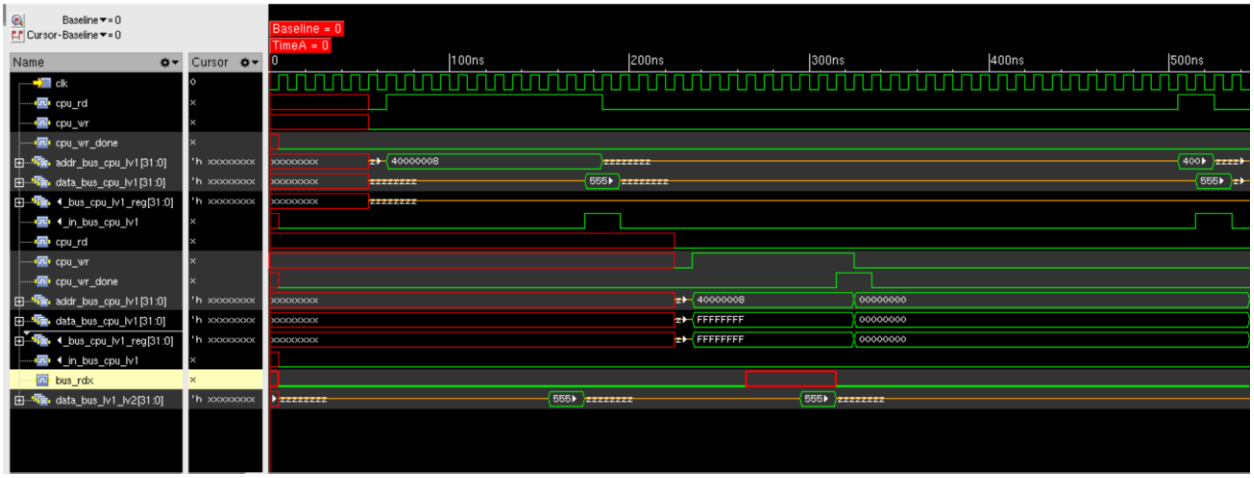
Team Number	#13
Bug Number	#1
Bug Location	File: main_func_lv1_dl.sv , Line No: 137
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_read_dcache
Checker/Assertion Failed	Scoreboard Error UVM_ERROR ../uvm/cache_scoreboard_c.sv(297) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] Data MISMATCH!!! expected = ffffffff received = 5555aaaa
Checker Description	When a write miss occurs, the bus_req_type needs to be set to “BUS_RDX” and this signal needs to be asserted. This should invalidate copies in any other caches. Failing to do so, will result in a stale copy of data, which results in data mismatch.
Bug Description/Debug Process	Sequence of operations: • Read Request to CPU1(Read miss) • Write Request to CPU0 (Write miss) • Read Request to CPU1 (Read miss) Initially, a read request to CPU1 with address as 32’h4000_0008 will result in a read miss, with the data being fetched from l2. This is followed by Write request to CPU0 with address 32’h4000_0008 and data as 32’hffff_ffff. This will generate a write miss and the bus_rdx signal needs to be asserted. However, this doesn’t happen in our case. Moreover, when a read request to CPU1 takes place, the data being read is 32’h5555_aaaa instead of 32’hffff_ffff. This implies, the copies in other CPUs were not invalidated resulting in a data mismatch.
Original Code	bus_rdx_reg <= 1'b0;
Code After Fix	bus_rdx_reg <= 1'bz;

Output from console showing scoreboard checker failure:

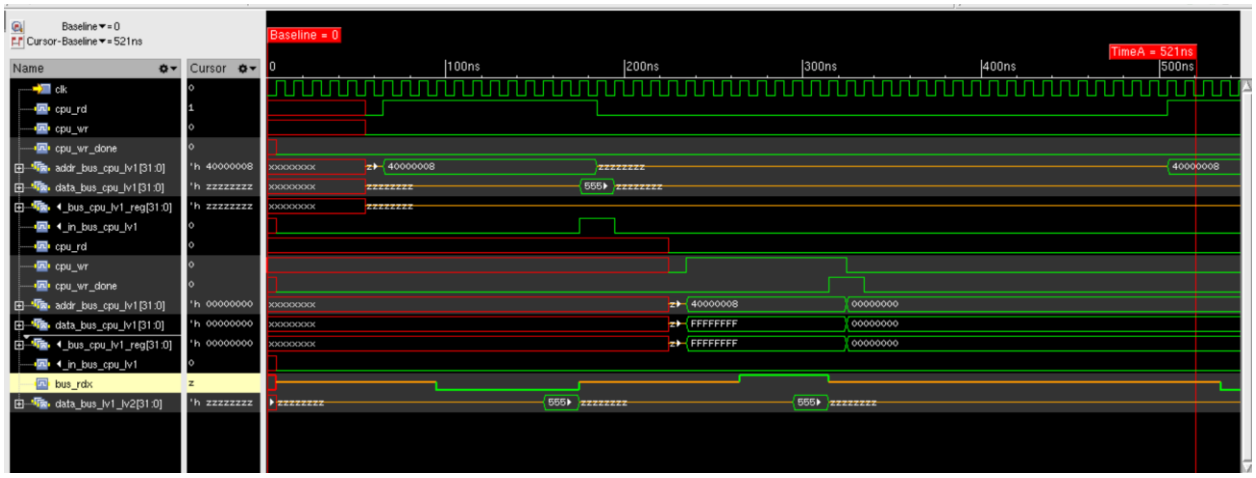
```
UVM_INFO ../uvm/cache_scoreboard_c.sv(523) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU1:
----------------------------------------------------
Name            Type              Size  Value
----------------------------------------------------
packet          cpu_mon_packet_c  -     @7626
  dat           integral          32    'h5555aaaa
  address       integral          32    'h40000008
  num_cycles    integral          32    'h0
  illegal       integral          1     'h0
  request_type  request_t         1     READ_REQ
  addr_type     addr_t            32    DCACHE
----------------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(293) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] CHECK_DATA CPU1
UVM_ERROR ../uvm/cache_scoreboard_c.sv(297) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] Data MISMATCH!!! expected = ffffffff received = 5555aaaa
UVM_ERROR ../uvm/cache_scoreboard_c.sv(414) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] Expected activity is not observed on the system bus
UVM_INFO ../uvm/cache_scoreboard_c.sv(416) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] Expected SBUS Packet
------------------------------------------------------------
Name                          Type             Size  Value
------------------------------------------------------------
expected                      sbus_packet_c    -     @7707
  bus_req_type                bus_req_t        32    BUS_RD
  bus_req_proc_num            bus_req_proc_t   32    REQ_PROC1
  req_address                 integral         32    'h40000008
  bus_req_snoop               integral         4     'h1
  req_serviced_by             serv_by_t        32    SERV_NONE
  rd_data                     integral         32    'hffffffff
  wr_data_snoop               integral         32    'hffffffff
  snoop_wr_req_flag           integral         1     'h1
  cp_in_cache                 integral         1     'h1
  shared                      integral         1     'h1
  service_time                integral         32    'h0
  proc_evict_dirty_blk_addr   integral         32    'h0
  proc_evict_dirty_blk_data   integral         32    'h0
  proc_evict_dirty_blk_flag   integral         1     'h0
------------------------------------------------------------
```

Waveform showing bus_rdx signal set to x during a write miss operation:



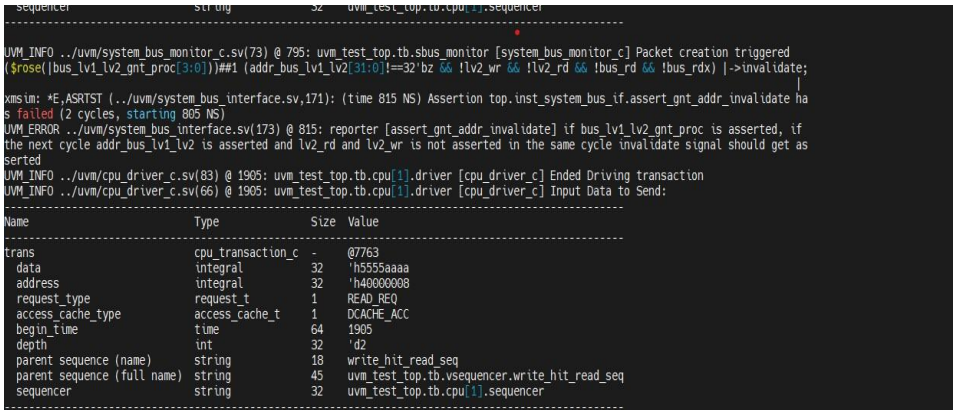
Waveform after fixing the bug:



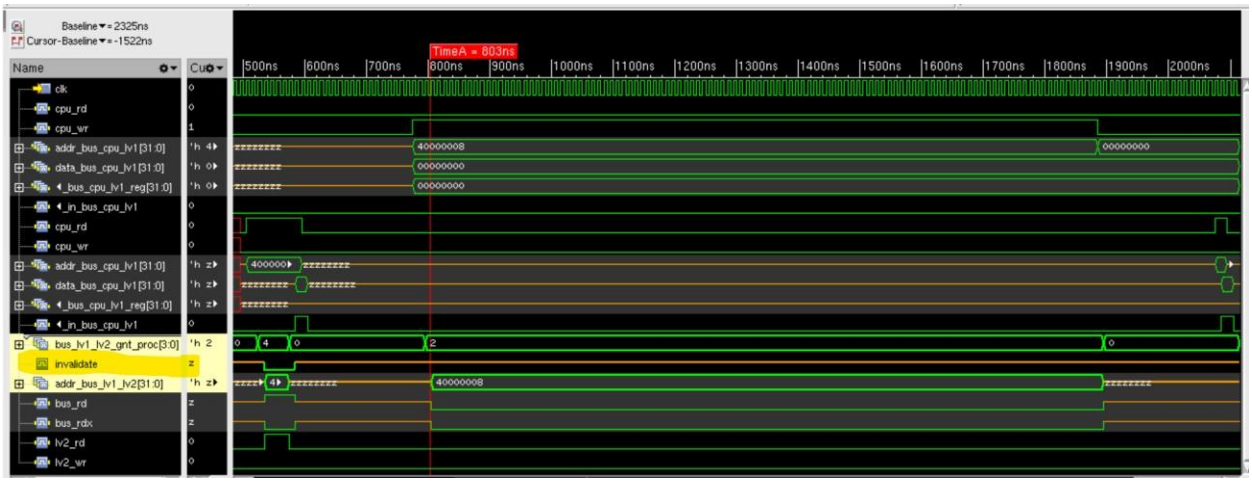
Team Number	#13
Bug Number	#2
Bug Location	File: main_func_lv1_dl.sv , Line No: 221
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_hit_read
Checker/Assertion Failed	Assertion failure UVM_INFO ../uvm/system_bus_monitor_c.sv(73) @ 795: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered ($rose(|bus_lv1_lv2_gnt_proc[3:0]))##1 (addr_bus_lv1_lv2[31:0]!==32'bz && !lv2_wr && !lv2_rd && !bus_rd && !bus_rdx) |->invalidate; | xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,171): (time 815 NS) Assertion top.inst_system_bus_if.assert_gnt_addr_invalidate ha s failed (2 cycles, starting 805 NS) UVM_ERROR ../uvm/system_bus_interface.sv(173) @ 815: reporter [assert_gnt_addr_invalidate] if bus_lv1_lv2_gnt_proc is asserted, if the next cycle addr_bus_lv1_lv2 is asserted and lv2_rd and lv2_wr is not asserted in the same cycle invalidate signal should get as serted
Checker Description	According to the HAS document, in a write hit scenario when the cache is in shared condition, it need to invalidate the copies in other caches. So for this to happen, the signal bus_lv1_lv2_gnt_proc once is made high, the address of the block to be invalidated is put in the addr_bus_lv1_lv2 in the next cycle after which the invalidate signal is asserted. Also, since it's a hit scenario, the request to bus_rd, bus_rdx, lv2_rd and lv2_wr is not asserted.
Bug Description/Debug Process	Sequence of operations: • Read request to CPU1 with address 32'h4000_0008.(read miss)

