Team-16 Bug 1 Report

Team Number	# 16
Bug Number	# 1
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 176
Bug Type	Functional Bug
SV Test Run to uncover the Bug	read_miss_dcache: We have added this test case to validate processors read miss on a DCACHE. This involves randomizing the request_type to READ_REQ and access_cache_type to DCACHE_ACC.

Checker/Assertion Failed	Assertion assert_data_in_bus_lv1_lv2_busrd_busrdx failed which is described as below in the system_bus_interface:
	property data_in_bus_lv1_lv2_busrd_busrdx;
	@(posedge clk)
	data_in_bus_lv1_lv2 |=> (bus_rd!==1'b1 & bus_rdx!==1'b1);
	endproperty
	assert_data_in_bus_lv1_lv2_busrd_busrdx : assert
	property(data_in_bus_lv1_lv2_busrd_busrdx)
	else
	`uvm_error("system_bus_interface",$sformatf("Assertion-20 assert_data_in_bus_lv1_lv2_busrd_busrdx Failed: bus_rd and/or bus_rdx asserted when data_in_bus_lv1_lv2 asserted"))

Checker Description	The signals bus_rd and bus_rdx should be deasserted when data_in_bus_lv1_lv2 is asserted.
Bug Description/ Debug Process	After running the test, we observed that the assertion **assert_data_in_bus_lv1_lv2_busrd_busrdx** is failing, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that the signal **bus_rd** remains high even after **data_in_bus_lv1_lv2** is asserted (as shown in fig.2), which violates the specifications outlined in the HAS document(as shown on fig.6). Due to this discrepancy, the above-mentioned assertion has failed. The value for bus_rd comes from **bus_rd_reg** and an incorrect logic for bus_rd_reg assignment is noticed under the read miss logic within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.3 and fig.4 respectively. After fixing the bug we observed that there is no assertion failure (implied by the UVM Summary report in fig.4). Further analysis using Simvision waveform shows the same shown below in fig.5
Original Code	bus_rd_reg <= 1'b1;
Code after Fix	bus_rd_reg <= 1'b0;

```
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,207): (time 1415 NS) Assertion top.inst_system_bus_if.assert_data_in_bus_
lv1_lv2_busrd_busrdx has failed (2 cycles, starting 1405 NS)
UVM_ERROR ../uvm/system_bus_interface.sv(209) @ 1415: reporter [system_bus_interface] Assertion-20 assert_data_in_bus_lv1_
lv2_busrd_busrdx Failed: bus_rd and/or bus_rdx asserted when data_in_bus_lv1_lv2 asserted
UVM_INFO ../uvm/cache_scoreboard_c.sv(539) @ 1425: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU3:
```

Fig.1

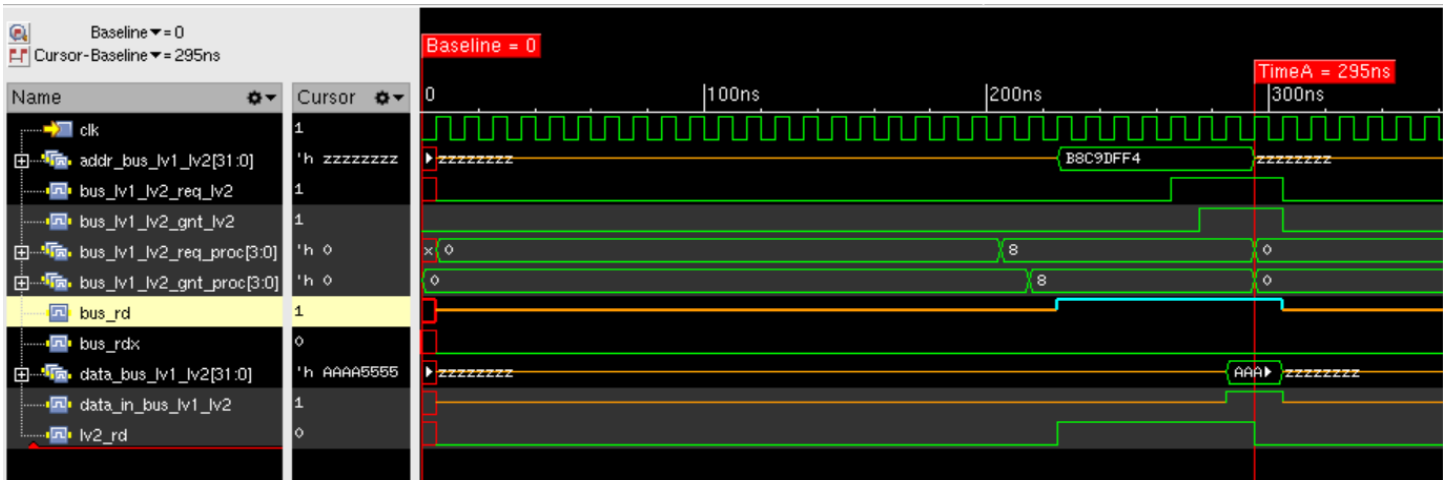


Fig.2 Waveform before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   68
UVM_WARNING :    0
UVM_ERROR :    5
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    26
[cpu_driver_c]    14
[cpu_monitor_c]     4
[read_miss_dcache]     1
[read_miss_dcache_seq]     3
[system_bus_interface]     5
[system_bus_monitor_c]    16
Simulation complete via $finish(1) at time 1445 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   68
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    26
[cpu_driver_c]    14
[cpu_monitor_c]     4
[read_miss_dcache]     1
[read_miss_dcache_seq]     3
[system_bus_monitor_c]    16
Simulation complete via $finish(1) at time 1445 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.4 UVM Report Summary after bug fix

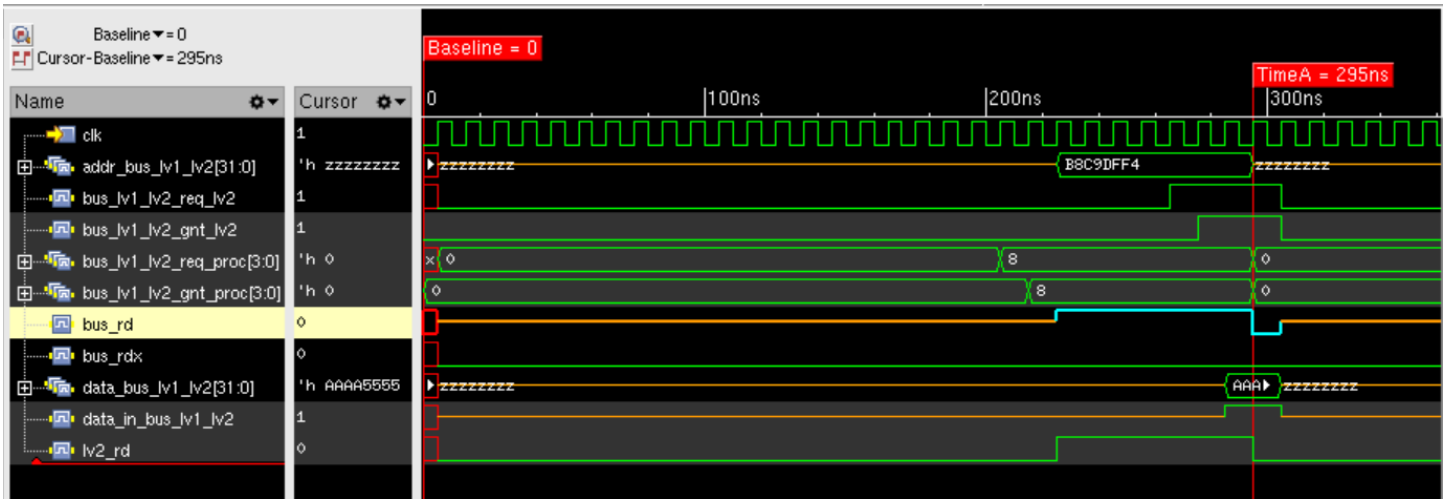


Fig.5 Waveform after bug fix

2. Processor Read Miss

If the Block is not hit then the following possibilities can occur

2.1 Free block available in the set

- Bus access is been requested (bus_lv1_lv2_req_proc made high)
- Wait till Access is granted (bus_lv1_lv2_gnt_proc to be made high by arbiter)
- Once access granted, bus_rd and lv2_rd is raised
- Address of the requested data block is put in addr_bus_lv1_lv2

- Wait till level 2 cache / other cache provides the data; communicated by making data_in_bus_lv1_lv2 high. Level 2 cache / other caches put the data in data_bus_lv1_lv2.
- Once data_in_bus_lv1_lv2 becomes high, cache_var [index_proc, blk_access_proc] is updated with value from data_bus_lv1_lv2; cache_proc_contr [index_proc, blk_access_proc] [MESI_range] is updated with updated_mesi_state from CC. Cache_proc_contr [index_proc, blk_access_proc] [Tag_range] is updated with tag_proc.

Fig.6 HAS Document Statement supporting failed Assertion.

Team-16 Bug 2 Report

Team Number	**# 16**
Bug Number	# 2
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 137
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss_dcache: We have added this test case to validate processors write miss scenario on a DCACHE. This involves randomizing the request_type to WRITE_REQ and access_cache_type to DCACHE_ACC.
Checker/Assertion Failed	Assertion assert_bus_rd_or_rdx_after_lv2_rd failed which is described as below in the system_bus_interface: assert_bus_rd_or_rdx_after_lv2_rd: assert property (@(posedge clk) ((lv2_rd) && (addr_bus_lv1_lv2[31:30] != 2'b0))|-> (bus_rd || bus_rdx)) else `uvm_error("system_bus_interface",$sformatf("Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Failed: bus_rd or bus_rdx not asserted when lv2_rd asserted"))
Checker Description	The signals bus_rd or bus_rdx should be asserted when lv2_wr is asserted and address is DCACHE address.
Bug Description/ Debug Process	After running the test, we observed that the assertion **assert_bus_rd_or_rdx_after_lv2_rd** is failing, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that the signal **bus_rd** or **bus_rdx** is not getting asserted even after **lv2_rd** is asserted (as shown in fig.2), which violates the specifications outlined in the HAS document (as shown on fig.6). Due to this discrepancy, the above-mentioned assertion has failed. We also noticed that the value of **bus_rdx** is initialized to **1'b0** whereas **bus_rd** is initialized to **1'bz**(which should not be the case as both signals should be initialized to **1'bz**). The value for **bus_rdx** comes from **bus_rdx_reg** and an incorrect initialization was done to the signal **bus_rdx_reg** as **1'b0** within the file main_func_lv1_dl under design/lv1/ folder. On every posedge of clk the **bus_rdx_reg** is assigned the values **1'b0** (from line 137) and value **1'b1**(from line 248) which resulted an overall value of **1'bx** as shown in Fig.2 waveform. UVM Summary report before and after the fix is shown in fig.3 and fig.5 respectively. After fixing the bug we observed that there is still an assertion **assert_data_in_bus_lv1_lv2_busrd_busrdx** failure (Fig.4) (also implied by the UVM Summary report in fig.5), which potentially showed a path to bug3. We will discuss more about this in Bug3 report. After fixing the bug2 the assertion **assert_bus_rd_or_rdx_after_lv2_rd** is not failing anymore and correct value is getting initialized to the signal **bus_rdx** and it is getting asserted after lv2_rd is asserted. (Refer Fig.5 and Fig.6)
Original Code	bus_rdx_reg <= 1'b0;
Code after Fix	bus_rdx_reg <= 1'bz;

```
UVM_INFO ../uvm/system_bus_monitor_c.sv(78) @ 205: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,168): (time 225 NS) Assertion top.inst_system_bus_if.assert_bus_rd_or_rdx_after_lv2_rd
has failed
UVM_ERROR ../uvm/system_bus_interface.sv(170) @ 225: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
UVM_ERROR ../uvm/system_bus_interface.sv(170) @ 235: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
UVM_ERROR ../uvm/system_bus_interface.sv(170) @ 245: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
UVM_ERROR ../uvm/system_bus_interface.sv(170) @ 255: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
UVM_ERROR ../uvm/system_bus_interface.sv(170) @ 265: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
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_ERROR ../uvm/system_bus_interface.sv(170) @ 275: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
UVM_INFO ../uvm/system_bus_monitor_c.sv(225) @ 285: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
UVM_ERROR ../uvm/system_bus_interface.sv(170) @ 285: reporter [system_bus_interface] Assertion-13 assert_bus_rd_or_rdx_after_lv2_rd Fail
ed: bus_rd or bus_rdx not asserted when lv2_rd asserted
UVM_INFO ../uvm/cache_scoreboard_c.sv(515) @ 305: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU0:
```

