An interactive environment for the teaching of Computer Architecture

P. S. Coe, L. M. Williams and R. N. Ibbett
Computer Systems Group
Department of Computer Science
University of Edinburgh
Edinburgh, EH9 3JZ, UK
{psc, lmw, rni}@dcs.ed.ac.uk

Abstract

The operation of a computer can be conceptualised as a large, discrete and constantly changing set of state information. However even for the simplest uni-processor the data set of such a model is very large and is subject to rapid change. We show how HASE provides an interactive animated display of a given computer system's components allowing visualisation of the data set (and consequently the mechanisms employed by computer architects when designing a computer system). Simulation models of both the DLX and DASH architectures are discussed, demonstrating HASE's suitability as a teaching tool for computer architecture students.

1 The problem

In trying to understand the detailed operation of a computer, the computer science student must consider the constantly changing state of its individual components. One natural way to represent this state information is by means of a set of diagrams. However even for the simplest uni-processor the set of data to be conveyed by such means is very large (memory contents, cache contents and registers for example) and is subject to rapid change.

This presents the scientist with a problem - how to represent a large, ever changing set of state information with a sufficiently fine time resolution to be useful. Given these requirements it is natural to look for some form of tool to help with this data management job.

2 HASE

At the University of Edinburgh Computer Science Department a high level GUI based tool suitable for the management of such information exists in the form of HASE [1] (**H**ierarchical **A**rchitecture design and **S**imulation **E**nvironment).

HASE allows for the rapid development and exploration of computer architectures at multiple levels of abstraction, encompassing both hardware and software. The user interacts with HASE via an X-Windows/Motif graphical interface, and one of the main forms of output is an animation of the design window.

As the components of a computer can be treated very naturally as objects, HASE itself is based on the object oriented simulation language Sim++ [2] and an object oriented database management system, ObjectStore [3].

The environment includes a design editor and object libraries appropriate to each level of abstraction in the hierarchy, plus instrumentation facilities to assist in the validation of the model. The overall structure of the HASE environment is shown in Figure 1. The system can thus be set up to return event traces and statistics which provide information about, for example, synchronisation, communication, memory latencies and cache hit/miss ratios.

Although originally aimed at the computer architect's requirement for a high level design and simulation environment, many of HASE's features lend themselves well to the teaching of computer

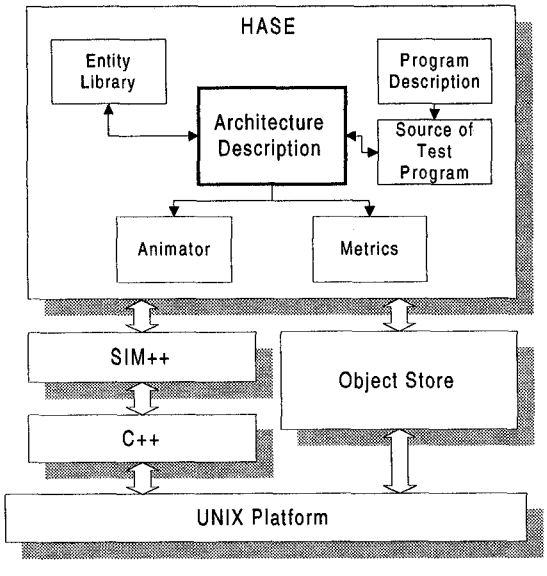

Figure 1 HASE System Overview

architecture principles (for example pipelining, cache coherency, microprocessor operation etc.). In order to extend HASE's suitability as a teaching/demonstration aid extensions to the HASE environment have been made. The main extensions are:

1. Support for the real-time annotation of an animation.

Integrating Tech. into C.S.E. 6/96 Barcelona, Spain
© 1996 ACM 0-89791-844-4/96/0009...$3.50

2. A set of restrictive mouse-bindings which prevent the end-user of a demonstration system from accessing HASE's low-level design functions.

3. A method by which the end-user can easily change the initial contents of memories and registers.

3 Typical simulations

At present various simulations are available which demonstrate many important architectural principles/techniques. Two such simulations are discussed below.

3.1 The DLX simulation

The DLX architecture is a generic RISC architecture described by Hennessy and Patterson [4]. The reasons why this architecture was chosen to be simulated are:

1 It is a simple architecture with a small instruction set.

2 It is free of the individual features of commercially available architectures which are included to improve performance, leaving the student to concentrate on some of the principles of computer architecture.

The DLX simulation model (as described in [5]) attempts to highlight several fundamental concepts of computer architecture along with a couple of more complex ones. The concepts concentrated on were pipelined execution of instructions (including a technique called *scoreboarding*), parallel arithmetic units and the effects of different branching policies.

