An Interactive, Visual Simulator for the DLX Pipeline [1]

Yinong Zhang and George B. Adams III
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana 47907

Abstract

We have built an interactive, visual pipeline simulator, called *dlxview*, for the DLX instruction set and pipeline described in *Computer Architecture A Quantitative Approach* by Hennessy and Patterson [1]. This software provides animated versions of key figures and tables from the text and allows the user to readily follow details of pipeline activity as a code is simulated, to vary pipeline implementation, and to compare performance across different pipeline designs.

The software package requires a system running Unix and X11, with Tcl/Tk installed, and using the GNU gcc compiler is recommended. A 256 color display with 1024x768 pixels is best for display, due to the detailed diagrams of the DLX pipeline. The software has been designed to run on a variety of platforms and has been tested on Solaris 2.3, SunOS 4.1.1, HP-UX 9.0, DEC OSF/1 4.0, and Linux kernel 1.2.1. DLXview is available at `http://yara.ecn.purdue.edu/~teamaaa/dlxview/`

1 Introduction

We have built an interactive, visual pipeline simulator, *dlxview*, for the DLX instruction set and pipeline described in *Computer Architecture A Quantitative Approach* by Hennessy and Patterson [1]. This software provides animated versions of key figures and tables from the text and allows the user to readily follow details of pipeline activity as a code is simulated, to vary pipeline implementation, and to compare performance across different pipeline designs. This complements the text and allows the user to explore DLX behavior more rapidly and extensively than possible otherwise.

DLXview is one component of a larger project in computer system education, the *CASLE* project, standing for Compiler/Architecture Simulation for Learning and Experimenting. CASLE provides an HTML forms-interface tool set that automatically produces an optimizing compiler, assembler, and architectural simulator based upon user specifications. Thus, users can experiment with the total-system effect of, say, changing the number of registers, instruction latencies, and compiler optimizations used, exploring the performance issues spanning compiler to hardware. CASLE software requires a system with a good C compiler (e.g., gcc) and that is capable of running the CASLE CGI scripts that constitute the user interface, for example, a UNIX workstation with httpd or PC hardware running Linux. More information is available at `http://purcell.ecn.purdue.edu/~casle/` .

2 DLXview Description

DLXview is based on *DLXsim*, a basic DLX pipeline simulator that is part of the software distribution supporting the Hennessy and Patterson text. Major dlxview capabilities include: step forward

[1]Supported by NSF Grant No. CDA-9312649.

or backward by clock cycle, step forward or backward by instruction, and run.

Three versions of the DLX pipeline may be studied: basic, scoreboard, and Tomasulo. For each version, parameters such as the number of function units, their latencies, and whether they are pipelined can be varied. Memory load and store latency can be controlled as well. The graphical depiction of the DLX pipeline is customized to reflect the specific DLX configuration chosen. For example, if the user chooses Tomasulo with three floating point multiplier reservation stations and no floating point divide unit, then the animated depiction of the CPU will show exactly that.

Once the user has selected a DLX configuration and a code sequence along with any input data and register value initialization, simulation and animation of the pipeline depiction may begin. Simulation can start with any instruction in the code; the default is the beginning. The code appears in a window with the memory address, machine language, assembly code, and comments shown for each line of the program; the active instruction is highlighted. The user clicks buttons to move through time in the simulation, make adjustments, or exit. As the user proceeds interactively the pipeline display presents detailed information relevant to the program and particular pipeline configuration. Each instruction and animation related to it are displayed in a unique color. The displays closely follow the relevant figures and tables from the Hennessy and Patterson text. Figure 1 shows an example of the screen presentation for the basic DLX pipeline.

The basic pipeline animation includes the following information: (1) the traditional pipeline time line table showing clock cycles, the instructions in the pipeline and their progress over time, and any stall cycles; (2) the pipeline schematic of either the integer data paths or the floating point data paths with active stages colored to match their resident instructions; and (3) for integer instructions, the specific active data paths (including forwarding) highlighted and color-coded, with the names of active registers appearing in the register file block.

The scoreboarding animation includes: (1) a scrolling table of instruction status with the clock cycle at which issue, read operands, execution complete, and write result occur, (2) highlighted active scoreboard entries, register file entries, data paths, and function units, (3) scoreboard with contents, and (4) current instruction status or bookkeeping for each scoreboard entry with bookkeeping showing the specific Boolean expressions controlling that entry.

The Tomasulo animation shows and makes available a like amount of information.

3 DLXview Additional Capabilities

DLXview provides a number of capabilities beyond those necessary to a basic interactive simulation of a given code to facilitate exploration.

During a simulation session, the configuration of the pipeline can be checked and can be changed by first aborting the current state.

Once configured, assembly code, data, and initial register values for the pipeline may be loaded from files. The output of dlxcc may be used as input for simulation or assembly may be written by hand. A register initialization file is valuable for situations such as examining the execution of a loop in which some instructions have pre-specified operands, and to avoid starting the interesting portion of a code segment at other than clock cycle one due to cycles spent on instructions to initialize register values.

At any time during a simulation session the DLX assembly code may be edited in a pop-up window. When editing is complete, the new file can be immediately loaded to start a new simulation. New data may also be loaded at this time, or data file used currently in use is loaded.

