Bug Report

CSCE 714: Advanced Hardware Design Functional Verification

Multicore MESI Based Cache Design HAS

Team 17: Hunter Britton, Christopher Muniz, Jaewoo Lee

Team Number	# 17
Bug Number	# 1
Bug Location	Main_func_lv1_dl.sv line: 176
Bug Type	Functional Bug
SV Test Run to uncover the Bug	Read_hit_dcache
Checker/Assertion Failed	assert_release_after_gnt
Checker Description	If the bus_rd or bus_rdx is asserted, lv_2 has to be asserted too
Bug Description/ Debug Process	Send a read request into the dcache on each processor. Since the read request hits multiple processors and the code does not return the value down to zero the rd_register is never freed. Read_hit_dcache ensures that there is a free block and that the read needs to be in the second cache, by reading to the dcache. Starting at the beginning of the readmiss and reading down to the free block if statement, the bus_rd_reg was not deasserted after the data was put into the bus for lv1/lv2.
Original Code	Line 176: bus_rd_reg <= 1'b0;
Code after Fix	bus_rd_reg <= 1'b1;

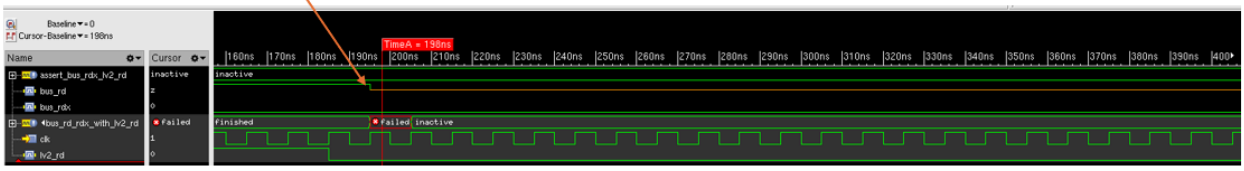
```
//ASSERTION 21: If bus_rd or bus_rdx asserted, then lv2_rd also asserted.
//property that checks if signal_1 is asserted when signal_2 is asserted
    property prop_sig2_with_sig1(signal_1, signal_2);
    @(posedge clk)
        signal_2 |-> signal_1;
    endproperty

    assert_bus_rd_rdx_with_lv2_rd: assert property (prop_sig2_with_sig1(lv2_rd,(bus_rd||bus_rdx)))
    else
        `uvm_error("system_bus_interface",$sformatf("Assertion assert_bus_rd_rdx_with_lv2_rd Failed: bus_rd/bus_rdx asserted without a lv2_rd"))
```

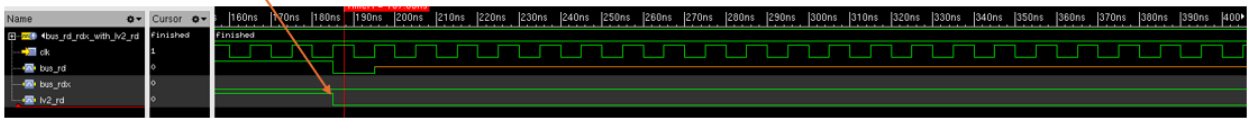
2.1 Free block available in the set

- 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

As lv2_rd gets deasserted, bus_rd goes to high impedance and throws an error, leading to a state of unsure where the lv2_rd should be.



Lv2_rd gets deasserted, and bus_rd gets deasserted



Team Number	# 17
Bug Number	# 2
Bug Location	Main_func_lv1_dl.sv @ 221
Bug Type	Functional Bug
SV Test Run to uncover the Bug	read_two_procs_force_write.sv
Checker/Assertion Failed	assert_wr_completion
Checker Description	Checks to see if wr_done is asserted within 100 cycles of cpu_wr being asserted to showcase that the write is done.
Bug Description/ Debug Process	When forcing two processors to read the same address to push the state into shared, and then force a write on the same address, the test was written to force an invalidate on the shared state, the state should move to INVALID and invalidate the write leading to a cpu_wr_done being scheduled.
Original Code	invalidate_reg <= 1'bz;
Code after Fix	invalidate_reg <= 1'b1;

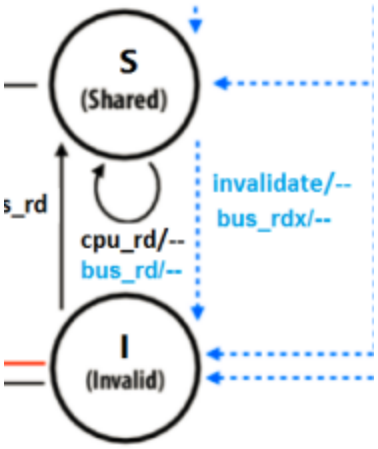


Figure 2-1

```
// ASSERTION9: write acknlwedgement time out
    property prop_wr_completion;
        @(posedge clk)
        (cpu_wr) |-> ##[0:100] (cpu_wr_done);
    endproperty

    assert_wr_completion: assert property (prop_wr_completion)
    else
        `uvm_error("cpu_lv1_interface", $sformatf("Assertion assert_wr_completion Failed: Write operation not completed by LV1 cache"));
```