Fig.1

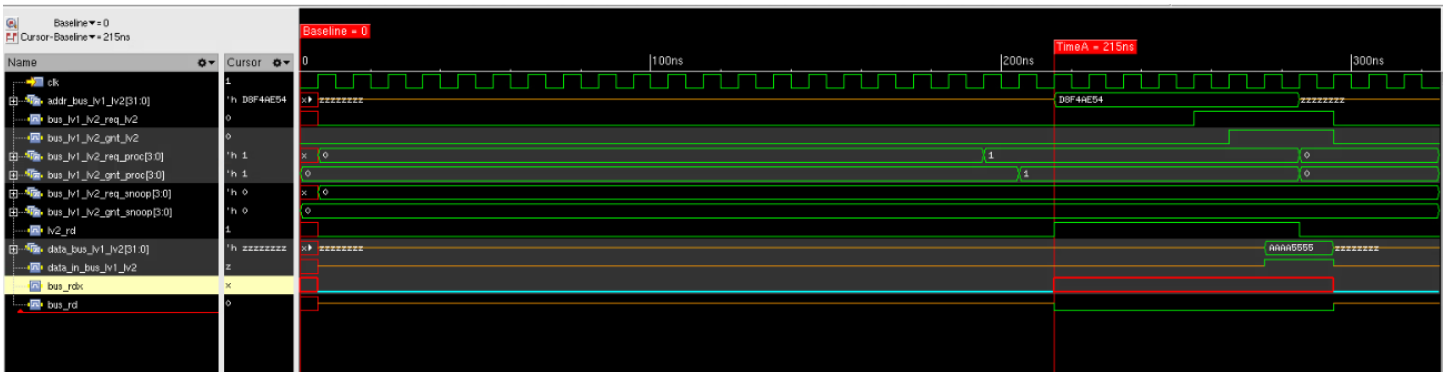


Fig.2 Waveform before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   27
UVM_WARNING :    0
UVM_ERROR :    7
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]     5
[cpu_driver_c]     6
[cpu_monitor_c]     4
[system_bus_interface]     7
[system_bus_monitor_c]     4
[write_miss_dcache]     1
[write_miss_dcache_seq]     3
Simulation complete via $finish(1) at time 325 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
UVM_INFO ../uvm/system_bus_monitor_c.sv(78) @ 205: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(194) @ 275: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] BUS_RDX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(225) @ 285: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
                    data_in_bus_lv1_lv2 ⟹ (bus_rd=≠1'b1 & bus_rdx=≠1'b1);
                                          |
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,207): (time 295 NS) Assertion top.inst_system_bus_if.assert_data_in_bus_lv1_lv2_busrd_b
usrdx has failed (2 cycles, starting 285 NS)
UVM_ERROR ../uvm/system_bus_interface.sv(209) @ 295: reporter [system_bus_interface] Assertion-20 assert_data_in_bus_lv1_lv2_busrd_busrd
x Failed: bus_rd and/or bus_rdx asserted when data_in_bus_lv1_lv2 asserted
UVM_INFO ../uvm/cache_scoreboard_c.sv(515) @ 305: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU0:
```

Fig.4

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   27
UVM_WARNING :    0
UVM_ERROR :    1
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]     5
[cpu_driver_c]     6
[cpu_monitor_c]     4
[system_bus_interface]     1
[system_bus_monitor_c]     4
[write_miss_dcache]     1
[write_miss_dcache_seq]     3
Simulation complete via $finish(1) at time 325 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.5 UVM Report Summary after bug fix

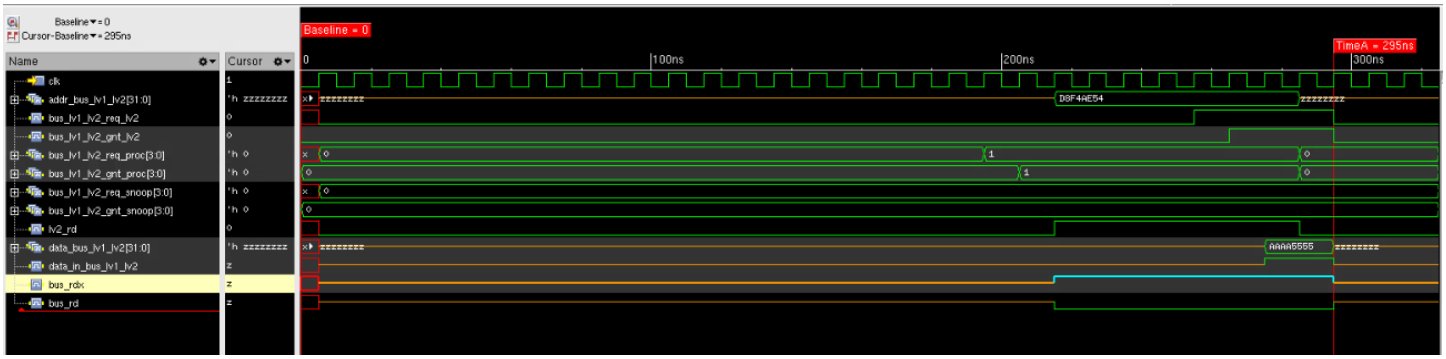


Fig.6 Waveform after bug fix

3.2 Processor Write miss

Similar to Read Miss, Write Miss also has two possibilities which are free block/line available and free block/line not available.

(1) Free block available

The following operations are carried out at Proc side of the requesting cache.

If a free line is available. (Processor Write miss in L1 Cache with Shared / Exclusive

/ Modified state)

- bus_lv1_lv2_req_proc is raised.
- Wait till bus_lv1_lv2_gnt_proc to be made high by arbiter.
- Once access granted, bus_rdx and lv2_rd is raised
- Address of the requested data block is put in addr_bus_lv1_lv2.
- Wait till level 2 cache provides the data. Communicated by making data_in_bus_lv1_lv2 high. ATTENTION: data will only be provided by level 2 cache in this case because other level 1 caches will first make their copies Invalid.
- Once data_in_bus_lv1_lv2 becomes high, cache_var [index_proc, blk_access_proc] is updated with value from data_bus_lv1_lv2. cache_proc_contr [index_proc, blk_access_proc] [MESI_range] is updated with updated_mesi_proc. cache_proc_contr [index_proc, blk_access_proc] [Tag_range] is updated with tag_proc.

Fig.6 HAS Document Statement supporting failed Assertion.

Team-16 Bug 3 Report

Team Number	**# 16**
Bug Number	# 3
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 259
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss_dcache: We have added this test case to validate processors read miss on a DCACHE. This involves randomizing the request_type to WRITE_REQ and access_cache_type to DCACHE_ACC.

Checker/Assertion Failed	Assertion assert_data_in_bus_lv1_lv2_busrd_busrdx failed which is described as below in the system_bus_interface:
	property data_in_bus_lv1_lv2_busrd_busrdx;
	@(posedge clk)
	data_in_bus_lv1_lv2 |=> (bus_rd!==1'b1 & bus_rdx!==1'b1);
	endproperty
	assert_data_in_bus_lv1_lv2_busrd_busrdx : assert
	property(data_in_bus_lv1_lv2_busrd_busrdx)
	else
	`uvm_error("system_bus_interface",$sformatf("Assertion-20 assert_data_in_bus_lv1_lv2_busrd_busrdx Failed: bus_rd and/or bus_rdx asserted when data_in_bus_lv1_lv2 asserted"))

Checker Description	The signals bus_rd and bus_rdx should be deasserted when data_in_bus_lv1_lv2 is asserted.
Bug Description/ Debug Process	After fixing bug2 we observed that the assertion **assert_data_in_bus_lv1_lv2_busrd_busrdx** is failing, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that the signal **bus_rdx** remains high even after **data_in_bus_lv1_lv2** is deasserted (as shown in fig.2), which is a violation similar to bug1. Due to this discrepancy, the above-mentioned assertion has failed. The value for **bus_rdx** comes from **bus_rdx_reg** and an incorrect logic for **bus_rdx_reg** assignment is noticed under the write miss logic within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.3 and fig.4 respectively. After fixing the bug we observed that there is no assertion failure (implied by the UVM Summary report in fig.4). Further analysis using Simvision waveform shows the expected behavior of the signal **bus_rdx** as shown below in fig.5
Original Code	bus_rdx_reg <= 1'b1;
Code after Fix	bus_rdx_reg <= 1'b0;

