VERIFICATION TEST PLAN

Team Members:

Poornima Gurushanthappa

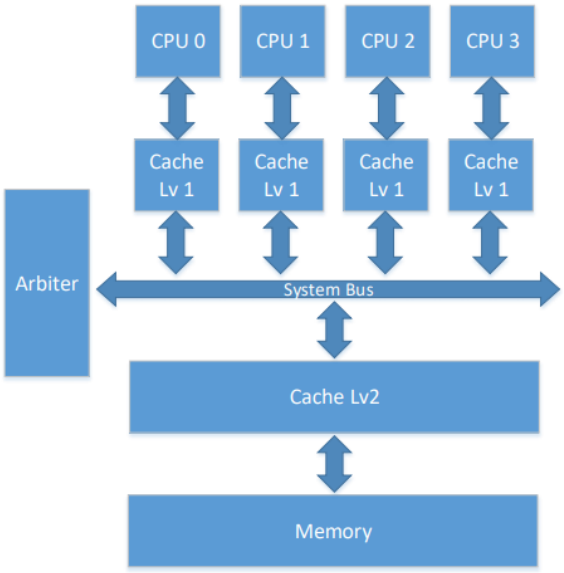
Sourav Sudhir

Prateek Sasikumar

Ganesh Prabhu Sathyanarayana

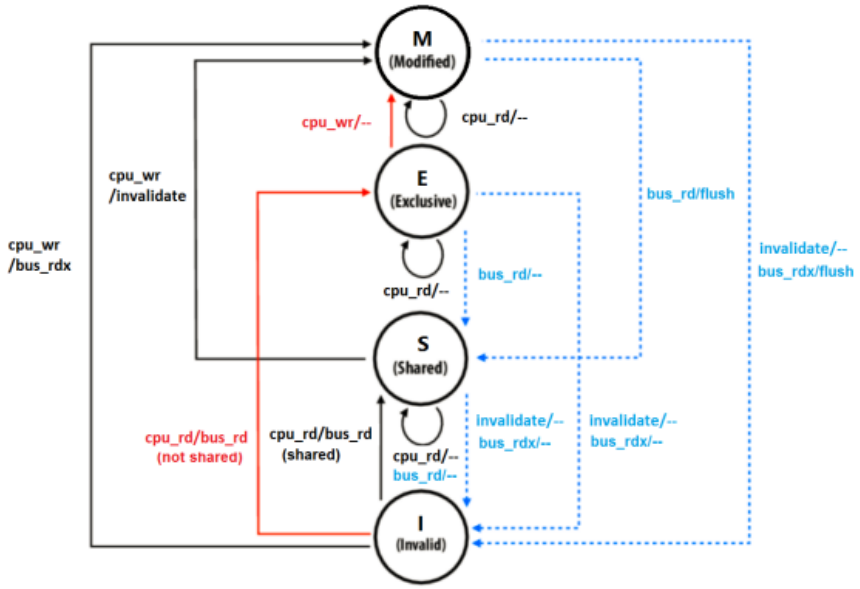
OVERVIEW OF THE DESIGN:

The design under test cache system in a chip-multiprocessor(cmp) with 4 processors, each having a L1 cache and all sharing a single L2 cache. The transactions between the caches is through the system bus. The arbiter determines access to the bus. Level 1 caches are divided into data and instruction caches and only data stored in data caches are shared among the unicore processors. Since many components interact through the same system bus, the transactions on the bus line need to be monitored. The figure below shows the system level interactions.

