RaVi: Interactive visualization of information system dynamics using a Java-based schematic editor and simulator

Peter Marwedel, Khac Dung Cong, Sergej Schwenk*

I. Introduction

One of the key limitations of traditional media, e.g. books, for teaching is their lack of visualizing the dynamic behavior of systems. Videos tapes and video distribution techniques have made it possible to show non-interactive media elements to students. However, one of the key advantages of multimedia systems over traditional media is their potential of providing interactiveness. This interactiveness, however, is not easy to obtain, since it requires the simulation of the system to be visualized. Designing simulators can be a challenging task that can not be solved within the time-frame usually allocated for multi-media projects. On the other hand, available simulators are not suitable for classroom use. They are frequently designed for optimum simulation speed and complex design projects. Ease of use, excellent visualization and portability have normally not been top goals for simulator design. Also, powerful simulators are typically proprietary and come at high costs, preventing their widespread deployment to class rooms and into the hands of students. In the following, we will describe how this problem was solved within the RaVi project. RaVi stands for "Rechnerarchitektur-Visualisierung" (German for "computer architecture visualization"). RaVi is part of the larger SIMBA-project[1] [1].

II. Object-oriented modeling

Hendrich, with his design of the HADES system [2], has paved a way out. HADES is a Java-based class library comprising a schematic editor and a simulator kernel for simulating hardware structures. It supports the IEEE 1164 logic value system and deterministic simulation of clocked parallel hardware structures. The class library includes classes for common hardware elements. The schematic editor can be used to create instances of these classes. Hence, each hardware component is an instance of the corresponding Java class representing its type. This is a nice example of exploiting object orientation in hardware design. Each instance comes with methods for simulating its behaviour as well as with methods for displaying itself on the screen and for showing and modifying its properties. Instances can be connected using drap and drop.

III. Multimedia components designed in RaVi

RaVi's objective is to create interest in computer architecture among students which traditionally are not very keen to study computer engineering, such as many women. RaVi's approach is to provide extensive motivational material and to visualize the dynamic behaviour that is found in computer architectures.

As a first example, we visualized dynamics within a microcoded version of the MIPS-processor, as described in the popular book by Hennessy/Patterson [3]. By default, HADES uses color to visualize signal values. Unfortunately, in each clock step many signals change their values in the actual architecture. However, many of these are redundant since they are not loaded into any memory element during that clock step. More precisely, many line segments used in the architecture schematics are not used to carry relevant information. In order to avoid confusing changes of the colors of almost all line segments, an algorithm had to be developed which identifies those line segments that carry relevant information. Starting from memory inputs that are enabled for a certain clock step, paths to selected memory outputs are found. Line segments on selected paths are marked for visualization. All other line segments will be displayed with a light color, representing an undefined value. This algorithm facilitates understanding the dynamic behaviour of architectures.

As a second example, we have implemented the MIPS-pipeline described in the same book. Current work includes visualizing multi-processor cache coherency protocols, such as the MESI-protocol. Future work will comprise the visualization of the dynamic instruction scheduling techniques found in high speed processors, such as score-boarding.

In all of the cases mentioned, it is possible to interactively modify memory contents and to change interconnections between components. Using non-predefined components requires specifying their behaviour in Java.

IV. Results

In RaVi, we were able to reduce the effort for generating visualization components for computer architecture classroom teaching. We found that the techniques employed in HADES extend beyond the scope of computer architecture, since the component symbols used for the schematic as well as their behaviors can be freely defined in Java.

*The authors are with the Dept. of Computer Science, University of Dortmund, Germany. E-mail: Peter.Marwedel@udo.edu

[1]We gratefully acknowledge the funding of the SIMBA-project by the German ministry of research and development (BMBF).

References

[1] S. Schubert et al., "SIMBA home page", *http://ddi.cs.uni-dortmund.de/simba*

[2] N. Hendrich, "A Java-based Framework for Simulation and Teaching", *Proceedings of the 3rd European Workshop on Microelectronics Education*, EWME 2000, Kluwer Academic Publishers, p. 285-288

[3] J. L. Hennessy and D. A. Patterson, "Computer Organization – The Hardware/Software Interface", *Morgan Kaufmann Publishers Inc.*, 1995