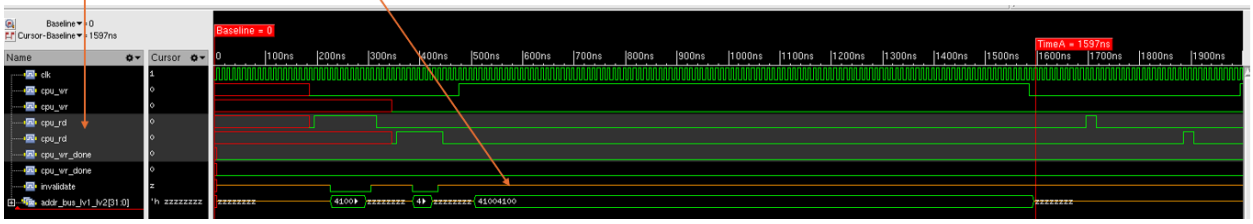
```
xmsim: *E,ASRTST (../uvm/cpu_lv1_interface.sv,120): (time 77315 NS) Assertion top.inst_cpu_lv1_if[0].assert_wr_completion has failed (101 cycles, starting 76315 NS)
UVM_ERROR ../uvm/cpu_lv1_interface.sv(122) @ 77315: reporter [cpu_lv1_interface] Assertion assert_wr_completion Failed: Write operation not completed by LV1 cache
        (cpu_wr) |-> ##[0:100] (cpu_wr_done);
```

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

Cpu0 wr, cpu3 wr, cpu0_rd, cpu3_rd, cpu0_wr_done, cpu3_wr_done, invalidate

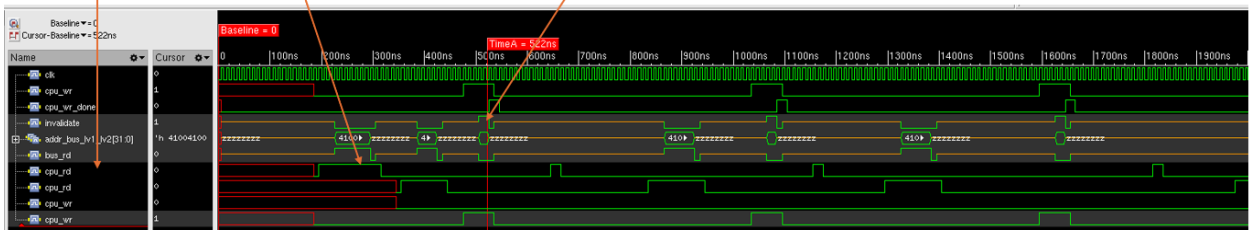
Invalidate goes to high impedance rather than high. Invalidate should go high to signal that cpu_wr_done needs to be asserted too



Cpu0 rd, cpu3 rd, cpu_wr0, cpu_wr3

Read, read on bus_rd and then cpu_wri

Invalidate goes to high, which then causes cpu_wr_done to be asserted.



