Using DLSim 3: A Scalable, Extensible, Multi-level Logic Simulator

Richard M. Salter and John L. Donaldson
Computer Science Department
Oberlin College
Oberlin, OH 44074
rms@cs.oberlin.edu, john.donaldson@oberlin.edu

ABSTRACT

Students of Computer Organization should be able to "learn by doing" at all levels of computer design. *DLSim 3* is a multilevel simulation system that provides a unified platform for studying system structure, from low level combinational and sequential circuits, through design of a complete CPU. Using DLSim 3, students recognize the uniformity of system structure, as well as the principles of abstraction that link the various levels of design.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education; K.3.1 [**Computers and Education**]: Computer Uses in Education—*computer-assisted instruction*

General Terms

Design, Experimentation

Keywords

Computer Architecture, Logic Circuit Simulation, Abstraction

1. INTRODUCTION

DLSim 3 is a logical circuit simulation system designed with several novel features that allow students to experience simulated computer operation at various levels. DLSim 3's features include:

Three different levels of abstraction. These are *cards*, *chips,* and *plug-ins*. Cards are self-contained, fully editable DLSim circuits. Chips are derived from DLSim circuits but are opaque. Plug-ins are Java-based extensions and as such can provide a higher level of functionality.

Cohesive structure. Unlike other simulators that support circuit abstraction, a DLSim 3 circuit is presented as a single entity in a multi-level format that permits uniform access to all visible levels.

XML export DLSim 3 stores its data in both a binary and XML-based format. The XML file provides open access, permitting DLSim to serve, for example, as the design front end for larger applications [1].

ITiCSE'08, June 30–July 2, 2008, Madrid, Spain.
ACM 978-1-60558-115-6/08/06.

DLSim's abstraction capability provides two crucial factors necessary for such a simulation system to achieve the desired level of expressiveness:

- **Extensibility.** Abstract circuits extend the palette of basic building blocks used to construct larger, more complex circuits.
- **Scalability.** Abstract circuits permit designers to focus on a particular level of design, needing only to understand the functional behavior of lower levels and not their implementation.

2. DLSIM IN THE CLASSROOM

DLSim 3's features are designed to support active student learning. An instructor might supply a simple circuit (e.g., decoder, multiplexer, etc.) as a chip to demonstrate functionality without revealing design. Students would then be assigned the task of fleshing out the details.

Chips and plug-ins support top-down design. A high-level view of a complex circuit is built using chips and/or plug-ins for simpler components. One by one, they are replaced by cards (showing a subcircuit's structure with, perhaps, new chips and plug-ins) until the design is fully revealed.

Another important role for the plug-in is when scalability would otherwise prohibit the use of DLSim's GUI interface (e.g., a large scale random access memory). Finally, because they are written in Java, plug-ins can be enhanced beyond the plug-in interface. For example, a plug-in can perform user I/O, or interact with files, keyboard and screen. Plug-ins can also determine their appearance on the DLSim canvas, so that an ALU can "look" like an ALU.

DLSim 3 comes equipped with an API that simplifies plug-in development and encourages users to extend the basic tool set. A diverse plug-in library is currently under development and will be distributed with the next release. Compilers for translating circuits specified using VHDL or Verilog into plug-in source code are also under consideration.

3. REFERENCES

[1] M. J. Jipping, K. Ludewig, S. Henry, and L. Tableman. How to integrate FPGAs into a computer organization course. In *SIGCSE'06: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 234–238. ACM Special Interest Group on Computer Science Education, March 2006.