	• Read request to CPU2 with address 32’h4000_0008. (read miss) • Write Request to CPU1 with the same address and data as 32’h0000_0000.(write hit) • Read request to CPU2 with the same address.(read miss) The first read operation will be a read miss and as a result the data will be fetched from data_bus_lv1_lv2 and after one cycle data_in_bus_cpu_lv1 and the data will be 32’h5555_AAAA. Similarly, for the read operation in the CPU2, it will results in a read miss fetching the data 32’h5555_AAAA.It will result in the state as Shared state in both CPUs. This is followed by a write operation to CPU1 with address 32’h4000_0008 and data as 32’h0000_0000. Before writing the data, it should invalidate copies in other CPUs. However, the invalidate signal is not asserted during this write operation. The read operation in CPU2, should result in a read miss. But, it’s resulting in a read hit, with the data at address 32’h4000_0008 as 32’h5555_AAAA.
Original Code	invalidate_reg <= 1'bz; (write hit condition)
Code After Fix	invalidate_reg <= 1'b1;

Output from console showing assertion failure:

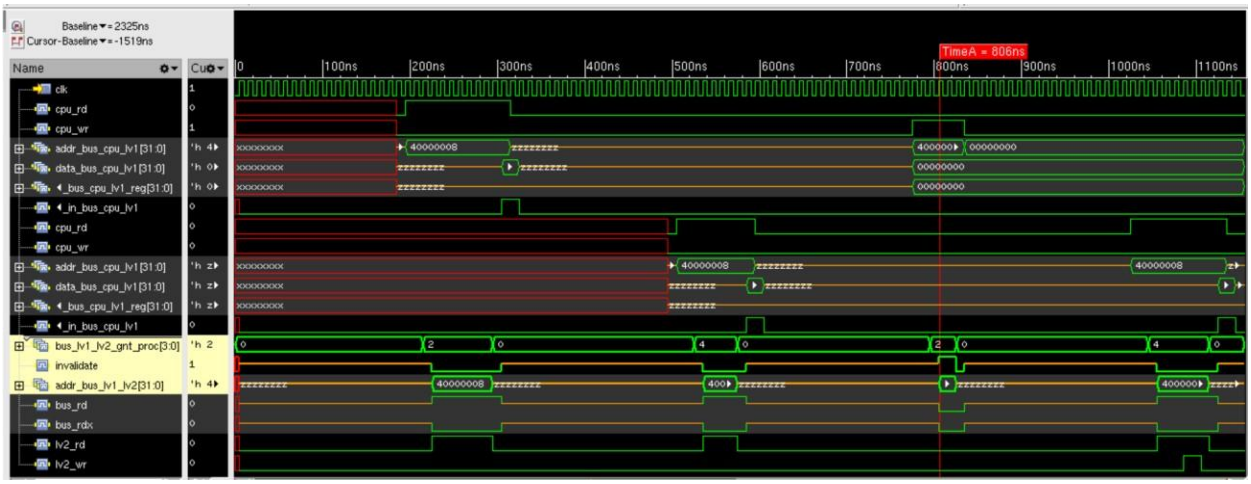


Waveform before fixing the bug:



Here, during the write hit operation, the invalidate signal is not asserted due to which the copies in other caches are not invalidated hence making the read operation a hit scenario.

Waveform after fixing the bug:



Here, during the read request to CPU1, it will result in a read miss.

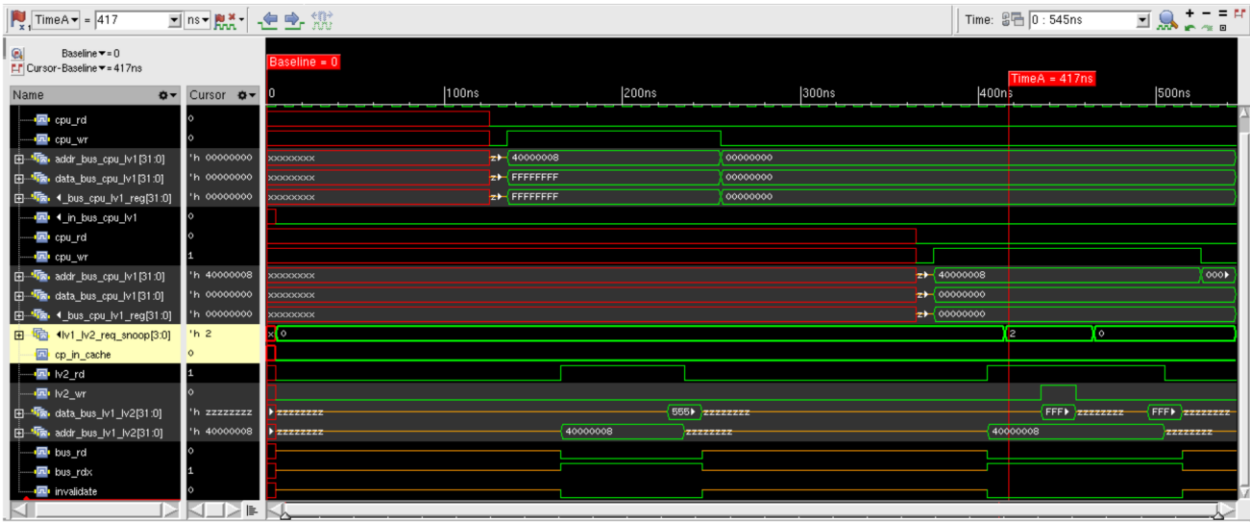
Team Number	#13
Bug Number	#3
Bug Location	File: main_func_lv1_dl.sv , Line No: 301
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_after_write
Checker/Assertion Failed	UVM_ERROR ../uvm/system_bus_interface.sv(162) @ 425: reporter [system_bus_interface] Assertion snoop_follow_cp_in_cache Failed: Once bus_lv1_lv2_req_proc is asserted, then cp_in_cache is not asserted
Checker Description	During a write miss scenario, when there are copies of the block in other caches, when bus_lv1_lv2_req_snoop snoop is asserted then the cp_in_cache should also be asserted.
Bug Description/Debug Process	Sequence of operations: • Write Request to CPU1 with the address 32'h4000_0008 and data as 32'hffff_ffff (write miss). • Write Request to CPU2 with the address 32'h4000_0008 and data as 32'h0000_0000 (write miss). The first write operation to CPU1 with the address as 32'h4000_0008 and data as 32'hffff_ffff will result in a write miss. This is followed by write operation to CPU2 with the same address but the data as 32'h0000_0000. The second write operation will also result in a miss. But since there is a copy of the block in CPU1, this will make the bus_lv1_lv2_req_snoop to be the processor 1 and cp_in_cache needs to be asserted. However, the cp_in_cache is not asserted.
Original Code	cp_in_cache <= 1'b0;
Code After Fix	cp_in_cache <= 1'b1;

Output from console showing assertion failure:

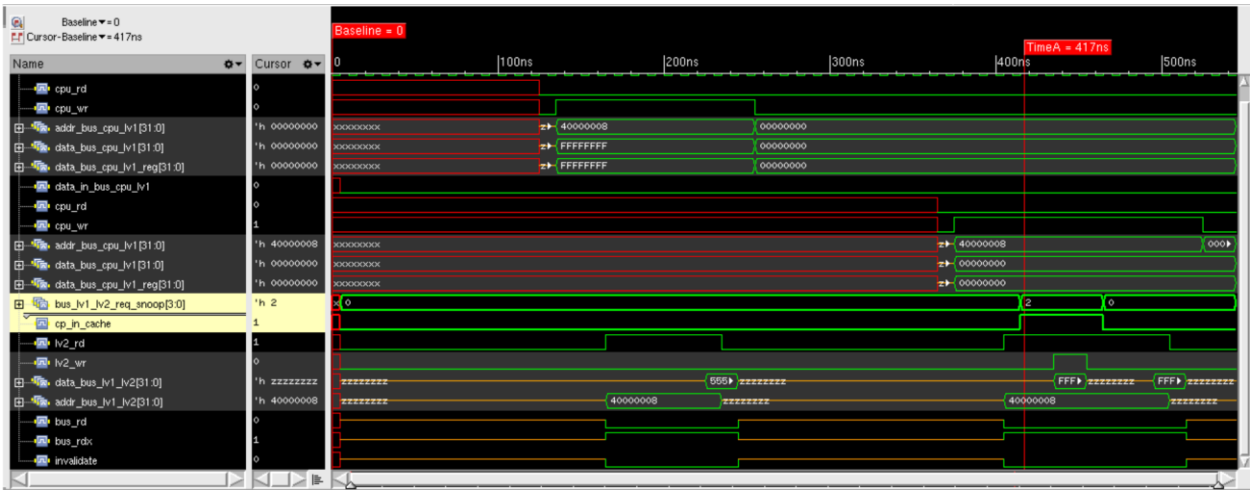
```
UVM_INFO ../uvm/cpu_driver_c.sv(83) @ 275: uvm_test_top.tb.cpu[1].driver [cpu_driver_c] Ended Driving transaction
UVM_INFO ../uvm/cpu_driver_c.sv(66) @ 275: uvm_test_top.tb.cpu[2].driver [cpu_driver_c] Input Data to Send:
----------------------------------------------------------------------------------------------------
Name                          Type                Size  Value
----------------------------------------------------------------------------------------------------
trans                         cpu_transaction_c   -     @7674
  data                        integral            32    'h0
  address                     integral            32    'h40000008
  request_type                request_t           1     WRITE_REQ
  access_cache_type           access_cache_t      1     DCACHE_ACC
  begin_time                  time                64    275
  depth                       int                 32    'd2
  parent sequence (name)      string              21    write_after_write_seq
  parent sequence (full name) string              48    uvm_test_top.tb.vsequencer.write_after_write_seq
  sequencer                   string              32    uvm_test_top.tb.cpu[2].sequencer
----------------------------------------------------------------------------------------------------

UVM_INFO ../uvm/system_bus_monitor_c.sv(73) @ 395: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,160): (time 425 NS) Assertion top.inst_system_bus_if.assert_snoop_follow_cp_in_cache has failed
UVM_ERROR ../uvm/system_bus_interface.sv(162) @ 425: reporter [system_bus_interface] Assertion snoop_follow_cp_in_cache Failed: Once bus_lv1_lv2_req_proc is asserted, then cp_in_cache is not asserted
UVM_INFO ../uvm/system_bus_monitor_c.sv(144) @ 495: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] BUS_RDX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(168) @ 505: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
UVM_INFO ../uvm/cache_scoreboard_c.sv(531) @ 525: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU2:
```

