

Sabbatical Leave Report
David A. Poplawski
Department of Computer Science
Leave Period: Spring Semester 2006

My proposed program of study consisted of two projects involving software used in computer organization/architecture classes. During my leave I completed one of the projects and got a start on the other.

The first proposed software package was a new logic simulation tool. I completed this project, writing approximately 30,000 lines of Java code and over 13,000 words of on-line documentation. The code was fairly thoroughly tested and is, in my opinion, ready to be used by students. A faculty member at Grand Valley State University has agreed to beta-test it in his computer organization course this fall. By early 2007 it will be made available to educators in other educational institutions, and I have worked with MTU's intellectual property office on the details of making this work. I have also written a paper, to be submitted to the 2007 Computer Science Education conference (SIGCSE), documenting (and promoting) the software. *attached*

Two versions of the package were implemented. One is the fully functional version which will be distributed. Another is a web-page accessible version (a Java applet), which has all the same functionality except the ability to save or print the circuit that was designed. This version makes it extremely easy for potential users to experiment with the package and, hopefully, find it something they will want to use at their site. It is available at <http://www.cs.mtu.edu/pop/jlsp/bin/JLS.html>.

The work on the first project took longer than expected because I decided to add several features that I had not originally planned on (or had even conceived of). These features make the software more robust and add capabilities found in no other comparable software package.

As a result I did not make as much progress on the second project, the assembly language IDE, as I would have liked. However the ground work was laid for future work on the project, either by myself or by students looking for independent study projects.

Unfortunately nothing could be worked out with Hope College, so the hoped for interaction with their faculty did not come to pass. I am very disappointed by this.

JLS: A Pedagogically Targeted Logic Design and Simulation Tool

David A. Poplawski
Department of Computer Science
Michigan Technological University
Houghton, Michigan
pop@mtu.edu

ABSTRACT

JLS is a GUI-based digital logic simulation tool specifically designed for use in a wide range of digital logic and computer organization courses. It is comparable in features and functionality with commercial products, but includes many student and instructor-friendly aspects not found in those products such as state-machine and truth table editors, extensive error checking, and multiple simulation-result views. Students quickly become proficient in its use, enabling them to concentrate on circuit design and debugging issues. The circuit drawing interface is convenient enough to allow instructors to use it for classroom presentations, and circuits can be modified and tested so quickly that it promotes exploring alternatives not prepared for in advance. Its non-GUI execution capability, with parameter settings, configuration files and textual output simplifies the grading of large numbers of student projects.

Categories and Subject Descriptors

K.3.2 [Computers and Education]: Computer and Information Science Education

Keywords

Logic Simulation

1. INTRODUCTION

Choosing a logic simulation tool for introductory courses in logic design and computer organization is problematic. On the one hand there are industrial strength tools that are expensive, have steep learning curves, and are generally used by experienced designers. Many of these tools use textual, as opposed to graphical, input.

On the other hand there are many free tools. Most are (very) limited in functionality. Few will work on multiple platforms.

JLS is a portable powerful digital logic simulation tool created primarily for educational use that addresses the prob-

lems listed above. It is written in Java and has been tested on many platforms. It is free. It consists of a simple to use yet powerful graphical editor that allows users to create and modify logic circuits, and a simulator that will show (in multiple ways) the operation of the circuit over a period of time. Circuits as simple as a few logic gates or as complex as complete CPU microarchitectures with embedded subcircuits have been created and simulated in JLS.

Logic circuits can contain the standard gate types: AND, OR, NOT, NAND, NOR, XOR, tri-state buffer and logically neutral time delay element; composite elements: decoder, multiplexor, and adder; memory elements: registers, SRAM, and ROM; a clock and various mechanisms for connecting gates and elements via wires and wiring elements. State machines can be created by using JLS's state machine editor. Combinational circuitry specified by a truth table can be generated by using JLS's truth table editor. Complex, synchronized multi-signal inputs can be specified. Circuits can include copies of other circuits (subcircuits), nested to an arbitrary depth. Circuits can be printed or exported as image (JPEG) files.

