

# ECE 5752 - Spring 2005

## Programming Exercise #3

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

Assigned: Wed 03/02/05

Due: Mon 03/21/05

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 3.0d, 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 this assignment an SRAM “main-memory” model should be developed and integrated into the SimpleScalar environment. This is only an accurate model of a high-cost machine such as the Cray C90. 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 and constraints specified by a industry-supplied SRAM data sheet
- Include a realistic bus model incorporating contention and the fact that only a single transaction may occupy the address or data portion of the bus during any cycle between the microprocessor BIU and SRAM device(s)
- 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 for the SimpleScalar-PISA 31-bit address space. If you want to truncate the memory space down to 256MByte that is considered legitimate for this exercise.
2. Incorporate a parameter into the model which allows a variable integer number of Processor cycles for each SRAM bus cycle. This will allow you to simulate multiple clock multipliers, effectively allowing flexible bus and processor frequency combinations.
3. Integrate that SRAM model into the `simplescalar` environment
4. Validate operation of both the bus and the SRAM model in a verbose mode

5. Compare the simulation of three benchmark/input set configurations between (a) the unmodified SimpleScalar simulator and your modified and instrumented SimpleScalar simulator running (b) clock multiplier 15 and (c) clock multiplier 10.

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 Synchronous SRAM data sheet from those available on-line and capable of multiple speeds (200 MHz & 133 MHz suggested) - A PDF of the datasheet must be available. Any product/datasheet is which meets these criterion is acceptable, however the model for some may be easier to write than others. A possible source for datasheets is listed below.  
<http://www.necel.com/memory/english/document/sram.html>  
<http://www-3.ibm.com/chips/techlib/techlib.nsf/products/SRAM>  
<http://www.samsung.com/Products/Semiconductor/SRAM/TechnicalInfo/datasheets.htm>
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 at this point that the timing diagrams required for the deliverables are suggested to be drawn. This is useful in the development of the model.  
It is suggested that the following procedure be followed on one of the COLOR machines in the btdpool or any other Linux machine you have available.
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. Your model should be capable of a flexible # of processor cycles per SRAM bus cycle. This is referred to as the clock multiplier. For the required deliverables you will be asked to simulate two processor/bus frequency pairings - again suggested pairings are 2GHz/133Mhz and a 2GHz/200Mhz.

6. 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.
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 four unique benchmark/input sets for at least 500 Million instructions each using an unmodified version of sim-outorder and the modified sim-outorder using both frequency pairings or bus multipliers. The only changes from the default configuration should be those parameters added for your SRAM model. 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 3.0 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 internal state of the SRAM itself, or some other element of the system? How often must this state data be replicated, if it must be replicated at all?
4. How many unique devices or ICs were simulated in your SRAM model? How does the variation between memory implementations encourage the design of memory simulations using object-oriented techniques?
5. An SRAM main-memory is significantly more expensive than a DRAM main memory. Explain how an SRAM main memory can improve performance when both SRAM and DRAM operate at the same 100/133/166/200 MHz frequencies.

## 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, Explain your results including generated figures, Conclusions should include the performance comparisons between the modified and unmodified sim-outorder configurations.

2. A diagram showing the implementation of your memory system, including the number of devices wide that are required to cover the bus and the total number of devices required. Please also provide a URL which will allow access to the datasheet of the SRAM you have modeled.
3. Provide timing diagrams showing {read\_to\_write, read\_to\_read, write\_to\_read and write\_to\_write} transitions for accesses which are destined for the same bank of devices.
4. Listing of the source code file which contains your SRAM model
5. A graphical representation of the results of the (12) 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.
6. Partial output from the (4) sim-outorder simulations, (8) modified sim-outorder simulations, and (1) PARTIAL verbose-mode modified sim-outorder simulation.