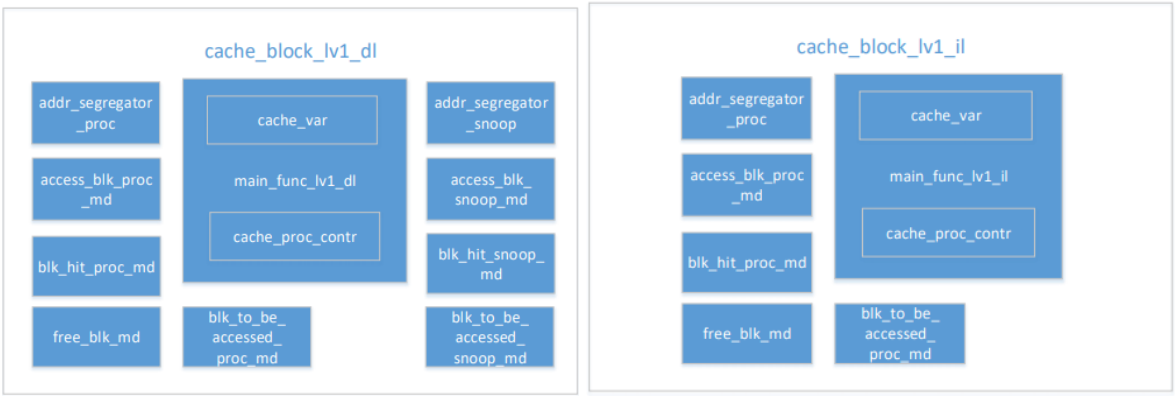


The L1 caches follow the MESI coherence protocol for sharing the data caches among the unicore processors. The state diagram below describes the protocol, and as we can say the cache can have one of four states - Modified, Exclusive, Shared or Invalid. The cache controller may get a read or write request from the processor or the bus and depending on the present state of the cache line, it must provide an output according to the MESI protocol. There are two sides to the Finite State Machines(FSM) for MESI: processor and snoop (or bus) side requests. The verification environment should focus on verifying the functionality in all possible states for all the possible stimuli. It maybe error prone due to the complexity of the state transitions.

The cache replacement policy used for all the caches is PLRU, although L1 and L2 differ in the associativity of the caches. The LRU functionality has two parts: one to determine the replacement block based on current LRU state and another to update LRU state based on what block being accessed in a set. The block replacements need to be verified, here along with writeback to memory and write allocate.



And the third major component of the design is the main functional block. This block is important because it coordinates the overall functionality of the cache system. It processes the hits and misses for read and write requests on the processor side. It also processes the snoop side requests from other caches (such as a hit on the snooping cache). The main functional is also responsible for the interactions with the system bus which are extremely important for proper communication among the different cache blocks. The complexity and the importance of this block in the co-ordination of the cache functions requires a good verification environment to identify the potential errors.



Along with the blocks mentioned above, we need to verify that the system follows all the rules that are set, such as the invalid blocks do not get a hit, or a valid block gets replaced in the presence of an invalid block.

The cache block and the cache controller in Level2 cache needs to be verified too. Features to be verified at Level2 cache is very similar to that of Level1 cache, except that Level2 cache doesn't have a MESI FSM and Instruction cache is writeable.

DESCRIPTION OF VERIFICATION LEVELS:

The provided design is best verified on a hierarchical level. If the entire design is considered to be an SoC, the breakdown would be as follows:

Block level	L1 LRU block, MESI FSM, snoop-side supporting modules of L1 cache block, proc-side supporting modules of L1 cache block, proc-side supporting modules of L2 cache block, L2 LRU block
IP level	L1 instruction cache controller, L1 data cache controller, L2 cache controller, L1 instruction cache block, L1 data cache block, L2 cache block
Sub-system level	L1 cache unicore, L2 cache
SoC level	Memory System.

The order of verifying the system is Block -> IP -> Sub-system -> SoC

At lower levels of verification, if grey box approach is used, we have access to some portion of the internal structure of the module and hence the available signals can be initialized easily and the transitions from those states can be driven and the responses can be tested. With this done, at higher levels, the exact transitions need not be tested. These signals thus provide controllability and reduce the time of testing.

For example, in the case of MESI FSM, if the module was tested at block level, a particular state of FSM can be initialized and the transitions from that state for a particular set of inputs can be tested. If this wasn't done at the block level and the MESI FSM had to be tested from L1 data cache controller, several read/write transactions would have to be performed to put the state machine in a particular state to test a functionality.