JLS has many features that are useful for classroom instruction, student use, and for the grading of students' circuit assignments. These include simple and intuitive drawing and editing, comprehensive error checking/reporting and extensive on-line help and undo/redo.

This paper discusses the pedagogical requirements of a digital logical simulation tool. It then describes JLS and, in particular, its features that address those requirements. Finally it compares JLS with other available tools.

2. PEDAGOGICAL ISSUES

A logic simulation tool designed for pedagogical use shares many attributes with large, expensive professional tools, but many attributes are peculiar to this use. Among the most important are cost (the cheaper the better, but free is best), extreme ease of use, convenient ways to view a circuit's dynamic operation, and mechanisms to simplify the grading of multiple student projects. These and other issues are addressed below.

2.1 Graphical User Interface

Instructors draw logic diagrams when they lecture, textbooks show logic diagrams when illustrating circuits, so a simulation tool must also have a graphical user interface so that students can experiment with circuits in the way they have seen them presented. Text-based circuit specification is too cumbersome and non-intuitive for students new to the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCSE '07 Covington, KY USA

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

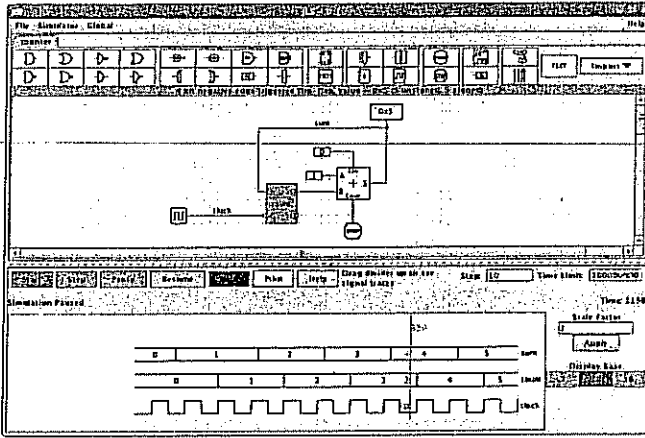


Figure 1: JLS window with simple circuit and paused simulation signal trace.

- Debugging should be simple, especially w.r.t. slowly stepping the simulation of a circuit and the interrogation and display of signal and memory element values.
- The tool should be portable so that the student can use it at their convenience on their platform of choice.

3. JLS

JLS addresses all the pedagogical issues discussed above. It is easy to learn and use but powerful for both instructor and student in a wide range of digital logic and computer organization courses. Rather than enumerate all its features and describe each, its more interesting and unique capabilities will be described here. Figure 1 shows the JLS GUI with a simple circuit and simulated signal trace.

3.1 Circuit Creation

Circuits are constructed by selecting, configuring and placing logic elements in the drawing area. The user simply clicks on an element on the tool bar, which brings up a dialog box with which the user can select characteristics such as the number of inputs to logic gates, the number of replications of the element orientation, names for registers, sizes and widths of memory, etc (see Figure 2). For convenience, numeric values can be entered by pulling down and clicking on a keypad. With a single click users can copy the characteristics of the previously created element of the same type or display the help page for the particular element.

When satisfied the resulting element is dragged to its desired location and placed. As it is being dragged, overlaps with existing elements are indicated, and placing an element on top of another is not allowed. Also, when inputs or outputs of the new element overlap unattached inputs or outputs (unattached inputs/outputs are small circles), or unattached wire ends, the circles turn green if a legal connection can be made or a message is displayed indicating why they cannot be made (for example, if the user attempts to directly connect the outputs of two non-tristate gates, or if the bit-widths differ).