Before viewing the simulation the student has to select one of the three available branching policies and the type of description they require (Figure 2 shows this dialog).

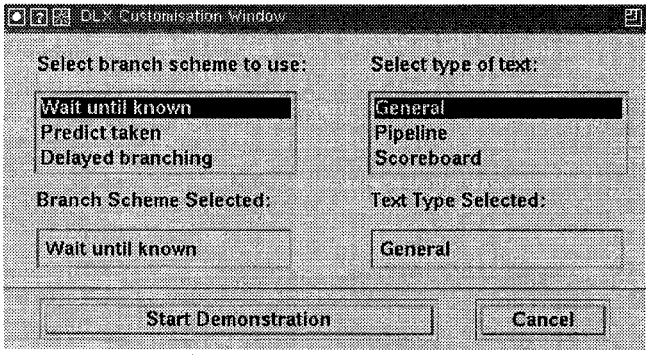

Figure 2 Typical HASE Dialog Window

The different types of description annotate the animation from different perspectives, for example one gives a general description of what is happening within the architecture at each stage and another attempts to describe the operations of the pipeline in more detail. During the simulation the student is able to display the contents of the memory, the register banks and the instructions queued in any of the pipeline stages, allowing the state of the architecture to be seen at any stage of the simulation.

A small piece of code (which comes in three forms, one for each of the branching policies) is supplied, which includes all relevant dependencies between instructions and sequences of instructions to illustrate the necessary principles. A set of textual annotations is also supplied with this code.

3.2 The Stanford DASH simulation

The DASH architecture [6] was built in the Computer Systems Laboratory at Stanford University. The main motivation underlying its inception was a desire to prove the feasibility of building a scaleable high performance machine with multiple coherent caches and a single address space.

The DASH hardware is organised hierarchically as follows. At the bottom of the architectural hierarchy is a set of processing nodes, grouped together in clusters of four and connected together via a common bus (the lower level interconnection mechanism). These buses are in turn connected together by a (dual) interconnection network (the higher level interconnection mechanism).

The rationale behind presenting this architecture as a demonstration model was as follows:

1 The problems outlined above for describing the operation of a uni-processor are further exacerbated in the case of a multi-processor and thus a visual description of such a system is invaluable when explaining its operation.

2 The Stanford DASH system offers a good platform with which to highlight the problems associated with the well known multi-processor multi-cache coherency problem.

The DASH simulation [7] takes advantage of HASE's abstraction facilities by structuring the simulation model at three distinct hierarchical levels.

At the lowest level of abstraction individual memory requests can be seen travelling between the low level simulation entities such as processors, buses, caches and memory units (these requests are presented as a series of animated icons travelling along links connecting simulation entities). At this low level it is possible to display the cache memory contents and examine the setting of, say, valid or shared bits.

The medium level of abstraction shows some of the low level entities being grouped into composite entities representing higher level architectural constructs. For example the processor and its associated caches are grouped together into the processing node entity. This technique is useful in providing a mechanism for 'filtering' the amount of on-screen data presented to a user. At this medium level the filtering proves useful in allowing the user to examine the cluster's bus arbitration policy (only messages between a processing node and the bus are seen rather than all the related cache hit/miss information present at the lower level). Finally the highest level of abstraction deals with the interconnection of DASH clusters. At this level each cluster is represented by a single icon. This provides a convenient method for examining inter-cluster transactions.

One of HASE's most powerful features is to allow the different levels of abstraction to be combined. This proves useful in examining how, say, the intra-cluster cache-coherency protocol (visible at the lowest level of simulation) interacts with the inter-cluster coherency mechanisms (visible at the higher level).