Plan per level:

Block level	Verify the functionality of each of the blocks
IP level	Verify the interconnect communication between the blocks and the functionality of the IP
Sub-system level	Verify the interconnect communication between the IPs and the functionality of the sub-system
SoC level	Verify the overall functionality

We cannot proceed to higher levels of hierarchy without verifying the functionalities at lower level as it increases the possibility of bug escape due to which cost of testing increases by a factor of 10 for each level. We intend to thoroughly check at the block levels and reduce the checking at the higher levels (Sub-System level or SoC) to reduce the time (which adds to the cost) of testing that yields a better ROI and a lower time to market. But the risk this entails is the possibility of bug escape at block level which is less likely to be caught at higher levels and this would reduce the coverage.

FEATURES TO BE VERIFIED:

Cache Block:

1. Verify that reads to cache result in the right data being retrieved in the case that data is available in level 1 of the cache. This should be verified for both data and instruction caches for the cases of serial addresses, same addresses and random addresses being accessed.
2. Verify that writes to data cache result in the right data being written to the right address in level 1 of the cache. This should be verified for the cases of serial addresses, same addresses and random addresses being accessed.
3. Verify that read miss in data cache or instruction cache result in the right data being retrieved from the right address in level 2 cache. This should be verified for both data and instruction caches for the cases of serial addresses, same addresses and random addresses being accessed. Also verify any successive reads or writes are not affected.
4. Verify that the invalid case of writes to instruction caches are not allowed. This should be verified for both data and instruction caches for the cases of serial addresses, same addresses and random addresses being accessed.

MESI FSM:

1. Verify that ALL the MESI state transitions occur as specified in the description. This is one of the most critical blocks that needs to be verified since any wrong transition or stuck at fault of any state would result in an incoherent cache. Random state transitions that are not allowed also need to be verified.

LRU replacement policy:

1. Verify that the block evictions are occurring on a miss and the right block address is being evicted. This affects performance of the cache however the cache would work correctly even if there were a bug in this block. Hence, it is secondary feature that needs to be tested

Critical Features:
Cache read and write, Coherency maintained across multiple CPU caches, Write back, All state transitions for MESI FSM.

Secondary Features:
Issuing multiple read and write accesses to cache, LRU policy, write allocate.

Non-verified Features:
Address segregator, Internal LRU state transitions for each cache, Initial Invalidated state of storage structures lru_var, cache_var, cache_proc_contr. Level 2 cache functionality and arbiter is not assumed under test.

TEST METHODS AND SCENARIOS:

Test Method: Grey Box Test method will be implemented for our verification. This provides us controllability to set up scenarios like initialising the MESI state of a cache block to the desired state. Having a grey box test approach will also help us in easier debug of issues as it gives us access to some internal signals.

Test Scenarios/Test cases:

Cache Block:

Intent: To identify any bugs in cache read and write transactions over the system bus. Enables us to cover L1 cache unicore, L1 cache controllers, L1 cache blocks on proc and snoop sides. We try to cover all the legal cases for all cores. This verifies the responsive feature of the cache block.

1. Cache reads
   Single to Multiple Cache reads are generated for both Instruction Cache and Data Cache from different CPU cores.
2. Cache writes
   Single to Multiple Cache writes are generated for Data Cache from different CPU cores.
3. Cache Read Miss followed by Cache Write
   Single to Multiple Cache read misses followed by a write are generated for both Instruction Cache and Data Cache from different CPU cores.
4. Cache Write miss followed by Cache Read
   Single to Multiple Cache write misses followed by a read are generated for Data Cache from different CPU cores.
5. Random reads and writes:
   Intent: To cover different addresses that were not covered previously and also Interactions between lv1 cache, systembus and lv2 cache can be verified.
   Generate Random read and write sequences to cover both Data and Instruction cache.