Waveform before fixing the bug:



Waveform after fixing the bug:



Team Number	#13
Bug Number	#4
Bug Location	File: main_func_lv1_dl.sv , Line No: 176
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_read_dcache
Checker/Assertion Failed	UVM_ERROR ../uvm/system_bus_interface.sv(184) @ 335: reporter [system_bus_interface] Assertion rd_rdx_assert_lv2_rd Failed: If either bus_rd or bus_rdx is asserted,then lv2_rd is asserted
Checker Description	As long as bus_rd or bus_rdx is asserted, lv2_rd should be asserted indicating a read miss scenario. property rd_rdx_assert_lv2_rd; @(posedge clk) (bus_rd || bus_rdx)|->lv2_rd; endproperty assert_rd_rdx_assert_lv2_rd: assert property(rd_rdx_assert_lv2_rd) else `uvm_error("system_bus_interface",$sformatf("Assertion rd_rdx_assert_lv2_rd Failed: If either bus_rd or bus_rdx is asserted,then lv2_rd is asserted")) endinterface
Bug Description/Debug Process	Sequence of operations: • Read Request to CPU1 with the address 32'h4000_0008 (read miss). During a read request for CPU1 with address as 32'h4000_0008, it will result in a read miss. This will assert the signal bus_rd and lv2_rd. However, the assertion got failed. After further inspection from the waveform, we found that the bus_rd was asserted for one additional cycle than the lv2_rd. Though there are no waveforms from the HAS documents to support that there shouldn't be no assertion of bus_rd for an additional cycle. During the read miss scenario, it has been mentioned that the lv2_rd and bus_rd is asserted together. Moreover, the reason for asserting bus_rd is for reading a block from L2. Once, the l2_rd signal

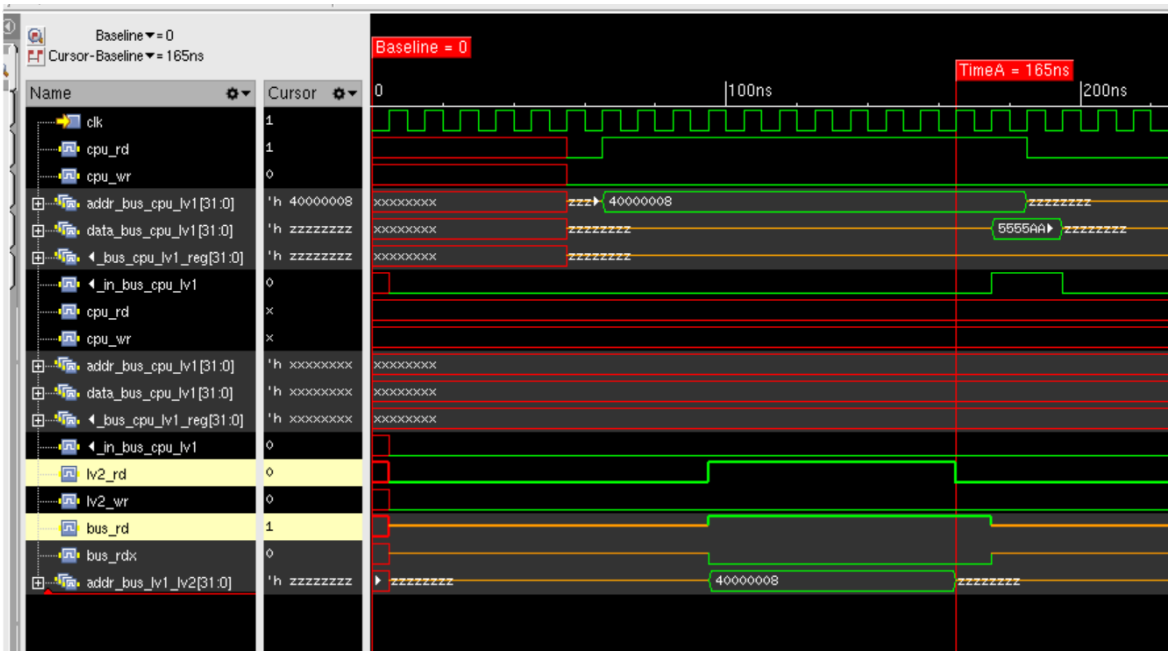
	goes low, there's no cause for the bus_rd signal to be asserted for an additional cycle.
Original Code	bus_rd_reg <= 1'b1;
Code After Fix	bus_rd_reg <= 1'b0;

Output from console showing assertion failure

```
----------------------------------------------------------------------------------------------
Name                          Type             Size  Value
----------------------------------------------------------------------------------------------
trans                         cpu_transaction_c  -    @7856
  data                        integral         32    'h5555aaaa
  address                     integral         32    'h40000008
  request_type                request_t        1     READ_REQ
  access_cache_type           access_cache_t   1     DCACHE_ACC
  begin_time                  time             64    0
  depth                       int              32    'd2
  parent sequence (name)      string           21    write_read_dcache_seq
  parent sequence (full name) string           48    uvm_test_top.tb.vsequencer.write_read_dcache_seq
  sequencer                   string           32    uvm_test_top.tb.cpu[1].sequencer
----------------------------------------------------------------------------------------------

UVM_INFO ../uvm/system_bus_monitor_c.sv(73) @ 85: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(123) @ 155: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Bus read or bus readX success
ful
UVM_INFO ../uvm/system_bus_monitor_c.sv(168) @ 165: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,182): (time 175 NS) Assertion top.inst_system_bus_if.assert_rd_rdx_assert_lv2_rd has
 failed
UVM_ERROR ../uvm/system_bus_interface.sv(184) @ 175: reporter [system_bus_interface] Assertion  rd_rdx_assert_lv2_rd Failed: If eithe
r bus_rd or bus_rdx is  asserted,then lv2_rd is asserted
UVM_INFO ../uvm/cache_scoreboard_c.sv(523) @ 185: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU1:
----------------------------------------------------
```

Waveform before fixing bug:



Waveform after fixing bug:

Team Number	#13
Bug Number	#5
Bug Location	File: main_func_lv1_dl.sv , Line No: 259
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_read_dcache
Checker/Assertion Failed	UVM_ERROR ../uvm/system_bus_interface.sv(184) @ 335: reporter [system_bus_interface] Assertion rd_rdx_assert_lv2_rd Failed: If either bus_rd or bus_rdx is asserted,then lv2_rd is asserted
Checker Description	As long as bus_rd or bus_rdx is asserted, lv2_rd should be asserted indicating a write miss scenario. property rd_rdx_assert_lv2_rd; @(posedge clk) (bus_rd || bus_rdx)|->lv2_rd; endproperty assert_rd_rdx_assert_lv2_rd: assert property(rd_rdx_assert_lv2_rd) else `uvm_error("system_bus_interface",$sformatf("Assertion rd_rdx_assert_lv2_rd Failed: If either bus_rd or bus_rdx is asserted,then lv2_rd is asserted")) endinterface
Bug Description/Debug Process	Sequence of operations: • Write Request to CPU0 with the address 32'h4000_0008 and data as 32'FFFF_FFFF (write miss). During a write request for CPU0 with address as 32'h4000_0008 and data as 32'hFFFF_FFFF, it will result in a write miss. This will assert the signal bus_rdx and lv2_rd. However, the assertion got failed. After further inspection from the waveform, we found that the bus_rdx was asserted for one additional cycle than the lv2_rd. Though there are no waveforms from the HAS documents to support that there shouldn't be no assertion of bus_rdx for an additional cycle. During the

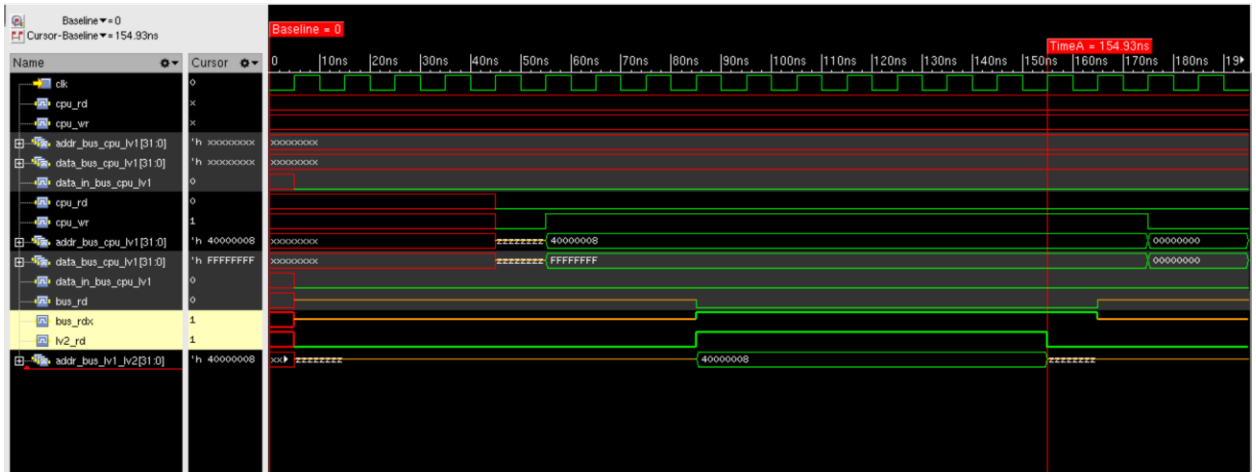
	write miss scenario, it has been mentioned that the lv2_rd and bus_rdx is asserted together. Moreover, the reason for asserting bus_rdx is for reading a block from L2 in the intention to modify it. Once, the l2_rd signal goes low, there's no cause for the bus_rdx signal to be asserted for an additional cycle.
Original Code	bus_rdx_reg <= 1'b1;
Code After Fix	bus_rdx_reg <= 1'b0;

