout, which is shown by the assertion failing. This indicates there is a chance of erroneous initialization of bus_rd and checking the initialization of bus_rd and found the bug. |
| **Original Code** | assign bus_rd = bus_rdx_reg |
| **Code after Fix** | assign bus_rd = bus_rd_reg |



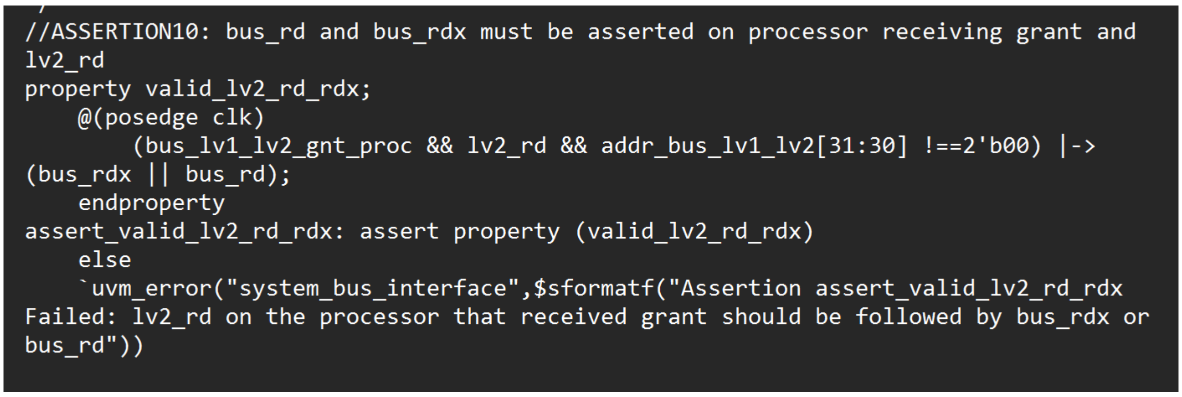
```
//ASSERTION14: bus_rd and bus_rdx should not be asserted simultaneously
property valid_bus_rd_rdx;
    @(posedge clk)
            not(bus_rd && bus_rdx);
        endproperty
assert_valid_bus_rd_rdx: assert property (valid_bus_rd_rdx)
    else
    `uvm_error("system_bus_interface",$sformatf("Assertion assert_valid_bus_rd_rdx
Failed: bus_rd and bus_rdx are asserted simultaneously"))
```



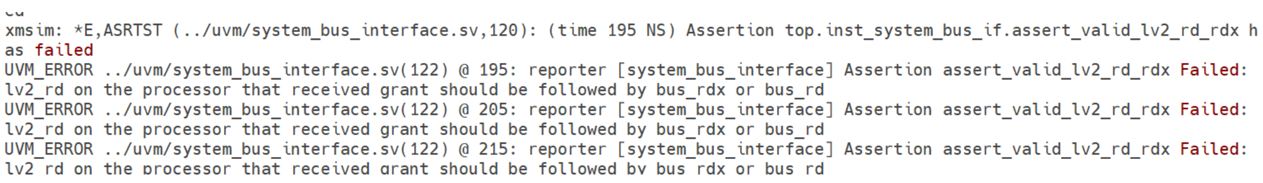
```
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,158): (time 2255 NS) Assertion top.inst_system_bus_if.assert_valid_bus_rd_rdx
has failed
UVM_ERROR ../uvm/system_bus_interface.sv(160) @ 2255: reporter [system_bus_interface] Assertion assert_valid_bus_rd_rdx Failed:
 bus_rd and bus_rdx are asserted simultaneously
UVM_ERROR ../uvm/system_bus_interface.sv(160) @ 2265: reporter [system_bus_interface] Assertion assert_valid_bus_rd_rdx Failed:
 bus_rd and bus_rdx are asserted simultaneously
UVM_ERROR ../uvm/system_bus_interface.sv(160) @ 2275: reporter [system_bus_interface] Assertion assert_valid_bus_rd_rdx Failed:
 bus_rd and bus_rdx are asserted simultaneously
```



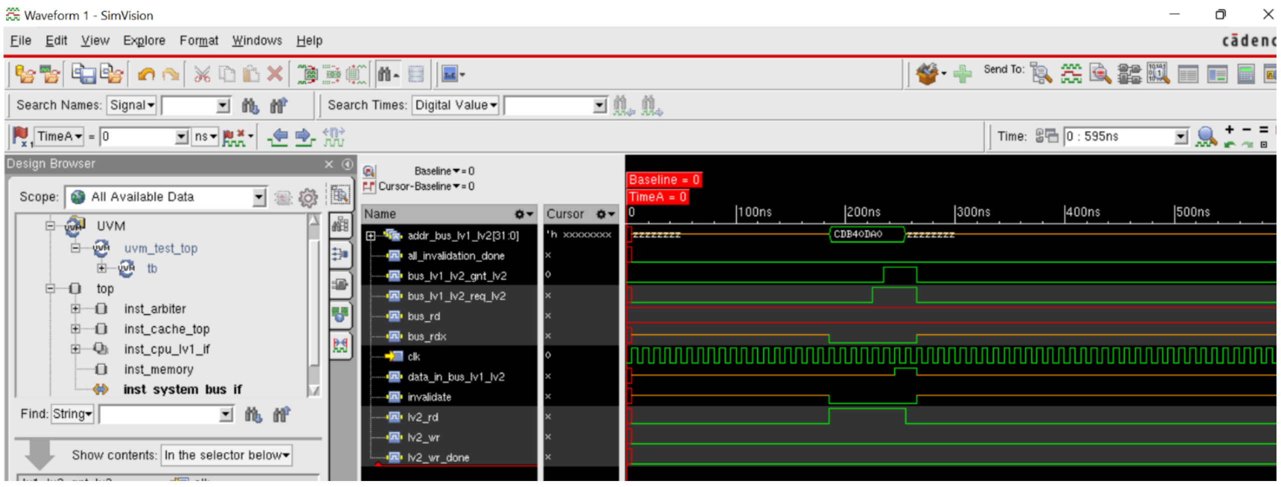
| Team Number | #16 |
|---|---|
| Bug Number | #2 |
| Bug Location | Error in line 136: In main_fun_lv1_dl |
| Bug Type | Functional Bug |
| SV Test Run to uncover the Bug | rd_rd_wr : choosing processors randomly , we read first read on one address followed by read on the same address, followed by writing the same address |
| Checker/Assertion Failed | is in system_bus_interface: assert_valid_lv2_rd_rdx |
| Checker Description | It is lv2_rd on the processor that received grant should be followed by bus_rdx or bus_rd. |
| Bug Description/ Debug Process | In this case the first read to the address is a miss and while observing the waveforms we could see the bus_rd is not changing for the entire simulation. This resulted in the assertion failing as the processor received grant for read but the bus_rd remained unchanged as it was uninitialized. Looking at the initialization part we observed the bus_rd_reg is not initialized. After making the correction looking at the waveforms the bus_rd was observed to be functioning correctly. |
| Original Code | bus_rdx_reg < = 1’bz |
| Code after Fix | bus_rd_reg <= 1’bz |



```
//ASSERTION10: bus_rd and bus_rdx must be asserted on processor receiving grant and
lv2_rd
property valid_lv2_rd_rdx;
    @(posedge clk)
        (bus_lv1_lv2_gnt_proc && lv2_rd && addr_bus_lv1_lv2[31:30] !==2'b00) |->
(bus_rdx || bus_rd);
    endproperty
assert_valid_lv2_rd_rdx: assert property (valid_lv2_rd_rdx)
    else
    `uvm_error("system_bus_interface",$sformatf("Assertion assert_valid_lv2_rd_rdx
Failed: lv2_rd on the processor that received grant should be followed by bus_rdx or
bus_rd"))
```



```
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,120): (time 195 NS) Assertion top.inst_system_bus_if.assert_valid_lv2_rd_rdx h
as failed
UVM_ERROR ../uvm/system_bus_interface.sv(122) @ 195: reporter [system_bus_interface] Assertion assert_valid_lv2_rd_rdx Failed:
lv2_rd on the processor that received grant should be followed by bus_rdx or bus_rd
UVM_ERROR ../uvm/system_bus_interface.sv(122) @ 205: reporter [system_bus_interface] Assertion assert_valid_lv2_rd_rdx Failed:
lv2_rd on the processor that received grant should be followed by bus_rdx or bus_rd
UVM_ERROR ../uvm/system_bus_interface.sv(122) @ 215: reporter [system_bus_interface] Assertion assert_valid_lv2_rd_rdx Failed:
lv2_rd on the processor that received grant should be followed by bus_rdx or bus_rd
```



