ECE 485/585
Winter 2022
Final Project Description
Notes are marked in blue color below.

Your team is responsible for the design and simulation of a split L1 cache for a new 32-bit processor which can be used with up to three other processors in a shared memory configuration. The system employs a MESI protocol to ensure cache coherence.

Notes: 32-bit processor = 32 address bits

Your L1 instruction cache is four-way set associative and consists of 16K sets and 64-byte lines. Your L1 data cache is eight-way set associative and consists of 16K sets of 64-byte lines. The L1 data cache is write-back using write allocate and is write-back except for the first write to a line which is write-through. Both caches employ LRU replacement policy and are backed by a shared L2 cache. The cache hierarchy employs inclusivity.

Notes: 64-byte lines = 2^6, so there are 6 address bits for Byte Select bits (bit [5:0])
16K sets cache = 2^14, so there are 14 address bits for index (bit[19:6])
So Tag bits = 32 – 6 – 14 = 12 bits (bit[31:20])

How do you know which cache (Instruction or Data cache) are you accessing? The trace file will tell you.

Inclusivity: to maintain inclusivity, there are a bunch of messages to be sent between L1 and L2 cache and you'll be responsible for generating those message appropriately.

Describe and simulate your cache in Verilog, C, or C++. If using Verilog, your design does not need to be synthesizable. Your simulation does not need to be clock accurate and does not need to store/retrieve data.

Maintain and report the following key statistics of cache usage for each cache and display them upon completion of execution of each trace:
Notes: "for each trace" above means for each trace file.

- Number of cache reads
- Number of cache writes
- Number of cache hits
- Number of cache misses
- Cache hit ratio

In order to maintain inclusivity and implement the MESI protocol the L1 caches may have to communicate with the shared L2 cache. To simulate this, you should display the following messages (where <address> is a hexadecimal address).

- Return data to L2 <address>
    - In response to a 4 in the trace file your cache should signal that it's returning the data for that line (if present and modified)
- Write to L2 <address>
    - This operation is used to write back a modified line to L2 upon eviction from the L1 cache. It is also used for an initial write through when a cache line is written for the first time so that the L2 knows it's been modified and has the correct data
- Read from L2 <address>
    - This operation is used to obtain the data from L2 on an L1 cache miss
- Read for Ownership from L2 <address>
    - This operation is used to obtain the data from L2 on an L1 cache write miss

Notes: If during your processing for your L1 cache, all the ways are full and you have a miss, you have to do evict from your L1 cache. What happens if that line is dirty? Answer: you have to write back to the next level cache which is L2 cache.

Outer most cache will do the snooping.

Modes

Your simulation must support at least two modes (without the need to recompile). In the first mode, your simulation displays only the required summary of usage statistics and responses to 9s in the trace file and nothing else. In the second mode, your simulation should display everything from the first mode but also display the communication messages to the L2 described above (and nothing else). You may choose to have additional modes or conditional compilation that display debug information but these should not be present in the version that you demonstrate for the final project demo. Your simulation should not require recompilation to change the name of the input trace file.

Project report

You must submit a brief report that includes relevant internal design documentation, assumptions and design decisions, the source code for your cache, any associated modules used in the validation of the cache (including the testbench if using Verilog), and your simulation results along with the usage statistics.

Traces

Your testbench must read cache accesses/events from a text file of the following format. You should not make any assumptions about alignment of memory addresses. You can assume that memory references do not span cache line boundaries.

n address

Where n is

0 read data request to L1 data cache
1 write data request to L1 data cache
2 instruction fetch (a read request to L1 instruction cache)
3 invalidate command from L2
4 data request from L2 (in response to snoop)
8 clear the cache and reset all state (and statistics)
9 print contents and state of the cache (allow subsequent trace activity)

The address will be a hex value. For example:

2 408ed4
0 10019d94
2 408ed8
1 10019d88
2 408edc

Notes: The above is an example of the trace file. The trace file will tell you whether it is accessing instruction cache or accessing data cache.

For n = 3, it means that any of the other processors moves the cache line from Shared to Modified state due to which all other processors need to be informed about this. Hence all the other processors will be sent an invalidate command from L2, hence our processor will also be sent an invalidate command from L2. Note this would only apply to data cache, not instruction cache since instruction cache cannot go to Modified state because we cannot write to instruction cache.

For n = 4, it means that we snooped an RFO by another processor.

For n = 8, it means to reset the entire cache.

When printing the contents and state of the cache in response to a 9 in the trace file, use a concise but readable form that shows only the valid lines in the cache along with way and appropriate state and LRU bits.

Grading

The project is worth 100 points. Your grade will be based upon:

- External specification
- Completeness of the solution (adherence to the requirements above)
- Correctness of the solution
- Quality and readability of the project report (e.g. specifications, design decisions)
- Validity of design decisions

- Quality of implementation
- Structure and clarity of design
- Readability
- Maintainability
- Testing
- Presentation of results

Notes:
Testing: need to have a test plan with various test cases. Write down what you expect to see on the test case and check results.

Project report: not looking for ~35 pages. Looking for what's the internal design, including what part of address field is used for cache index, byte select and so on plus any design decision you made.

Demo: Each team has ~20 minutes to do the demo for the final project in Week 10.