Output from console showing assertion failure:

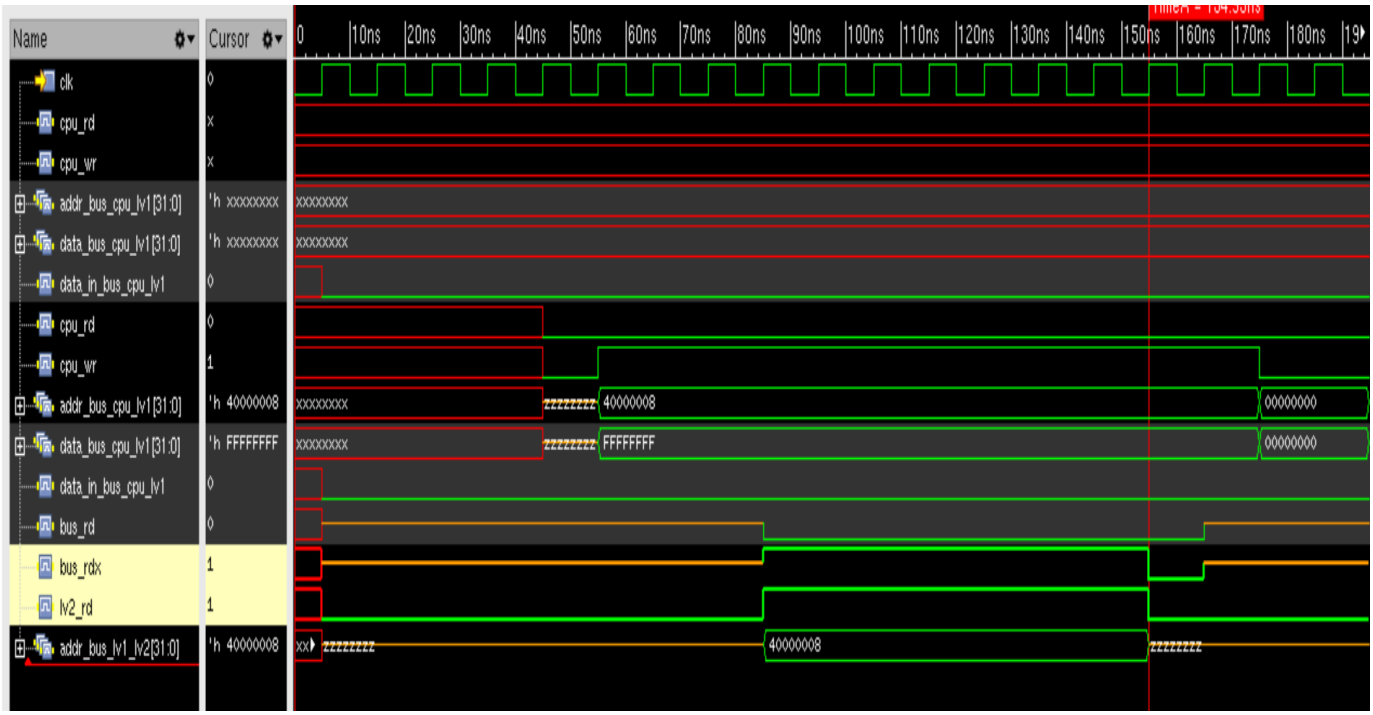
```
----------------------------------------------------------------------------------------------------
Name                        Type             Size  Value
----------------------------------------------------------------------------------------------------
trans                       cpu_transaction_c  -    @7856
  data                      integral         32    'hffffffff
  address                   integral         32    'h40000008
  request_type              request_t        1     WRITE_REQ
  access_cache_type         access_cache_t   1     DCACHE_ACC
  begin_time                time             64    0
  depth                     int              32    'd2
  parent sequence (name)    string           21    write_read_dcache_seq
  parent sequence (full name) string         48    uvm_test_top.tb.vsequencer.write_read_dcache_seq
  sequencer                 string           32    uvm_test_top.tb.cpu[0].sequencer
----------------------------------------------------------------------------------------------------

UVM_INFO ../uvm/system_bus_monitor_c.sv(73) @ 75: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(144) @ 145: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] BUS_RDX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(168) @ 155: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,182): (time 165 NS) Assertion top.inst_system_bus_if.assert_rd_rdx_assert_lv2_rd has failed
UVM_ERROR ../uvm/system_bus_interface.sv(184) @ 165: reporter [system_bus_interface] Assertion  rd_rdx_assert_lv2_rd Failed: If either bus_rd or bus_rdx is  asserted,then lv2_rd is asserted
UVM_INFO ../uvm/cache_scoreboard_c.sv(515) @ 175: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU0:
----------------------------------------------------------------------------------------------------
```

Waveform before correcting the bug:



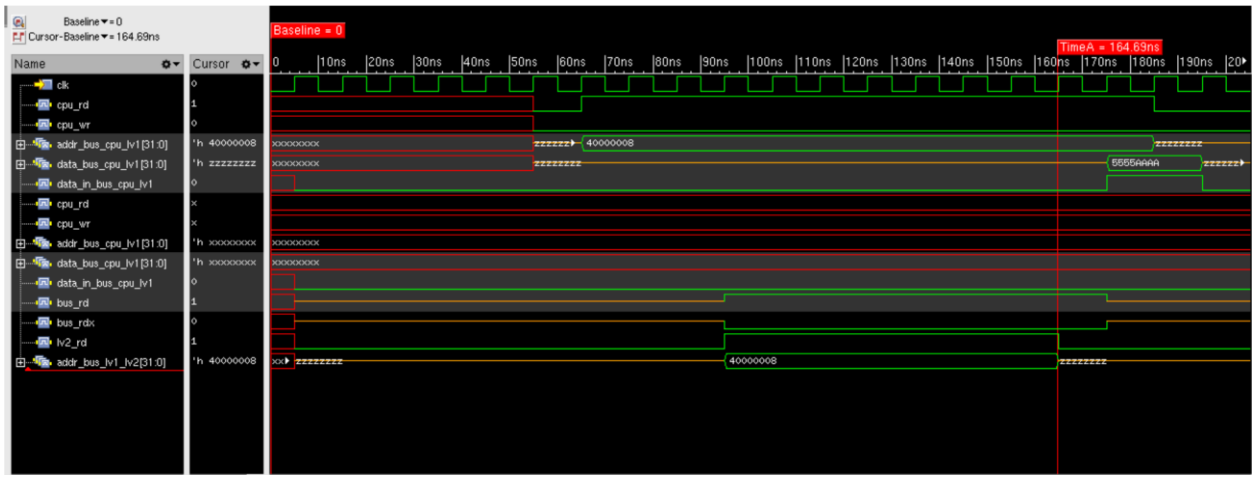
Waveform after fixing the bug:



Additional Information for re-running the same test case:

After fixing the bug for bus_rdx_reg(bug 5), we see that there's no assertion failure for the write_read_dcache. This is followed by removing the fix for bus_rd_reg(bug 4), the test case was modified to do the read operation to CPU1. While rerunning the simulation, it was observed that the test case failed. This is supported by the screenshots below.

Waveform before fixing bus_rd_reg bug(Bug 4) while bus_rdx_reg bug being fixed.



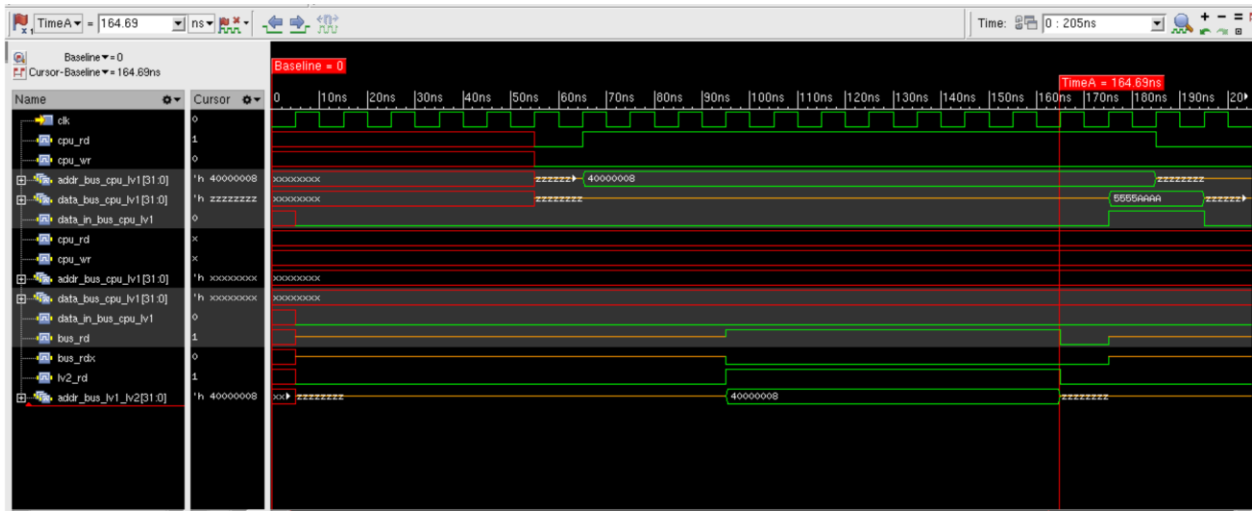
After fixing the bus_rd_reg bug(while bus_rdx_reg bug fixed)