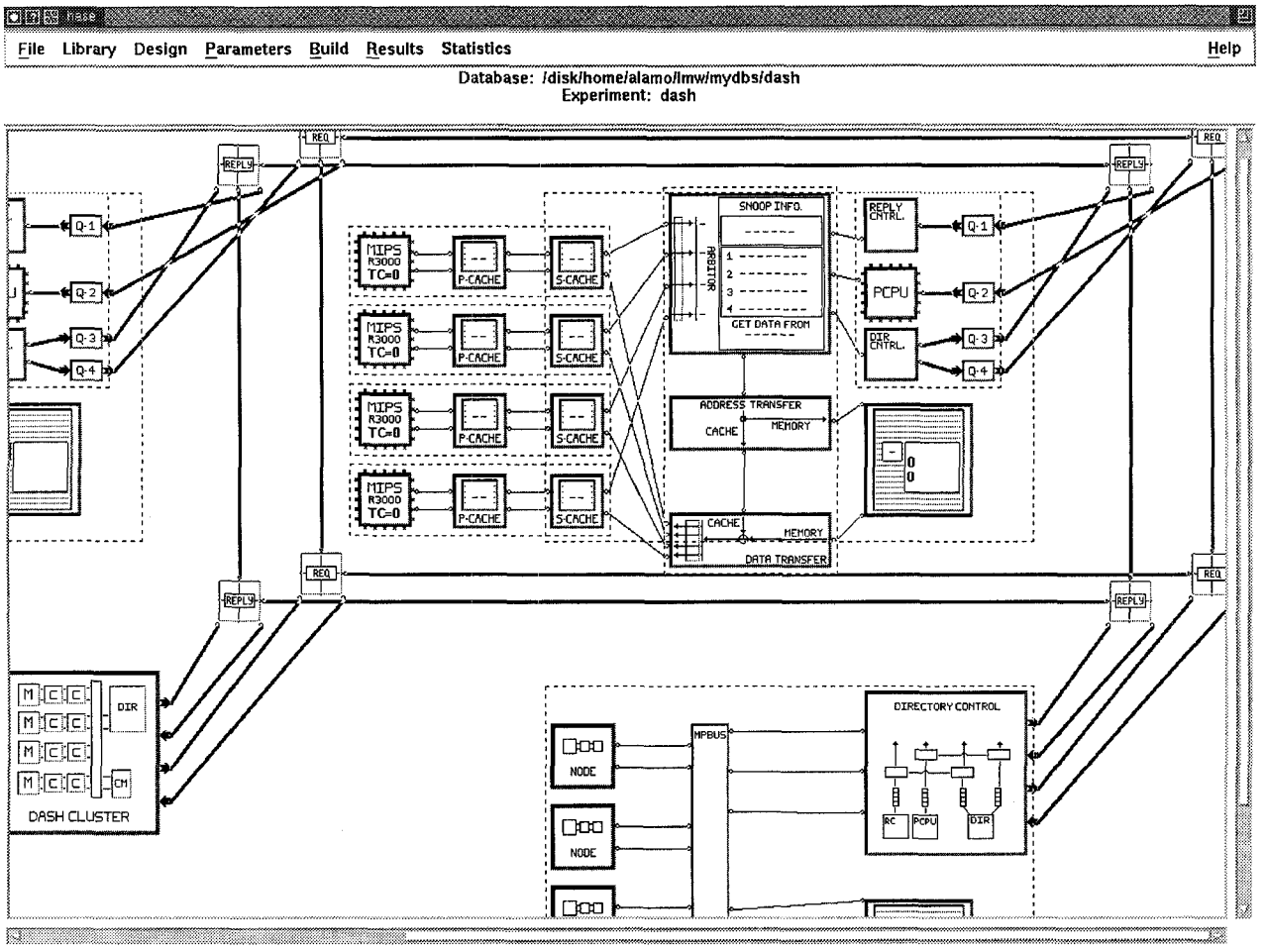

Figure 3 Screenshot of DASH Simulation

A typical screenshot from the DASH simulation showing the simultaneous use of (four) clusters at various levels of abstraction is shown in Figure 3.

4 Conclusions

We have shown that HASE provides an effective environment for the simulation and exploration of various types of computer architecture.

The animation facilities found in HASE prove useful for allowing the computer science student to visualise the mechanisms employed by computer architects when designing a computer system. This visual insight is complemented by the ability to filter out extraneous on-screen data by presenting architectures at multiple levels of abstraction. Furthermore animations can be textually annotated to draw the attention of the student to salient features of the architecture.

By combining these attributes of the HASE display with the ability to parameterise the animation taking place on-screen, the student may interactively experiment with the architectural model. For example this allows performance trade-offs to be examined.

References

1 R. Ibbett, P. Heywood and F. Howell. HASE: A Flexible Toolset for Computer Architects. *(to appear) The Computer Journal,* 1996.

2 Jade Simulations Inc., Sim++ Reference Manual.

3 ObjectStore Release 3.0, User Guide, Object Design Inc., Burlington

4 J. Hennessy and D. Patterson. *Computer Architecture, A Quantitative Approach.*, Morgan Kaufmann Publishers, Inc., 1990.

5 Paul Coe, Simulation of the DLX Architecture, University of Edinburgh 4th Year Project Report, June, 1995.

6 Daniel E. Lenoski, The Design and Analysis of DASH: A Scaleable Directory-Based Multi-processor, *TR:CSL-TR-92-507* Computer Systems Laboratory, Stanford University, 1992.

7 L. Williams and R. N. Ibbett, Simulating the DASH Architecture in HASE, *(to appear) in Proc. 29th Annual Simulation Symposium*, April 1996.

Acknowledgements

The HASE project is supported by the UK EPSRC.