ECE 485/585
Fall 2013
Final Project Description

Your team is responsible for the simulation of a unified L2 cache design for a new 32-bit processor that 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 cache uses 64-byte lines, is 8-way set associative, and has 16K sets. It uses a true LRU replacement policy and write allocate.

Although you do not need to model them, the processor's (split) L1 instruction and data caches both use 32-byte lines and are 2-way and 4-way set associative respectively. The L2 cache is backed by a DRAM-based memory subsystem that reads and writes 64 bytes at a time. Your system must maintain inclusivity.

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.

Output

Your model must produce the following output. For each operation your model performs on the shared bus, you must output a line as follows:

R address
W address
M address
I address

Where R indicates a memory R, W indicates a memory write, M indicates a read with intent to modify, I indicates an invalidate, and address is a hexadecimal memory address.

Your model must also output a line for any communication with the L1 caches. You can create your own syntax for this but it must begin with 'L1' and be consistent with the format above. Describe the syntax in your final report.

When your model needs to obtain the snoop result for any operation it performs on the bus it should call the int function GetSnoopResult(int Address, char Operation) where Address is the address and Operation is either 'R', 'W', 'M', or 'I'. The return value is 0 for no hit, 1 for HIT, and 2 for HITM (Hit on a Modified Line). Your function implementation can use whatever (reproducible) method you want for generating the return value but your model must be capable of handling any of the three possible values.

Similarly, your model must provide and call an int function PutSnoopResult(int Address, char Operation) to report your result of snooping the bus operations initiated by other processors on the shared bus (see codes 3-6 under Traces below). This function should write your response to the output using the format

SR no HIT
SR HIT
SR HITM

to indicate snoop responses of "no hit", "hit", or "hit to a modified line" respectively.

You should have a method to either turn off all of the above output or direct it to a file.

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

Testing

Testing is a crucial part of your project. Create an explicit plan for how you will ensure that all aspects of your model is operating correctly and include it in your final report. Describe each of the tests you perform on the model.

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 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 request from L1 data cache
1 write request from L1 data cache
2 read request from L1 instruction cache
3 snooped invalidate command
4 snooped read request
5 snooped write request
6 snooped read with intent to modify
8 clear the cache and reset all state
9 print contents and state of each valid cache line (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