Search Names: Signal

Search Times: Digital Value

TimeA = 0 ns

Time: 0 : 595ns

Scope: All Available Data

Find: String

Show contents: In the selector below

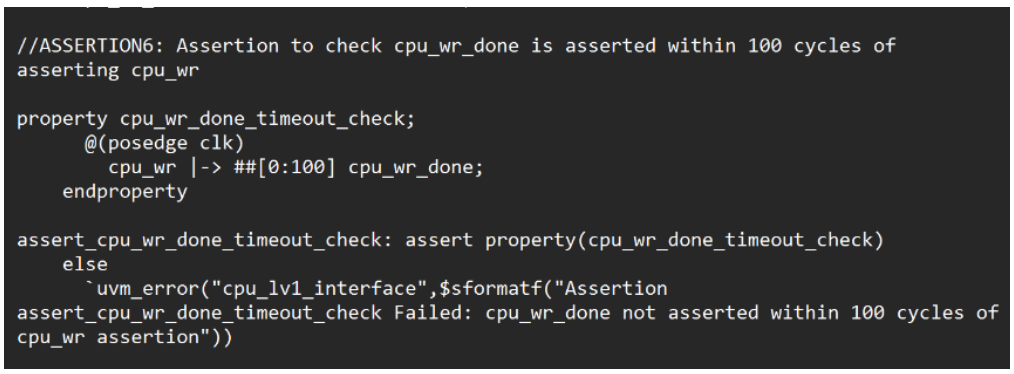
Baseline = 0

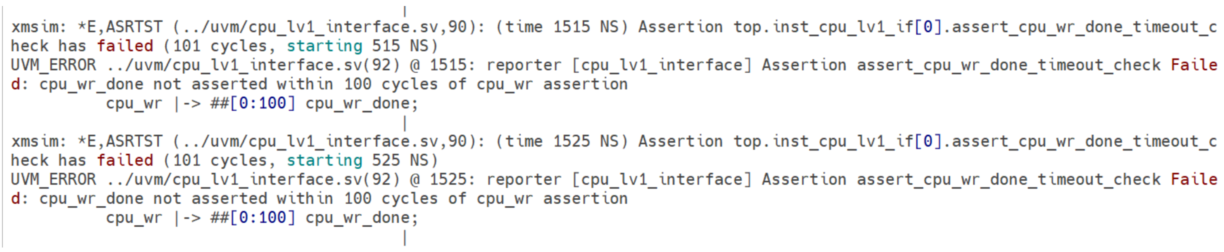
Cursor-Baseline = 0

Baseline = 0

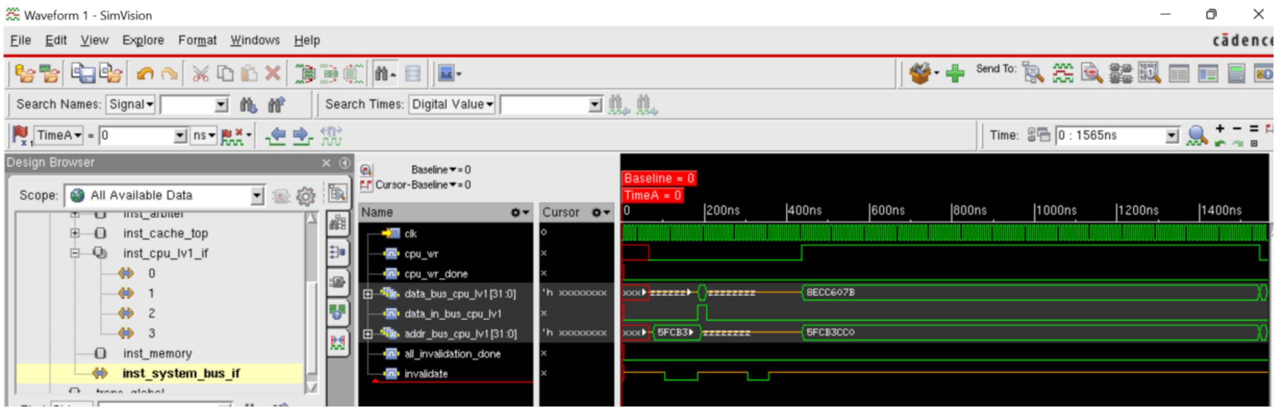
TimeA = 0

| Team Number | #16 |
|---|---|
| Bug Number | #3 |
| **Bug Location** | Error line 221: In main_fun_lv1_dl.sv |
| **Bug Type** | Functional Bug |
| **SV Test Run to uncover the Bug** | invalid_check: in this test we are trying to check the invalidation function of MESI protocol by making a processor invalidate the data which was earlier in shared state. For this test case processor selection is done randomly and we made sure the two processors are reading the same address by adding a constraint in our randomization. The sequence of operations in this test are read on processor1 , then read on processor2 on the same address. Finally, the processor1 is made to write to the same address. Before the write operation the address block in processor 1 and 2 is in shared state and write operation is invalidating the other processor which is not writing. |
| **Checker/Assertion Failed** | in cpu_lv1_interface : assert_cpu_wr_done_timeout_check |
| **Checker Description** | Cpu_wr_done is not asserted within 100 cycles of cpu write assertion. Whenever cpu write is made high the data is expected to reach cpu from lv1 or lv2 or from memory and then the cpu write done should be made high. |
| **Bug Description/ Debug Process** | The assertion indicated the cpu_wr_done is not asserted even after cpu_wr is asserted. This resulted in failure of test and from waveforms we have seen that cpu_wr_done failed to get asserted even after cpu_wr is asserted. As per HAS when the cache data is in shared condition cpu_wr_done is asserted after cpu_wr after invalidating the copies of the block and making all_invalidation_done high and copying the data into cache_var. From the waveform we can see that invalidate is in high impedance which is making all_invalidation_done not getting asserted. Going back and looking at the code for write hit in shared condition we observed invalidate_reg is not made high. |
| **Original Code** | invalidate_reg <= 1'bz |
| **Code after Fix** | invalidate_reg <= 1'b1 |


```
//ASSERTION6: Assertion to check cpu_wr_done is asserted within 100 cycles of
asserting cpu_wr

property cpu_wr_done_timeout_check;
      @(posedge clk)
        cpu_wr |-> ##[0:100] cpu_wr_done;
    endproperty

assert_cpu_wr_done_timeout_check: assert property(cpu_wr_done_timeout_check)
    else
      `uvm_error("cpu_lv1_interface",$sformatf("Assertion
assert_cpu_wr_done_timeout_check Failed: cpu_wr_done not asserted within 100 cycles of
cpu_wr assertion"))
```