Output from console showing the test case pass

```
Name                          Type              Size  Value
trans                         cpu_transaction_c -     @7856
  data                        integral          32    'h5555aaaa
  address                     integral          32    'h40000008
  request_type                request_t         1     READ_REQ
  access_cache_type           access_cache_t    1     DCACHE_ACC
  begin_time                  time              64    0
  depth                       int               32    'd2
  parent sequence (name)      string            21    write_read_dcache_seq
  parent sequence (full name) string            48    uvm_test_top.tb.vsequencer.write_read_dcache_seq
  sequencer                   string            32    uvm_test_top.tb.cpu[1].sequencer

UVM_INFO ../uvm/system_bus_monitor_c.sv(73) @ 85: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(123) @ 155: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Bus read or bus readX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(168) @ 165: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
UVM_INFO ../uvm/cache_scoreboard_c.sv(523) @ 185: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU1:
```

Waveform after fixing the bug:



Team Number	#13
Bug Number	#6
Bug Location	File: main_func_lv1_dl.sv , Line No: 80
Bug Type	Functional Bug
SV Test Run to uncover the Bug	eviction
Checker/Assertion Failed	Scoreboard Error
Checker Description	UVM_ERROR ../uvm/cache_scoreboard_c.sv(450) @ 3335: uvm_test_top.tb.sb [cache_scoreboard_c] Additional System Bus activity is observed for CPU1
Bug Description/Debug Process	Sequence of operations: • Read request to CPU1 with address 32'h4001_0000. • Read request to CPU1 with address as 32'h4002_0000. • Read request to CPU1 with address as 32'h4003_0000. • Read request to CPU1 with address as 32'h4004_0000. • Read request to CPU1 with address as 32'h4005_0000. • Read request to CPU1 with address as 32'h4005_0000. The first read request is made to CPU1 with address as 32'h4001_0000 will result in a read miss and the data 32'haaaa_5555 is fetched from lv2. This is followed by 3 read operations to the remaining ways of the set with addresses as 32'h4002_0000, 32'h4003_0000 and 32'h4004_0000. This completely fills the set. When another read operation happens to the same set with address as 32'h4005_0000, the first way in the same set ,i.e, the block with address 32'h4001_0000 should be evicted to accommodate the block with address 32'h4005_0000. But the eviction of the block doesn't take place as this can be observed from the blk_free signal. After the fourth read

	operation to CPU1, the blk_free_signal is never asserted. This implies that the first block didn't get invalidated. On further examination, the current_mesi_proc signal gets a value of 2'bzz. When the invalid is 2'bzz, the eviction doesn't take place. Moreover, when we do a read operation to CPU1 with address 32'h4005_0000, it should result in a read hit. However, since the block with 32'h4005_0000 is not invalidated and the blk_free signal is not asserted, no eviction takes place. When the invalid parameter is changed to 2'b00, the eviction takes place properly.
Original Code	parameter INVALID = 2'bzz;
Code After Fix	parameter INVALID = 2'b00;

BEFORE FIXING BUG:

Output from console showing scoreboard checker failure:

```
UVM_INFO ../uvm/system_bus_monitor_c.sv(168) @ 3325: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
UVM_INFO ../uvm/cpu_driver_c.sv(83) @ 3335: uvm_test_top.tb.cpu[1].driver [cpu_driver_c] Ended Driving transaction
UVM_INFO ../uvm/virtual_seqs.sv(38) @ 3335: uvm_test_top.tb.vsequencer@@replacement_seq [replacement_seq] drop_objection
UVM_INFO /opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_objection.svh(1268) @ 3335: reporter [TEST_DONE] 'r
un' phase is ready to proceed to the 'extract' phase
UVM_ERROR ../uvm/cache_scoreboard_c.sv(450) @ 3335: uvm_test_top.tb.sb [cache_scoreboard_c] Additional System Bus activity is observe
d for CPU1
UVM_INFO ../uvm/cache_scoreboard_c.sv(459) @ 3335: uvm_test_top.tb.sb [cache_scoreboard_c] Received SBUS Packet 0:
----------------------------------------------------------------
Name                          Type            Size  Value
----------------------------------------------------------------
s_packet                      sbus_packet_c   -     @7770
  bus_req_type                bus_req_t       32    BUS_RD
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC1
  req_address                 integral        32    'h0
  bus_req_snoop               integral        4     'h0
  req_serviced_by             serv_by_t       32    SERV_NONE
  rd_data                     integral        32    'h0
  wr_data_snoop               integral        32    'h0
  snoop_wr_req_flag           integral        1     'h0
  cp_in_cache                 integral        1     'h0
  shared                      integral        1     'h0
  service_time                integral        32    'h0
  proc_evict_dirty_blk_addr   integral        32    'h0
  proc_evict_dirty_blk_data   integral        32    'h0
  proc_evict_dirty_blk_flag   integral        1     'h0
----------------------------------------------------------------
```

Fig 1: Output console from scoreboard

Waveforms:

Fig 2: Waveform for 5th read operation to CPU1 with address 32'h4005_0000

Here, for 5th read operation to CPU1 with address 32'h4005_0000, since the set is filled, the first block should get evicted. However, the first block doesn't get evicted and signal blk_free is not asserted and also current_mesi_proc signal is also 2'bzz.

Fig 3: Waveform for 6th read operation to CPU1 with address 32'h4005_0000

Here, for the next read operation to CPU1 with address 32'h4005_0000, it should result in a read hit, but we can see no data is being read from CPU1. Also, from the signal data_bus_cpu_lv1, we see that no data is being read from the bus. This is because the block in the set didn't get evicted.

AFTER FIXING BUG:

Output from console:

```
UVM_INFO ../uvm/cache_scoreboard_c.sv(523) @ 1235: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU1:
-----------------------------------------------
Name            Type              Size  Value
-----------------------------------------------
packet          cpu_mon_packet_c  -     @7682
  dat           integral          32    'haaaa5555
  address       integral          32    'h40050000
  num_cycles    integral          32    'h0
  illegal       integral          1     'h0
  request_type  request_t         1     READ_REQ
  addr_type     addr_t            32    DCACHE
-----------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(293) @ 1235: uvm_test_top.tb.sb [cache_scoreboard_c] CHECK_DATA CPU1
UVM_INFO ../uvm/cache_scoreboard_c.sv(295) @ 1235: uvm_test_top.tb.sb [cache_scoreboard_c] Data match!!! expected = aaaa5555 received
 = aaaa5555
```

Fig 4: Output console from scoreboard

Waveforms:

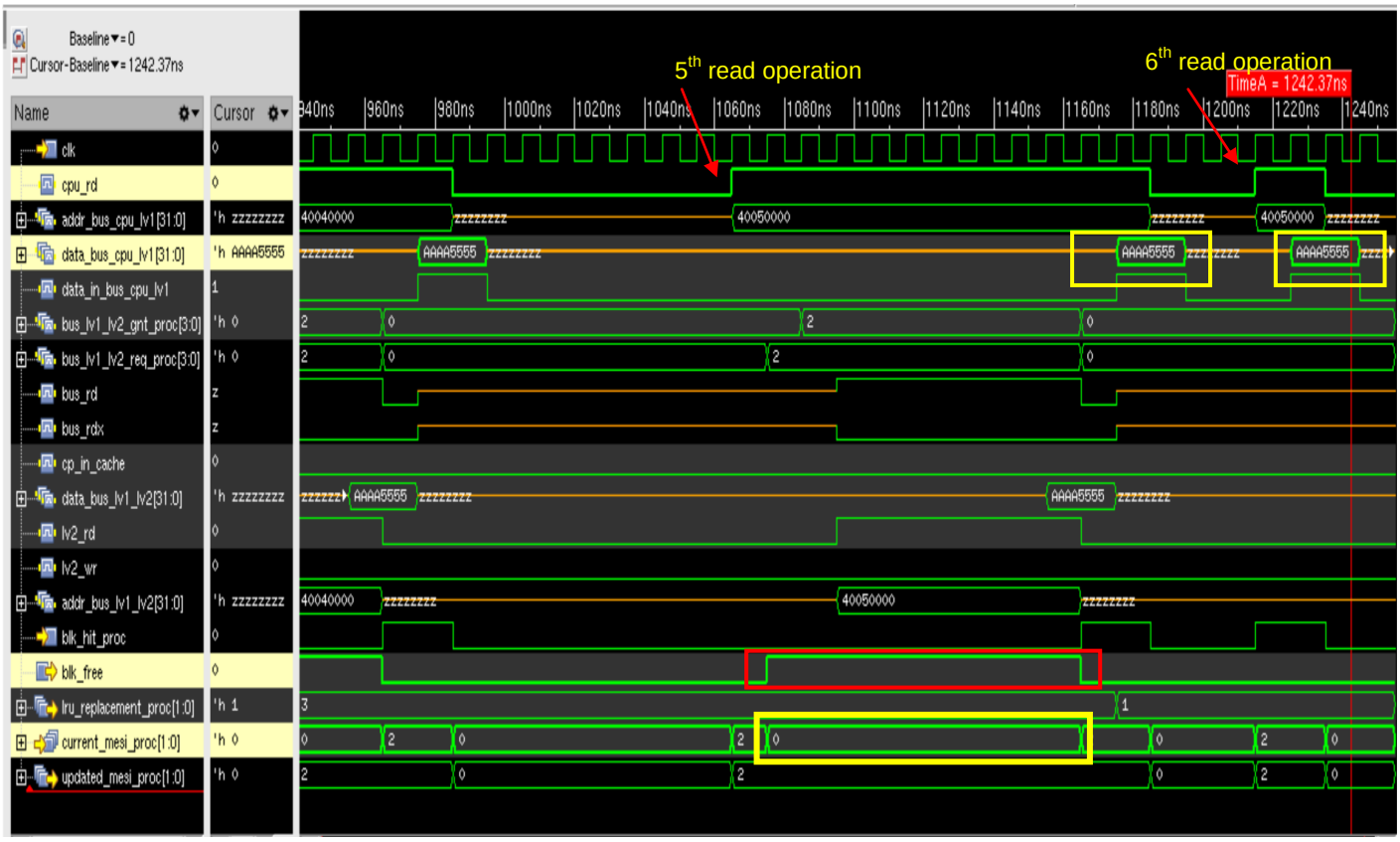


Fig 5: Waveform for 5th & 6th read operation to CPU1 with address 32'h4005_0000

After making changes to the invalid parameter from 2’bzz to 2’b00, it can be observed that blk_free signal is asserted after the 5th read operation to CPU1 with address 32’h4005_0000. This means the first block is invalidated and the data is also put in the bus data_bus_cpu_lv1. In the next read operation to CPU1 with address 32’h4005_0000, it results in a read hit, with the data being received at data_bus_cpu_lv1 as 32’haaaa_55555.