```
UVM_INFO ../uvm/system_bus_monitor_c.sv(78) @ 205: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(194) @ 275: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] BUS_RDX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(225) @ 285: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
                    data_in_bus_lv1_lv2 |-> (bus_rd=/=1'b1 & bus_rdx=/=1'b1);
                                        |
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,207): (time 295 NS) Assertion top.inst_system_bus_if.assert_data_in_bus_lv1_lv2_busrd_b
usrdx has failed (2 cycles, starting 285 NS)
UVM_ERROR ../uvm/system_bus_interface.sv(209) @ 295: reporter [system_bus_interface] Assertion-20 assert_data_in_bus_lv1_lv2_busrd_busrd
x Failed: bus_rd and/or bus_rdx asserted when data_in_bus_lv1_lv2 asserted
UVM_INFO ../uvm/cache_scoreboard_c.sv(515) @ 305: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU0:
--------------------------------------------------
```

Fig.1

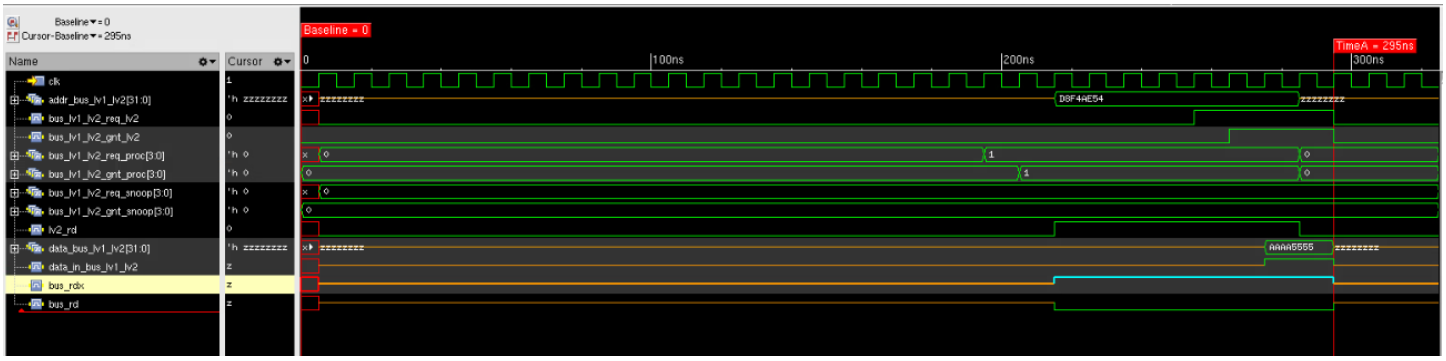


Fig.2 Waveform before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   27
UVM_WARNING :    0
UVM_ERROR :    1
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]     5
[cpu_driver_c]     6
[cpu_monitor_c]     4
[system_bus_interface]     1
[system_bus_monitor_c]     4
[write_miss_dcache]     1
[write_miss_dcache_seq]     3
Simulation complete via $finish(1) at time 325 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   27
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]     5
[cpu_driver_c]     6
[cpu_monitor_c]     4
[system_bus_monitor_c]     4
[write_miss_dcache]     1
[write_miss_dcache_seq]     3
Simulation complete via $finish(1) at time 325 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.4 UVM Report Summary after bug fix

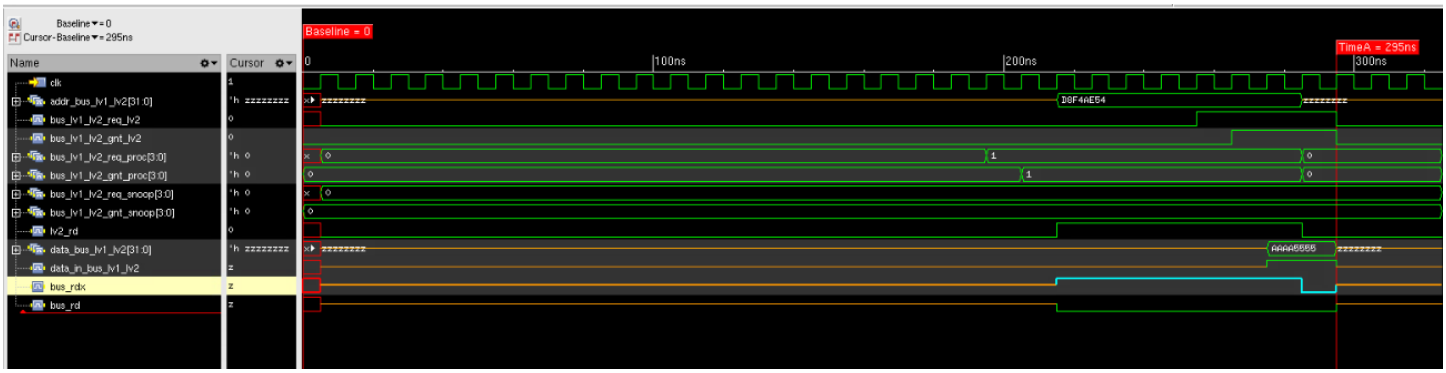


Fig.5 Waveform after bug fix

Team-16 Bug 4 Report

Team Number	**# 16**
Bug Number	# 4
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 221
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_request_shared: We have included this test case to verify the handling of write requests when the data is in a shared state. This involves reading data from an address that is currently present on another core, followed by a write request to the same address.
Checker/Assertion Failed	Assertion assert_cpu_wr_done_cpu_wr failed which is described as below in the cpu_lv1_interface: property prop_cpu_wr_done_cpu_wr; @(posedge clk) cpu_wr |-> ##[0:100] cpu_wr_done; endproperty assert_cpu_wr_done_cpu_wr : assert property(prop_cpu_wr_done_cpu_wr) else `uvm_error("cpu_lv1_interface",$sformatf("Assertion-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined time of 100 cycles"))
Checker Description	The signals cpu_wr_done should be asserted within defined time after the signal cpu_wr is asserted.
Bug Description/ Debug Process	After running the test, we observed that the assertion **assert_cpu_wr_done_cpu_wr** is failing, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that the signal **invalidate** remains **1'bz** even after **bus_lv1_lv2_gnt_proc** is asserted (as shown in fig.2), which violates the specifications outlined in the HAS document(as shown on fig.6). Due to this discrepancy, the above-mentioned assertion has failed. The value for invalidate comes from **invalidate_reg** and an incorrect logic for **invalidate_reg** assignment is noticed under the write hit shared logic within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.3 and fig.4 respectively. After fixing the bug we observed that there is no assertion failure (implied by the UVM Summary report in fig.4). Further analysis using Simvision waveform shows the same result as shown below in fig.5
Original Code	invalidate_reg <= 1'bz;
Code after Fix	invalidate_reg <= 1'b1;

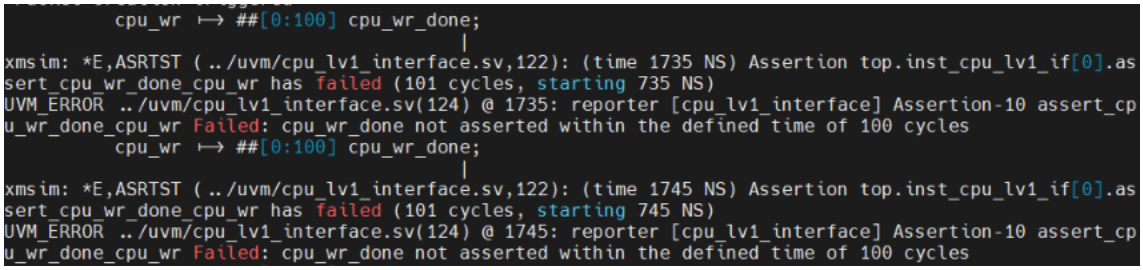
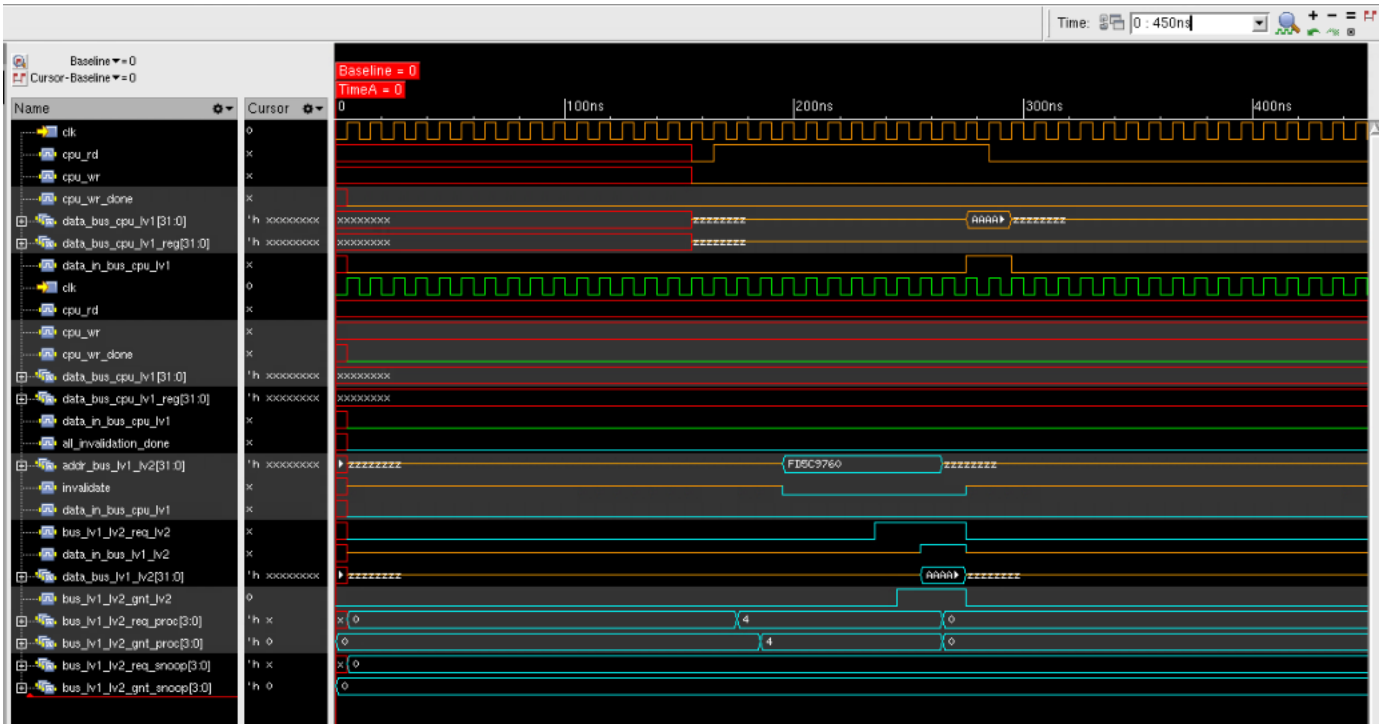
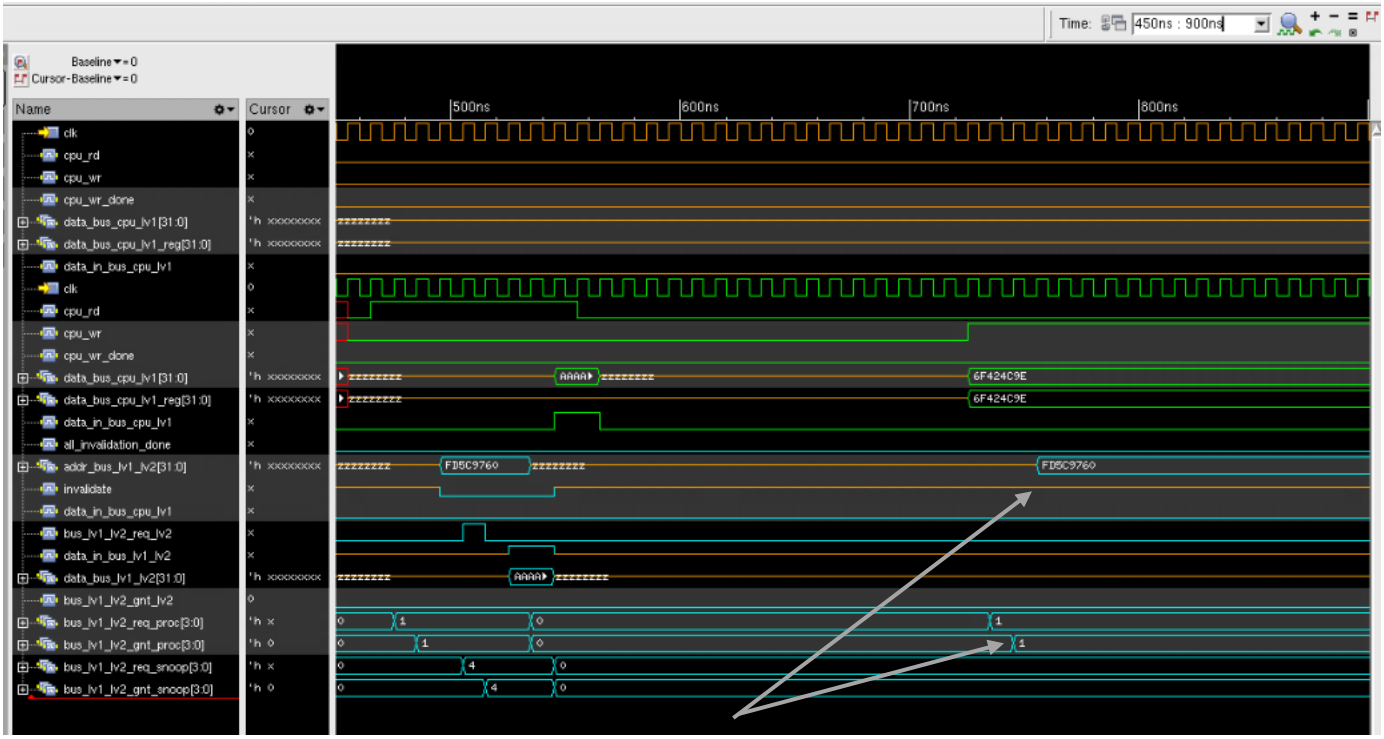


Fig.1



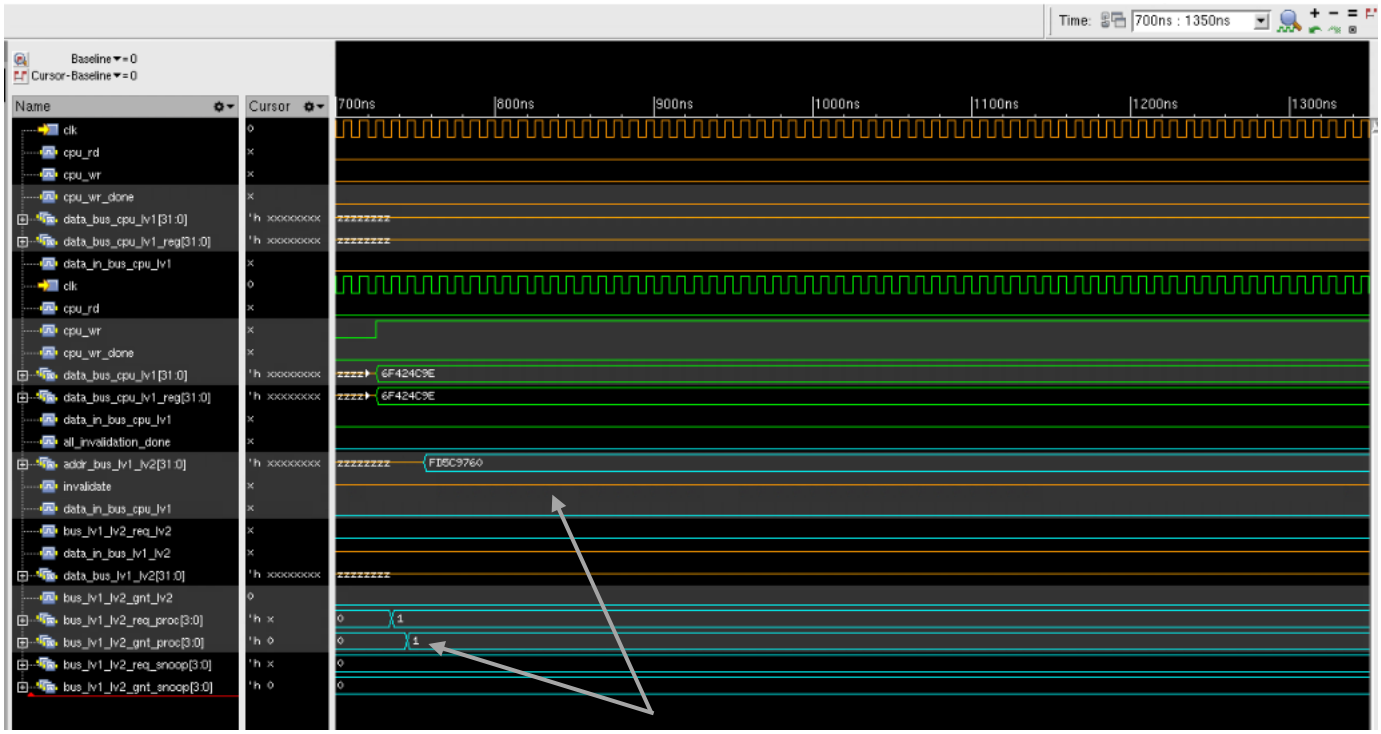
Time Frame: 0:450ns

Fig.2a Signals in orange, green and blue represent signals of cpu[2], cpu[0] and system bus interface respectively



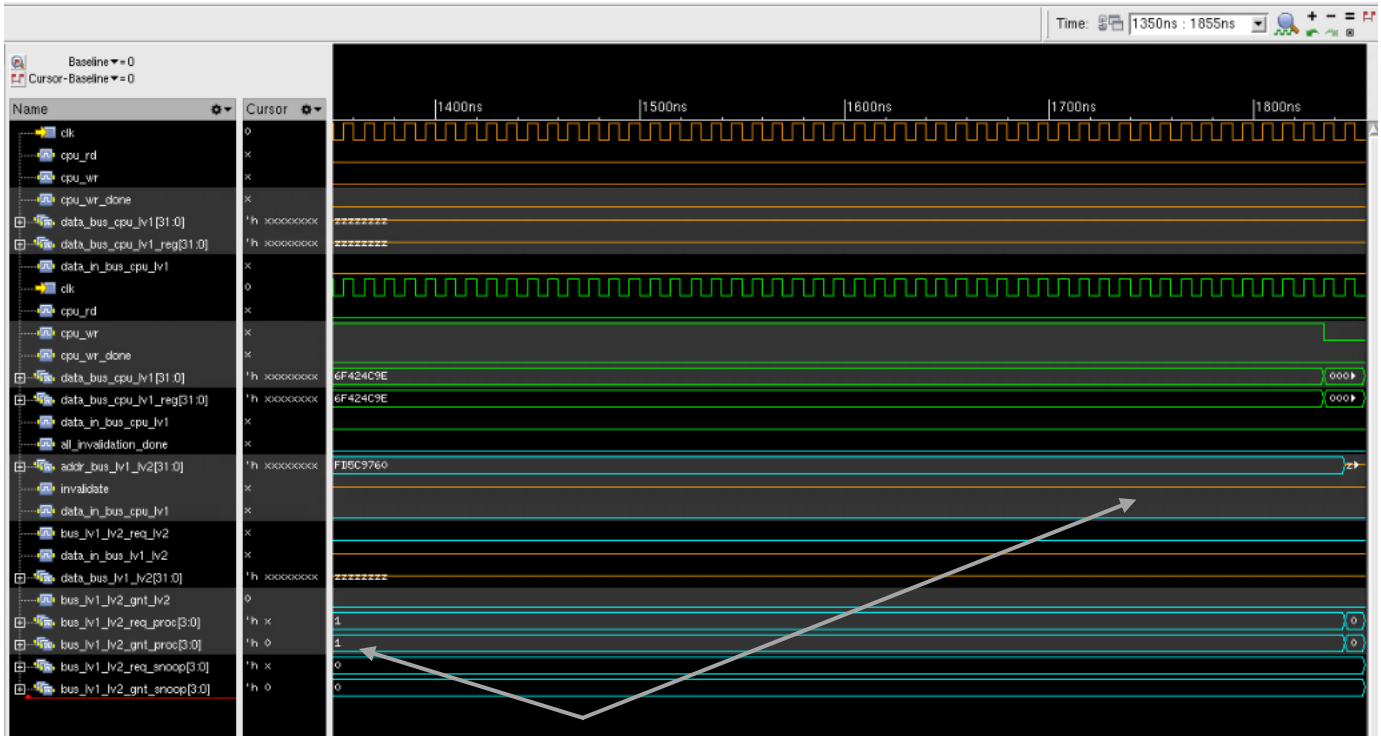
Time Frame: 450:900ns

Fig.2b Signals in orange, green and blue represent signals of cpu[2], cpu[0] and system bus interface respectively



Time Frame: 900:1350ns

Fig.2c Signals in orange, green and blue represent signals of cpu[2], cpu[0] and system bus interface respectively



Time Frame: 1350:1855ns

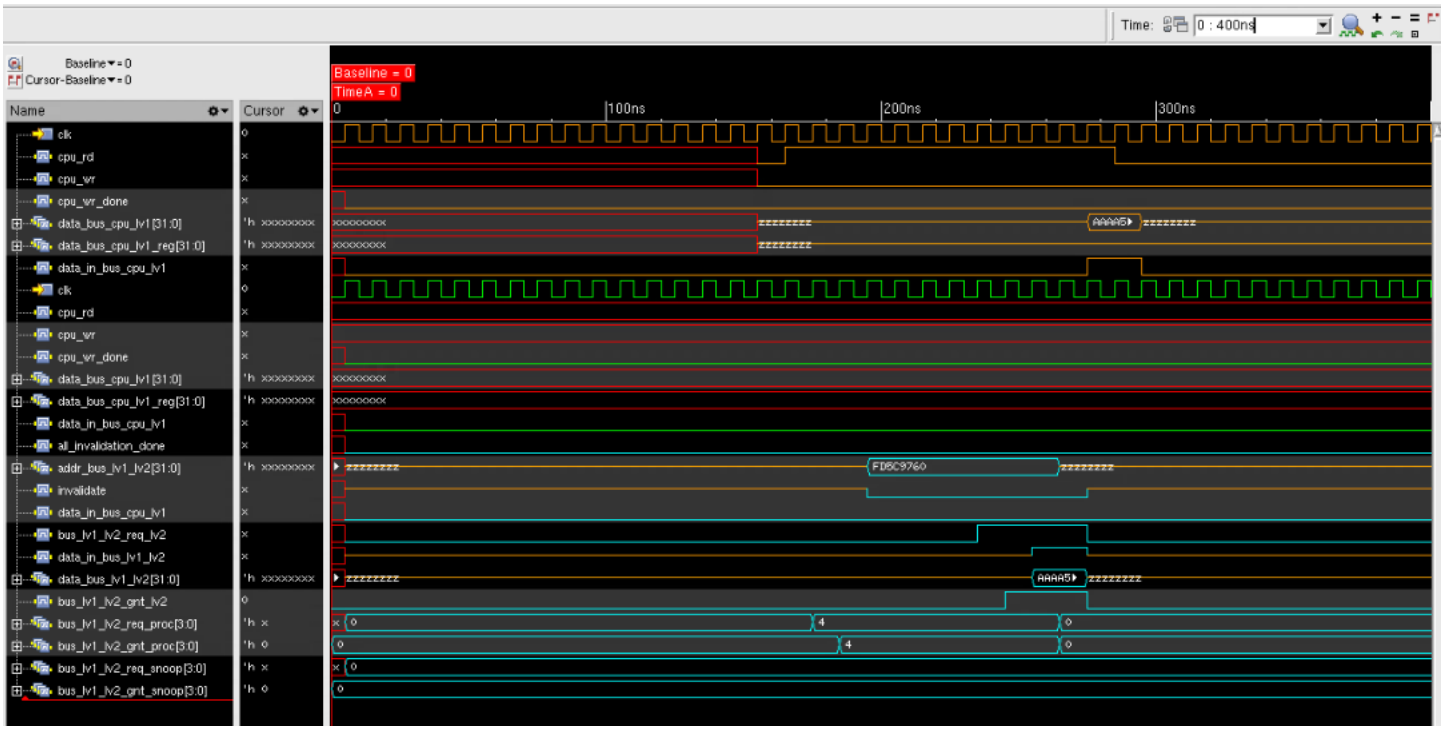
Fig.2e Signals in orange, green and blue represent signals of cpu[2], cpu[0] and system bus interface respectively

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   41
UVM_WARNING :    0
UVM_ERROR :   12
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    11
[cpu_driver_c]    10
[cpu_lv1_interface]    12
[cpu_monitor_c]     4
[system_bus_monitor_c]     8
[write_request_shared]     1
[write_request_shared_seq]     3
Simulation complete via $finish(1) at time 1855 NS + 51
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   47
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    15