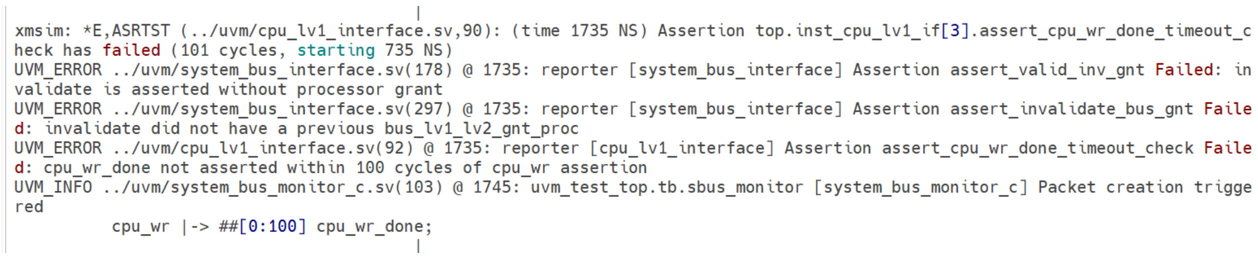


```
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,90): (time 1515 NS) Assertion top.inst_cpu_lv1_if[0].assert_cpu_wr_done_timeout_c
heck has failed (101 cycles, starting 515 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(92) @ 1515: reporter [cpu_lv1_interface] Assertion assert_cpu_wr_done_timeout_check Faile
d: cpu_wr_done not asserted within 100 cycles of cpu_wr assertion
          cpu_wr |-> ##[0:100] cpu_wr_done;
                                            |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,90): (time 1525 NS) Assertion top.inst_cpu_lv1_if[0].assert_cpu_wr_done_timeout_c
heck has failed (101 cycles, starting 525 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(92) @ 1525: reporter [cpu_lv1_interface] Assertion assert_cpu_wr_done_timeout_check Faile
d: cpu_wr_done not asserted within 100 cycles of cpu_wr assertion
          cpu_wr |-> ##[0:100] cpu_wr_done;
                                            |
```

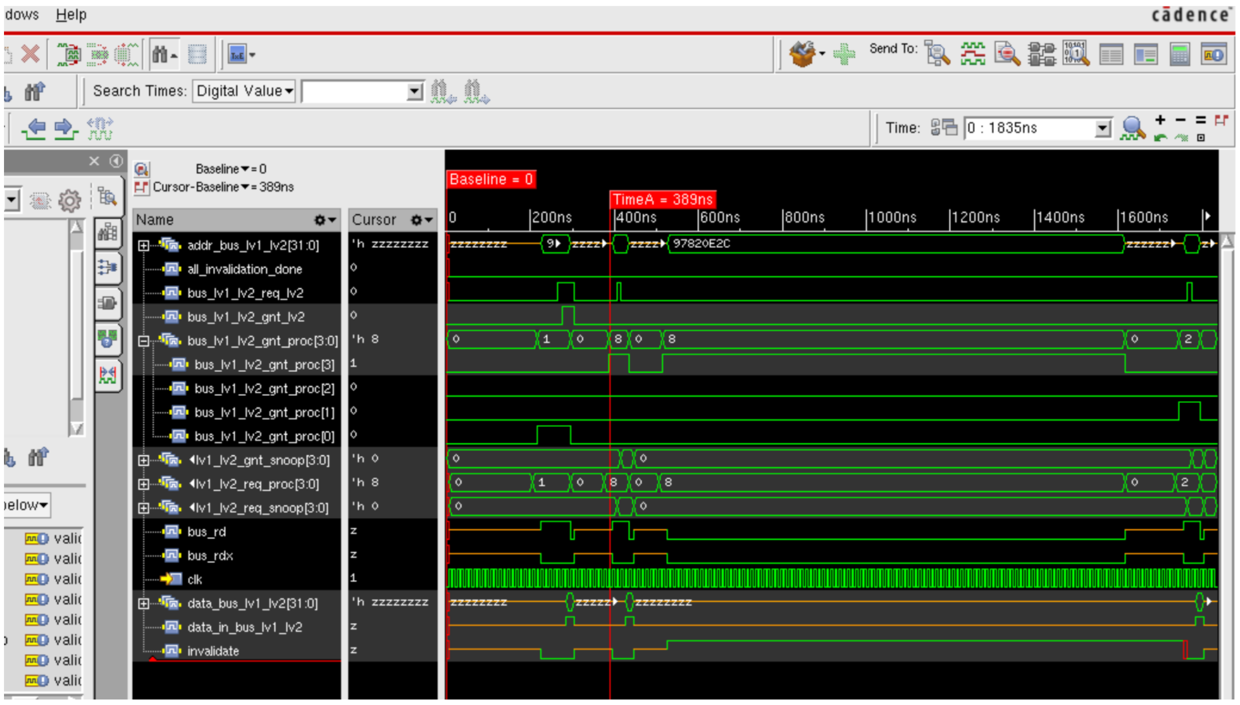


| Team Number | #16 |
|---|---|
| Bug Number | #4 |
| Bug Location | Error line 353: In main_fun_lv1_dl.sv |
| Bug Type | Functional Bug |
| SV Test Run to uncover the Bug | inv_to_shared_check: In this test we are trying to check the invalidation to shared function of MESI protocol. For this test case processor selection is done randomly and we made sure the two processors are reading the same address by adding a constraint in our randomization. The sequence of operations in this test are read on processor1 , then read on processor2 on the same address, then write on processor2 on same address. Finally, read o processor1 on same address. Before the final read operation the address block in processor 1 is in invalid state and final read operation is making the address to be in shared state. |
| Checker/Assertion Failed | in system_bus_interface : assert_ valid_inv_gnt and<br><br>in cpu_lv1_interface : assert cpu_wr_done_timeout_check |
| Checker Description | Invalidate is asserted without processor grant. Cpu_wr_done is not asserted within 100 cycles of cpu write assertion. Whenever cpu write is made high the data is expected to reach cpu from lv1 or lv2 or from memory and then the cpu write done should be made high. |
| Bug Description/ Debug Process | The assertions as indicated above are failing  we opened the simvision to check for the failing conditions. It is observed that even after invalidate is getting asserted the invalidation_done is not getting asserted.  According to HAS when invalidate is asserted and grant processor is deasserted the invalidation done should be high indicating no copy in cache. So we started looking for the code fragment for snoop side respons to invalidate when no copy is present and found that invalidate_reg is asserted instead of invalidation_done. |
| Original Code | invalidate_reg <= 1’b1 |
| Code after Fix | invalidation_done <= 1’b1 |

```
//ASSERTION16: Invalidate should not be asserted without proccessor getting grant in
previous cycle
property valid_inv_gnt;
@(posedge clk)
invalidate |-> $past(bus_lv1_lv2_gnt_proc) ;
endproperty
assert_valid_inv_gnt: assert property (valid_inv_gnt)
    else
     `uvm_error("system_bus_interface",$sformatf("Assertion assert_valid_inv_gnt
Failed: invalidate is asserted without processor grant"))
```



```
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,90): (time 1735 NS) Assertion top.inst_cpu_lv1_if[3].assert_cpu_wr_done_timeout_c
heck has failed (101 cycles, starting 735 NS)
UVM_ERROR ../uvm/system_bus_interface.sv(178) @ 1735: reporter [system_bus_interface] Assertion assert_valid_inv_gnt Failed: in
validate is asserted without processor grant
UVM_ERROR ../uvm/system_bus_interface.sv(297) @ 1735: reporter [system_bus_interface] Assertion assert_invalidate_bus_gnt Faile
d: invalidate did not have a previous bus_lv1_lv2_gnt_proc
UVM_ERROR ../uvm/cpu_lv1_interface.sv(92) @ 1735: reporter [cpu_lv1_interface] Assertion assert_cpu_wr_done_timeout_check Faile
d: cpu_wr_done not asserted within 100 cycles of cpu_wr assertion
UVM_INFO ../uvm/system_bus_monitor_c.sv(103) @ 1745: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation trigge
red
          cpu_wr |-> ##[0:100] cpu_wr_done;
```