Team Number	#13
Bug Number	#7
Bug Location	File: main_func_lv1_dl.sv , Line No: 194
Bug Type	Functional Bug
SV Test Run to uncover the Bug	replacement
Checker/Assertion Failed	UVM_ERROR ../uvm/cache_scoreboard_c.sv(450) @ 3305: uvm_test_top.tb.sb [cache_scoreboard_c] Additional System Bus activity is observed for CPU1
Checker Description	When there shouldn't be any activity on the sbus activity, but an additional activity is received then this error occurs.
Bug Description/Debug Process	Sequence of operations: • Write Request to CPU1 with address 32'h4001_0000 and data as 32'haaaa_eeee(1st). • Read request to CPU1 with address as 32'h4002_0000(2nd). • Read request to CPU1 with address as 32'h4003_0000(3rd). • Read request to CPU1 with address as 32'h4004_0000(4th). • Read request to CPU1 with address as 32'h4005_0000(5th). • Read request to CPU2 with address as 32'h4005_0000(6th). The first write request is made to CPU1 with address as 32'h4001_0000 and data as 32'haaaa_eeee will result in a write miss and the data 32'haaaa_5555 is fetched from lv2, and this overwritten by the 32'haaaa_eeee and the state will be modified. This is followed by 3 read operations to the remaining ways of the set with addresses as 32'h4002_0000, 32'h4003_0000 and 32'h4004_0000. This completely fills the set. When another read operation happens to the same set with address as 32'h4005_0000, the first way in the same set ,i.e, the block with address 32'h4001_0000 is replaced by 32'h4005_0000, and the data will writeback to level 2. But it is observed

	that the since the current_mesi_protocol is kept as modified, the block present in the address is 32'h4001_0000 is not getting invalidated as continuous activity in the bus is observed. So, when a read request is made to CPU2 with address 32'h4005_0000, it is served by L2 which is not correct. This implies the replacement didn't occur properly as the previous block was not invalidated properly.
Original Code	`CACHE_CURRENT_MESI_PROC <= MODIFIED
Code After Fix	`CACHE_CURRENT_MESI_PROC <= INVALID;

BEFORE FIXING THE BUG:

Output from console before fixing the bug:

Name	Type	Size	Value
s_packet	sbus_packet_c	-	@7819
bus_req_type	bus_req_t	32	BUS_RD
bus_req_proc_num	bus_req_proc_t	32	REQ_PROC1
req_address	integral	32	'h0
bus_req_snoop	integral	4	'h0
req_serviced_by	serv_by_t	32	SERV_L2
rd_data	integral	32	'haaaa5555
wr_data_snoop	integral	32	'h0
snoop_wr_req_flag	integral	1	'h0
cp_in_cache	integral	1	'h0
shared	integral	1	'h0
service_time	integral	32	'h0
proc_evict_dirty_blk_addr	integral	32	'h0
proc_evict_dirty_blk_data	integral	32	'h0
proc_evict_dirty_blk_flag	integral	1	'h1

Fig 1: Output Console

Waveform:

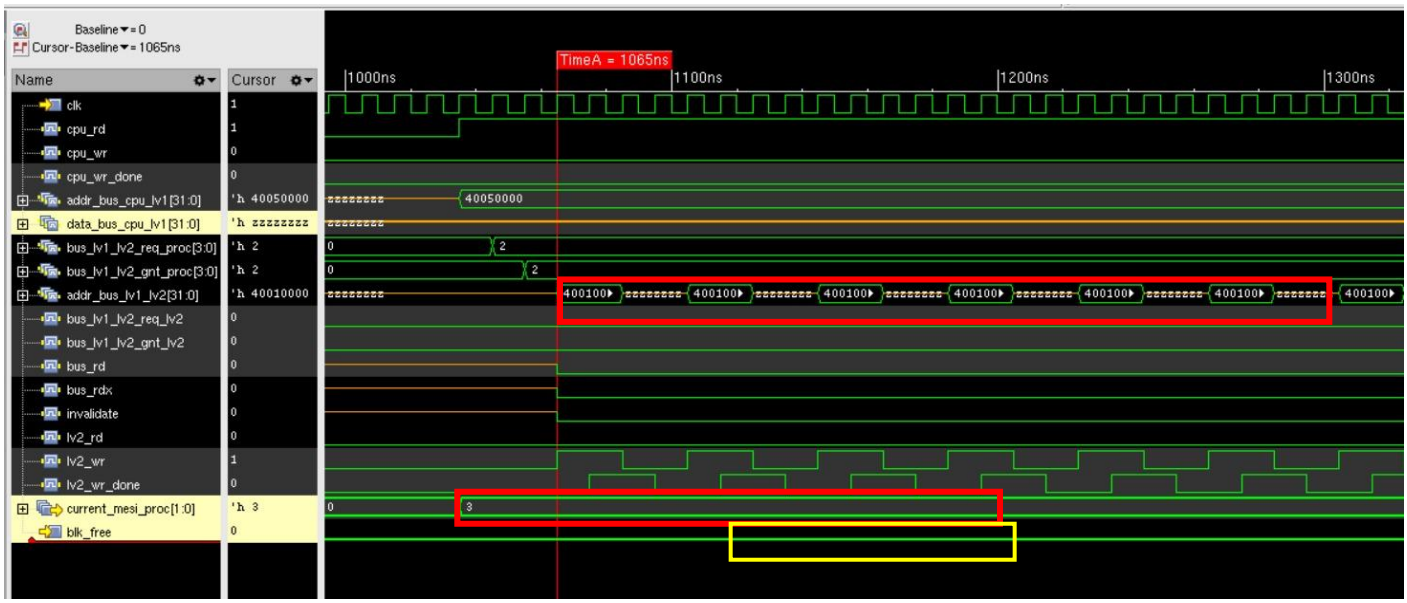


Fig 2: Waveform for read operation to CPU1(5th operation)

From the above waveform, during the read operation to CPU1 with address 32'h4005_000, the modified block present in the address 32'h4001_0000 should be written back to lvl2. However, since the current_messi_proc signal is not changed from modified, it continuously tries to write to l2. Moreover, the blk_free signal is also not asserted here.

AFTER FIXING THE BUG:

Output console after fixing the bug:

```
--------------------------------------------------------------------
Name                          Type             Size  Value
--------------------------------------------------------------------
s_packet                      sbus_packet_c    -     @7710
  bus_req_type                bus_req_t        32    BUS_RD
  bus_req_proc_num            bus_req_proc_t   32    REQ_PROC2
  req_address                 integral         32    'h40050000
  bus_req_snoop               integral         4     'h0
  req_serviced_by             serv_by_t        32    SERV_SNOOP1
  rd_data                     integral         32    'haaaa5555
  wr_data_snoop               integral         32    'h0
  snoop_wr_req_flag           integral         1     'h0
  cp_in_cache                 integral         1     'h1
  shared                      integral         1     'h1
  service_time                integral         32    'h0
  proc_evict_dirty_blk_addr   integral         32    'h0
  proc_evict_dirty_blk_data   integral         32    'h0
  proc_evict_dirty_blk_flag   integral         1     'h0
--------------------------------------------------------------------
```

Fig 3: Output console

Waveform:

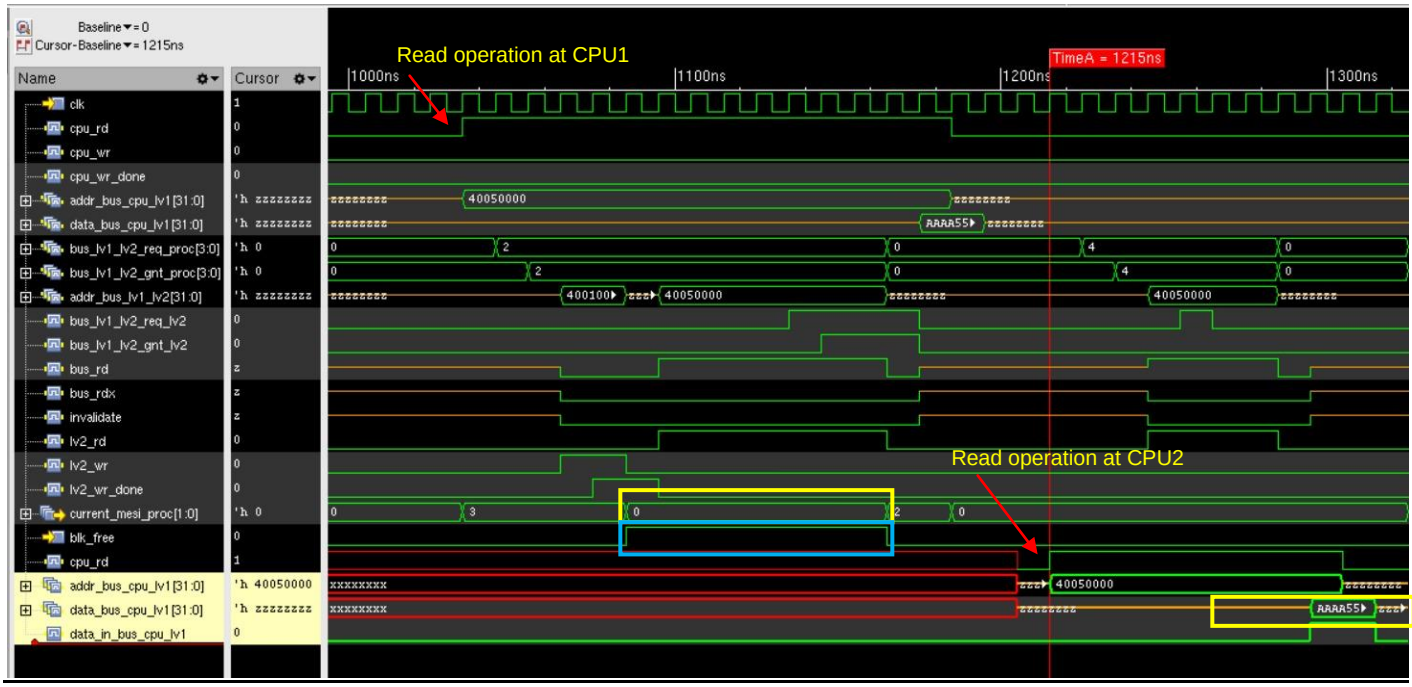


Fig 4: Waveform for read operation at CPU1 and read operation at CPU2.