[cpu_driver_c]    10
[cpu_monitor_c]     4
[system_bus_monitor_c]    10
[write_request_shared]     1
[write_request_shared_seq]     3
Simulation complete via $finish(1) at time 805 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

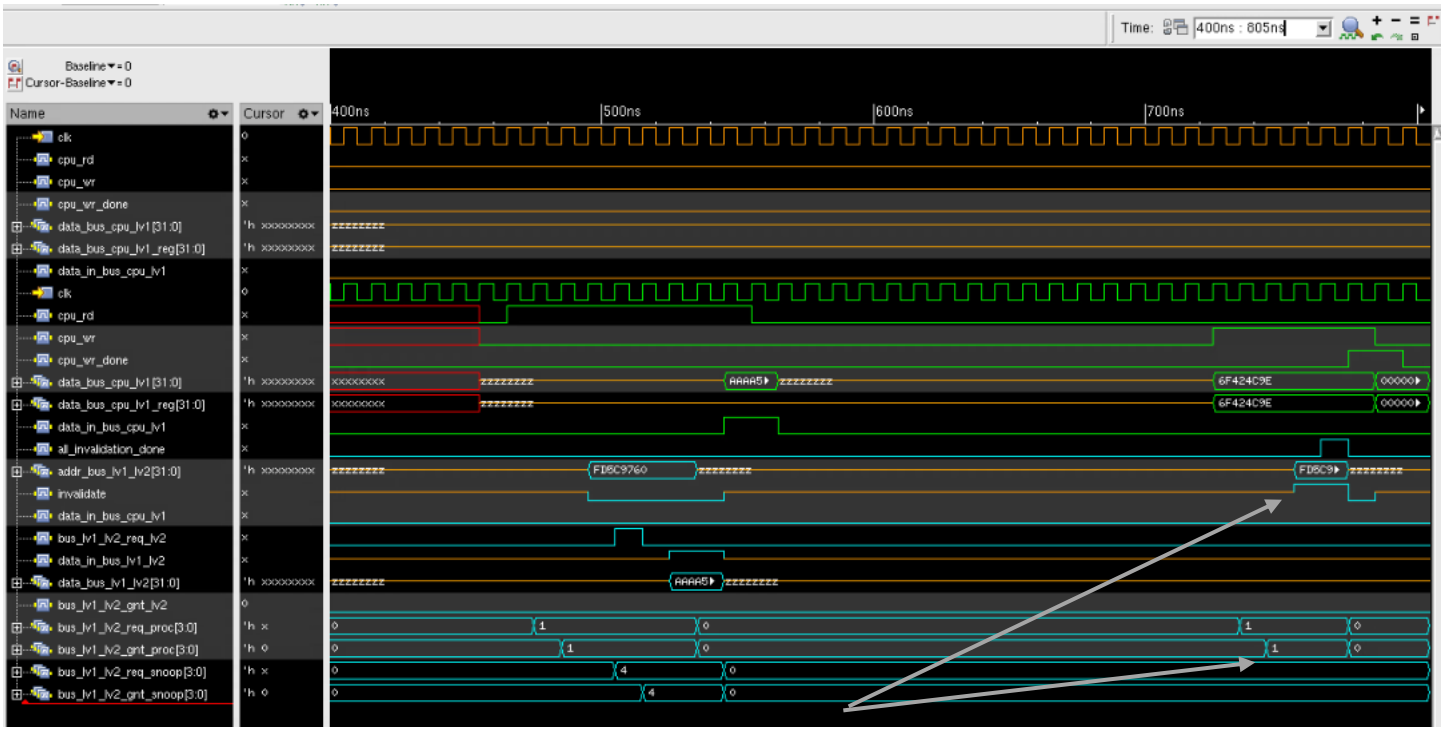
Fig.4 UVM Report Summary after bug fix



Time Frame: 0:400ns

Fig.5a Waveform after bug fix

Signals in orange, green and blue represent signals of cpu[2], cpu[0] and system bus interface respectively



Time Frame: 400: 805ns

Fig.5b Waveform after bug fix

Signals in orange, green and blue represent signals of cpu[2], cpu[0] and system bus interface respectively

If the Cache data is in Shared condition then the following is carried out

- Once bus_lv1_lv2_gnt_proc is high;
- Address of the block to be invalidated is regenerated from its TAG (stored in cache_proc_contr) and index_proc value and is put in addr_bus_lv1_lv2 bus.
- Signal "invalidate" is made high asking other level 1 caches to make their copy, if present, to be Invalid.
- When all such copies are invalidated, all_invalidation_done is made high.
- The cache_var is then updated with data_bus_lv1_lv2 value.
- cache_proc_contr [index_proc, blk_access_proc][MESI] value is updated with updated_mesi_proc value.
- cpu_wr_done is raised.
- bus_lv1_lv2_req_proc is deasserted

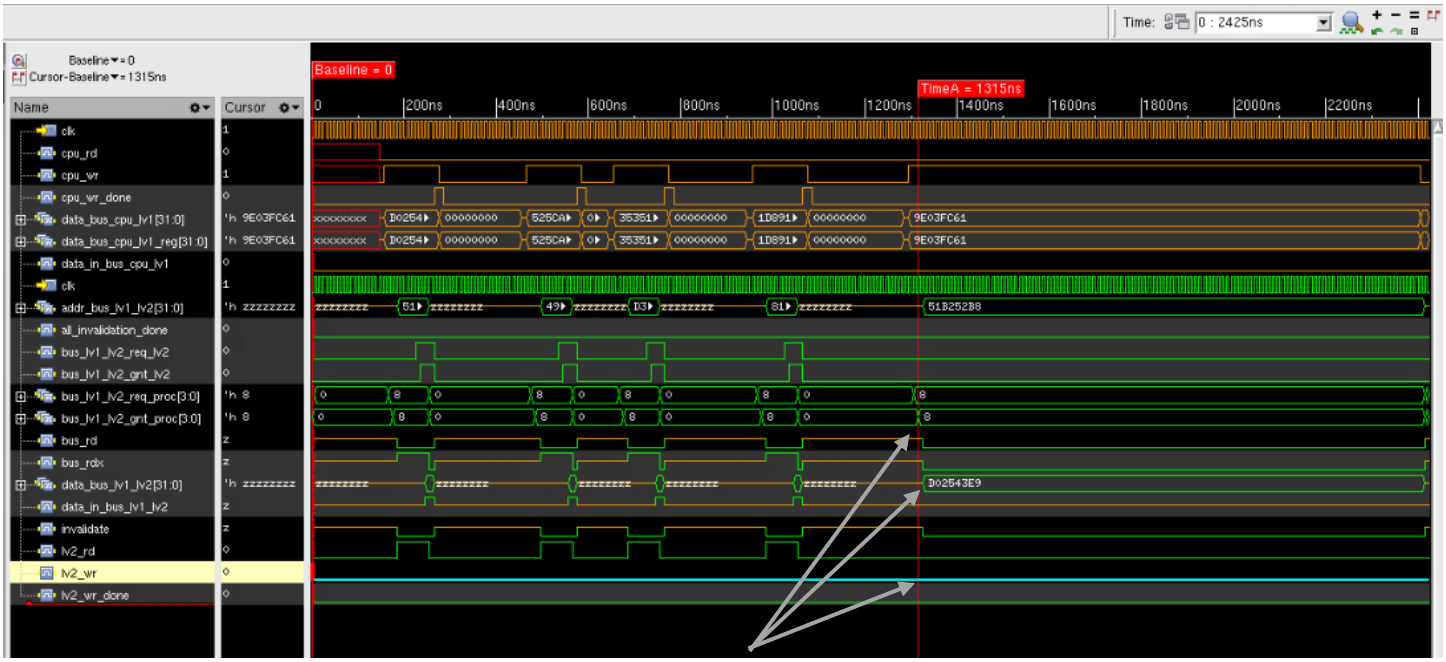
Fig.6 HAS Document Statement supporting failed Assertion.

Team-16 Bug 5 Report

Team Number	**# 16**
Bug Number	# 5
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 280
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss_dcache_replace: We have included this test case to verify the handling of write miss request on DCACHE when there is no free block available i.e., replacement needed. The process involves initially writing different data into the available free blocks of the cache. Subsequently, another write request is made to a different address, ultimately requiring the replacement of one of the existing blocks. All these transactions occur on the same core.
Checker/Assertion Failed	Assertion assert_cpu_wr_done_cpu_wr failed which is described as below in the cpu_lv1_interface: property prop_cpu_wr_done_cpu_wr; @(posedge clk) cpu_wr |-> ##[0:100] cpu_wr_done; endproperty assert_cpu_wr_done_cpu_wr : assert property(prop_cpu_wr_done_cpu_wr) else `uvm_error("cpu_lv1_interface",$sformatf("Assertion-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined time of 100 cycles"))
Checker Description	The signal cpu_wr_done should be asserted within the defined time after the signal cpu_wr is asserted.
Bug Description/ Debug Process	After running the test, we observed that the assertion **assert_cpu_wr_done_cpu_wr** is failing, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that the signal **lv2_wr** remains **1'b0** even after **bus_lv1_lv2_gnt_proc** is asserted and **data_bus_lv1_lv2** is loaded with dirty data(as shown in fig.2a,2b,2c), which violates the specifications outlined in the HAS document(as shown on fig.6). We observed an incorrect logic for **lv2_wr** assignment under the Write Miss Replacement logic within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.3 and fig.4 respectively. Unexpectedly, even after addressing the bug, the assertion still failed, prompting us to investigate another potential bug (which we are currently exploring). However, post-bug fix, we observed the expected behavior of the signal **lv2_wr** using Simvision waveform, as depicted in fig. 5a, 5b, and 5c.
Original Code	lv2_wr <= 1'b0;
Code after Fix	lv2_wr <= 1'b1;

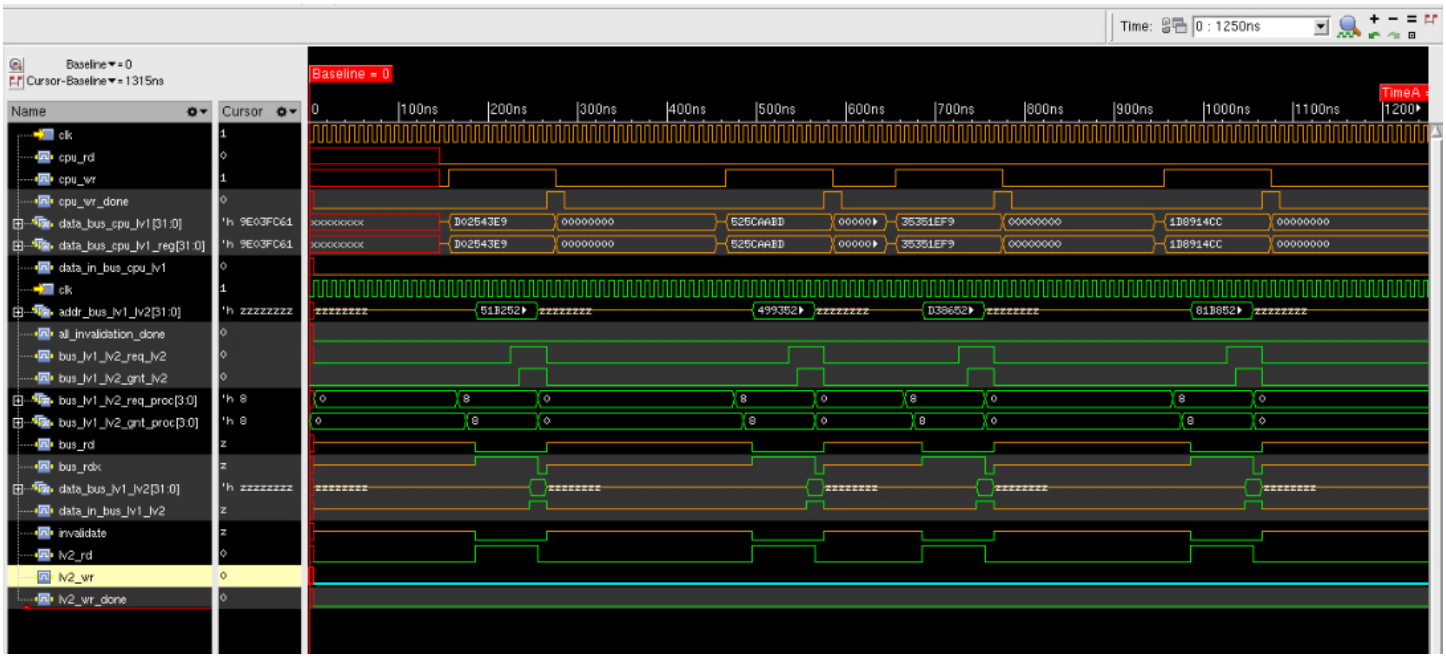
```
          cpu_wr ⟼ ##[0:100] cpu_wr_done;
                                    |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,122): (time 2305 NS) Assertion top.inst_c
pu_lv1_if[3].assert_cpu_wr_done_cpu_wr has failed (101 cycles, starting 1305 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(124) @ 2305: reporter [cpu_lv1_interface] Asserti
on-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined tim
e of 100 cycles
          cpu_wr ⟼ ##[0:100] cpu_wr_done;
                                    |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,122): (time 2315 NS) Assertion top.inst_c
pu_lv1_if[3].assert_cpu_wr_done_cpu_wr has failed (101 cycles, starting 1315 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(124) @ 2315: reporter [cpu_lv1_interface] Asserti
on-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined tim
e of 100 cycles
```

Fig.1



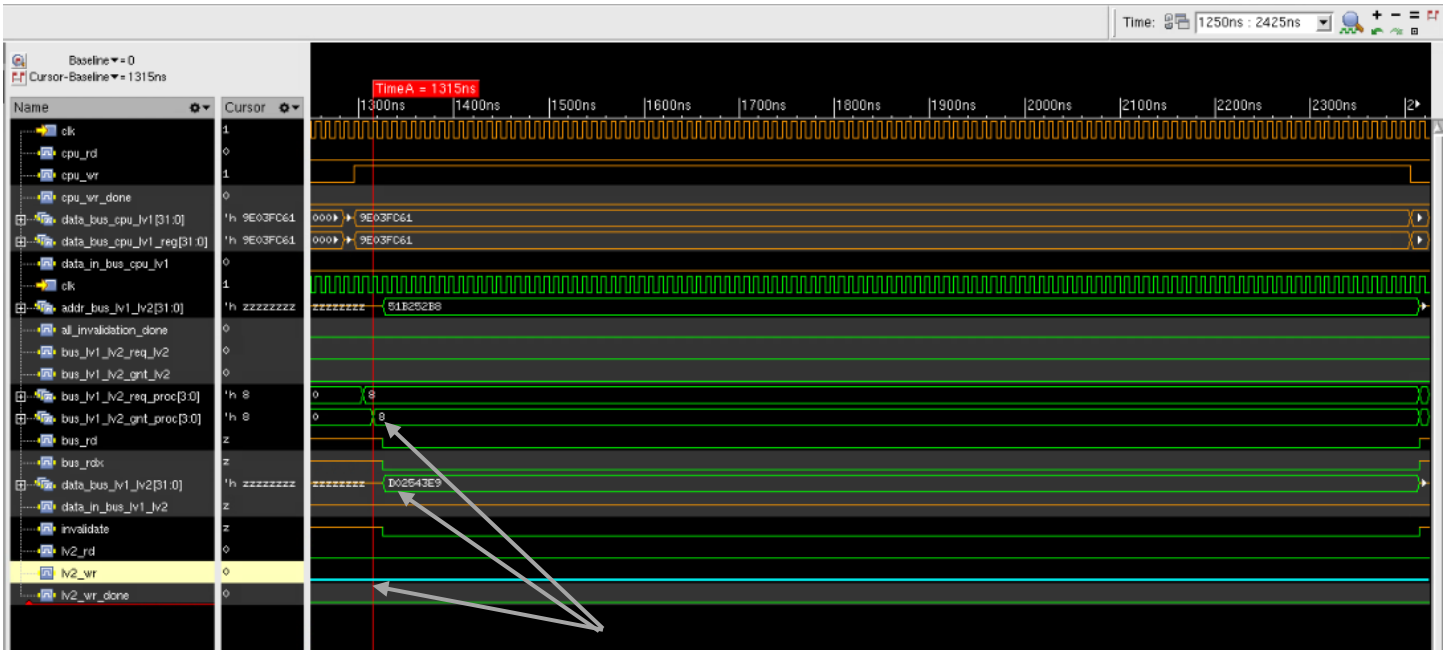
Full time frame :0:2425ns

Fig.2a Signals in orange, green and blue represent signals of cpu[3], system bus interface and unexpected signal behavior respectively



Time Frame1: 0:1250ns

Fig.2b Signals in orange, green and blue represent signals of cpu[3], system bus interface and unexpected signal behavior respectively



Time Frame2: 1250:2425ns

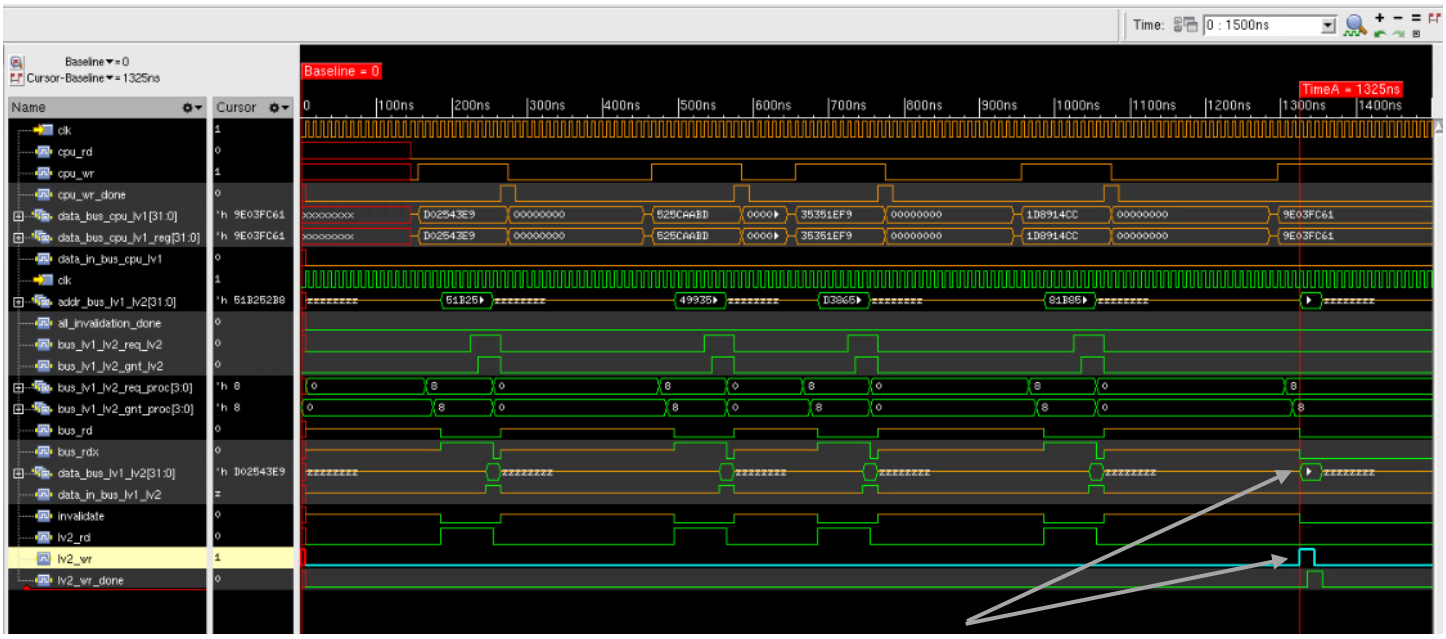
Fig.2c Signals in orange, green and blue represent signals of cpu[3], system bus interface and unexpected signal behavior respectively