Search Times: Digital Value

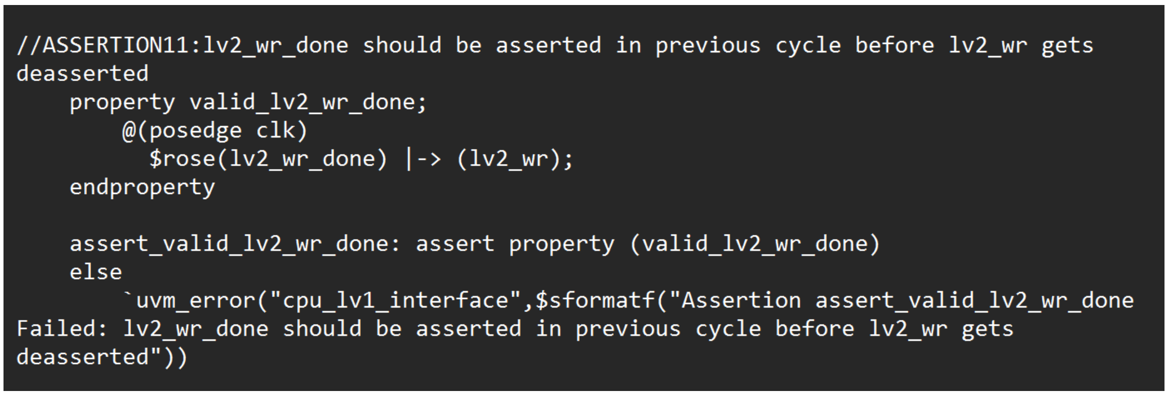
Time: 0 : 1835ns

Baseline = 0
Cursor-Baseline = 389ns

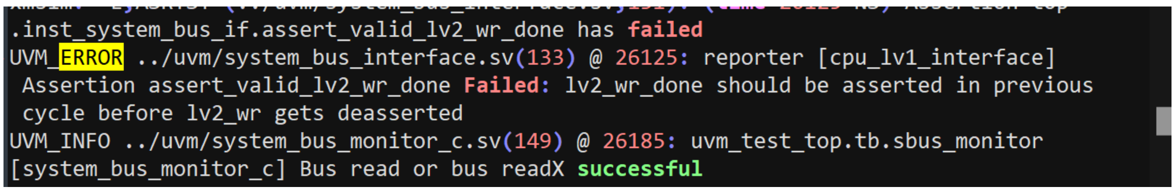
Baseline = 0
TimeA = 389ns

| Name | Cursor |
|---|---|
| addr_bus_lv1_lv2[31:0] | 'h zzzzzzzz |
| all_invalidation_done | 0 |
| bus_lv1_lv2_req_lv2 | 0 |
| bus_lv1_lv2_gnt_lv2 | 0 |
| bus_lv1_lv2_gnt_proc[3:0] | 'h 8 |
| bus_lv1_lv2_gnt_proc[3] | 1 |
| bus_lv1_lv2_gnt_proc[2] | 0 |
| bus_lv1_lv2_gnt_proc[1] | 0 |
| bus_lv1_lv2_gnt_proc[0] | 0 |
| lv1_lv2_gnt_snoop[3:0] | 'h 0 |
| lv1_lv2_req_proc[3:0] | 'h 8 |
| lv1_lv2_req_snoop[3:0] | 'h 0 |
| bus_rd | z |
| bus_rdx | z |
| clk | 1 |
| data_bus_lv1_lv2[31:0] | 'h zzzzzzzz |
| data_in_bus_lv1_lv2 | z |
| invalidate | z |

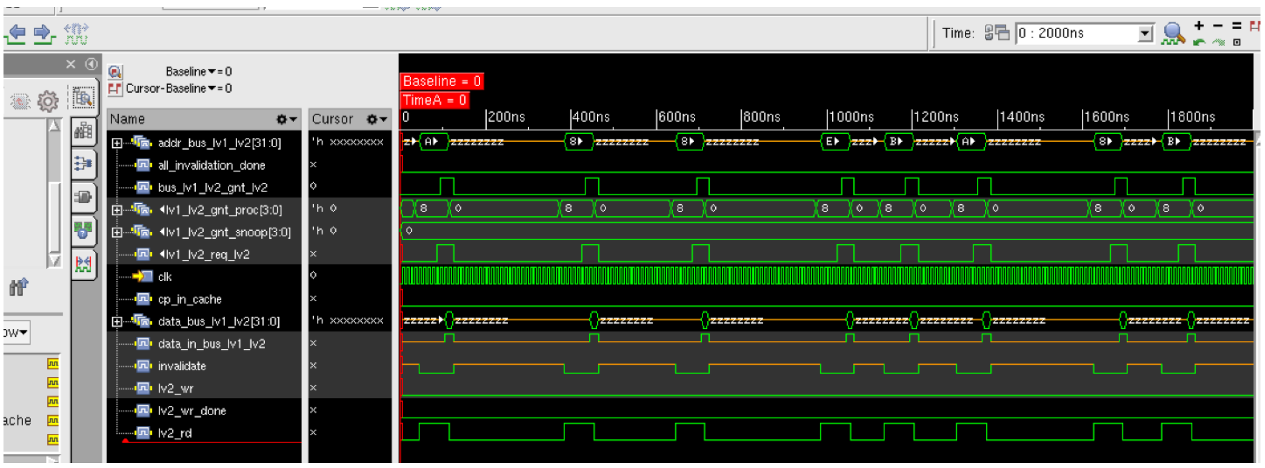
| **Team Number** | **#16** |
|---|---|
| **Bug Number** | **#5** |
| **Bug Location** | Error line 193: In main_fun_lv1_dl.sv |
| **Bug Type** | Functional Bug |
| **SV Test Run to uncover the Bug** | multi_rd_wr_rd_test : choosing processors randomly , we read first do multiple reads on one index followed by write on the same index, followed by multiple write on the same index of address. |
| **Checker/Assertion Failed** | is in system_bus_interface: assert_valid_lv2_wr_done |
| **Checker Description** | It is lv2_wr_done should be asserted in previous cycle before lv2_wr gets deasserted. |
| **Bug Description/ Debug Process** | In this case the first read to the address is a miss and multiple reads on the same index resulted in no free block. This followed by a write results in need for replacing the block for further reads on the same index. After the assertion has failed by looking at the waveform we observed the lv2_wr is never getting asserted. This led to looking at the part of code that requires replacement on a read miss. It is observed that lv2_wr is checked instead of lv2_wr_done not giving enough time for writing the missed value. After making the correction looking at the waveforms the lv2_wr and lv2_wr_done was observed to be functioning correctly. |
| **Original Code** | If(lv2_wr) begin |
| **Code after Fix** | If(lv2_wr_done) begin |



```
//ASSERTION11:lv2_wr_done should be asserted in previous cycle before lv2_wr gets
deasserted
    property valid_lv2_wr_done;
        @(posedge clk)
          $rose(lv2_wr_done) |-> (lv2_wr);
    endproperty

    assert_valid_lv2_wr_done: assert property (valid_lv2_wr_done)
    else
        `uvm_error("cpu_lv1_interface",$sformatf("Assertion assert_valid_lv2_wr_done
Failed: lv2_wr_done should be asserted in previous cycle before lv2_wr gets
deasserted"))
```



```
.inst_system_bus_if.assert_valid_lv2_wr_done has failed
UVM_ERROR ../uvm/system_bus_interface.sv(133) @ 26125: reporter [cpu_lv1_interface]
 Assertion assert_valid_lv2_wr_done Failed: lv2_wr_done should be asserted in previous
 cycle before lv2_wr gets deasserted
UVM_INFO ../uvm/system_bus_monitor_c.sv(149) @ 26185: uvm_test_top.tb.sbus_monitor
[system_bus_monitor_c] Bus read or bus readX successful
```