From the above waveform, during the read operation to CPU1 with address 32'h4005_000, the modified block present in the address 32'h4001_0000 should be written back to lvl2. The current_messi_proc signal is changed from modified to invalid, and writes to l2. The blk_free signal is also asserted here. During the read operation to CPU2 with address 32'h4005_0000 will result in the data being served by processor 1.

Team Number	#13
Bug Number	#8
Bug Location	File: main_func_lv1_dl.sv , Line No: 280
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss_replacement
Checker/Assertion Failed	Scoreboard Error
Checker Description	UVM_ERROR ../uvm/cache_scoreboard_c.sv(414) @ 2585: uvm_test_top.tb.sb [cache_scoreboard_c] Expected activity is not observed on the system bus
Bug Description/Debug Process	Sequence of operations: • Write Request to CPU1 with address 32'h4001_0000 and data as 32'h0000_aaaa(1st). • Write Request to CPU1 with address 32'h4002_0000 and data as 32'h0000_aaaa(2nd). • Write Request to CPU1 with address 32'h4003_0000 and data as 32'h0000_aaaa(3rd). • Write Request to CPU1 with address 32'h4004_0000 and data as 32'h0000_aaaa(4th). • Write Request to CPU1 with address 32'h4005_0000 and data as32'h0000_bbbb(5th). • Read request to CPU3 with address 32'h4005_0000(6th). Initially four different write operations to CPU1 with addresses as 32'h4001_0000, 32'h4002_0000, 32'h4003_0000 and 32'h4004_0000 and data as 32'h0000_aaaa will fill a set and all four operations result in a write miss. This is followed by a 5th write operation to the same set in CPU1, however, since the set is full, the first block with address 32'h4001_0000 needs to be evicted. Since, it's in a modified state, this dirty block needs to be written back to L2. However, the block is not written back to l2 as lv2_wr is not asserted. As we proceed with a read operation to CPU3 with address 32'h4005_0000, it should result in the data being served by CPU1,however it is observed that the data is being served by L2. Hence, resulting in abus mismatch

Original Code	lv2_wr	<= 1'b0;
Code After Fix	lv2_wr	<= 1'b1;

BEFORE FIXING BUG:

Output from console showing scoreboard checker failure:

```
-------------------------------------------------------------
Name                          Type            Size  Value
-------------------------------------------------------------
s_packet                      sbus_packet_c   -     @7819
  bus_req_type                bus_req_t       32    BUS_RD
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC1
  req_address                 integral        32    'h0
  bus_req_snoop               integral        4     'h0
  req_serviced_by             serv_by_t       32    SERV_L2
  rd_data                     integral        32    'haaaa5555
  wr_data_snoop               integral        32    'h0
  snoop_wr_req_flag           integral        1     'h0
  cp_in_cache                 integral        1     'h0
  shared                      integral        1     'h0
  service_time                integral        32    'h0
  proc_evict_dirty_blk_addr   integral        32    'h0
  proc_evict_dirty_blk_data   integral        32    'h0
  proc_evict_dirty_blk_flag   integral        1     'h0
-------------------------------------------------------------
```

Fig 1: Output console

Waveform before fixing the bug:

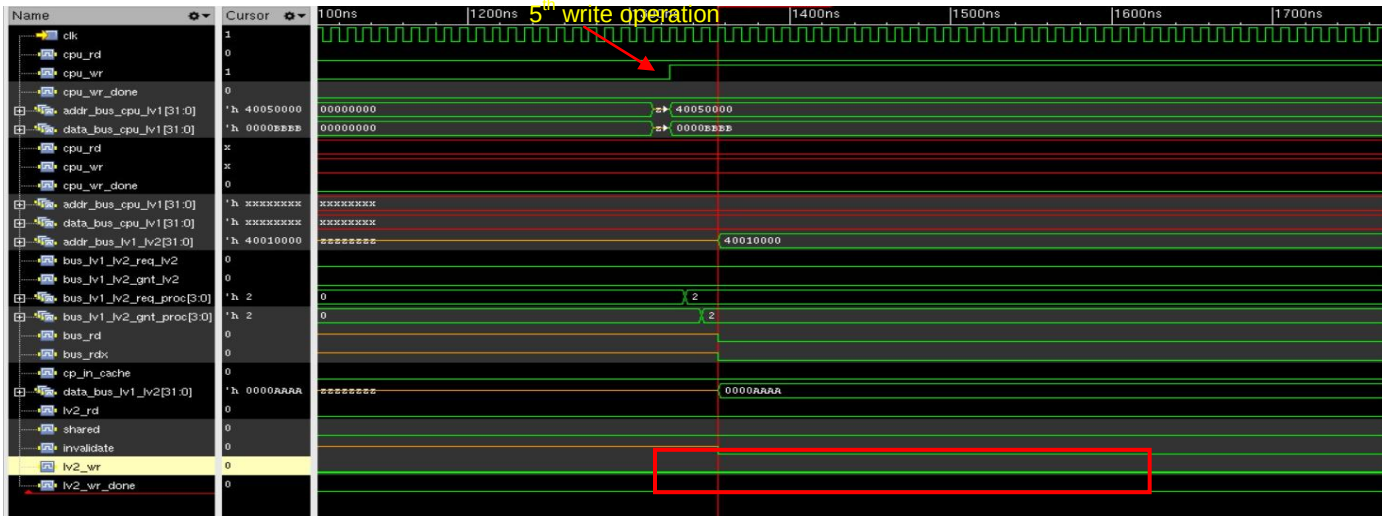


Fig 2: Waveform for 5th write operation

During 5th write operation to the same set in CPU1, since the set is full, the first block with address 32'h4001_0000 needs to be evicted. However, the block is not written back to l2 as lv2_wr is not asserted.

AFTER FIXING BUG:

Output console after fixing the bug:

```
------------------------------------------------------------------
Name                          Type            Size  Value
------------------------------------------------------------------
s_packet                      sbus_packet_c   -     @7714
  bus_req_type                bus_req_t       32    BUS_RD
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC2
  req_address                 integral        32    'h40050000
  bus_req_snoop               integral        4     'h0
  req_serviced_by             serv_by_t       32    SERV_SNOOP1
  rd_data                     integral        32    'hbbbb
  wr_data_snoop               integral        32    'h0
  snoop_wr_req_flag           integral        1     'h0
  cp_in_cache                 integral        1     'h1
  shared                      integral        1     'h1
  service_time                integral        32    'h0
  proc_evict_dirty_blk_addr   integral        32    'h0
  proc_evict_dirty_blk_data   integral        32    'h0
  proc_evict_dirty_blk_flag   integral        1     'h0
------------------------------------------------------------------
```

Fig 3: Output console

Waveform after fixing the bug:

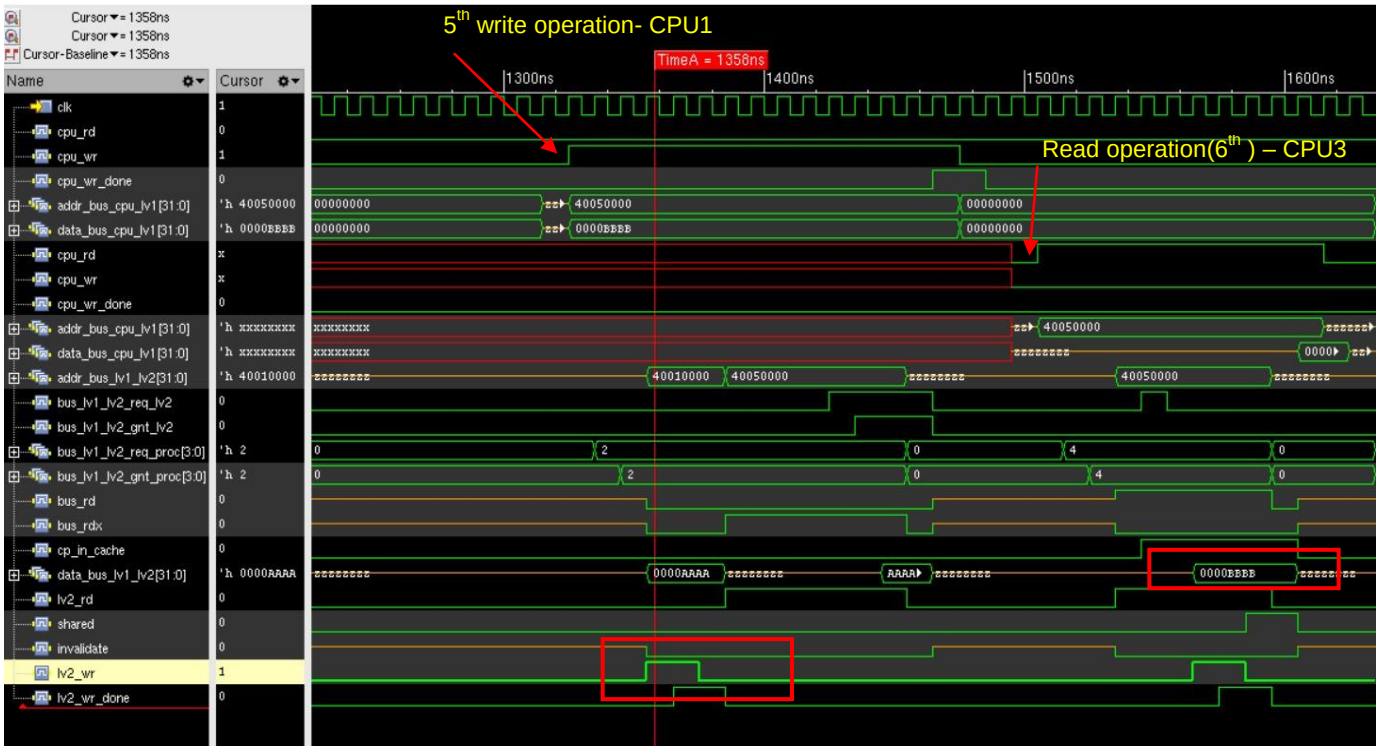


Fig 4: Waveform for 5th write operation(CPU1) and read operation(CPU3)

In the above waveform, during the 5th write operation to CPU1, the block is written back to l2 as lv2_wr is asserted. As we proceed with a read operation to CPU3 with address 32’h4005_0000, it results in the data being served by CPU1. As the data 32’h0000_bbbb is being read from the data_bus_cpu_lv1.