```
** Report counts by severity
UVM_INFO :   57
UVM_WARNING :    0
UVM_ERROR :   12
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    17
[cpu_driver_c]    14
[cpu_lv1_interface]    12
[cpu_monitor_c]     4
[system_bus_monitor_c]    14
[write_miss_dcache_replace]     1
[write_miss_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 2425 NS + 51
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457
   $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
** Report counts by severity
UVM_INFO :   57
UVM_WARNING :    0
UVM_ERROR :   12
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    17
[cpu_driver_c]    14
[cpu_lv1_interface]    12
[cpu_monitor_c]     4
[system_bus_monitor_c]    14
[write_miss_dcache_replace]     1
[write_miss_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 2425 NS + 51
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457
   $finish;
xcelium> exit
```

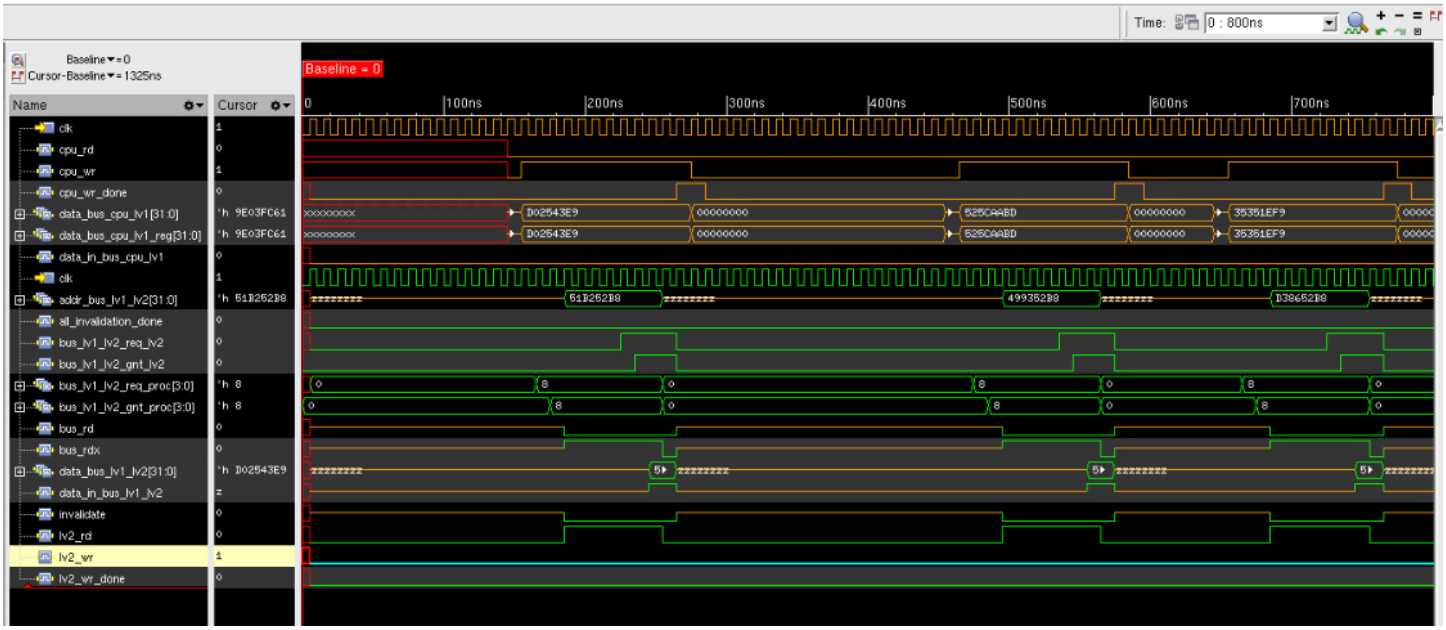
Fig.4 UVM Report Summary after bug fix



Time Frame: 0:1500ns

Fig.5a Waveform after bug fix

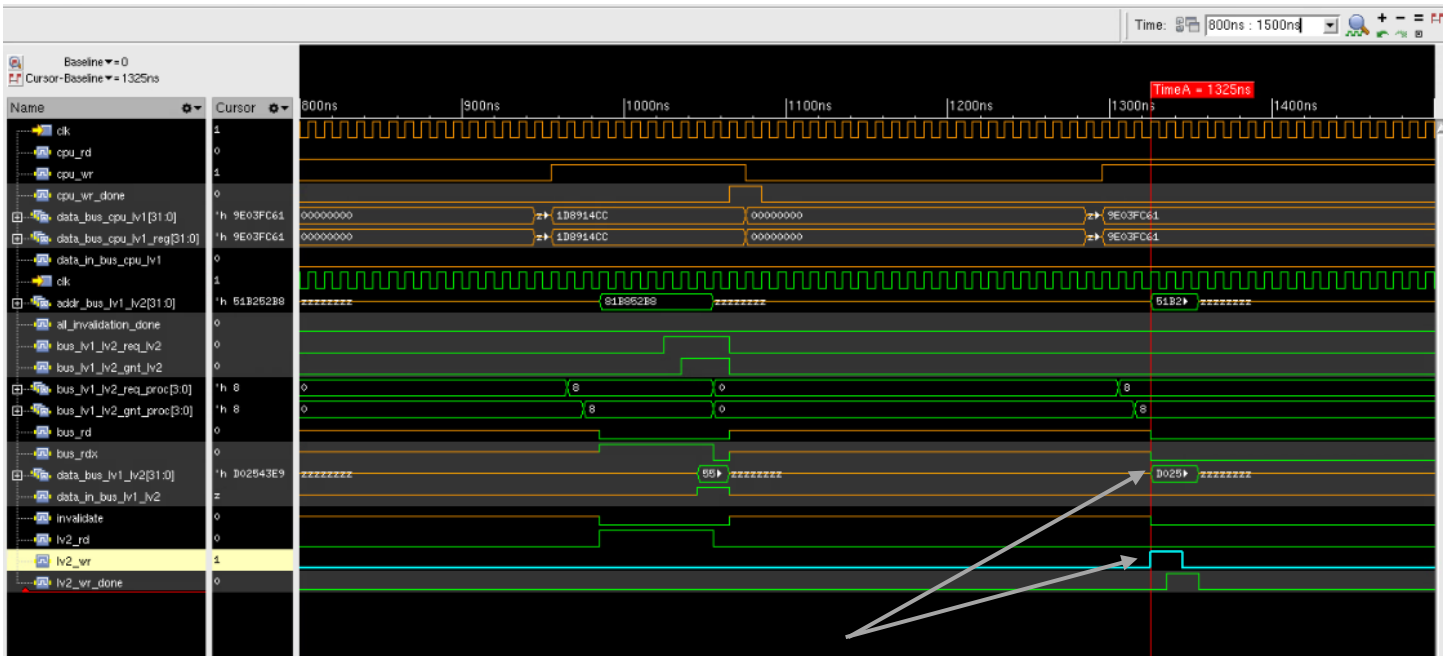
Signals in orange, green and blue represent signals of cpu[3], system bus intf and expected signal after fix respectively



Time Frame1: 0: 800ns

Fig.5b Waveform after bug fix

Signals in orange, green and blue represent signals of cpu[3], system bus intf and expected signal after fix respectively



Time Frame2: 800:1500ns

Fig.5c Waveform after bug fix

Signals in orange, green and blue represent signals of cpu[3], system bus intf and expected signal after fix respectively

(2) Free block not available, replacement needed.

If the replacement to be done is Shared/Exclusive then the cache_proc [Index_proc, blk_access_proc][MESI_range] is made Invalid.

If the replacement is in Modified state then the following is carried out (as dirty replacement needs to be updated in level 2 cache).

- When bus_lv1_lv2_gnt_proc is made high, address of the replacement block is regenerated from its TAG (stored in cache_proc_contr) and Index_proc value and is put in addr_bus_lv1_lv2.
- data_bus_lv1_lv2 is loaded with the dirty data.
- lv2_wr is made high asking level 2 cache to update its value.
- Once lv2_wr_done is made high by level 2 cache, cache_proc [index_proc, blk_access_proc] [MESI_range] is made Invalid.

Fig.6 HAS Document Statement supporting the bug.

Team-16 Bug 6 Report

Team Number	**# 16**
Bug Number	# 6
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 301
Bug Type	Functional Bug
SV Test Run to uncover the Bug	modified_to_invalid, exclusive_to_invalid are tests that failed in finding this bug. **modified_to_invalid**: This test case verifies the transition of a data block from the modified state to the invalid state. It begins with a write request to a specific address on Core X. Subsequently, a write and read request are issued on a different core (Core Y) for the same address as the initial write request on Core X. **exclusive_to_invalid:** This test case validates the transition of a data block from the exclusive to the invalid state. It begins with a read request to a specific address on Core X. Then, a write and read request occur on a different core (Core Y) for the same address as the initial read request on Core X.
Checker/Assertion Failed	expected = expected_sbus[i].pop_front(); received = received_sbus[i].pop_front(); if(expected.compare(received)) begin `uvm_info(get_type_name(), $sformatf("System bus activity matched for this request"),UVM_LOW) `uvm_info(get_type_name(), $sformatf("Expected & Received SBUS Packet \n%s", received.sprint()),UVM_LOW) end else begin `uvm_error(get_type_name(), $sformatf("System bus activity MISMATCH!!!")) `uvm_info(get_type_name(), $sformatf("Expected SBUS Packet \n%s", expected.sprint()),UVM_LOW) `uvm_info(get_type_name(), $sformatf("Received SBUS Packet \n%s", received.sprint()),UVM_LOW) end
Checker Description	This essentially verifies whether the packet details received on the system bus match the expected values or not.
Bug Description/ Debug Process	After running the test, we observed that there is a MISMATCH in the system bus activity, as indicated in the log file (refer to fig.1). There is a mismatch in the signal cp_in_cache(where expected is 1'b1 and received is 1'b0), as shown in fig.2. Upon analyzing the waveform in Simvision, we noticed that the signal **cp_in_cache** is not getting asserted even after **bus_lv1_lv2_req_snoop** is asserted (as shown in fig.3), which violates the specifications outlined in the HAS document (as shown on fig.7). We also noticed that an incorrect assignment was done to the signal **cp_in_cache** as **1'b0,** which instead should be **1'b1** within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.4 and fig.5 respectively. After fixing the bug we observed that there is no mismatch in the packet details (implied by the UVM Summary report in fig.5). Further analysis using Simvision waveform shows the correct behavior of cp_in_cache as shown below in fig.6
Original Code	cp_in_cache <= 1'b0;
Code after Fix	cp_in_cache <= 1'b1;