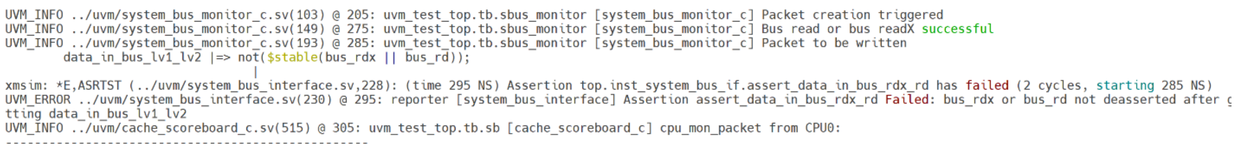


Baseline = 0
Cursor-Baseline = 0

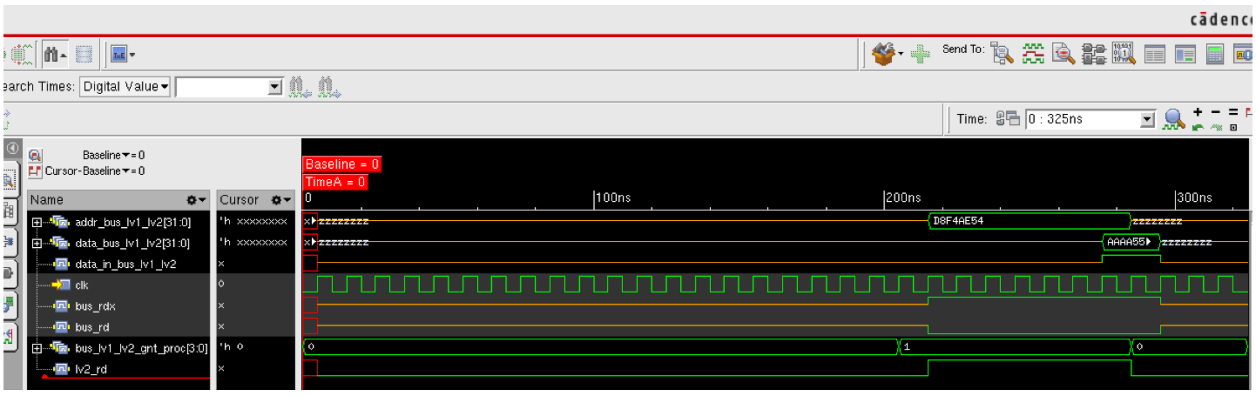
| Name | Cursor |
|---|---|
| addr_bus_lv1_lv2[31:0] | 'h xxxxxxxx |
| all_invalidation_done | x |
| bus_lv1_lv2_gnt_lv2 | 0 |
| lv1_lv2_gnt_proc[3:0] | 'h 0 |
| lv1_lv2_gnt_snoop[3:0] | 'h 0 |
| lv1_lv2_req_lv2 | x |
| clk | 0 |
| cp_in_cache | x |
| data_bus_lv1_lv2[31:0] | 'h xxxxxxxx |
| data_in_bus_lv1_lv2 | x |
| invalidate | x |
| lv2_wr | x |
| lv2_wr_done | x |
| lv2_rd | x |

| **Team Number** | **#16** |
|---|---|
| **Bug Number** | #6 |
| **Bug Location** | Error line 259: In main_fun_lv1_dl.sv |
| **Bug Type** | Functional Bug |
| **SV Test Run to uncover the Bug** | write_miss_dcache: choosing processors randomly , we write to the dcache causing the first write to a cache as miss. |
| **Checker/Assertion Failed** | is in system_bus_interface:  assert_data_in_bus_rdx_rd |
| **Checker Description** | It is . lv2_rd, bus_rdx and bus_rd should be deasserted after getting data_in_bus_lv1_lv2. |
| **Bug Description/ Debug Process** | In this case the first write to the randomly chosen address is a miss. This write resulted in the failure of assertion indicated above. According to HAS once grant is obtained lv2_rd and either of bus_rd or bus_rdx are asserted until data_in_bus_lv1_lv2 is high. This showed that lv_rd, bus_rd and bus_rdx one of the signals didn't function as per the requirement. After the assertion has failed by looking at the waveform we observed the bus_rdx is never getting deasserted. This led to looking at the part of code that corresponds to a simple write miss. It is observed that bus_rd is deasserted twice instead of bus_rdx resulting in anomalous behavior. After making the correction looking at the waveforms the bus_rd and bus_rdx were observed to be functioning correctly. |
| **Original Code** | bus_rd_reg <= 1'b0 |
| **Code after Fix** | bus_rdx_reg <= 1'b0 |

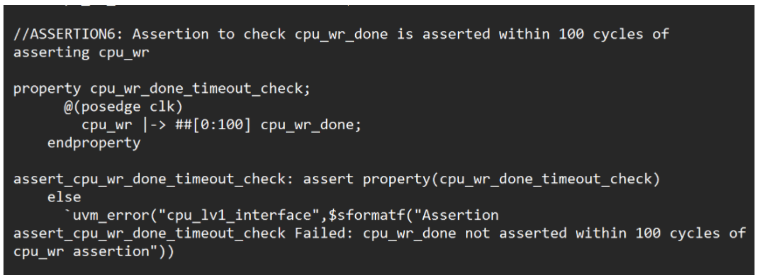
```
//ASSERTION21: lv2_rd, bus_rdx or bus_rd should be deasserted after data_in_bus_lv1
_lv2 being asserted.
    property data_in_bus_rdx_rd;
        @(posedge clk)
      data_in_bus_lv1_lv2 |=> not($stable(lv2_rd || bus_rdx || bus_rd));
    endproperty

    assert_data_in_bus_rdx_rd: assert property (data_in_bus_rdx_rd)
    else
        `uvm_error("system_bus_interface",$sformatf("Assertion
assert_data_in_bus_rdx_rd Failed: lv2_rd or bus_rdx or bus_rd not deasserted after
getting data_in_bus_lv1_lv2 "))
```



