

# SATSim: A Superscalar Architecture Trace Simulator Using Interactive Animation

Mark Wolff   Linda Wills  
School of Electrical and Computer Engineering  
Georgia Institute of Technology  
{wolff,linda.wills}@ece.gatech.edu

*Abstract – This paper describes an interactive animation tool, SATSim, which conveys superscalar architecture concepts. It has been used in an advanced undergraduate computer architecture course to visualize the complicated behavioral patterns of superscalar architectures, such as out-of-order execution, in-order commitment, and the impact of branch mispredictions and cache misses. SATSim allows students to interactively change hardware configuration parameters and to observe their effects visually in a more accessible manner than is currently possible with existing simulators or with traditional static media.*

## Introduction

As the complexity of superscalar architectures increases, the underlying concepts of dynamic scheduling and speculative execution become increasingly difficult to explain without some form of visualization. These architectures typically exhibit complex behavioral patterns involving concurrent, out-of-order execution of instructions on multiple functional units and the resolution of data and control dependencies on the fly. Static visual aids, such as diagrams on whiteboards, slides, or in textbooks, are limited in their ability to simultaneously convey both the structural relationships between components of a superscalar architecture and the temporal relationships between instructions as they progress through the pipeline and utilize architectural resources.

An alternative is to use a superscalar architectural simulator. However, existing simulators [6,7,8,9] have been designed primarily for performing research, in which the focus is on accurately modeling the *effects* of architectural

mechanisms so that their performance characteristics can be studied. Most of these simulators do not attempt to visually convey the behavior of the architectural mechanisms themselves and how they interact. They are often designed to accurately model a particular architecture and are often too complex to be suitable for students who are new to superscalar architecture concepts. More importantly, these simulators are typically not interactive: a machine configuration is specified and the simulation summarizes performance and resource utilization results as output. It is not easy to observe what is going on during the simulation (e.g., what the components are doing and how they interact), and it is not possible to interactively experiment with the simulation (e.g., to force a branch misprediction or a cache miss at any point during simulation in order to observe its effect).

This paper describes SATSim: Superscalar Architecture Trace Simulator. SATSim is being used at Georgia Tech in an undergraduate senior-level course on advanced computer architecture. SATSim provides an animated and interactive visualization aid for teaching some of the important concepts associated with superscalar architectures. These concepts include out-of-order execution, in-order commitment, dynamic resolution of data dependencies using register renaming and reservation stations, and the performance effects of branch prediction accuracy and cache hit rates.

## Simulator Details

SATSim reads instructions from a trace file and displays them as they progress through a simulated microarchitecture. The user can configure the microarchitecture by selecting the superscalar factor,

the number of reservation stations per execution unit, the number of reorder and rename buffer entries, and the number of each type of execution unit. In addition, the user can specify branch prediction accuracy, cache miss rates, and cache miss penalties, using the dialog box shown in Figure 1.

SATSim models the behavior of the specified microarchitecture, avoiding processor-specific idiosyncrasies that tend to confuse the concepts. Branch prediction and cache functionality are generated statistically with manual override to illustrate the effect of a misprediction. The trace file input is in MIPs format. The simulator decodes all instructions into one of only five types, integer, floating point, branch, load, and store. Source and destination registers are decoded so that data dependencies and renaming resource utilization can be accurately portrayed. Figure 2 shows a screen shot of the animation in progress. Figure 3 shows the animation window as it progresses through six simulated clock cycles. In particular, instruction 5 is shown going from issue to commitment.

Each instruction is given a three-digit name so that it can be displayed and tracked by the user as it progresses through the simulated processor. The simulator displays the status of all in-flight instructions. Instructions are shown as they occupy locations in the pipeline, rename buffer, reorder buffer, reservation stations, and execution units. Instructions acquire color as their destination register is renamed, and lose color once their result has been broadcast. The color, or lack of color, for the instruction providing data to each source register, is displayed for instructions that occupy a reservation station. The current color assignment, if any, for each of the registers is also displayed. The status of each instruction is updated on each simulated clock cycle. Running performance metrics are also updated and displayed on each clock cycle.