MESI Protocol:

Intent: To identify the bugs in MESI transitions, cpu and system bus interactions. This covers all the legal and illegal cases. The feature to be tested is cache coherency.

1. MESI FSM
   Verify the following MESI state transitions. This functionality would be best covered with randomized volume exercise.
   - MESI Modified State to Modified State transition
   - MESI Modified State to Shared State transition
   - MESI Modified State to Invalid State transition
   - MESI Exclusive State to Shared State transition
   - MESI Exclusive State to Invalid State transition
   - MESI Exclusive State to Modified State transition

- MESI Exclusive State to Exclusive State transition
- MESI Shared State to Invalid State transition
- MESI Shared State to Shared State transition
- MESI Shared State to Modified State transition
- MESI Invalid State to Modified State transition
- MESI Invalid State to Shared State transition
- MESI Invalid State to Exclusive State transition

LRU:

Intent: To identify bugs in lru state transitions and transactions with blk_hit_snoop_md, free_block_md, block_to_be_accessed_snoop_md. The feature to be tested here is the lru policy of the cache. We identify the legal and corner cases.

1. Cache line replacement
   Fill up all the cache blocks and issue a new read/write to the same block which results in replacement of one of the lines. Write back and write allocate scenarios are also covered in this test case.

Instruction cache:

Intent: To test the read-only feature of instruction cache and hence cover an illegal case.

1. Writes to Instruction cache
   Cache Write is issued for both Level1 and Level2 caches. Writes for Level1 cache shouldn't be processed as it is not allowed, whereas Level2 cache writes needs to be processed.

Checking:

The correctness of the design is verified to through implementing checkers at both internal signal level and at the data output level.

Some signal Checkers:

1. bus_lv1_lv2_gnt_proc should be one hot encoded.
2. cpu_rd and cpu_wr shouldn't be asserted simultaneously.
3. cpu_wr_done should be asserted after cpu_wr has been asserted.
4. lv2_wr_done should be asserted after lv2_wr has been asserted.
5. Data should be available on data_bus_cpu_lv1 after cpu_rd has been asserted.
6. bus_lv1_lv2_gnt_proc needs to be asserted after bus_lv1_lv2_req_proc has been asserted.
7. all_invalidation_done should be asserted after a invalidate signal was being asserted
8. data_in_bus_cpu_lv1 needs to be asserted after cpu_rd has been asserted.
9. data_in_bus_lv1_lv2 needs to be asserted after lv2_rd has been asserted.
10. Data should be available on data_bus_lv1_lv2 after lv2_rd has been asserted.
11. Proc MESI state should be updated from 'x' to 'y' once 'z' was asserted
12. Snoop MESI state not updated from 'x' to 'y' once 'z' was asserted
13. access_blk_proc is one hot encoded.
14. block_hit_proc should be asserted after clk_hit_proc/snoop_md has been asserted.

Some Data Output checkers:

1. On a Cache read, the data on the data bus needs match the expected value.
2. When a cache is read after a cache write, the data read should match with the data written.

DESIGN FUNCTIONALITY COVERAGE

All the functional features have been addressed in our verification plan across all the level of hierarchy. The plan also covers complex interactions like all the MESI state transitions along with cache line replacement and interconnect communication after integration of different levels of hierarchy. Randomizing multiple read and write access and also the sequence of read and writes to cache from multiple CPU cores would cover the corner cases. We expect a functional coverage of 95% due to complexity of features of the design, a block, expression, and toggle coverage of 95%.

Multicore

Code Coverage Achieved:
Block : 81%
Expression: 75%
Toggle : 50%

Functional Coverage Achieved: 55%

Single Core

Code Coverage Achieved
Block : 87%
Expression : 83%
Toggle : 70%

Functional Coverage Achieved: 84%

Coverage is low at multicore level since we have not exercised all the cores equally. This can be safely ignored since each core is just instantiated in the design. Achieving a high coverage in single core would be good.

Request Service to time is one of the cover groups which can be removed from over all coverage since the request to service is not an important factor.