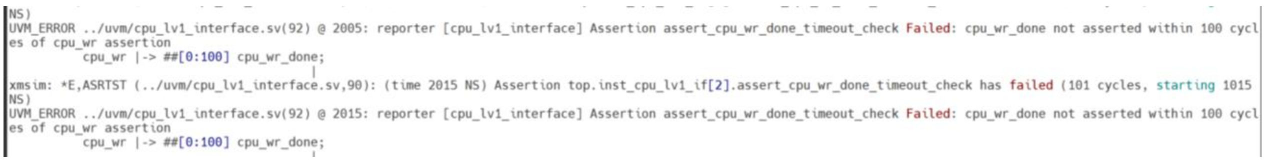
```
UVM_INFO ../uvm/system_bus_monitor_c.sv(103) @ 205: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(149) @ 275: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Bus read or bus readX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(193) @ 285: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
        data_in_bus_lv1_lv2 |=> not($stable(bus_rdx || bus_rd));
                                  |
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,228): (time 295 NS) Assertion top.inst_system_bus_if.assert_data_in_bus_rdx_rd has failed (2 cycles, starting 285 NS)
UVM_ERROR ../uvm/system_bus_interface.sv(230) @ 295: reporter [system_bus_interface] Assertion assert_data_in_bus_rdx_rd Failed: bus_rdx or bus_rd not deasserted after g
tting data_in_bus_lv1_lv2
UVM_INFO ../uvm/cache_scoreboard_c.sv(515) @ 305: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU0:
--------------------------------------------------
```


cādence
Search Times: Digital Value
Send To:
Time: 0 : 325ns
Baseline = 0
Cursor-Baseline = 0
Name
Cursor
Baseline = 0
TimeA = 0
0
100ns
200ns
300ns
addr_bus_lv1_lv2[31:0]
data_bus_lv1_lv2[31:0]
data_in_bus_lv1_lv2
clk
bus_rdx
bus_rd
bus_lv1_lv2_gnt_proc[3:0]
lv2_rd
'h xxxxxxxx
'h xxxxxxxx
x
0
x
x
'h 0
x
zzzzzzzz
D8F4AE54
zzzzzzzz
zzzzzzzz
AAAA55
zzzzzzzz
0
1
0

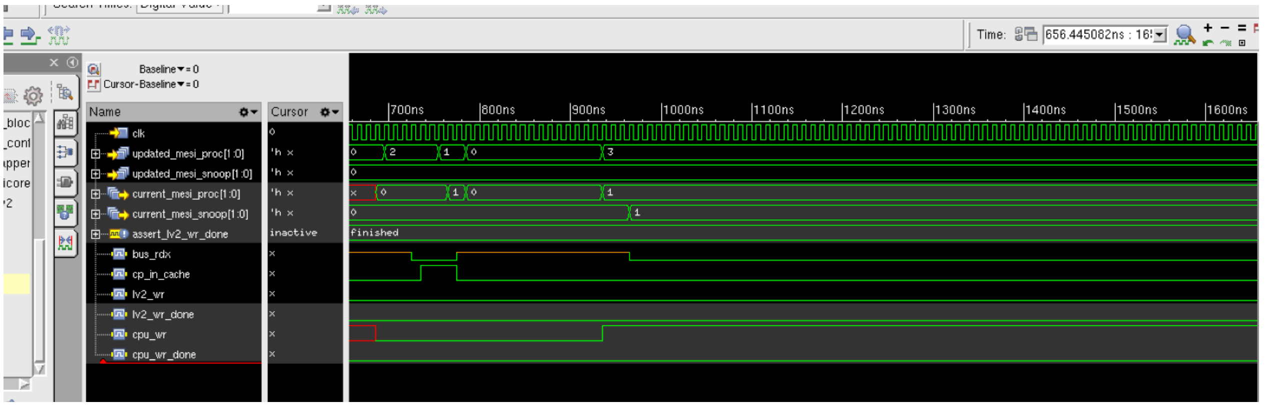
| Team Number | #16 |
|---|---|
| **Bug Number** | **#7** |
| **Bug Location** | Error line 313: In main_fun_lv1_dl.sv |
| **Bug Type** | Functional Bug |
| **SV Test Run to uncover the Bug** | modified_to_invalid_check: In this test we are trying to check the modified to invalid snoop side function of MESI protocol by making a processor invalidate the data which was earlier in modified state. For this test case processor1 writes on an address and followed by read on same address by processors1 and 2 . Finally, write on the same address by processor2 and looping this process 5 times for regression. The sequence of operations in this test are write on processor1 , then read on processor2 on the same address which makes it shared. Finally, the processor2 is made to write to the same address making it modified and next write by processor1 in the loop makes the block invalid in processor2. |
| **Checker/Assertion Failed** | In cpu_lv1_interface : assert cpu_wr_done_timeout_check. |
| **Checker Description** | Cpu_wr_done is not asserted within 100 cycles of cpu write assertion. Whenever cpu write is made high the data is expected to reach cpu from lv1 or lv2 or from memory and then the cpu write done should be made high. |
| **Bug Description/ Debug Process** | The assertion indicated the cpu_wr_done is not asserted even after cpu_wr is asserted. This resulted in failure of test and from waveforms we have seen that cpu_wr_done failed to get asserted even after cpu_wr is asserted. As per HAS when the cache data is in modified condition lv2_wr_done is asserted after lv2_wr invalidating the copies of the block and updating the current_mesi_snoop. From the waveform we observed the current_mesi_snoop is following the behavior of update_mesi_proc instead of update_mesi_snoop. Going back and looking at the code for write miss on snoop side we observed in modified condition `CACHE_CURRENT_MESI_SNOOP is assigned updated_mesi_proc instead of updated_mesi_snoop. |
| **Original Code** | `CACHE_CURRENT_MESI_SNOOP <= updated_mesi_proc; |
| **Code after Fix** | `CACHE_CURRENT_MESI_SNOOP <= updated_mesi_snoop; |



```
//ASSERTION6: Assertion to check cpu_wr_done is asserted within 100 cycles of
asserting cpu_wr

property cpu_wr_done_timeout_check;
      @(posedge clk)
        cpu_wr |-> ##[0:100] cpu_wr_done;
    endproperty

assert_cpu_wr_done_timeout_check: assert property(cpu_wr_done_timeout_check)
    else
      `uvm_error("cpu_lv1_interface",$sformatf("Assertion
assert_cpu_wr_done_timeout_check Failed: cpu_wr_done not asserted within 100 cycles of
cpu_wr assertion"))
```



