Bug-9 Report
Group-8

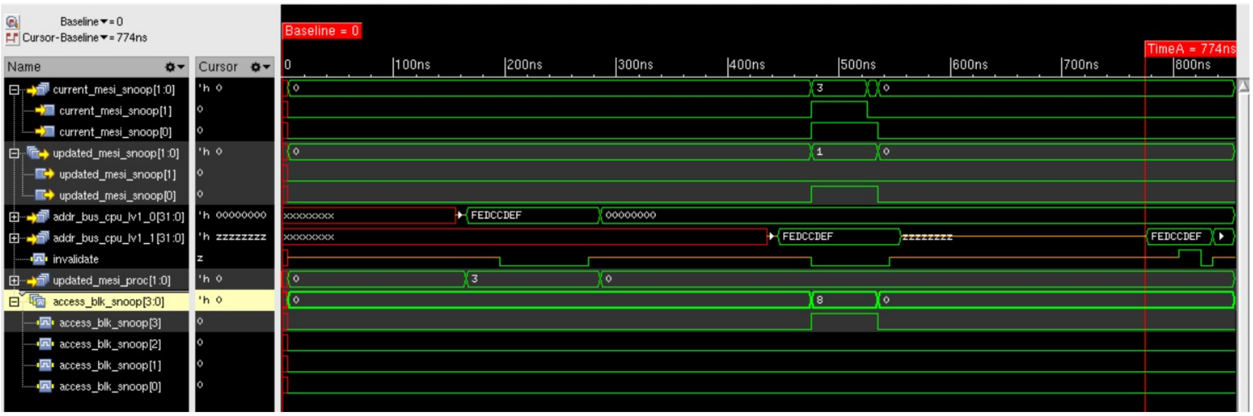
1. **Failing Test Name:** MESI Testing for the data cache, MSI sequence testing

2. **Test Description(Describe the planned scenario and the expected result):**

Testing the MESI protocol by simulating multiple scenarios such as Modified to Shared, Exclusive to Invalid sequence etc. In the MSI test, we write to an address through 1st core, causing write miss, then read that address through 2nd core, causing read miss and modify the data through the 2nd core, causing read hit, to test if the data becomes invalid in 1st core.

3. **Failing assertion that helped you identify the bug**

In the scoreboard we noticed that there was a system bus mismatch while expected data was received. We looked into the waveforms and looked into the MESI states of all processors involved in the test case. We noticed that the current_mesi_snoop signal never transitions to show shared state even though the updated_mesi_snoop signal is changed. We looked at other signals related to MESI transactions. When looking at the access_blk_snoop, we noticed the signal was zero when it was supposed to be high. According to the HAS document, the access_blk_snoop signal is supposed to go high whenever the data in the cache is not in an invalid state and the cache tag matches the tag address being snooped. In this situation, the test case matched the requirements but the signal was still low.


Baseline = 0
Cursor-Baseline = 774ns

Name	Cursor
current_mesi_snoop[1:0]	'h 0
current_mesi_snoop[1]	0
current_mesi_snoop[0]	0
updated_mesi_snoop[1:0]	'h 0
updated_mesi_snoop[1]	0
updated_mesi_snoop[0]	0
addr_bus_cpu_lv1_0[31:0]	'h 00000000
addr_bus_cpu_lv1_1[31:0]	'h zzzzzzzz
invalidate	z
updated_mesi_proc[1:0]	'h 0
access_blk_snoop[3:0]	'h 0
access_blk_snoop[3]	0
access_blk_snoop[2]	0
access_blk_snoop[1]	0
access_blk_snoop[0]	0

4. **Debugging:**

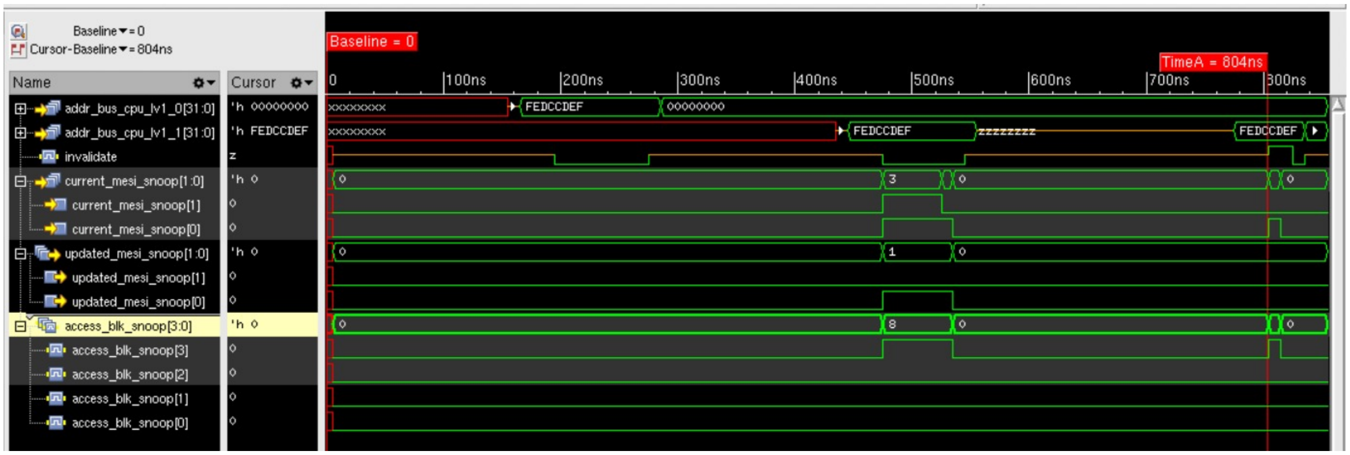
The access_blk_snoop signal is controlled by the access_blk_snoop_md file which ensures that the signal goes high when above conditions are met. This is triggered by signals such as bus read and bus_readx in the original file. However, according to the HAS document, the Invalidate signal is also responsible for triggering the snoop mechanism in the design.This meant that the snoop mechanism was not getting triggered by Invalidate. Hence, we added the Invalidate signal in the sensitivity list of the if loop in access_blk_snoop_md to ensure access_blk_snoop is high every time the Invalidate signal goes high and the conditions are met.

5. **Erroneous RTL file name –** access_blk_snoop_md.sv

6. **Lines of RTL file responsible for the bug:** if(bus_rd || bus_rdx) begin

7. **Corrected RTL code:** if(bus_rd || bus_rdx || invalidate) begin

8. **Corrected Wave Form:**


Baseline = 0
Cursor-Baseline = 804ns
Baseline = 0
TimeA = 804ns
Name
Cursor
0
100ns
200ns
300ns
400ns
500ns
600ns
700ns
800ns
addr_bus_cpu_lv1_0[31:0]
'h 00000000
addr_bus_cpu_lv1_1[31:0]
'h FEDCCDEF
invalidate
z
current_mesi_snoop[1:0]
'h 0
current_mesi_snoop[1]
0
current_mesi_snoop[0]
0
updated_mesi_snoop[1:0]
'h 0
updated_mesi_snoop[1]
0
updated_mesi_snoop[0]
0
access_blk_snoop[3:0]
'h 0
access_blk_snoop[3]
0
access_blk_snoop[2]
0
access_blk_snoop[1]
0
access_blk_snoop[0]
0
xxxxxxxx
FEDCCDEF
00000000
xxxxxxxx
FEDCCDEF
zzzzzzzz
FEDCCDEF
0
3
0
0
1
0
0
8
0
0