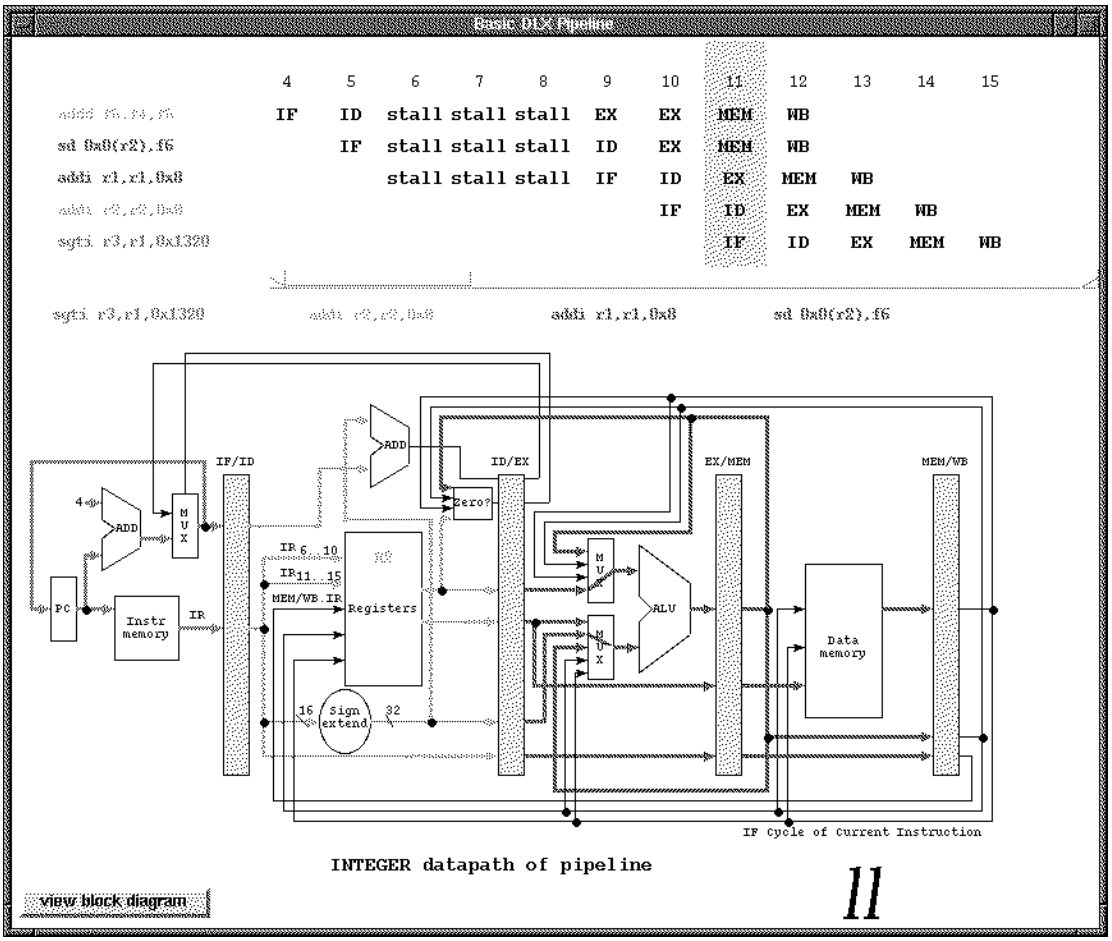

Figure 1: Example screen display (dithered black and white rather than color)

If no instruction has entered the pipeline yet, the *step forward* function request the starting address. Otherwise, simulation resumes from where previously stopped to execute the next instruction. The clock cycle count will advance to that when the next instruction reaches the first pipeline stage. The *next cycle* function will also request the starting address if no instruction has yet entered the pipeline. Otherwise, simulation advances by one clock cycle. Using next cycle continuously will show every detail in the simulated pipeline.

The *go* function begins simulation and continues until the execution terminates naturally via a `trap` instruction.

The *step back* function allows the user to return to the pipeline state of the previous instruction. Stepping forward and back supports close examination of the state changes taking place throughout the pipeline. The *previous cycle* function allows single cycle retreats in time.

The *trace* function allows for output of an instruction trace or memory reference trace. Traces are in a format suitable for available cache simulators such as tycho and dineroIII [2]. Trace collection can begin at any time during a simulation session and will start from the current instruction. The trace file is not available until the current simulation session ends; i.e., the simulation stops naturally or is interrupted by a reset command from the user. To avoid producing a garbled trace,

the tracing process is stopped automatically when either the step back or previous cycle functions are used.

4 Conclusions

We have provided a brief introduction to the capabilities and system requirements of dlxview, an interactive, configurable, highly detailed, visual simulator for the DLX instruction set and DLX pipeline variants, including scoreboard and Tomasulo. The software package requires a system running Unix and X11, with Tcl/Tk installed, and using the GNU gcc compiler is recommended. It has been designed to run on a variety of platforms and has been tested on Solaris 2.3, SunOS 4.1.1, HP-UX 9.0, DEC OSF/1 4.0, and Linux kernel 1.2.1.

Further development will be guided by feedback from users. More information and a brief tour of a Tomasulo simulation is available at `http://yara.ecn.purdue.edu/~teamaaa/dlxview/`

References

[1] J. L. Hennessy and D. A. Patterson. *Computer Architecture A Quantitative Approach.* Morgan Kaufmann Publishers, Inc., 1990 and 1996 (either edition).

[2] Mark D. Hill and Alan Jay Smith. Evaluating Associativity in CPU Caches. *IEEE Transactions on Computers*, C-38:1612–1630, Dec. 1989.