

# ECE 5900\_04 - Spring '01

## Laboratory Exercise #3

### Writing a SRAM simulation model & integrating that model into SimpleScalar 3.0

Assigned: Wed 3/14/01

Due: Fri 3/30/01

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

### Purpose:

The purpose of this exercise is to develop an understanding of how extensions are added to the simplescalar environment, and repair a significant drawback of the standard distribution. 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. A SRAM model should be developed and integrated into the SimpleScalar environment. The SRAM model should meet the following requirements, either independently or 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 data sheet
- Have both a verbose and quiet mode which can be set using a simplescalar configuration file

### Problem Specification:

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

1. Generate a SRAM model
2. Integrate that SRAM model into the simplescalar environment
3. Validate operation of the model in a verbose mode
4. Compare the simulation of three benchmark/input set configurations between the newly instrumented SimpleScalar simulator and the un-modified SimpleScalar simulator.

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 SRAM 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. Two possible sources for datasheets are listed below.

<http://www-3.ibm.com/chips/techlib/techlib.nsf/productfamilies/SRAM>

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

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 on one of the “kirchhoff#” machines in the 7th floor lab.

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 SRAM devices. 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 statistics via *mem\_reg\_stats()*, and thus report statistics at the end of simulation. These statistics should include the number of accesses and the average latency. 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 three unique benchmark/input sets using both the modified and unmodified sim-outorder. For one (recommended fewest main-memory accesses) benchmark/input set run your modified sim-outorder using the verbose mode used for model validation.

## Considerations:

1. What did you feel was the hardest element of this assignment, be specific. Not simply “development of the SRAM model” but what aspect of that process?
2. Do you feel that the SRAM 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”?

3. What, if any, state is maintained by your model, does it pertain to the state of the bus between the microprocessor core and the SRAM, the internals of the SRAM itself, or some other element of the system?

## **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 SRAM you have modeled.
3. Listing of the source code file which contains your SRAM model
4. Output from the (3) sim-outorder simulations, (3) modified sim-outorder simulations, and (1) PARTIAL verbose-mode modified sim-outorder simulation.