```
UVM_INFO ../uvm/cache_scoreboard_c.sv(274) @ 555: uvm_test_top.tb.sb [cache_scoreboard_c] CHECK_DATA CPU0
WRITE Request
UVM_INFO @ 555: reporter [MISCMP] Miscompare for expected.wr_data_snoop: lhs = 'ha3dcddb9 : rhs = 'h0
UVM_INFO @ 555: reporter [MISCMP] 2 Miscompare(s) (1 shown) for object s_packet@7964 vs. expected@7827
UVM_ERROR ../uvm/cache_scoreboard_c.sv(433) @ 555: uvm_test_top.tb.sb [cache_scoreboard_c] System bus activity MISMATCH!!!
UVM_INFO ../uvm/cache_scoreboard_c.sv(434) @ 555: uvm_test_top.tb.sb [cache_scoreboard_c] Expected SBUS Packet
```

Fig.1

```
----------------------------------------------------------------
Name                          Type            Size  Value
----------------------------------------------------------------
expected                      sbus_packet_c   -     @7827
  bus_req_type                bus_req_t       32    BUS_RDX
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC0
  req_address                 integral        32    'h6d2c336c
  bus_req_snoop               integral        4     'h2
  req_serviced_by             serv_by_t       32    SERV_L2
  rd_data                     integral        32    'ha3dcddb9
  wr_data_snoop               integral        32    'ha3dcddb9
  snoop_wr_req_flag           integral        1     'h1
  cp_in_cache                 integral        1     'h1
  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
----------------------------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(435) @ 555: uvm_test_top.tb.sb [cache_scoreboard_c] Received SBUS Packet
----------------------------------------------------------------
Name                          Type            Size  Value
----------------------------------------------------------------
s_packet                      sbus_packet_c   -     @7964
  bus_req_type                bus_req_t       32    BUS_RDX
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC0
  req_address                 integral        32    'h6d2c336c
  bus_req_snoop               integral        4     'h2
  req_serviced_by             serv_by_t       32    SERV_L2
  rd_data                     integral        32    'ha3dcddb9
  wr_data_snoop               integral        32    'h0
  snoop_wr_req_flag           integral        1     'h1
  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
```

Fig. 2 Mismatch in packet details

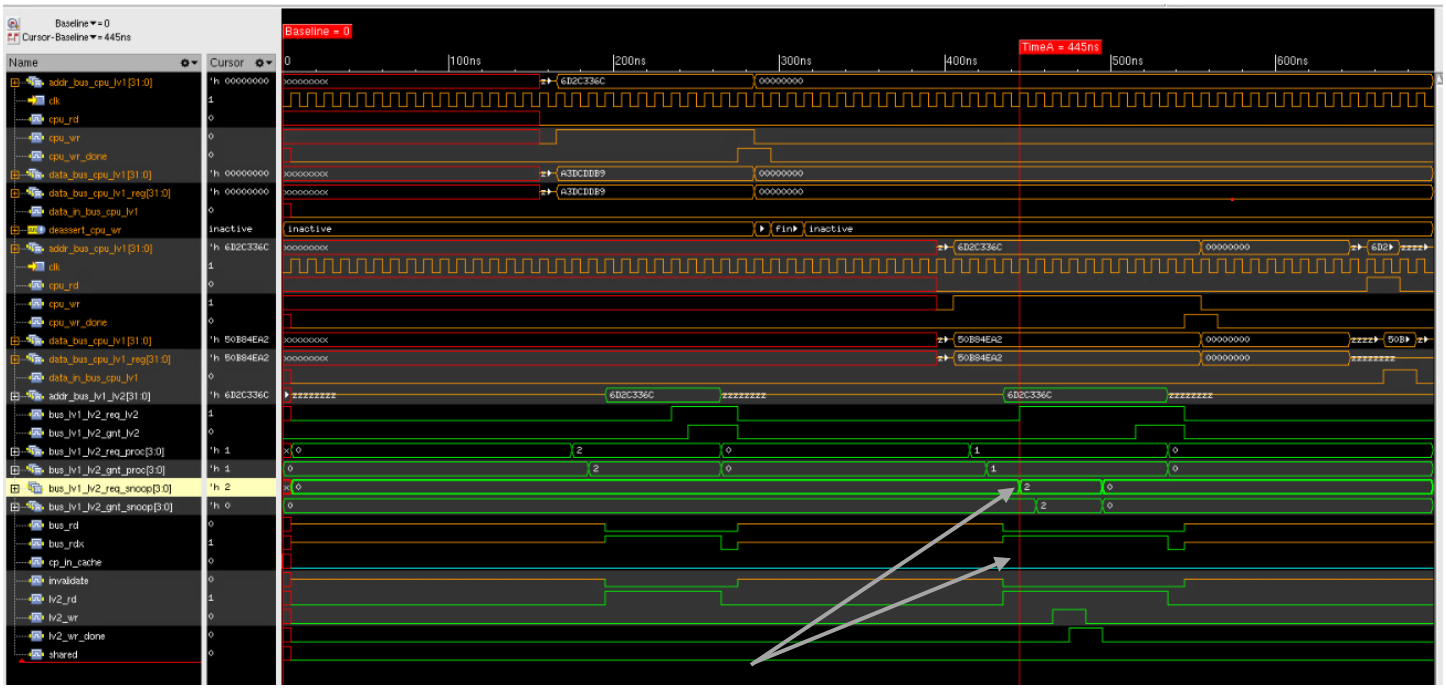


Fig.3 Waveform before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   44
UVM_WARNING :    0
UVM_ERROR :    1
UVM_FATAL :    0
** Report counts by id
[MISCMP]     2
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    14
[cpu_driver_c]    10
[cpu_monitor_c]     4
[modified_to_invalid]     1
[modified_to_invalid_seq]     3
[system_bus_monitor_c]     7
Simulation complete via $finish(1) at time 695 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.4 UVM Report Summary before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   42
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    13
[cpu_driver_c]    10
[cpu_monitor_c]     4
[modified_to_invalid]     1
[modified_to_invalid_seq]     3
[system_bus_monitor_c]     7
Simulation complete via $finish(1) at time 695 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.5 UVM Report Summary after bug fix

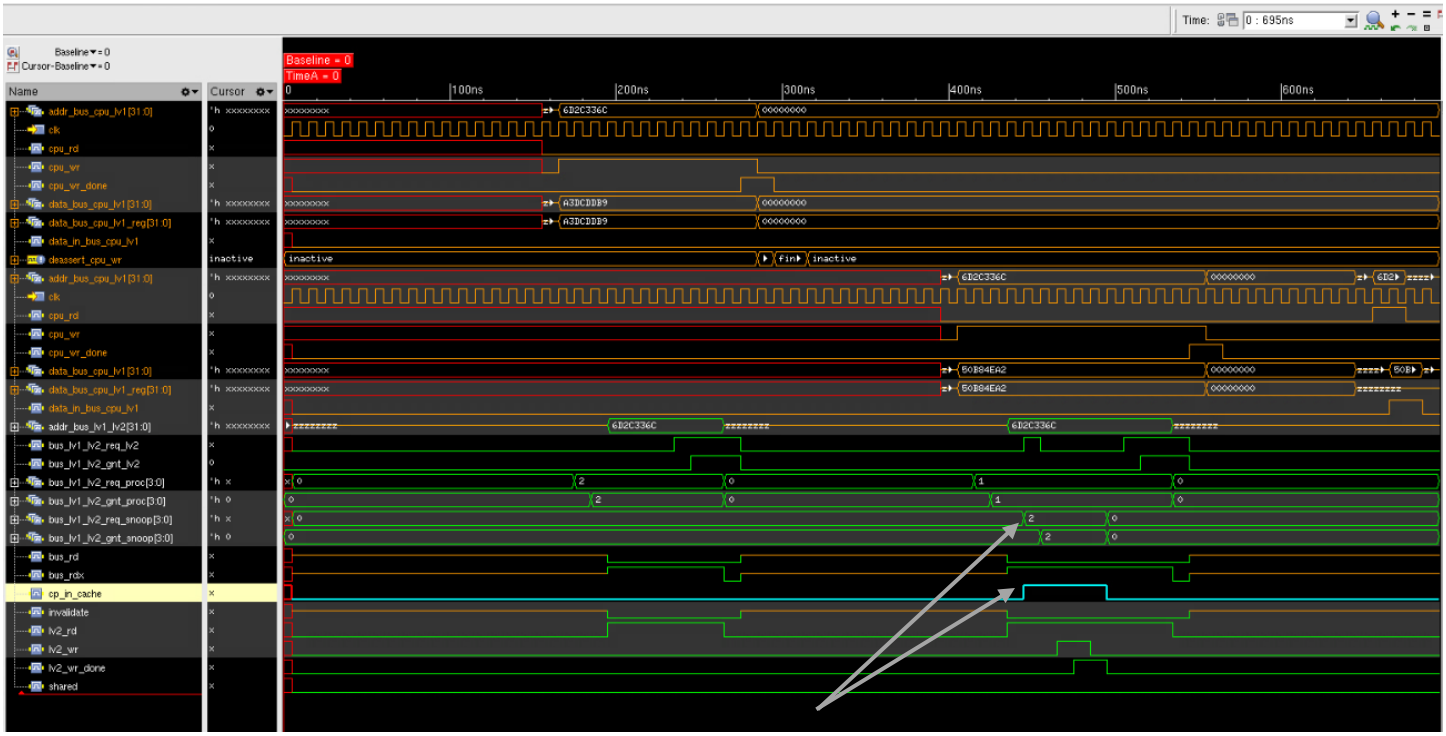


Fig.6 Waveform after bug fix

Snoop Side

On the snoop side of the level 1 caches with copies following happens (The block is hit in other snooping cache)

- cp_in_cache is raised.

a) If Copy is in Shared State

- Signal shared_local is made high.
- cache_proc_contr [index_snoop, blk_access_snoop] [MESI_range] value is updated

b) If Copy is in Exclusive state

- cache_proc_contr [index_snoop, blk_access_snoop] [MESI_range] value is updated

c) If Copy is in Modified state

- bus_lv1_lv2_req_snoop is raised to ask for access to the bus.
- Wait for bus_lv1_lv2_gnt_snoop to be high.
- data_bus_lv1_lv2 is loaded with data from Modified copy.
- Signal lv2_wr is asserted to make level 2 cache update its value.

- Wait for lv2_wr_done to be high.
- cache_proc_contr [index_snoop, blk_access_snoop] [MESI_range] value is updated
- bus_lv1_lv2_req_snoop is deasserted.

Only level 2 cache provides the data. It informs the requestor by making data_in_bus_lv1_lv2 high. After this operation, the block will become block hit and previous operation is carried out.

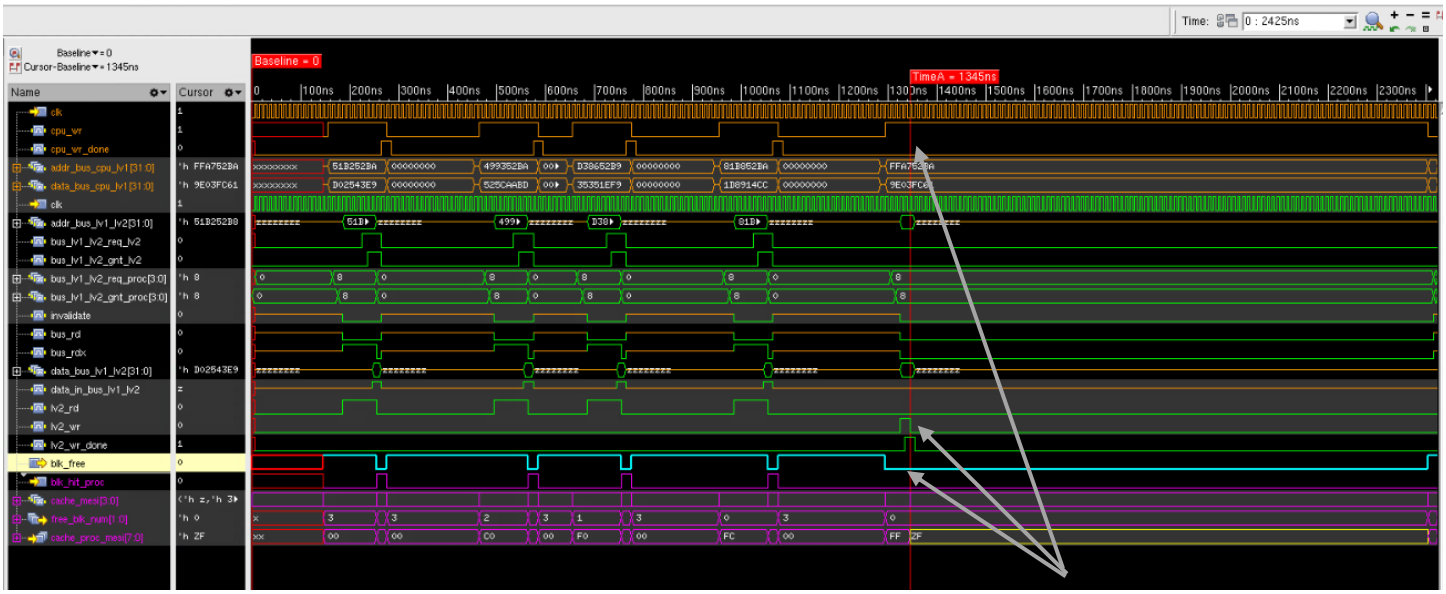
Fig.7 HAS Document Statement supporting failed Assertion.

Team-16 Bug 7 Report

Team Number	# 16
Bug Number	# 7
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 80
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss_dcache_replace: We have included this test case to verify the handling of write miss request on DCACHE when there is no free block available i.e., replacement needed. The process involves initially writing different data into the available free blocks of the cache. Subsequently, another write request is made to a different address, ultimately requiring the replacement of one of the existing blocks. All these transactions occur on the same core.
Checker/Assertion Failed	Assertion assert_cpu_wr_done_cpu_wr failed which is described as below in the cpu_lv1_interface: property prop_cpu_wr_done_cpu_wr; @(posedge clk) cpu_wr |-> ##[0:100] cpu_wr_done; endproperty assert_cpu_wr_done_cpu_wr : assert property(prop_cpu_wr_done_cpu_wr) else `uvm_error("cpu_lv1_interface",$sformatf("Assertion-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined time of 100 cycles"))
Checker Description	The signal cpu_wr_done should be asserted within the defined time after the signal cpu_wr is asserted.
Bug Description/ Debug Process	Even after fixing bug5 and running this test, we observed that the assertion **assert_cpu_wr_done_cpu_wr** is failing again, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that there is an issue on replacement write which is evicting the existing block and replacing. Even after the **lv2_wr** is deasserted, the **blk_free** is not getting asserted (Refer fig.2a,2b,2c). When traced back in design/common/free_blk_md, we found that the **blk_free** is assigned **1'b1** if **cache_mesi** is **INVALID** which comes from the signal **cache_proc_mesi** in design/lv1/ main_func_lv1_dl. We found that the parameter **INVALID** is assigned with **2'bzz** instead of **2'b00** within the file main_func_lv1_dl under design/lv1/ folder. Due to this violation **cpu_wr_done** is not getting asserted as there is no free block to replace the new write request and caused the above assertion to fail. UVM Summary report before and after the fix is shown in fig.3 and fig.4 respectively. Unexpectedly, even after addressing the bug, we still got errors which can be seen in UVM Summary Report(fig.4) but this time the error is not due to the assertion failure instead it an error due to system bus activity mismatch (as shown in fig.5) which prompted us to investigate another potential bug (We will discuss that in Bug8 Report). However, post-bug fix, we observed the expected behavior of the signal **cpu_wr** and **cpu_wr_done** as **blk_free** is asserted after **lv2_wr** is done using Simvision waveform, as depicted in fig. 6a, 6b, and 6c.
Original Code	parameter INVALID = 2'bzz;
Code after Fix	parameter INVALID = 2'b00;

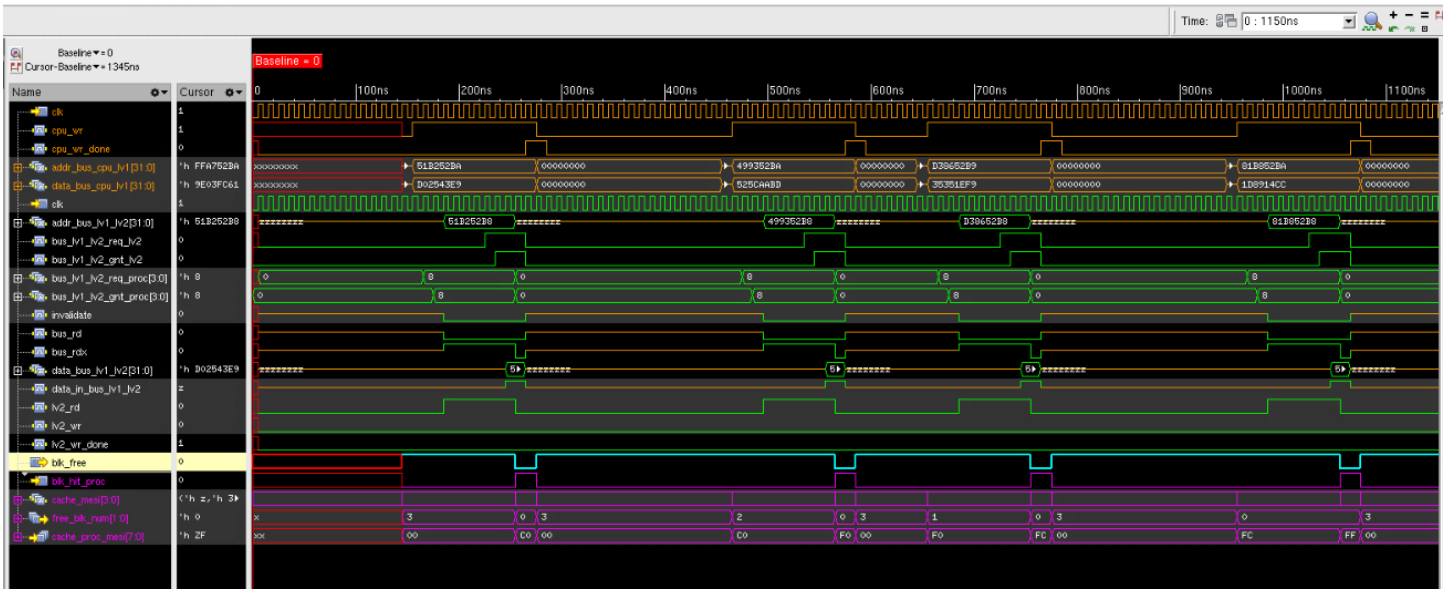
```
          cpu_wr ⟼ ##[0:100] cpu_wr_done;
                                    |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,122): (time 2305 NS) Assertion top.inst_c
pu_lv1_if[3].assert_cpu_wr_done_cpu_wr has failed (101 cycles, starting 1305 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(124) @ 2305: reporter [cpu_lv1_interface] Asserti
on-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined tim
e of 100 cycles
          cpu_wr ⟼ ##[0:100] cpu_wr_done;
                                    |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,122): (time 2315 NS) Assertion top.inst_c
pu_lv1_if[3].assert_cpu_wr_done_cpu_wr has failed (101 cycles, starting 1315 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(124) @ 2315: reporter [cpu_lv1_interface] Asserti
on-10 assert_cpu_wr_done_cpu_wr Failed: cpu_wr_done not asserted within the defined tim
e of 100 cycles
```

Fig.1



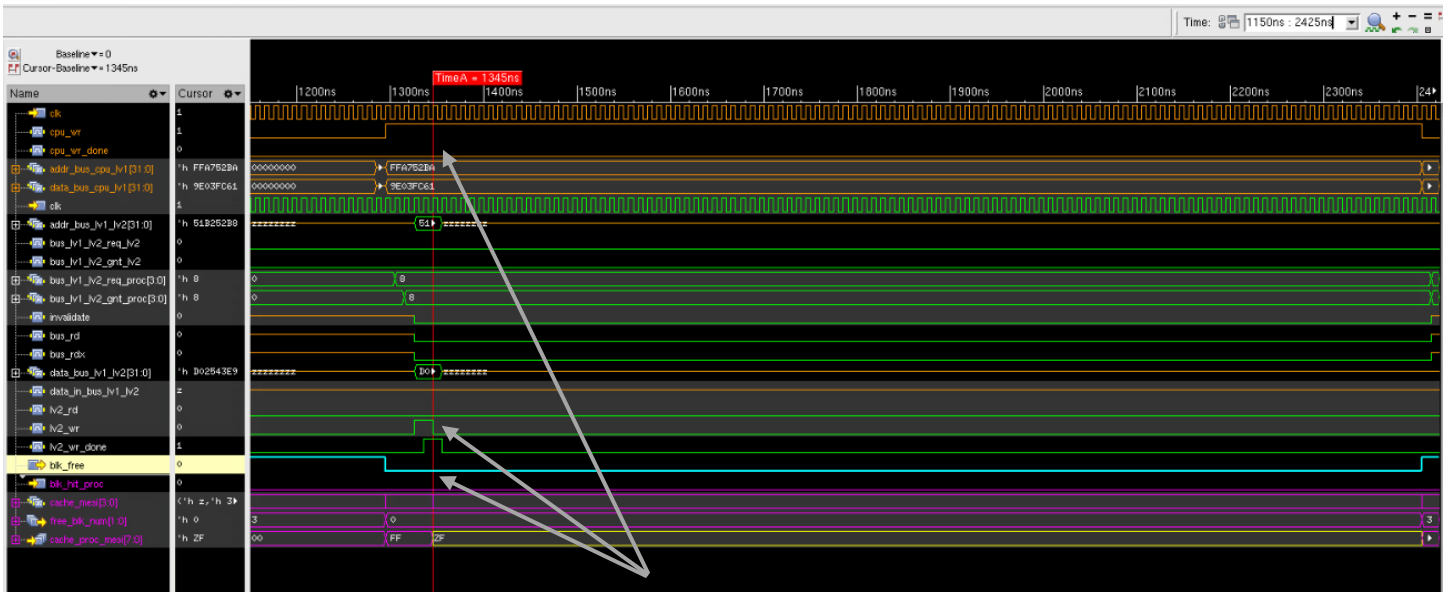
Full time frame :0:2425ns

Fig.2a Signals in orange, green, pink and blue represent signals of cpu[3], system bus interface, blk_free_md and unexpected signal behavior respectively



Time Frame1: 0:1150ns

Fig.2b Signals in orange, green, pink and blue represent signals of cpu[3], system bus interface, blk_free_md and unexpected signal behavior respectively



Time Frame2: 1150:2425ns

Fig.2c Signals in orange, green, pink and blue represent signals of cpu[3], system bus interface, blk_free_md and unexpected signal behavior respectively

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   57
UVM_WARNING :    0
UVM_ERROR :   12
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    17
[cpu_driver_c]    14
[cpu_lv1_interface]    12
[cpu_monitor_c]     4
[system_bus_monitor_c]    14
[write_miss_dcache_replace]     1
[write_miss_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 2425 NS + 51
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

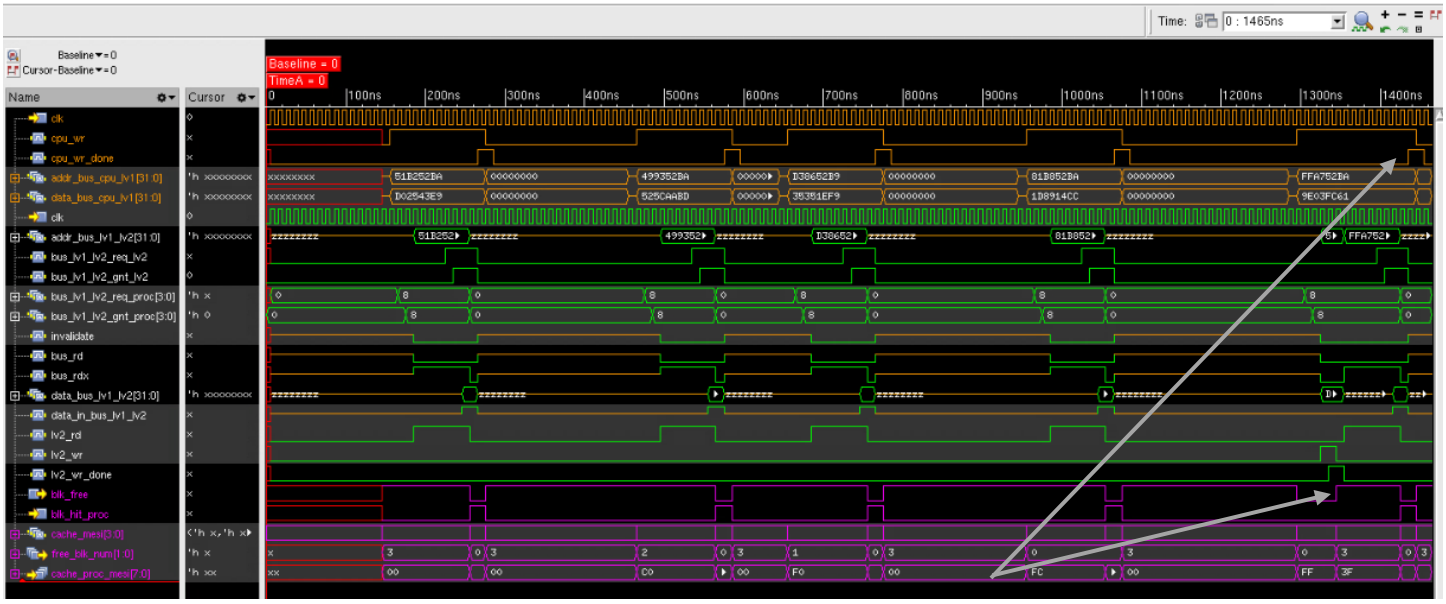
Fig.3 UVM Report Summary before bug fix

```
** Report counts by severity
UVM_INFO :   65
UVM_WARNING :    0
UVM_ERROR :    1
UVM_FATAL :    0
** Report counts by id
[MISCMP]     2
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    22
[cpu_driver_c]    14
[cpu_monitor_c]     4
[system_bus_monitor_c]    16
[write_miss_dcache_replace]     1
[write_miss_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 1465 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.4 UVM Report Summary after bug fix

```
UVM_INFO ../uvm/cache_scoreboard_c.sv(274) @ 1445: uvm_test_top.tb.sb [cache_scoreboard_c] CHECK_DATA CPU3
WRITE Request
UVM_INFO @ 1445: reporter [MISCMP] Miscompare for expected.proc_evict_dirty_blk_addr: lhs = 'h51b252b8 : rhs = 'h0
UVM_INFO @ 1445: reporter [MISCMP] 3 Miscompare(s) (1 shown) for object s_packet@7879 vs. expected@7956
UVM_ERROR ../uvm/cache_scoreboard_c.sv(433) @ 1445: uvm_test_top.tb.sb [cache_scoreboard_c] System bus activity MISMATCH!!!
UVM_INFO ../uvm/cache_scoreboard_c.sv(434) @ 1445: uvm_test_top.tb.sb [cache_scoreboard_c] Expected SBUS Packet
```