Multi-segment wires are drawn by simply clicking them in place. The wires automatically assume the bit width of the input or output or other wire they become connected to. The end of a wire can be connected to another wire at any

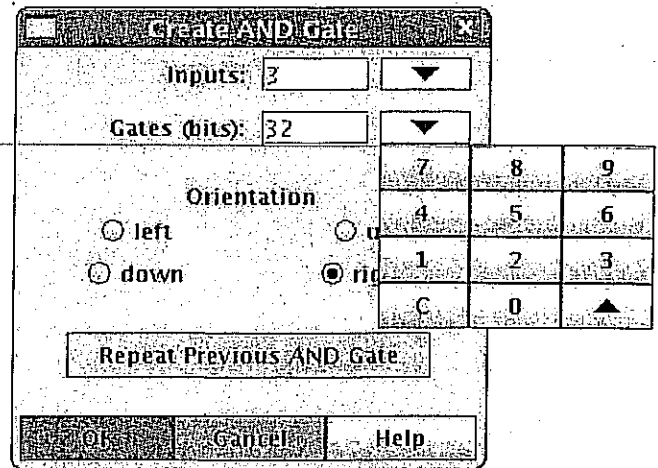


Figure 2: Gate creation dialog (32 3-input AND gates).

point along that wire. Illegal connections are detected (for example, creating a wire loop) and displayed as the wire is being drawn and then not allowed.

The characteristics of an element already placed are displayed by simply moving the cursor over the element. Wire widths and type (i.e., tri-state or not) are also displayed when the cursor is over any part of the wire.

Single elements and/or wires can be dragged to new positions; wires attached to moving elements move with the elements. Collections of elements and wires can be selected by simply tracing a rectangle over them, then moved (dragged), cut, copied or deleted en-mass. As collections are being moved, overlap and possible connections are detected and reported. Cut or copied collections can be pasted at other places in the circuit being edited, or into other circuits that are currently open (JLS can have several circuits open at the same time, using tabs to select the currently visible, edited one).

3.2 Subcircuit Inclusion

Subcircuits are a powerful abstraction tool, and JLS supports the nesting of subcircuits to an arbitrary depth. Subcircuits are imported (i.e., a copy is made), not linked. Imported subcircuits can be opened for editing without leaving JLS or having to re-import modified versions. Imports can come from existing circuit files, or from circuits just created but not yet saved.

3.3 Truth Table Editor

Rather than specify complicated logic using many individual gates and complex wiring, a user can use the truth table editor to specify the behavior of the equivalent circuit. Truth tables with an arbitrary number of single-bit inputs and outputs can be specified. Both input and output don't-cares are supported.

Figure 3 shows the truth table for a full adder. The user enters the input and output signal names, then clicks on any value in the truth table to toggle among 0, 1 and don't-care. Inconsistent outputs when an input don't care is "proposed" are prevented. The effect of the equivalent logic is simulated rather than the actual gates and wiring being generated.

3.7 Batch Operation

JLS can be run in "batch" mode, in which a circuit is loaded and simulated to completion (or time limit) without the GUI appearing or any other user intervention. When simulation ends the final contents of all watched elements (registers, memories, output pins) are printed to the standard output. In the case of memory elements, only those "words" that changed during simulation are printed.

When a circuit is run in batch mode several parameter files may be specified:

- A signal generator file that will send the specified signals to the input pins of the loaded circuit during simulation.
- A parameter file in which element propagation delays can be changed, element watches can be set or reset, registers can be given initial values, and memories can be told to read their initial contents from specific files.

There are also command line flags for setting the simulation time limit and generating a printable image file of the circuit.

3.8 Locking Circuit Elements

It is often useful to be able to give students a partially completed circuit and have them finish it. However there may be parts of the circuit that the instructor doesn't want students to be able to modify. JLS supports a locking mechanism on individual elements enabling the instructor to specify those parts of the circuit they don't want modified. JLS then prohibits the modification of any properties or deletion of any locked element.

