Project Requirements

18 November 2024 13:04

Use GitHub as a code repository to facilitate **code sharing and revisions**.

Simulation of the last level cache (LLC) for a new processor that can be used with **up to three other processors** in a **shared memory configuration.**

CACHE:

- Total capacity = 16MB
- 64-byte lines
- 16-way set associative
- Write allocate policy
- MESI protocol
- Replacement policy: pseudo-LRU scheme

Next higher level cache (you do not need to model it):

- 64-byte lines
- 4-way set associative
- The first write to a line is write-once policy: the first write to a line is write through;
- Subsequent writes to the same line are write-back
- Must maintain inclusivity

Model and Simulation:

- Python, C, or C++ or Verilog or SystemVerilog
- Does not need to be clock accurate
- Does not need to be synthesizable
- Do not need to store data in the cache as the cache behavior (hits, misses, evictions, etc.) is independent of the data.

Model communication between your **LLC and the next higher level cache**
Bus operations that your LLC performs
Snoop results that your LLC reports on the bus in response to snooping the simulated bus operations of other processors and their caches.

Implement the **following functions and use them as appropriate in your simulation**.
Put the #defines in a separate include file so they can be used by your main code.

Use whatever method you want to simulate the snoop result returned by other LLCs in response to your LLC's bus operations.
But **it must be repeatable and easily changed** so that you can test that your simulated cache behaves properly **for all return values.**

Output:
Maintain and report the following key statistics of cache usage for each cache and print them upon completion of execution of each trace:

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

Modes:

Must support **at least two** modes (without the need to recompile):

- **Silent**
- **Normal**

- In **silent mode**, your simulation displays only the required summary of usage statistics and responses to 9s in the trace file and nothing else.
- In **normal mode**, your simulation should display everything from the first mode but also display the bus operations, reported snoop results, and communication messages to the higher level cache described above.

Additional modes (e.g. verbose) or conditional compilation
That display debug information **but these should not be turned on in the version that you demonstrate to me**

Your simulation **should not require recompilation** to change the name of the input trace file.

- In C, you should use command line arguments (e.g. argc, argv[]) to specify the mode and input file.
- In SV, consider using the $value$plusargs function.

Testing:

- Crucial part of your project.
- **Create an explicit written plan** for how you will ensure that your model is operating correctly **and include it** in your final report.
- **Describe each of the tests** you perform on the model.

Project report:
Submit a report that includes:

- Design specification that describes the interface to the cache module (to the next level in the memory hierarchy, and any shared buses) relevant internal design documentation
- The source modules for your cache,
- any associated modules used in the validation of the cache (including the testbench if using Verilog/SystemVerilog), and
- Simulation results along with the usage statistics.
- Make (and document) reasonable assumptions about the processor and memory subsystem and their interfaces.
- The report should justify these assumptions as well as any design decisions you make.
- Be sure to state any assumptions about the L1 cache as well.