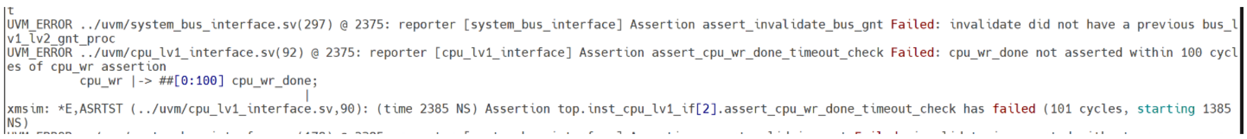
```
NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(92) @ 2005: reporter [cpu_lv1_interface] Assertion assert_cpu_wr_done_timeout_check Failed: cpu_wr_done not asserted within 100 cycl
es of cpu_wr assertion
          cpu_wr |-> ##[0:100] cpu_wr_done;
                                            |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,90): (time 2015 NS) Assertion top.inst_cpu_lv1_if[2].assert_cpu_wr_done_timeout_check has failed (101 cycles, starting 1015
NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(92) @ 2015: reporter [cpu_lv1_interface] Assertion assert_cpu_wr_done_timeout_check Failed: cpu_wr_done not asserted within 100 cycl
es of cpu_wr assertion
          cpu_wr |-> ##[0:100] cpu_wr_done;
```



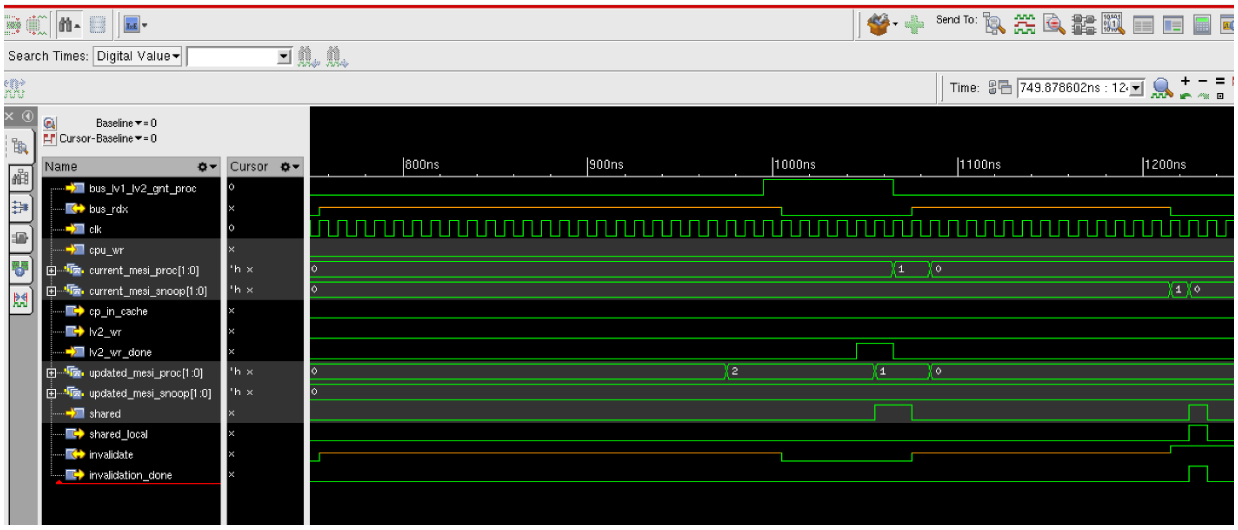
| Team Number | #16 |
|---|---|
| Bug Number | #8 |
| **Bug Location** | Error line 297: In main_fun_lv1_dl.sv |
| **Bug Type** | Functional Bug |
| **SV Test Run to uncover the Bug** | test_write_miss_snoop: In this test we are trying to check the write miss snoop side of the MESI protocol in case there is a block hit. For this test case processor0 writes on an address and followed by read on same address by processor1 and followed write by processor2, read by processor0 . Finally, write on the same address by processor2, followed by processor1. The sequence of operations in this test are write on processor0 making it exclusive , then read on processor1 on the same address which makes it shared. Finally, the processor2 is made to write to the same address making it modified and next write by processor1 making it modified. |
| **Checker/Assertion Failed** | in cpu_lv1_interface : assert_invalidate_bus_gnt and assert_cpu_wr_done_timeout_check. |
| **Checker Description** | The first one check if the invalidate has a bus processor grant in the previous cycle. Cpu_wr_done is not asserted within 100 cycles of cpu write assertion. Whenever cpu write is made high the data is expected to reach cpu from lv1 or lv2 or from memory and then the cpu write done should be made high. |
| **Bug Description/ Debug Process** | The assertion indicated the invalidate is asserted without previous grant and cpu_wr_done is not asserted even after cpu_wr is asserted. This resulted in failure of test and from waveforms we have seen that when invalidate gets asserted there is not grant in the previous cycle. As per HAS when the cached data is found on a write miss the current_mesi_snoop should be updated followed by the invalidation_done. From the waveform we observed the current_mesi_snoop is continuously invalid instead of update_mesi_proc before the invalidation done is asserted. Going back and looking at the code for write miss on snoop side when there is a block hit we observed in modified condition `CACHE_CURRENT_MESI_SNOOP is assigned INVALID instead of updated_mesi_proc. |
| **Original Code** | `CACHE_CURRENT_MESI_SNOOP <= INVALID; |
| **Code after Fix** | `CACHE_CURRENT_MESI_SNOOP <= updated_mesi_proc; |

```
//ASSERTION30: invalidate should be asserted only if previous bus_lv1_lv2_gnt_proc

assert_invalidate_bus_gnt: assert property (prop_sig1_before_sig2(bus_lv1_lv2
_gnt_proc,invalidate))
    else
     `uvm_error("system_bus_interface",$sformatf("Assertion assert_invalidate_bus_gnt
Failed: invalidate did not have a previous bus_lv1_lv2_gnt_proc"))
```

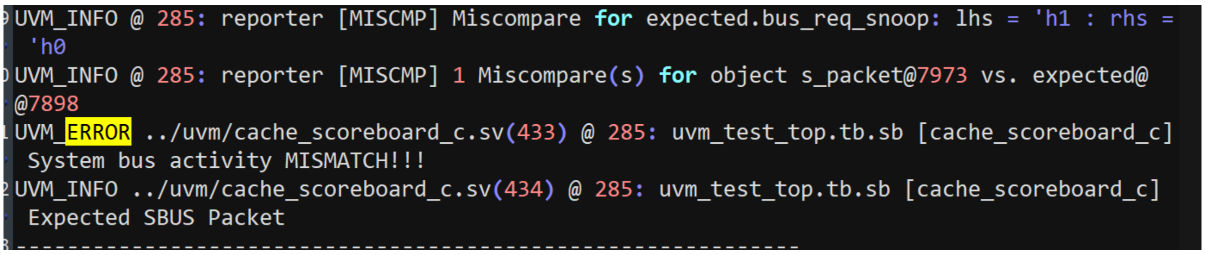
```
//ASSERTION6: Assertion to check cpu_wr_done is asserted within 100 cycles of
asserting cpu_wr

property cpu_wr_done_timeout_check;
      @(posedge clk)
        cpu_wr |-> ##[0:100] cpu_wr_done;
    endproperty

assert_cpu_wr_done_timeout_check: assert property(cpu_wr_done_timeout_check)
    else
      `uvm_error("cpu_lv1_interface",$sformatf("Assertion
assert_cpu_wr_done_timeout_check Failed: cpu_wr_done not asserted within 100 cycles of
cpu_wr assertion"))
```



