

ECE 5752- Spring '05

Programming Exercise #4

Writing a DRAM simulation model & integrating that model into SimpleScalar 3.0

Assigned: Monday 03/21/05

Due: Monday 04/18/05

All work should be done individually, no group work allowed for this assignment

Purpose:

This assignment is very similar to the previous assignment, however it involves the generation of a more complex model, namely that of a DRAM device. The current main memory model for the sim-outorder tool in SimpleScalar 3.0, as defined in `sim-outorder.c`, is contained in the function `mem_access_latency()`. This fixed latency model is not realistic for any reasonable type of memory system. For completion of this assignment, a realistic synchronous DRAM model should be developed and integrated into the SimpleScalar environment.

- Follow the timing parameters specified by a industry-supplied synchronous DRAM data sheet. Must be capable of a minimum 64-bit bandwidth of 1GB/s.
- Complied with a given memory system architecture - specified in the parameterized configuration. The diagramed memory architecture must cover the full 2GB SimpleScalar address space.
- Have a verbose which can be enabled using a SimpleScalar configuration file or command line parameter
- Be parameterized to allow for the following variables:
 - Processor / DRAM bus multiplier
 - At least 2 unique sets of timing characteristics (i.e. 2-2-2 SDRAM vs. 3-3-3 SDRAM)
 - Various implementations of the required 2GB address space (i.e. 1 single sided 2GB DIMM, 2 double sided 1GB DIMM, etc)
- Provide the following statistics specifically from your DRAM simulation engine:
 - Number of DRAM accesses (#)
 - Average Access Latency (nS)
 - Data Bus occupancy (%)

Problem Specification:

The DRAM model should meet the following requirements, both independently and when integrated into the SimpleScalar environment.

For completion of this assignment, each student will be required to:

1. Generate a DRAM model
2. Integrate that DRAM model into the SimpleScalar environment
3. Validate operation of the model in a verbose mode
4. Compare the simulated performance of three benchmark/input set between:
 - un-modified SimpleScalar
 - (4) unique DRAM-enabled SimpleScalar configurations varying {multiplier, address space & timing characteristics}

It is recommended that you follow the suggested procedure, but if you are able to complete the requirements of the problem, the procedure is secondary.

Suggested Procedure:

1. Select a DRAM data sheet from those available online - one available in PDF is mandatory. Any product/datasheet which is capable of greater than 1GB/s bandwidth is acceptable, however the model for some may be easier to write than others. A possible source for datasheets is listed below.

<http://www.micron.com/products/category.jsp?path=/DRAM>

2. Once you have selected a datasheet, it is recommended that you write an object-oriented diagram of the software model which you intend to implement. If you prefer a flow-chart approach, that is also acceptable, but since the model will not be a stand alone program, the object-oriented diagram is recommended.

It is suggested that the following procedure be followed using one (or more) of the COLOR Linux machines in the btdpool, but if you have a Linux machine that is personally available, you are free to use that machine as well.

3. In modifying the SimpleScalar toolset, it is advised that you NOT destroy the stock configuration. Therefore you can either work wisely in the original source directory creating a new binary from mildly modified *.c files, or more simply, copy the entire source tree to a new directory in which you will make your modifications.
4. Using the flow-chart/diagram you developed in step 2, write a model for the DRAM device. This model can be written in any language, but must be able to be integrated into the SimpleScalar environment. If you intend to integrate your model into the sim-outorder tool it is most likely that you are going to have to change the parameters to and code in the following function.

```
static unsigned int mem_access_latency()
```

Your model should be written such that it is encapsulated in a new *.c file. You will be required to modify the Makefile to include this new *.c file into the binary you specify (either sim-outorder or a new name).

5. Your model should be capable simulating: (a) A flexible # of processor cycles per DRAM bus cycle; (b) A flexible set of timing characteristics; (c) A flexible implementation of the required

2GB address space (i.e. (16) 1 Gbit devices organized as 64M x4 or 32M x8 devices; or (32) 512 Mbit devices organized as 32M x4 or 16M x8 devices). These parameters to the simulation should be verified as well as you are able prior to turning in the results for the completion of the programming assignment.

6. In addition, your model should be written such that it can be run in either a “verbose” mode, or a quiet mode. In either mode, the model should register the statistics given above via *mem_reg_stats()*, and thus report statistics at the end of simulation. In verbose mode, it should display, for each access, the time of access, the address, the block size, and the latency returned. In quiet mode, the model returns the appropriate latency to SimpleScalar, and only reports results via the registered statistics.
7. Verify that after your source-code changes the simple-scalar 3.0 toolset still builds properly. Run your modified simulation tool, using verbose mode, on a single configuration file - use the same tool & configuration file for which you have results from prior labs or executions. Verify that your source-code changes generate modified output from the simulation execution.
8. Simulate your (4) desired benchmark/input sets (which executes at least 500 million instructions each) using minimally five unique configurations of your definition with varied parameters.

Considerations:

1. What did you feel was the hardest element of this assignment, be specific. Not simply “development of the DRAM model” but what aspect of that process?
2. What, if any, state is maintained by your model, does it pertain to the state of the bus between the microprocessor core and the DRAM, the internals of the DRAM devices, or some other element of the system?
3. Do you feel that the DRAM model you have developed is more accurate than the “fixed latency” model of the original SimpleScalar 3.0b distribution? Why or why not? What elements of your model provide a more realistic latency than the “fixed latency”?
4. Why did you choose the DRAM (or datasheet) which you did? In hindsight do you feel this was a good choice?
5. How would you expand your simulator to enable the simulation of a wider variety of DRAM technologies?

Deliverables

Many of the deliverables - the text file outputs - can be included in the report two-pages to each side of a piece of paper. The report should remain full page size text.

1. Lab report, format specified in lab guidelines, Results should include and explain all generated figures. Conclusions should include the performance comparisons between the modified and unmodified sim-outorder configurations. Explain the results you generate.

2. A diagram showing the implementation of your memory system, and a URL which will allow access to the datasheet of the DRAM you have modeled.
3. Listing of the source code file which contains your DRAM model
4. Generate charts of the results produced by your simulation. A graphical representation of the results of the (15) simulation runs showing:
 - a). Execution Time (sim_cycle)
 - b). Cycles per instruction (sim_CPI)
 - c). Average main-memory access-timeEach statistic will require a unique figure.
5. Output from:
 - (2) PARTIAL verbose-mode modified sim-outorder simulations showing different simulated bus multipliers.