ECE 485/585
Fall 2012
Final Project Description

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.
Your L1 instruction cache is two-way set associative and consists of 16K sets and 64-byte lines. Your L1 data cache is four-way set associative and consists of 16K sets of 64-byte lines. The L1 data cache is write-back using write allocate. Both caches employ LRU replacement policy and are backed by a shared L2 cache.

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.

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

Modeling the next level in the memory hierarchy

You must have a model of the next level in the memory hierarchy so that you can properly handle misses.   You can simply have a "stub" module for a unified L2 cache.   It will present an interface to the L1 caches and respond but not really model a cache.

Project report

You must submit a report that includes a 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), and your 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.

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 cross cache line boundaries.

```
n address
```

Where n is

0	read data request from L1 cache
1	write data request from L1 cache
2	instruction fetch (treated as a read request from L1 cache)
3	invalidate command from L2
8	clear the cache and reset all state
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
```

When printing the contents and state of the cache use a concise but readable form that shows only the valid lines in the cache along with any state 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