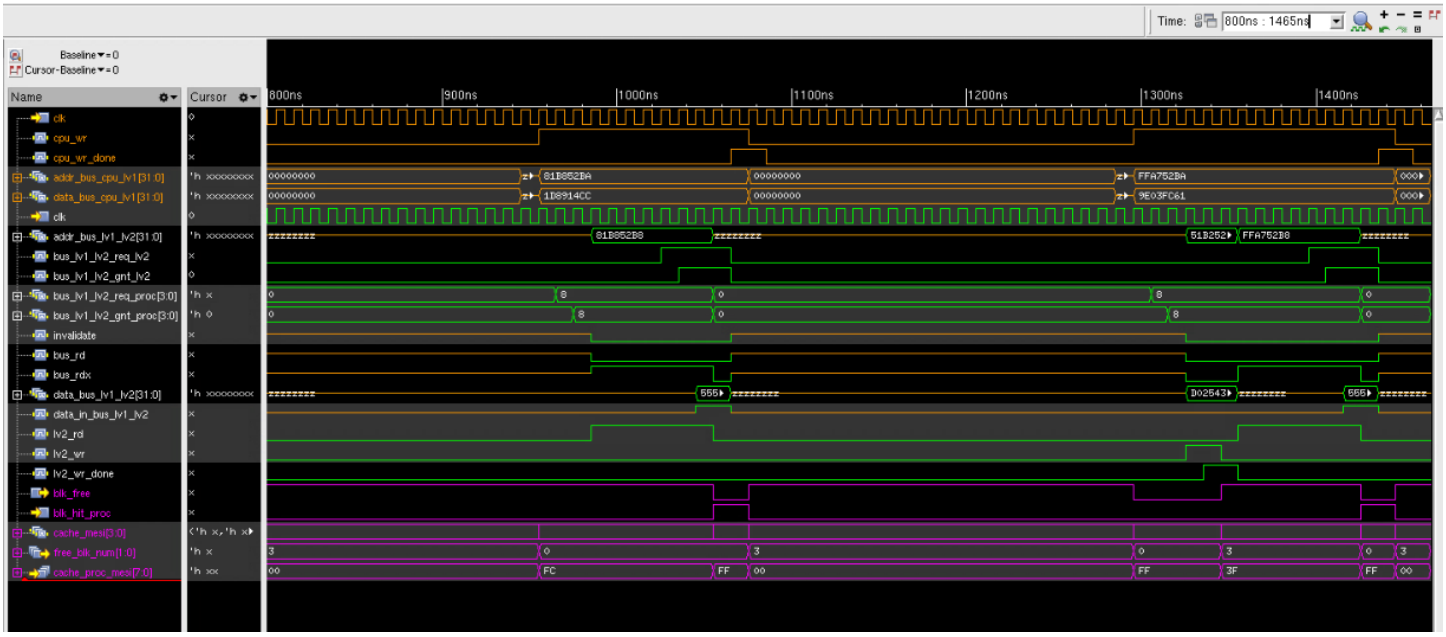
Fig.5 Error post bug fix



Time Frame: 0:1465ns

Fig.6a Waveform after bug fix

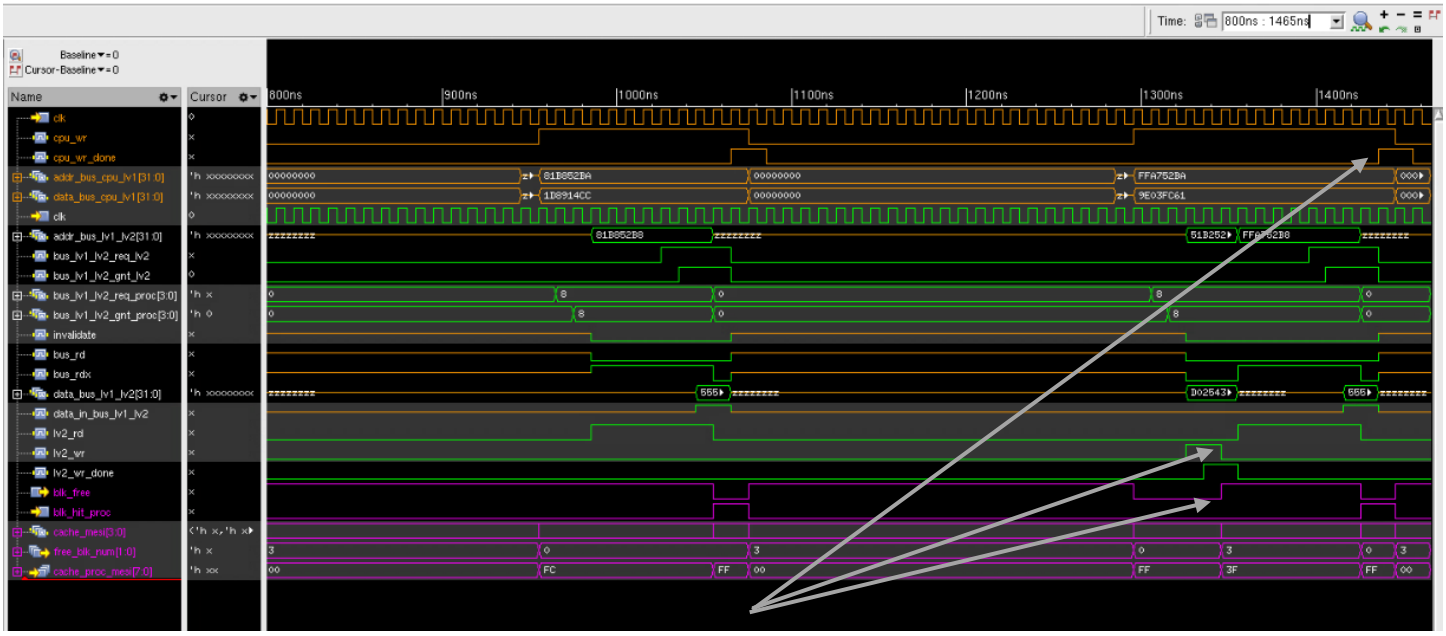
Signals in orange, green and pink represent signals of cpu[3], system bus intf and free_blk_md respectively



Time Frame1: 0: 800ns

Fig.6b Waveform after bug fix

Signals in orange, green and pink represent signals of cpu[3], system bus intf and free_blk_md respectively



Time Frame2: 800:1465ns

Fig.6c Waveform after bug fix

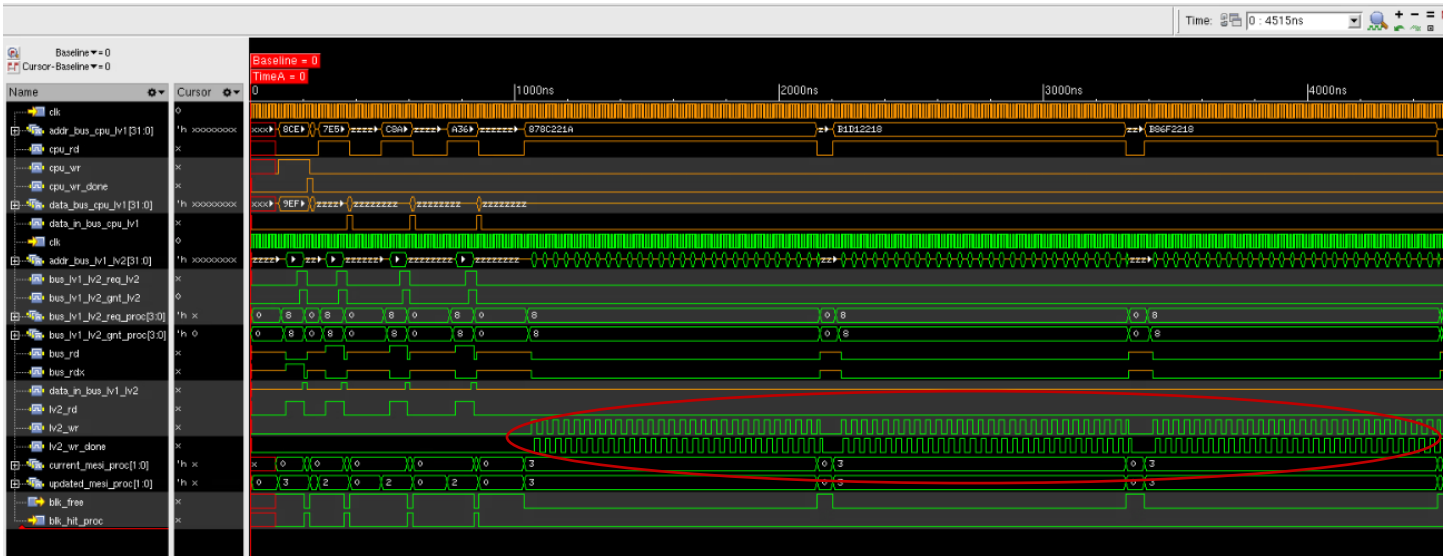
Signals in orange, green and pink represent signals of cpu[3], system bus intf and free_blk_md respectively

Team-16 Bug 8 Report

Team Number	**# 16**
Bug Number	# 8
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 194
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_read_dcache_replace: We have included this test case to verify the handling of read miss request on DCACHE when there is no free block available i.e., replacement needed. The process involves a write request followed by multiple read requests ultimately requiring the replacement of first write request. All these transactions occur on the same core.
Checker/Assertion Failed	Assertion deassert_data_in_bus_cpu_rd failed which is described as below in the cpu_lv1_interface: property prop_data_in_bus_cpu_rd; @(posedge clk) $rose(cpu_rd) \|-> ##[1:$]$fell(cpu_rd) |=> $fell(data_in_bus_cpu_lv1); endproperty deassert_data_in_bus_cpu_rd: assert property(prop_data_in_bus_cpu_rd) else `uvm_error("cpu_lv1_interface",$sformatf("Assertion-11 deassert_data_in_bus_cpu_rd Failed: data_in_bus_cpu_lv1 not deasserted in the next cycle after cpu_rd deasserted"))
Checker Description	The signal data_in_bus_cpu_lv1 should be deasserted one cycle after the signal cpu_rd is deasserted.
Bug Description/ Debug Process	After running the test, we observed that the assertion **deassert_data_in_bus_cpu_rd** is failing, as indicated in the log file (refer to fig.1). Upon analyzing the waveform in Simvision, we noticed that the signal **cpu_rd** and **data_in_bus_cpu_lv1** has unusual behavior. The read request for replacement is not happening as expected. It is observed that multiple **lv2_wr** are happening which should not be the case (as shown in fig.2a,2b,2c), which violates the specifications outlined in the HAS document (as shown on fig.6). We observed an incorrect logic for MESI state transition where the state should be assigned as INVALID instead of MODIFIED assignment under the Read Miss Replacement Modified logic within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.3 and fig.4a respectively. Unexpectedly, the test failed again but this time the error is not due to the assertion failure instead it an error due to system bus activity mismatch (as shown in fig.4b) which prompted us to investigate another potential bug (We will discuss that in Bug9 Report). However, post-bug fix, we observed the expected behavior of the signal **cpu_rd** and **data_in_bus_cpu_lv1** using Simvision waveform, as depicted in fig.5. and lv2_wr is not happening multiple times as the state is not in the MODIFIED loop.
Original Code	`CACHE_CURRENT_MESI_PROC <= MODIFIED;
Code after Fix	`CACHE_CURRENT_MESI_PROC <= INVALID;

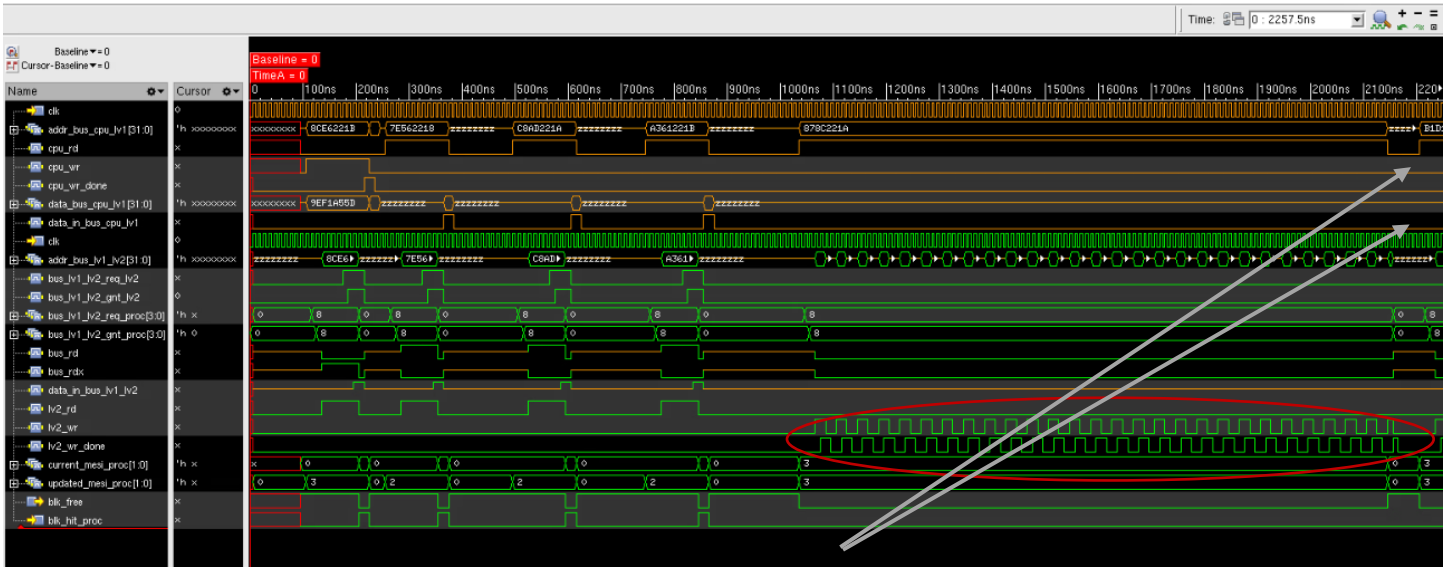
```
                    $rose(cpu_rd) |-> ##[1:$]$fell(cpu_rd) |=> $fell(data_in_bus_cpu_lv1);
                                                                |
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,131): (time 3335 NS) Assertion top.inst_cpu_lv1_if[3].deassert_data_in_bu
s_cpu_rd has failed (113 cycles, starting 2215 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(133) @ 3335: reporter [cpu_lv1_interface] Assertion-11 deassert_data_in_bus_cpu_r
d Failed: data_in_bus_cpu_lv1 not deasserted in the next cycle after cpu_rd deasserted
        assert_lv2_wr_after_lv2_wr_done: assert property (@(posedge clk) $rose(lv2_wr_done) |-> ##1 $fell(lv2_wr))
                                                                                               |
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,212): (time 3345 NS) Assertion top.inst_system_bus_if.assert_lv2_wr_af
ter_lv2_wr_done has failed (2 cycles, starting 3335 NS)
UVM_ERROR ../uvm/system_bus_interface.sv(214) @ 3345: reporter [system_bus_interface] Assertion-21 assert_data_address_
bus_after_lv2_wr_done Failed: lv2_wr did not deassert one cycle after lv2_wr assertion
UVM_INFO ../uvm/system_bus_monitor_c.sv(78) @ 3405: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation
 triggered
UVM_INFO ../uvm/cpu_driver_c.sv(83) @ 4515: uvm_test_top.tb.cpu[3].driver [cpu_driver_c] Ended Driving transaction
UVM_INFO ../uvm/virtual_seqs.sv(38) @ 4515: uvm_test_top.tb.vsequencer@@write_read_dcache_replace_seq [write_read_dcach
e_replace_seq] drop_objection
UVM_INFO /opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_objection.svh(1268) @ 4515: reporter
[TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_ERROR ../uvm/cache_scoreboard_c.sv(450) @ 4515: uvm_test_top.tb.sb [cache_scoreboard_c] Additional System Bus activ
ity is observed for CPU3
UVM_INFO ../uvm/cache_scoreboard_c.sv(459) @ 4515: uvm_test_top.tb.sb [cache_scoreboard_c] Received SBUS Packet 0:
```

Fig.1



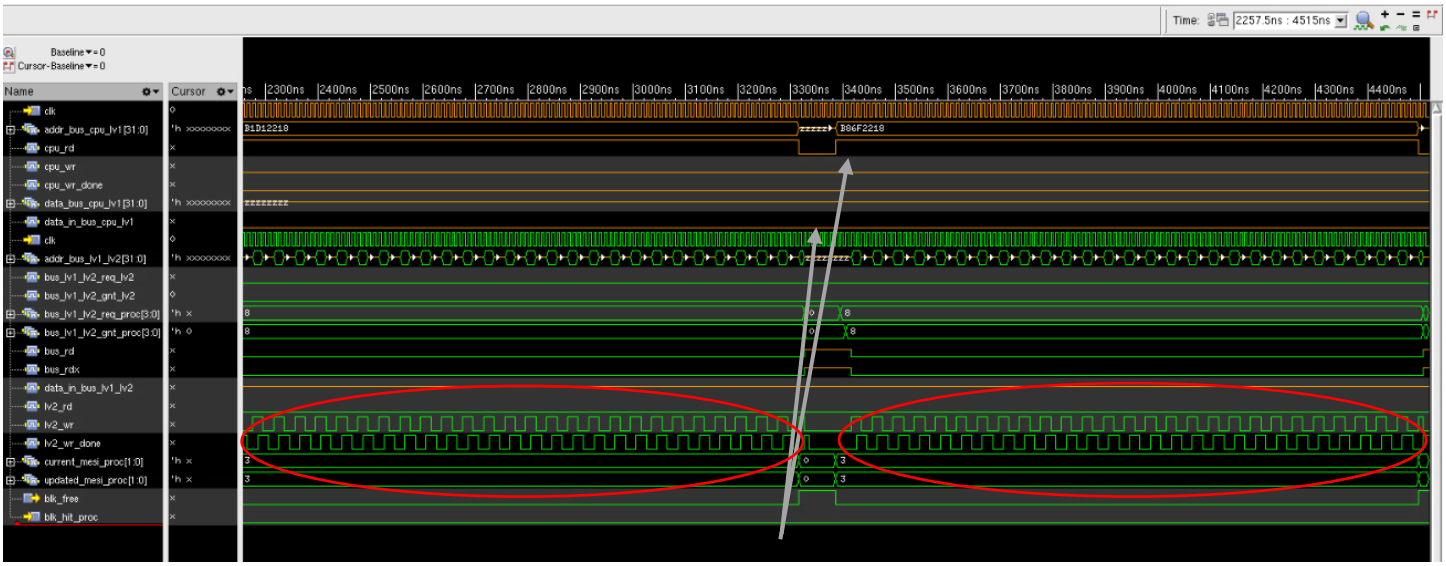
Full time frame :0:2425ns

Fig.2a Signals in orange, green represent signals of cpu[3], system bus interface respectively



Time Frame1: 0:2257.5ns

Fig.2b Signals in orange, green represent signals of cpu[3], system bus interface respectively