4. JLS VERSUS OTHER TOOLS

Burch [1] and Wolfe [6] contain excellent, although slightly dated, summaries of the existence and features of many free, GUI based circuit simulators. Most of those listed are platform specific (mainly Windows and Macintosh). A few are Java applets and hence restricted from loading and saving files, and generally have very restricted functionality. Two are Java application programs and hence available on all platforms supporting the JVM: LogicSim [5] and Logisim [1].

LogicSim has an intuitive circuit drawing mechanism and the ability to display current signal values. It also has a subcircuit (module) inclusion mechanism. However it only supports basic logic gates, flip-flops, and clock. Wires cannot be bundled. Simple inputs values come from switch and number-input elements. Outputs are displayed by LED and LCD-like elements. There is no concept of time and hence no propagation delays. There is no batch execution mechanism and hence little to assist with grading of students' circuits.

Logisim also has an easy-to-use drawing mechanism and the ability to display current signal values. It supports basic logic gates, tri-state gates, flip-flops, constant value sources, and has a subcircuit inclusion. There are no other complex elements, no wire bundling, no propagation delays and no batch execution capability.

The tools most comparable to JLS in functionality are LogicWorks 5 [2] and TkGate [3]. LogicWorks is a commercial product (i.e., not free) that runs on Windows and Mac platforms only. It has little support for grading purposes.

TkGate is free and designed to run in a Linux environment. It has a fairly intuitive drawing mechanism, but does not detect error conditions like wire width mismatches or directly connecting the outputs of two ordinary gates on the fly. Logic elements can overlap, which is confusing. It has a wide range of logic elements from simple gates to multiplexers, adders and memory, but no state machines or truth-table specifications. Documentation is available only from the tkgate web site, not built into the tool. There is no text-based output making batch grading difficult.

5. AVAILABILITY

JLS is a free Java (version 5) application program packaged in two jar files and can be obtained from the author by sending an email request. It has been thoroughly tested on Windows and Linux based platforms. It has been tested to a lesser extent on Sun-Solaris and Apple Macintosh-OS X. It will not work with one-button mice.

An applet version of JLS exists at

www.cs.mtu.edu/~pop/jlsp/bin/JLS.html

A simple tutorial accompanies the web page that leads new users through the basics of constructing and simulating three increasingly complex circuits: a simple combinational circuit for the boolean expression $A \oplus B$, a 4-bit counter constructed from a register, adder and clock, and a full adder that imports half adder subcircuits. All the drawing and simulation features work as in the application program version. However, since it is an applet, existing circuits can not be loaded and user-drawn circuits cannot be saved.

6. ACKNOWLEDGMENTS

JLS began as a student project many years ago after several failed attempts. Eric Simonton designed the first practical GUI version that generated a textual representation that was then simulated with a separate tool. His simple circuit drawing mechanism, slightly modified, is the basis for the current version. Eric Dalquist, Benny Evans and Jonathan Even, in a senior software engineering project, added a subcircuit inclusion mechanism and an integrated time-based simulator, thereby avoiding the extra steps required to use a separate tool and the ability to query signal and memory element values during, and not just at the end of, circuit simulation.

7. REFERENCES

- [1] C. Burch. Logisim: a graphical system for logic circuit design and simulation. *J. Educ. Resour. Comput.*, 2(1):5-16, 2002.
- [2] Capilano Computing. *LogicWorks 5 Interactive Software*. Prentice Hall, 2003.
- [3] J. Hansen, 2006. <http://www.tkgate.org>.
- [4] D. Patterson and J. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, third edition, 2005.
- [5] A. Tetzl, 2006. http://www.tetzl.de/java_logic_simulator.html.
- [6] G. S. Wolfe, W. Yurcik, H. Osborne, and M. A. Holliday. Teaching computer organization/architecture with limited resources using simulators. In *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 176-180, New York, NY, USA, 2002. ACM Press.