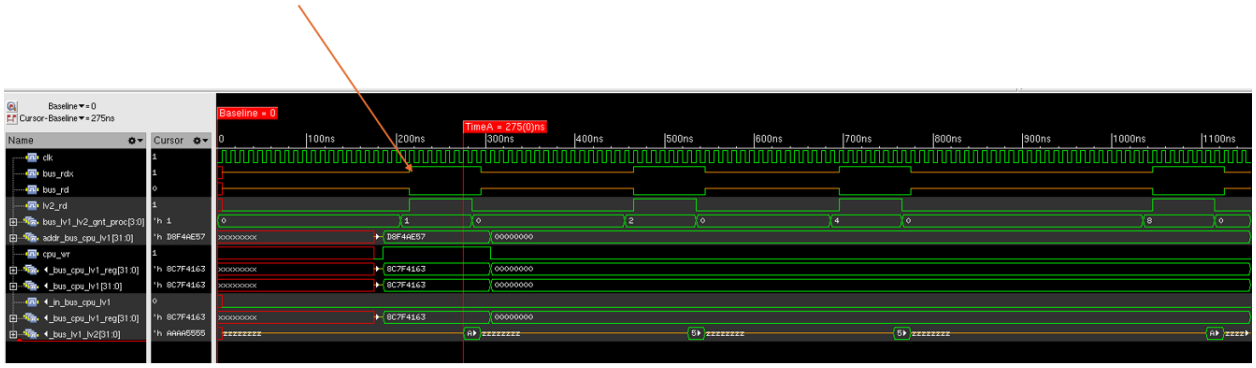
Team Number	# 17
Bug Number	# 3
Bug Location	Main_func_lv1_dl.sv @ 259
Bug Type	Functional Bug
SV Test Run to uncover the Bug	write_miss.sv
Checker/Assertion Failed	Assertion assert_bus_rd_rdx_with_lv2_rd Failed: bus_rd/bus_rdx asserted without a lv2_rd
Checker Description	Checks to see if bus_rd or bus_rdx is asserted without lv2_rd
Bug Description/ Debug Process	Writing randomly to an address in the d_cache to verify that signals are properly deasserted, and asserted When accessing lv2 cache, at the end of the write all signals need to be deasserted. The assertion failed, and by looking at the waveform bus_rdx_reg is not deasserted properly, leading to the assertion hitting, rather it is asserted high.
Original Code	bus_rdx_reg <= 1'b1;
Code after Fix	bus_rdx_reg <= 1'b0;

```
UVM_INFO ../uvm/system_bus_monitor_c.sv(249) @ 65: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet creation triggered
UVM_INFO ../uvm/system_bus_monitor_c.sv(404) @ 135: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] BUS_RDX successful
UVM_INFO ../uvm/system_bus_monitor_c.sv(437) @ 145: uvm_test_top.tb.sbus_monitor [system_bus_monitor_c] Packet to be written
xmsim: *E,ASRTST (../uvm/system_bus_interface.sv,359): (time 155 NS) Assertion top.inst_system_bus_if.assert_bus_rd_rdx_with_lv2_rd has failed
UVM_ERROR ../uvm/system_bus_interface.sv(361) @ 155: reporter [system_bus_interface] Assertion assert_bus_rd_rdx_with_lv2_rd Failed: bus_rd/bus_rdx asserted without a lv2_rd
UVM_INFO ../uvm/cache_scoreboard_c.sv(523) @ 165: uvm_test_top.tb.sb [cache_scoreboard_c] cpu_mon_packet from CPU1:
----------------------------------------------------
Name              Type               Size  Value
----------------------------------------------------
packet            cpu_mon_packet_c   -     @7964
  dat             integral           32    'h56e2143a
  address         integral           32    'he39f1a46
  num_cycles      integral           32    'h0
  illegal         integral           1     'h0
  request_type    request_t          1     WRITE_REQ
  addr_type       addr_t             32    DCACHE
----------------------------------------------------
```

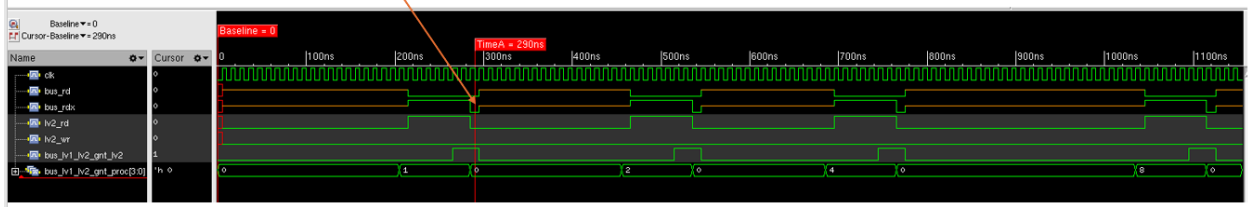
```
//ASSERTION 21: If bus_rd or bus_rdx asserted, then lv2_rd also asserted.
//property that checks if signal_1 is asserted when signal_2 is asserted
    property prop_sig2_with_sig1(signal_1, signal_2);
    @(posedge clk)
        signal_2 |-> signal_1;
    endproperty

    assert_bus_rd_rdx_with_lv2_rd: assert property (prop_sig2_with_sig1(lv2_rd,(bus_rd||bus_rdx)))
    else
        `uvm_error("system_bus_interface",$sformatf("Assertion assert_bus_rd_rdx_with_lv2_rd Failed: bus_rd/bus_rdx asserted without a lv2_rd"))
```

Bus_rdx gets asserted when bus_rd is asserted low



Bus_rdx gets asserted low when bus_rd is asserted low



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.

After this operation, the block will automatically become block hit and previously mentioned Processor Write Hit operation is carried out.

```
---Final Test Status---

Test PASS
 ____   _    ____ ____
|  _ \ / \  / ___/ ___|
| |_) / _ \ \___ \___ \
|  __/ ___ \ ___) |__) |
|_| /_/   \_\____/____/


--- UVM Report catcher Summary ---


Number of demoted UVM_FATAL reports  :    0
Number of demoted UVM_ERROR reports  :    0
Number of demoted UVM_WARNING reports:    0
Number of caught UVM_FATAL reports   :    0
Number of caught UVM_ERROR reports   :    0
Number of caught UVM_WARNING reports :    0

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   54
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
```