Time Frame2: 2257.5:4515ns

Fig.2c Signals in orange, green and blue represent signals of cpu[3], system bus interface and unexpected signal behavior respectively

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   67
UVM_WARNING :    0
UVM_ERROR :    5
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    22
[cpu_driver_c]    18
[cpu_lv1_interface]     2
[cpu_monitor_c]     4
[system_bus_interface]     2
[system_bus_monitor_c]    16
[write_read_dcache_replace]     1
[write_read_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 4515 NS + 52
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   89
UVM_WARNING :    0
UVM_ERROR :    1
UVM_FATAL :    0
** Report counts by id
[MISCMP]     2
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    36
[cpu_driver_c]    18
[cpu_monitor_c]     4
[system_bus_monitor_c]    22
[write_read_dcache_replace]     1
[write_read_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 1575 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.4a UVM Report Summary after bug fix

```
UVM_INFO @ 1185: reporter [MISCMP] Miscompare for expected.proc_evict_dirty_blk_data: lhs = 'h9ef1a55d : rhs = 'h5555aaaa
UVM_INFO @ 1185: reporter [MISCMP] 1 Miscompare(s) for object s_packet@7856 vs. expected@7963
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  -     @7963
  bus_req_type                bus_req_t      32    BUS_RD
  bus_req_proc_num            bus_req_proc_t 32    REQ_PROC3
  req_address                 integral       32    'h878c2218
  bus_req_snoop               integral       4     'h0
  req_serviced_by             serv_by_t      32    SERV_L2
  rd_data                     integral       32    'h5555aaaa
  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    'h8ce62218
  proc_evict_dirty_blk_data   integral       32    'h9ef1a55d
  proc_evict_dirty_blk_flag   integral       1     'h1
----------------------------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(435) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] Received SBUS Packet
----------------------------------------------------------------
Name                          Type           Size  Value
----------------------------------------------------------------
s_packet                      sbus_packet_c  -     @7856
  bus_req_type                bus_req_t      32    BUS_RD
  bus_req_proc_num            bus_req_proc_t 32    REQ_PROC3
  req_address                 integral       32    'h878c2218
  bus_req_snoop               integral       4     'h0
  req_serviced_by             serv_by_t      32    SERV_L2
  rd_data                     integral       32    'h5555aaaa
  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    'h8ce62218
  proc_evict_dirty_blk_data   integral       32    'h5555aaaa
  proc_evict_dirty_blk_flag   integral       1     'h1
```

Fig.4b System bus activity mismatch error

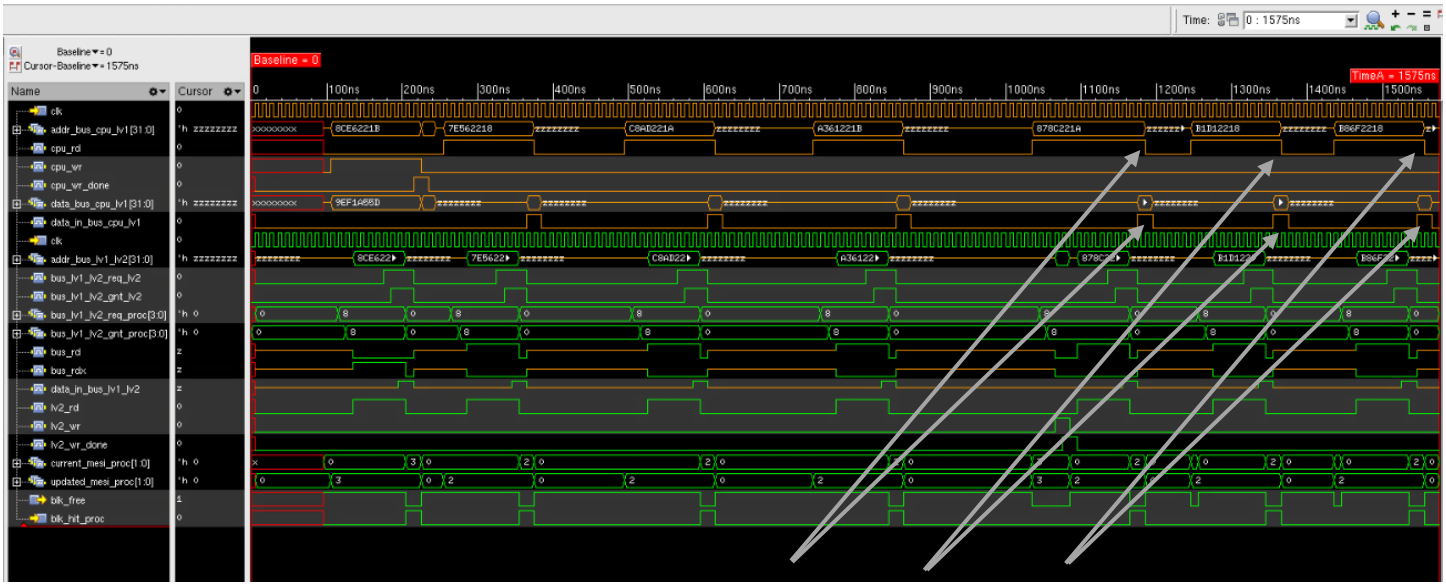


Fig.5 Waveform after bug fix

Signals in orange, green represent signals of cpu[3], system bus interface respectively

2.2 Processor Read Miss with no free block, replacement needed.

a) If the replacement to be done is Shared/Exclusive then the cache_proc [Index, blk_access_proc] [MESI_range] is made Invalid.

b) If the replacement is in Modified state then the following is carried out.

- bus_lv1_lv2_req_proc is already high (refer aforementioned steps)
- When bus_lv1_lv2_gnt_proc is made high, address of the replacement block is regenerated from its TAG (stored in cache_proc_contr) and Index_proc value and is put in addr_bus_lv1_lv2.
- data_bus_lv1_lv2 is loaded with the dirty data.
- lv2_wr is made high asking level 2 cache to update its value.
- Once lv2_wr_done is made high by level 2 cache, cache_proc [index_proc, blk_access_proc] [MESI_range] is made Invalid.

These set of operations will free that block and which in turn will trigger the free block operations explained above. Then once the above free block operation bring the data from level 2 cache or other level 1 caches it is seen as hit and Processor Read Hit operation is carried out.

Fig.6 HAS Document Statement supporting the bug.

Team-16 Bug 9 Report

Team Number	**# 16**
Bug Number	# 9
Bug Location	design/lv1/ main_func_lv1_dl – Line No. 192
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss_dcache_replace: We have included this test case to verify the handling of write miss request on DCACHE when there is no free block available i.e., replacement needed. The process involves initially writing different data into the available free blocks of the cache. Subsequently, another write request is made to a different address, ultimately requiring the replacement of one of the existing blocks. All these transactions occur on the same core. write_read_dcache_replace: We have included this test case to verify the handling of read miss request on DCACHE when there is no free block available i.e., replacement needed. The process involves a write request followed by multiple read requests ultimately requiring the replacement of first write request. All these transactions occur on the same core.
Checker/Assertion Failed	expected = expected_sbus[i].pop_front(); received = received_sbus[i].pop_front(); if(expected.compare(received)) begin `uvm_info(get_type_name(), $sformatf("System bus activity matched for this request"),UVM_LOW) `uvm_info(get_type_name(), $sformatf("Expected & Received SBUS Packet \n%s", received.sprint()),UVM_LOW) end else begin `uvm_error(get_type_name(), $sformatf("System bus activity MISMATCH!!!")) `uvm_info(get_type_name(), $sformatf("Expected SBUS Packet \n%s", expected.sprint()),UVM_LOW) `uvm_info(get_type_name(), $sformatf("Received SBUS Packet \n%s", received.sprint()),UVM_LOW) end
Checker Description	This essentially verifies whether the packet details received on the system bus match the expected values or not.
Bug Description/ Debug Process	After fixing bug7 and bug8 and running the test, we observed that the test is failing again, as indicated in the log file (refer to fig.1a). There is a mismatch in the data of evicted block as shown in fig.1b. Upon analyzing the waveform in Simvision, we noticed that during replacement, data for dirty blk is incorrect as shown in fig.1b, fig2. During 4th read since there is no free block available the first read request block has to be evicted and written to lv2. During **lv2_wr** we observed that **data_bus_lv1_lv2** is loaded with incorrect data as **5555aaaa** instead of **9ef1a55d** ,which violates the specifications outlined in the HAS document(as shown on fig.6). The evicted block has incorrect data as 5555aaaa instead of 9ef1a55d. This is happening because an incorrect logic for **data_bus_lv1_lv2_reg** under the Write Miss Replacement logic within the file main_func_lv1_dl under design/lv1/ folder. UVM Summary report before and after the fix is shown in fig.3 and fig.4 respectively. After fixing the bug we observed that there is no system bus activity mismatch (implied by the UVM Summary report in fig.4). Further analysis using Simvision waveform shows the same evicted block has correct data as shown below in fig.5
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}];

```
UVM_INFO ../uvm/cache_scoreboard_c.sv(293) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] CHECK_DATA CPU3
UVM_INFO ../uvm/cache_scoreboard_c.sv(295) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] Data match!!! expected = 5555aaaa received = 5555aaaa
UVM_INFO @ 1185: reporter [MISCMP] Miscompare for expected.proc_evict_dirty_blk_data: lhs = 'h9ef1a55d : rhs = 'h5555aaaa
UVM_INFO @ 1185: reporter [MISCMP] 1 Miscompare(s) for object s_packet@7856 vs. expected@7963
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
----------------------------------------------------------
```

Fig.1a

```
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   -     @7963
  bus_req_type                bus_req_t       32    BUS_RD
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC3
  req_address                 integral        32    'h878c2218
  bus_req_snoop               integral        4     'h0
  req_serviced_by             serv_by_t       32    SERV_L2
  rd_data                     integral        32    'h5555aaaa
  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    'h8ce62218
  proc_evict_dirty_blk_data   integral        32    'h9ef1a55d
  proc_evict_dirty_blk_flag   integral        1     'h1
----------------------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(435) @ 1185: uvm_test_top.tb.sb [cache_scoreboard_c] Received SBUS Packet
----------------------------------------------------------
Name                          Type            Size  Value
----------------------------------------------------------
s_packet                      sbus_packet_c   -     @7856
  bus_req_type                bus_req_t       32    BUS_RD
  bus_req_proc_num            bus_req_proc_t  32    REQ_PROC3
  req_address                 integral        32    'h878c2218
  bus_req_snoop               integral        4     'h0
  req_serviced_by             serv_by_t       32    SERV_L2
  rd_data                     integral        32    'h5555aaaa
  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    'h8ce62218
  proc_evict_dirty_blk_data   integral        32    'h5555aaaa
  proc_evict_dirty_blk_flag   integral        1     'h1
----------------------------------------------------------
```

Fig.1b

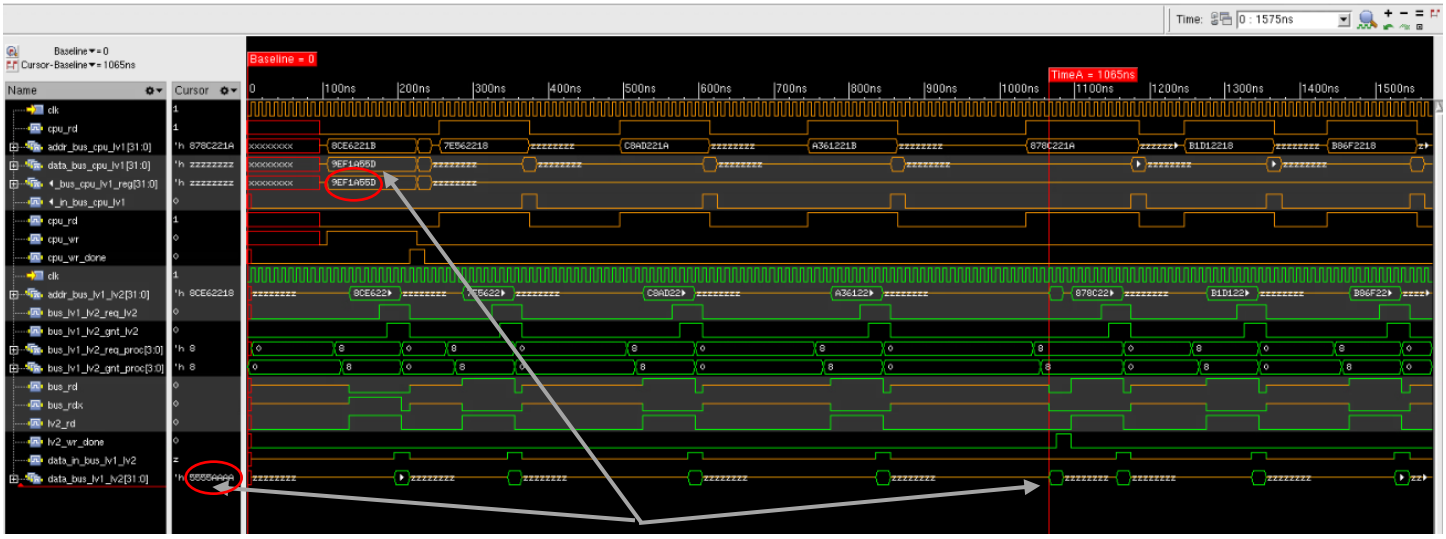


Fig.2 Signals in orange, green represent signals of cpu[3] and system bus interface respectively

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   65
UVM_WARNING :    0
UVM_ERROR :    1
UVM_FATAL :    0
** Report counts by id
[MISCMP]     2
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    22
[cpu_driver_c]    14
[cpu_monitor_c]     4
[system_bus_monitor_c]    16
[write_miss_dcache_replace]     1
[write_miss_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 1465 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.3 UVM Report Summary before bug fix

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   87
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[TEST_DONE]     1
[TOP]     1
[UVMTOP]     1
[cache_scoreboard_c]    35
[cpu_driver_c]    18
[cpu_monitor_c]     4
[system_bus_monitor_c]    22
[write_read_dcache_replace]     1
[write_read_dcache_replace_seq]     3
Simulation complete via $finish(1) at time 1575 NS + 50
/opt/coe/cadence/XCELIUM/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
xcelium> exit
```

Fig.4 UVM Report Summary after bug fix

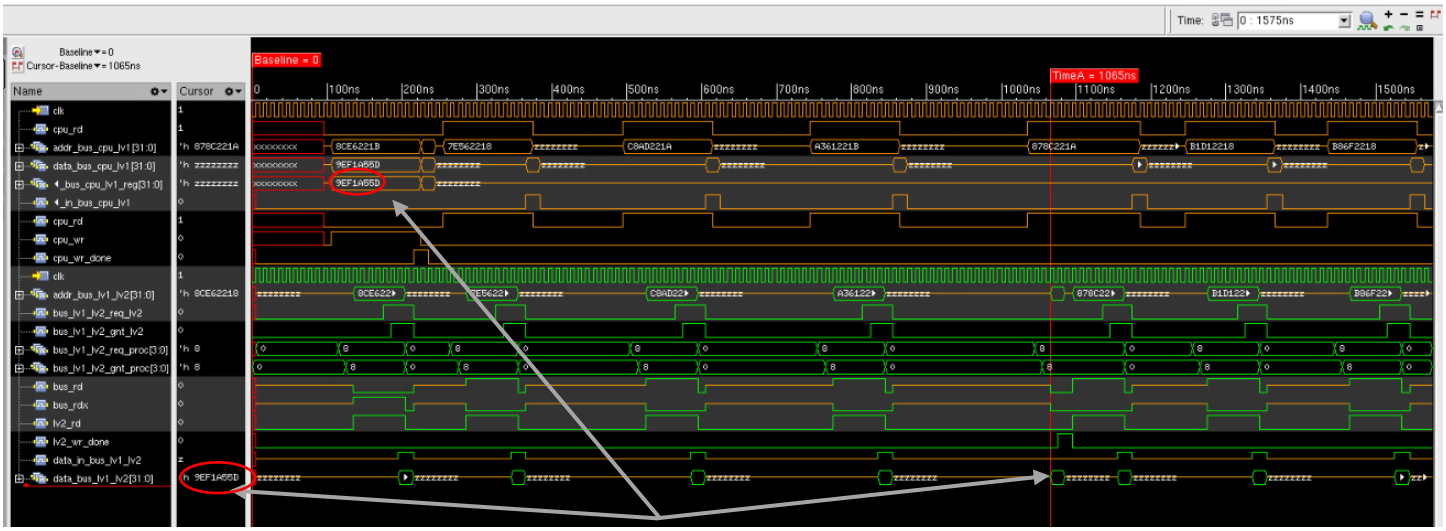


Fig.5 Waveform after bug fix

Signals in orange and green represent signals of cpu[3] and system bus interface after fix respectively

(2) Free block not available, replacement needed.

If the replacement to be done is Shared/Exclusive then the cache_proc [Index_proc, blk_access_proc][MESI_range] is made Invalid.

If the replacement is in Modified state then the following is carried out (as dirty replacement needs to be updated in level 2 cache).

- When bus_lv1_lv2_gnt_proc is made high, address of the replacement block is regenerated from its TAG (stored in cache_proc_contr) and Index_proc value and is put in addr_bus_lv1_lv2.
- data_bus_lv1_lv2 is loaded with the dirty data.
- lv2_wr is made high asking level 2 cache to update its value.
- Once lv2_wr_done is made high by level 2 cache, cache_proc [index_proc, blk_access_proc] [MESI_range] is made Invalid.

Fig.6 HAS Document Statement supporting the bug.