The animation has three interactive features that are useful for explaining and understanding superscalar architecture concepts. The user can force the next fetched branch instruction to be mispredicted, and then observe the resulting performance impact. The user can force the next

fetched line of instructions to miss in the instruction cache. And, the user can force the next load instruction that executes to miss in the data cache.

During the simulation, the user has control of the animation speed. The animation can be set to update the display every 1, 10, 100, or 1000 clock cycles. The user can single step through the animation, set the animation to 1, 2, or 10 screen updates per second, or let it run as fast as possible. The user can also allow the simulation to run to the end of the trace file without updating the screen.

When the simulation ends, the program writes pertinent performance and utilization data to a tab-delimited text file. Table 1 lists the data that is written to disk after each simulation.

■ Date and Time	■ Rename Utilization
■ All Simulation Parameters	■ Integer Execution Utilization
■ Trace File Name	■ Floating Point Execution Utilization
■ Total Cycles	■ Branch Execution Utilization
■ Number Instructions Committed	■ Memory Execution Utilization
■ Number of Each Instruction Type Fetched	■ Integer Reservation Utilization
■ Total Icache Misses	■ Floating Point Reservation Utilization
■ Total Miss Penalty	■ Branch Reservation Utilization
■ Total Stall Cycles	■ Memory Reservation Utilization
■ Total Dcache Misses	
■ Reorder Utilization	

Table 1. Data Collection

SATSim runs on Windows 9x or NT. The program was developed using Microsoft Visual C++ Version 6.

### Simulator Use

The simulator is currently used in an advanced undergraduate computer architecture course to assist students with understanding superscalar architecture concepts. One associated assignment asks the students to discuss the effects of branch prediction accuracy and cache hit rates on the performance of superscalar architectures. Another assignment asks the student to explore the design space by varying the architectural parameters, and then discuss the tradeoffs between chip-area cost (based on a given cost model), cycle time, and architectural resources.

The simulator allows students to explore the design space. Then, the archived data allows for more thorough analysis of design tradeoffs. Through these assignments, the students were able to explore the difficult concepts of dynamic scheduling and speculative execution.

### Conclusions and Future Work

SATSim and the associated assignments assist students with understanding superscalar architecture concepts. Significant improvement was observed, through quizzes and class participation, in the students' comprehension, as compared to previous courses.

SATSim conveys important fundamental concepts in superscalar architecture design and their associated nomenclature. Once this conceptual foundation is in place, more advanced concepts can be discussed, such as trace caches, branch predication, data prediction, vector processors, SIMD ISA extensions, and multiprocessor systems.

Enhancements to SATSim include: an online help system will be incorporated, a browse function will be added for selecting input and output files, and a utility for running simulations in batch mode will be included. Real-time visualization of resource utilization information will also be added.

The simulator can be downloaded from <http://ece.gatech.edu/research/pica/SATSim/satsim.html>

### References

- [1] Hennessey, J.L., Patterson, D.A., "*Computer Architecture A Quantitative Approach*," 2<sup>nd</sup>. Ed., Morgan Kaufmann Publishers, 1996.
- [2] Flynn, M.J., "*Computer Architecture Pipelined and Parallel Processor design*," Jones and Bartlett Publishers, 1995.
- [3] Song, S.P., Denman, M., Chang, J., The PowerPC 604 RISC Microprocessor, *IEEE Micro*, October 1994, p.8-17.
- [4] Kessler, R.E., The Alpha 21264 Microprocessor, *IEEE Micro*, March 1999, p.24-36.
- [5] Yeager, K.C., The Mips R10000 Superscalar Microprocessor, *IEEE Micro*, April 1996, p.28-40.
- [6] Douglas C. Burger and Todd M. Austin, The SimpleScalar Tool Set, Version 2.0, *Computer Architecture News*, 25 (3), pp. 13-25, June, 1997
- [7] Moura, C., SuperDLX A Generic Superscalar Simulator, ACAPS Technical Memo 64, McGill University School of Computer Science.
- [8] DLXview, <http://yara.ecn.purdue.edu/~teamaaa/dlxview/>
- [9] Larry B. Hostetler & Brian Mirtich. DLXsim --- A Simulator for DLX