Team Number	#13
Bug Number	#9
Bug Location	File: main_func_lv1_dl.sv , Line No: 192
Bug Type	Functional Bug
SV Test Run to uncover the Bug	writeback_read
Checker/Assertion Failed	UVM_ERROR ../uvm/cache_scoreboard_c.sv(297) @ 1305: uvm_test_top.tb.sb [cache_scoreboard_c] Data MISMATCH!!! expected = aaaa_eeee received = aaaa_5555
Checker Description	When a read miss occurs to CPU3 with address as 32'h4001_0000, the data to be received should be 32'haaaa_eeee, however we receive data as 32'haaaa_5555. Hence, we get a data mismatch.
Bug Description/Debug Process	Sequence of operations: • Write Request to CPU1 with address 32'h4001_0000 and data as 32'haaaa_eeee(1st). • Read request to CPU1 with address as 32'h4002_0000(2nd). • Read request to CPU1 with address as 32'h4003_0000(3rd). • Read request to CPU1 with address as 32'h4004_0000(4th). • Read request to CPU1 with address as 32'h4005_0000(5th). • Read request to CPU3 with address as 32'h4001_0000(6th). Initially, a write request is made to CPU1 with address as 32'h4001_0000 and data as 32'haaaa_eeee. This followed by 4 read operations to CPU1 within the same set with addresses as 32'h4002_0000, 32'h4003_0000, 32'h4004_000. These 4 operations to a set will fill the set. When we do a 5th read operation to CPU1 it doesn't have any more set available. Hence replacement occurs. The block with address as 32'h4001_0000 gets replaced and since it is a write operation, the block will be in a modified state. Since it's a dirty block, a write block should

	occur for this block. So, when we do the next read operation to CPU3 with address as 32'h4001_0000, it will result in a read miss and it must fetch the data as 32'haaaa_eeee. However, the data returned to CPU3 was 32'haaaa_5555. Hence causing data mismatch at CPU3. This shows that the writeback to the L2 didn't occur.
Original Code	data_bus_lv1_lv2_reg <= cache_var[{index_proc,2'b00}];
Code After Fix	data_bus_lv1_lv2_reg <= cache_var[{index_proc,blk_access_proc}];

The fix was made to facilitate the data to be fetched with address involving the block needs to be accessed. Hence, the second part of the index was changed from 2'b00 to blk_access_proc.

BEFORE FIXING THE BUG:

Output from console showing assertion failure:

```
UVM_INFO ../uvm/system_bus_monitor_c.sv(168) @ 1165: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
UVM_INFO ../uvm/cache_scoreboard_c.sv(523) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU1:
----------------------------------------------------
Name            Type              Size  Value
----------------------------------------------------
packet          cpu_mon_packet_c  -     @7696
  dat           integral          32    'haaaa5555
  address       integral          32    'h40050000
  num_cycles    integral          32    'h0
  illegal       integral          1     'h0
  request_type  request_t         1     READ_REQ
  addr_type     addr_t            32    DCACHE
----------------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(293) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] CHECK_DATA CPU1
UVM_INFO ../uvm/cache_scoreboard_c.sv(295) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] Data match!!! expected = aaaa5555 received
 = aaaa5555
UVM_INFO @ 1185: reporter [MISCMP] Miscompare for expected.proc_evict_dirty_blk_addr: lhs = 'h40010000 : rhs = 'h0
UVM_INFO @ 1185: reporter [MISCMP] 3 Miscompare(s) (1 shown) for object s_packet@7770 vs. expected@7779
UVM_ERROR ../uvm/cache_scoreboard_c.sv(433) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] System bus activity MISMATCH!!!
UVM_INFO ../uvm/cache_scoreboard_c.sv(434) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] Expected SBUS Packet
------------------------------------------------------------
Name                        Type            Size  Value
------------------------------------------------------------
expected                    sbus_packet_c   -     @7779
  bus_req_type              bus_req_t       32    BUS_RD
  bus_req_proc_num          bus_req_proc_t  32    REQ_PROC1
  req_address               integral        32    'h40050000
  bus_req_snoop             integral        4     'h0
  req_serviced_by           serv_by_t       32    SERV_L2
  rd_data                   integral        32    'haaaa5555
  wr_data_snoop             integral        32    'h0
  snoop_wr_req_flag         integral        1     'h0
  cp_in_cache               integral        1     'h0
  shared                    integral        1     'h0
  service_time              integral        32    'h0
  proc_evict_dirty_blk_addr integral        32    'h40010000
  proc_evict_dirty_blk_data integral        32    'haaaaeeee
  proc_evict_dirty_blk_flag integral        1     'h1
------------------------------------------------------------
```

Fig 1: Output console

Waveform:

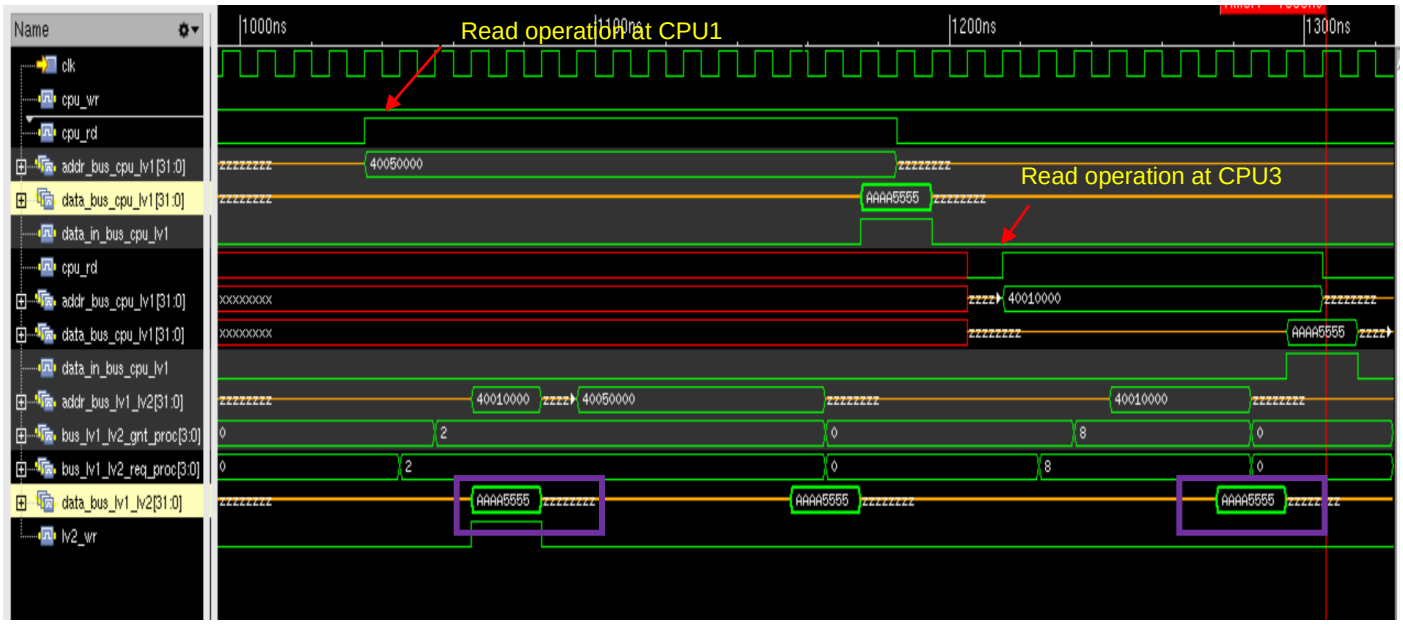


Fig 2: Waveform for read operation at CPU1(5th) and CPU3(6th)

The waveform represents the read miss in CPU3, with the address 32'h4001_0000. The expected data should have been 32'haaaa_eeee, but this is 32'haaaa_5555. This is because in the previous read request to CPU1 with the address as 32'h4005_0000, the data to be written back to be L2 should be 32'haaaa_eeee. But 32'haaaa_5555 is getting written back. Hence, causing data mismatch.

AFTER FIXING BUG:

Waveform:

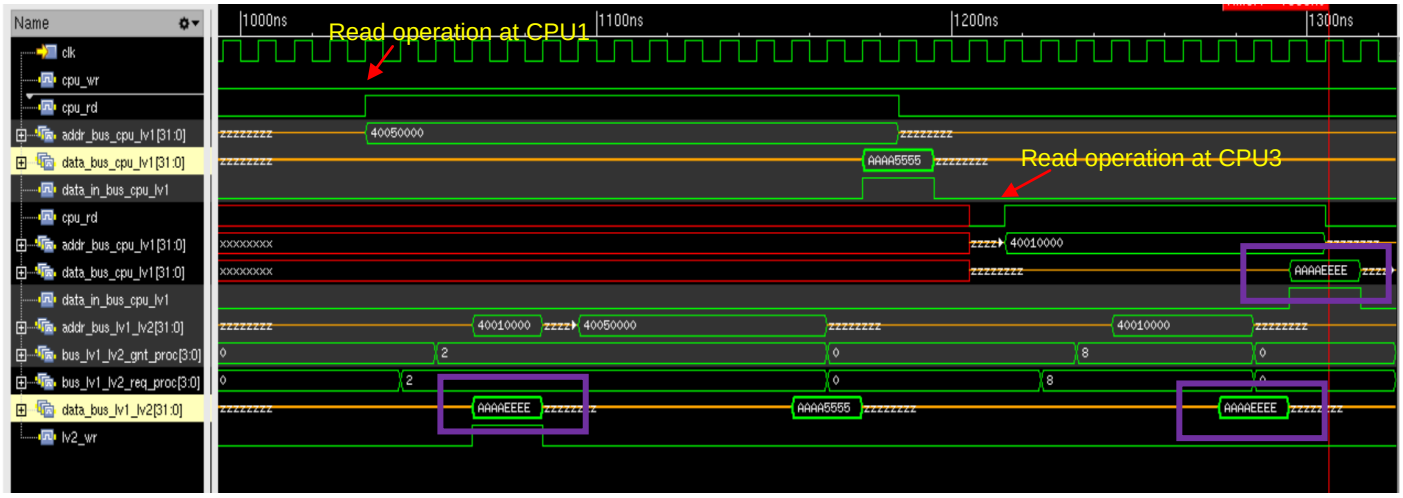


Fig 3: Waveform for read operation at CPU1(5th) and CPU3(6th)

The waveform represents the read miss in CPU3, with the address 32’h4001_0000. The expected data should have been 32’haaaa_eeee, and the data received is also 32’haaaa_eeee. This is can be seen from the previous read request to CPU1 with the address as 32’h4005_0000, where the data to be written back to be L2 is also 32’haaaa_eeee. Hence, upon a read miss to CPU3, the data received is the write back data to L2.