Team 1

Bug Report

Bug 1

Test Case: Read processor 0, Write processor 1, Read processor 1. (All at the same address)

Assertion failed: @posedge(clk) $fell (cpu_wr) |=> $fell (cpu_wr_done),

Explanation: When we encountered the above assertion failure, we looked into the waveforms and observed the same. i.e. when cpu_wr falls, cpu_wr_done does not fall. In fact, cpu_wr_done was never asserted. Since CPU_wr_done was never asserted, from the HAS pdf, we can confirm that cache_var and cache_proc_contr are not updated with the values that were passed by write proc1.
In order to find out why these two variables are not updated, we looked at the waveforms of address_bus_cpu_lv1[processor 1] and data_bus_cpu_lv1[processor 1]. We figured out that address_bus_cpu_lv1 is only 15 bits which ideally should be 32 bits.

Since the address bus is of only 15 bits, passing an address of 32h AABB_CCDD, the address gets truncated to 4CDD. This truncated address is less than the d_cache address and thus write operation is not permitted.

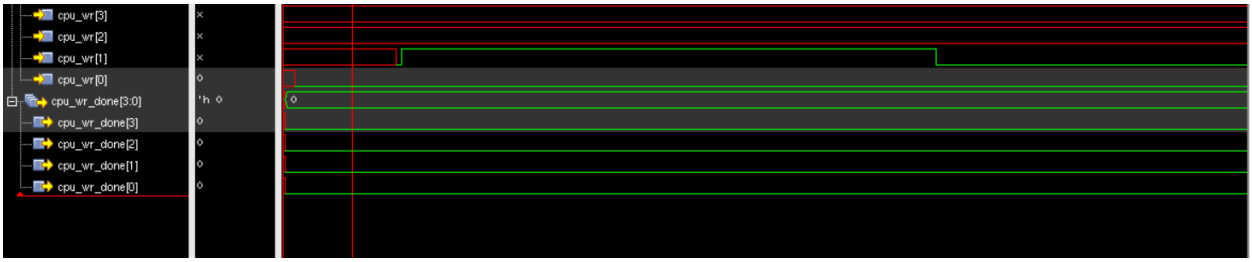
Solution: In cache_lv1_multicore line 140 (.ADDR_WID(INDEX_MSB),) should be replaced by .ADDR_WID(ADDR_WID).

Please find the waveforms as attached and let us know your inputs.

```
                    $fell(cpu_wr)|=>$fell(cpu_wr_done);
                                  |
ncsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,103): (time 1375 NS) Assertion top.inst_cpu_lv1_if[1].assert_chceking_cpu_wr_follow_cpu_wr_done has failed (2 cycles,
 starting 1365 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(105) @ 1375: reporter [cpu_lv1_interface] Assertion 7 checking_sequence_type_3 Failed: cpu_wr and cpu_wr_done don't follow ea
ch other
UVM_INFO ../uvm/cpu_driver_c.sv(83) @ 2555: uvm_test_top.tb.cpu[1].driver [cpu_driver_c] Ended Driving transaction
UVM_INFO ../uvm/virtual_seqs.sv(38) @ 2555: uvm_test_top.tb.vsequencer@@r0_w1_r1_same_addr_dcache_seq [r0_w1_r1_same_addr_dcache_seq] drop_objection
UVM_INFO /softwares/Linux/cadence/INCISIVE152/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_objection.svh(1268) @ 2555: reporter [TEST_DONE] 'run' phase is ready
 to proceed to the 'extract' phase
---Test Summary---

---Final Test Status---

Test FAIL
ncsim: *E,ERRSEV (../test/base_test.sv,65): (time 2555 NS).
top.base_test.report_phase
```



Bug 2

Test Case: Four writes to same processor i.e. processor 0 with different addresses (having different tags but same index number) followed by two reads (again having different tags but same index number)