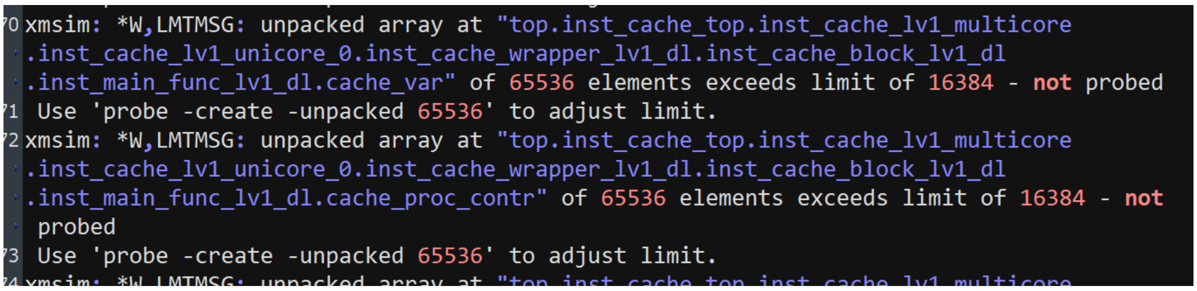
```
t
UVM_ERROR ../uvm/system_bus_interface.sv(297) @ 2375: reporter [system_bus_interface] Assertion assert_invalidate_bus_gnt Failed: invalidate did not have a previous bus_l
v1_lv2_gnt_proc
UVM_ERROR ../uvm/cpu_lv1_interface.sv(92) @ 2375: reporter [cpu_lv1_interface] Assertion assert_cpu_wr_done_timeout_check Failed: cpu_wr_done not asserted within 100 cycl
es of cpu_wr assertion
          cpu_wr |-> ##[0:100] cpu_wr_done;
                                          |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,90): (time 2385 NS) Assertion top.inst_cpu_lv1_if[2].assert_cpu_wr_done_timeout_check has failed (101 cycles, starting 1385
NS)
```



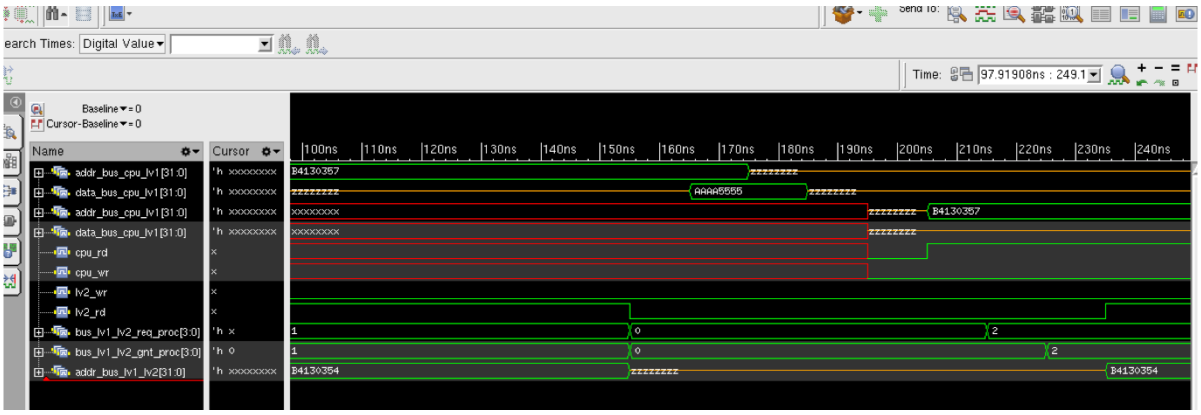
| Team Number | #16 |
|---|---|
| Bug Number | #9 |
| Bug Location | Error line 17: In main_fun_lv1_dl.sv |
| Bug Type | Functional Bug |
| SV Test Run to uncover the Bug | shared_test: choosing processors randomly with equal probability, we read using the first processor followed by a read by processor2 and finally a write by the first processor. |
| Checker/Assertion Failed | is in cache_scoreboard_.sv:System bus activity MISMATCH!!. |
| Checker Description | Different data is read and written than intended by the design. |
| Bug Description/ Debug Process | In this case the first read to the randomly chosen address is a miss. This is followed by another read making it shared and followed by a write. There, is a system bus mismatch and no other assertion has failed. Looking at the waveforms everything seemed to be fine except for the mismatch of the data that is logged in the run. There were array overflows observed in the log while instantiating cache top of design. Going back and looking at logic for write hit everything is logically correct but the updation of address might have resulted in reading from a wrong block and looking at the code we observed that the ADDR_WID is assigned to `INDEX_WID_LV1 instead of `ADDR_WID_LV1. |
| Original Code | parameter ADDR_WID = `INDEX_WID_LV1 , |
| Code after Fix | parameter ADDR_WID = `ADDR_WID_LV1, |



```
UVM_INFO @ 285: reporter [MISCMP] Miscompare for expected.bus_req_snoop: lhs = 'h1 : rhs =
 'h0
UVM_INFO @ 285: reporter [MISCMP] 1 Miscompare(s) for object s_packet@7973 vs. expected@
@7898
UVM_ERROR ../uvm/cache_scoreboard_c.sv(433) @ 285: uvm_test_top.tb.sb [cache_scoreboard_c]
 System bus activity MISMATCH!!!
UVM_INFO ../uvm/cache_scoreboard_c.sv(434) @ 285: uvm_test_top.tb.sb [cache_scoreboard_c]
 Expected SBUS Packet
------------------------------------------------------------
```



```
xmsim: *W,LMTMSG: unpacked array at "top.inst_cache_top.inst_cache_lv1_multicore
.inst_cache_lv1_unicore_0.inst_cache_wrapper_lv1_dl.inst_cache_block_lv1_dl
.inst_main_func_lv1_dl.cache_var" of 65536 elements exceeds limit of 16384 - not probed
 Use 'probe -create -unpacked 65536' to adjust limit.
xmsim: *W,LMTMSG: unpacked array at "top.inst_cache_top.inst_cache_lv1_multicore
.inst_cache_lv1_unicore_0.inst_cache_wrapper_lv1_dl.inst_cache_block_lv1_dl
.inst_main_func_lv1_dl.cache_proc_contr" of 65536 elements exceeds limit of 16384 - not
 probed
 Use 'probe -create -unpacked 65536' to adjust limit.
xmsim: *W,LMTMSG: unpacked array at "top.inst_cache_top.inst_cache_lv1_multicore
```



| Name | Cursor |
|---|---|
| addr_bus_cpu_lv1[31:0] | 'h xxxxxxxx |
| data_bus_cpu_lv1[31:0] | 'h xxxxxxxx |
| addr_bus_cpu_lv1[31:0] | 'h xxxxxxxx |
| data_bus_cpu_lv1[31:0] | 'h xxxxxxxx |
| cpu_rd | x |
| cpu_wr | x |
| lv2_wr | x |
| lv2_rd | x |
| bus_lv1_lv2_req_proc[3:0] | 'h x |
| bus_lv1_lv2_gnt_proc[3:0] | 'h 0 |
| addr_bus_lv1_lv2[31:0] | 'h xxxxxxxx |