

ECE 5900_04 - Spring '01

Laboratory Exercise #4

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

Assigned: Mon 4/9/01

Due: Fri 4/27/01

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 3.0b, as defined in `sim-outorder.c` is 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.

Problem Specification:

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

- Comply with a given memory system architecture - specified in a diagram you provide
- Follow the timing parameters specified by a industry-supplied synchronous DRAM data sheet
- Have both a verbose and quiet mode which can be set using a `simplescalar` configuration file
- Be parameterized to allow for the following variables:
 - Controller policy either Close-Page-Autoprecharge or Open Page
 - At least 2 unique timing models (i.e. 2-2-2 SDRAM vs. 3-3-3 SDRAM or 800MHz DRDRAM vs. 1066 MHz DRDRAM)
- Provide the following outputs specifically from your DRAM simulation engine:
 - Number of DRAM accesses (#)
 - Average Access Latency (nS)
 - Data Bus occupancy (%)

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 one benchmark/input set between:
 - un-modified SimpleScalar
 - DRAM-enabled SimpleScalar using Open Page policy
 - DRAM-enabled SimpleScalar using Close-Page-Autprecharge policy
 - DRAM-enabled SimpleScalar using Open Page policy using secondary timing model
 - DRAM-enabled SimpleScalar using Close-Page-Autprecharge policy using secondary timing model

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. It is also suggested that the following procedure be followed using one (or more) of the ‘color’ Linux machines such as ‘red.ee.mtu.edu’.

Suggested Procedure:

1. Select a DRAM data sheet from those available online - one available in PDF is suggested. Any product/datasheet is acceptable, however the model for some may be easier to write than others.
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.
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 outline/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. 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 in the problem specification 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.
6. 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.

7. Simulate your desired benchmark/input set (which executes at least 100 million instructions) using minimally five configurations, as defined above.

Considerations:

1. Generate a chart of the results produced by your simulation. Include the overall execution time as well as the statistics generated by your DRAM model in the chart.
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. 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?

Deliverables

1. Lab report, format specified in lab guidelines, 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. Output from the (1) sim-outorder simulations, (4) modified sim-outorder simulations, and (1) PARTIAL verbose-mode modified sim-outorder simulation.