Test Intent: To check block replacement policy (LRU) working

UVM_ERROR ../uvm/cache_scoreboard_c.sv(414) @
1725: uvm_test_top.tb.sb [cache_scoreboard_c] Expected activity is not observed on the system bus

```
UVM_ERROR ../uvm/cache_scoreboard_c.sv(433) @ 1675: uvm_test_top.tb.sb [cache_scoreboard_c] System bus activity MISMATCH!!!
UVM_INFO ../uvm/cache_scoreboard_c.sv(434) @ 1675: uvm_test_top.tb.sb [cache_scoreboard_c] Expected SBUS Packet
----------------------------------------------------------
Name                        Type            Size  Value
----------------------------------------------------------
expected                    sbus_packet_c   -     @7826
  bus_req_type              bus_req_t       32    BUS_RD
  bus_req_proc_num          bus_req_proc_t  32    REQ_PROC0
  req_address               integral        32    'h70000000
  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    'h55550000
  proc_evict_dirty_blk_data integral        32    'h5011959
  proc_evict_dirty_blk_flag integral        1     'h1
----------------------------------------------------------

UVM_INFO ../uvm/cache_scoreboard_c.sv(435) @ 1675: uvm_test_top.tb.sb [cache_scoreboard_c] Received SBUS Packet
----------------------------------------------------------
Name                        Type            Size  Value
----------------------------------------------------------
s_packet                    sbus_packet_c   -     @7952
  bus_req_type              bus_req_t       32    BUS_RD
  bus_req_proc_num          bus_req_proc_t  32    REQ_PROC0
  req_address               integral        32    'h70000000
  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    'h44440000
  proc_evict_dirty_blk_data integral        32    'h11111111
  proc_evict_dirty_blk_flag integral        1     'h1
----------------------------------------------------------
```

Observation:

Initially, as the cache was empty, the first four write instructions caused write miss and the data was written in the cache and all of them were in a modified state. When the fifth instruction, which was a read instruction belonging to the same set (as all had the same index) was executed, the first block of the cache was replaced and it matched with the expected result. However, when the sixth instruction which was also read having addresses belonging to the same set was executed, the expected block that must have been replaced was 3rd block (i.e. block 2). But as seen from the above image the expected and received did not match. In our test case, the received packet from the monitor showed that the 2nd block (i.e. block 3). This clearly indicated that there was some error in the flow of blocks getting replaced.

Solution: line 38 and 39 of file lru_block_lv1.sv in design/lv1 folder. The values of block to be replaced where exchanged for BLK_1 and BLK_2 replacement. The correct value is as follows.

BLK1_REPLACEMENT: lru_replacement_proc = 2'b01;

BLK2_REPLACEMENT: lru_replacement_proc = 2'b10;

Bug 3

Test Case: Write Processor 3 , Read Processor 0, Write Processor 3 , Read Processor 0

Test Intent: To verify MESI state by targeting same address in reads and writes

UVM_ERROR:/uvm/cache_scoreboard_c.sv(297) @
735: uvm_test_top.tb.sb [cache_scoreboard_c] Data MISMATCH!!! **expected = 29292929 received = 28282828**

Observation

Data when read from processor0 reads correctly the first time but when the second write is made at the same address, the processor0 reads stale and old value causing the DATA Mismatch.That is because the invalidation to the processor 0 did not happen and it keeps the old data . Hence we see the received data as 28282828 which ideally should be the updated value i.e. 29292929.

Since we saw that the received data in processor 0 is the same as the old data that has been received, we assumed that there has been a read hit in processor 0. To verify it, we checked at the invalidation signal that should have been high since we had a write to processor 3 in the third step. But the Invalidation signal was not set high. Since invalidation was not set high, we looked into the MESI states of processors.

When the processor 3 is writing data in step 3, the previous MESI states of processor 0 and processor 3 are both shared. As we are modifying the data in Processor 3 (shared to modified), the processor 0 should switch from shared to invalid. But as it is evident from the waveform that, processor 0 state is not being changed to Invalid(invalidation signal not high). Hence when we give the read command for the same address to processor 0 in step 4, as the current state of processor 0 is still shared, a read hit occurs and an old value is being displayed.

This confirmed that the mesi state of the processors are not getting updated correctly.

Solution: In main_func_lv1_dl.sv the SHARED value should be set 01 and EXCLUSIVE value should be set 10 so as to align it with the values set in cache_block_lv1_dl.sv file.

Waveform: