Hierarchical Architecture Design and Simulation Environment

F.W. Howell, R. Williams, R.N. Ibbett
Dept of Computer Science
University of Edinburgh
Edinburgh, Scotland, EH9 3JZ
email: fwh@dcs.ed.ac.uk

Abstract

The Hierarchical Architecture Design and Simulation Environment (HASE) is a tool for modelling and simulating computer architectures.

Using HASE, designers can create and explore architectural designs at different levels of abstraction through a graphical interface based on X-Windows/Motif and can view the results of the simulation through animation of the design drawings.

1 The Motivation

Advanced simulation tools are available for low level electronic design, such as Spice for analogue circuits, and VLSI layout tools. However, tools for rapid prototyping of architectural ideas are few and far between. Simulation languages (such as SIMULA, SIMSCRIPT, DEMOS etc) can be used to model computer architectures, but the user has to be an expert on simulation. This is also the problem of general purpose simulation tools (e.g. SES/Workbench), where icons represent 'queues', 'servers' etc., and the link between a queueing model of an architecture and the architecture itself is not immediately apparent to the engineer not fluent in queueing theory.

Conventional languages (C, C++, occam) are often used to construct simulators, but this approach involves starting from scratch for each new project. User interface aspects are often neglected as the tool will be thrown away with the next architecture. This is very wasteful, as many aspects of computers are constant between different architectures. The object oriented approach offers a solution. Standard components (such as memories, microprocessors and interconnection networks) can be held in a library. They can be constructed and linked together graphically on screen to create a simulation of an architecture, in much the same way that standard components can be wired together in a semi-custom VLSI tool. The difference is that the simulation is not fixed to low level wires and chip pins, but is free to choose the appropriate abstraction level.

HASE addresses one of the four "Grand Challenges in Computer Architecture" identified by the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing [1]:

> *... to develop sufficient infrastructure to allow rapid prototyping of hardware ideas and the associated software in a way that permits realistic evaluation.*

Any tool which attempts to bridge the hardware/software gap must provide models which are as understandable to the compiler writers working on instruction scheduling as they are to the computer engineers. Hierarchy is a fundamental concept in managing complexity for both hardware and software, so the simulation tool should be hierarchical to allow detailed simulations in areas of interest, and higher levels for the other parts of the architecture. This also helps to balance accuracy with fast simulation time. Computer engineering is concerned with cost as much as performance, so simulation models should be able to incorporate cost and other factors such as power consumption.

Complexity of parallel systems has led to the development of "performance tuning" as a step in parallel program development. However, we believe that such "post-mortem" analysis is too late, and that "pre-natal" parallel program design based on a simulation modelling tool is a better solution. The "right first time" approach of VLSI design (and Total Quality Management) should be extended to cover parallel software.

Some simulation tools employ "direct execution" to evaluate performance of parallel programs on parallel architectures (e.g. MIT/Proteus [11], Stanford/Tango [12], WWT [13]). This is an excellent approach for obtaining fast, realistic simulations if the processor you're running the simulation on is very similar to the processor used in the parallel machine - but we believe that this assumption is sometimes too restrictive and a hierarchical tool can find wider application.

2 The Tool

The HASE tool is based on SIM++, a discrete event simulation language built as an extension to C++. SIM++ is used to describe the behaviour of basic components of a simulation. The user can link together these components on screen, and HASE produces the SIM++ initialisation code necessary for simulating the network. New components can be constructed by linking together standard components. Each component can be simulated at any level of abstraction. A register transfer level simulation will produce the most accurate simulation results; behavioural level simulations run more swiftly. The tool allows different parts of the simulation to run at different abstraction levels, so the user can 'zoom in' to specific parts of the design to simulate that at a low abstraction level, and run the rest of the design at a high level of abstraction.

HASE is more flexible than many parallel architecture simulators, and can model any parallel system, not just a restricted variety of MIMD/SIMD machines. The use of SIM++ allows high performance distributed simulation on networks, using a Timewarp algorithm based on Virtual Time [10], [5]. The hierarchy also aids development of fast simulations, as typically only some of the components need to be simulated at a low level. The object-oriented simulation language makes it easy to construct reusable components, and a library tool lets users share components with each other.

Once a simulation has been constructed, users need the answers to many performance questions quickly. HASE incorporates automatic graphical results analysis, and the hierarchy is used to get detailed or global performance summaries. The design can be animated at any level, with the state of each component changing on screen, and data packets moving down links. This can highlight bottlenecks, and is an excellent presentation aid. Design is an iterative process, so once changes have been made the simulation can be run again rapidly. A hierarchical timing diagram display of the states of each component can also be displayed, and logic-analyser style measurements can be taken. There are also facilities for overall performance statistics such as total cost, average performance, and power dissipation.

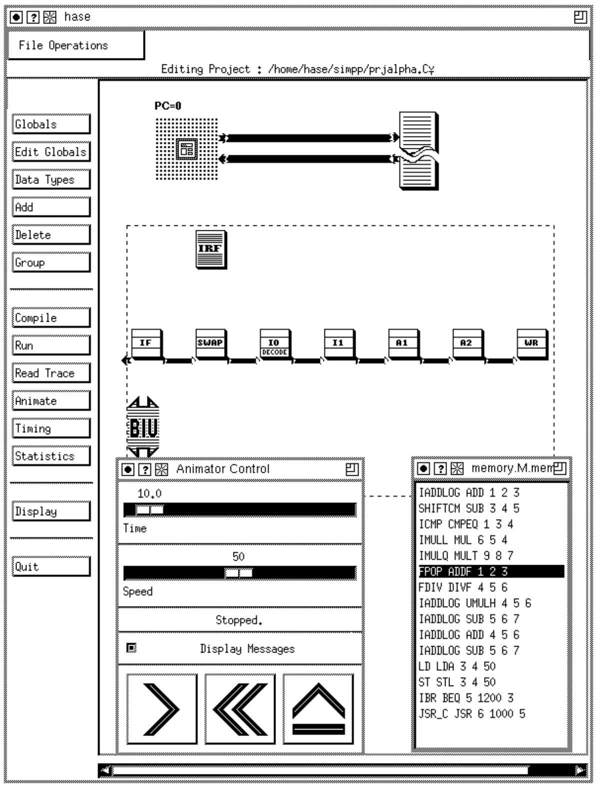

Figure 1: The HASE user interface.

HASE incorporates significant support for simulating single processors. Once a new assembly code has been defined, a parser is automatically linked in with the simulation, so that "instruction interpreting" objects can be written very easily. All "memory objects" are initialised at simulation run time with the appropriate program, fully parsed. These programs can be edited on screen, and whilst the simulation is running, the currently executing instruction can be highlighted. At higher levels, the overhead of interpreting instructions can be too great, so programs can be modelled as delays interspersed with communications events. Again parsers for the higher level language (for example consisting of `COMPUTE <time>`, `SEND <proc#>`, `RECV <proc#>`) can be generated automatically.

Simulation is particularly useful for parallel computers. Replication is a key feature of parallel systems, so HASE includes templates for the common structures. The user can slot any component into the template (including hierarchical components), and HASE does the rest. Current templates include a linear array, a 2D mesh, and an omega network (figure 2). More multistage networks and general k-ary n-cubes are under development. All have their size as a parameter, which can be set by moving a slider widget. Connections between neighbouring processors can be specified.

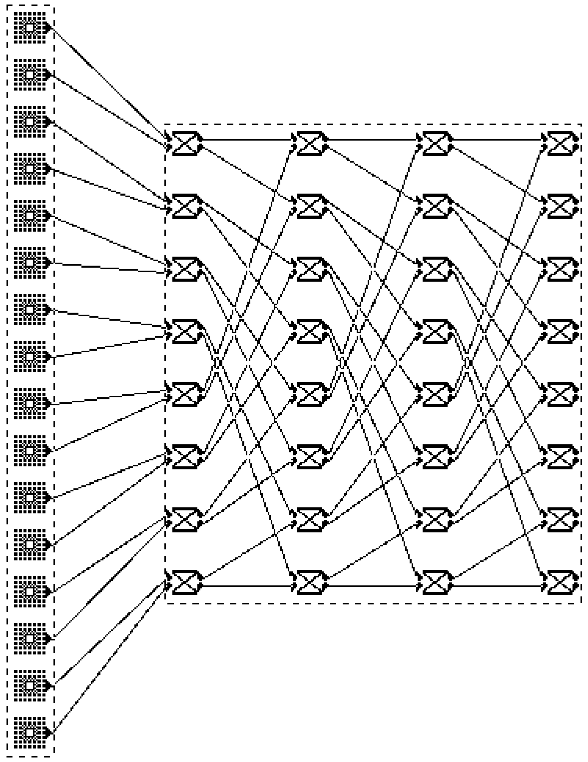

Figure 2: A HASE omega network.

In addition to simulating architectures, HASE can be applied to simulating parallel algorithms. The graphical interface and results analysis built into HASE can be used to visualise and evaluate parallel algorithms. Work has been done on process simulation for message passing systems, based on the Edinburgh Parallel Computer Centre (EPCC) CHIMP interface for portable message passing algorithms [9]. Work has also been done on routing algorithms for meshes and on evaluating multiprocessor WAN routers and LAN bridges [3].

3 Applications

3.1 DEC Alpha 21064 AXP microprocessor

1. At the detailed pipeline level, showing the movement of instructions through the functional units, blocking and exceptions (figure 1).
2. At the next level up, the instruction level. This provides a working 'instruction interpretting' simulation, with times for each instruction in clock cycles extracted from data sheets.
3. At the PMS level. Instructions are no longer interpreted - instead times for each process are modelled by delays, with measurements of total instruction counts, cache/main memory references, and I/O.

3.2 DEC Alpha-PC AXP

This demonstrates the use of HASE to evaluate complete systems. Cost/performance charts can be obtained. Note that the processor in this full system simulation is the one produced above, illustrating the reuse of components. A different processor can be substituted (e.g. Pentium) for performance comparisons.

3.3 Parallel Systems

This illustrates the HASE facilities for simulating parallel architectures. Processors and interconnection networks can be swapped in and out and performance of the resulting system can be measured.

Various direct and indirect networks are used to connect the processing elements. These include omega and mesh structures. The effects of varying processor and interconnect performance on parallel algorithms can be measured.

3.4 Parallel Software

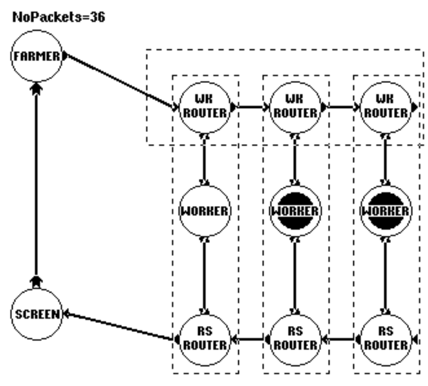

Figure 3: A task farm.

At the next level up are the applications. The demonstration shows the use of nested algorithmic skeletons [8] for developing parallel code.

4 HASE development

HASE has been developed by the Computer Systems Group at Edinburgh University. An advanced prototype using DEMOS was developed, and most of the ideas were tested on a multi level simulation of the Motorola 88110 microprocessor [2]. Recent work has converted HASE to the language SIM++ to allow faster simulations on networks of workstations. The current version of HASE runs on a Sun SPARCstation with X-windows and makes use of the commercial simulation language SIM++.

Currently work is in progress to include provision for evaluation of architectural support for PRAM and HPRAM computational models [6], and to evaluate process mapping strategies. Several PhD and MSc projects are using HASE as a simulation support tool. There are plans to make HASE freely available to the academic community.

References

[1] H.J. Siegel, S. Abraham, *et al "Report of the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing"*, Journal of Parallel and Distributed Computing, 16, pp. 199-211, 1992.

[2] A.R. Robertson & R.N. Ibbett *"Simulation of the MC88000 Microprocessor System on a Transputer Network"*, Lecture Notes in Computer Science, Distributed Memory Computing, 2nd European Conference, EDMCC2, April 1991, Springer-Verlag.

[3] J.S. Westerman *"Parallel Processing for a Communications Node"*, M.Sc Dissertation, Univ. of Edinburgh Dept. of Computer Science, September 1993

[4] J. Birtwistle *"DEMOS: Discrete Event Modelling On Simula"*, Prentice-Hall, 1985

[5] D.R. Jefferson *"Virtual Time"* ACM Transactions on Programming, 7(3),(July 1985) pp404-425.

[6] T. Heywood & S. Ranka *"A Practical Hierarchical Model of Parallel Computation I: The Model"*, Journal of Parallel and Distributed Computing, 16, pp. 212-232, 1992.

[7] *"A Workbench for Computer Architects"*, IEEE Design & Test of Computers, Feb, 1988.

[8] M.I. Cole *"Algorithmic Skeletons: Structured Management of Parallel Computation"*, Pitman & MIT Press, 1989.

[9] J.G. Mills, Lyndon J. Clarke, Arthur S. Trew *"CHIMP Concepts"*, Technical Report EPCC-KTP-CHIMP-CONC, Edinburgh Parallel Computing Centre, April 1991.

[10] Jade Simulations Inc. *"SIM++ v3.8 Reference Manual"*.

[11] E.A. Brewer, C.N. Dellarocas, A.Colbrook, W.E.Weihl *"PROTEUS: A High Performance Parallel-Architecture Simulator"*, Technical Report MIT/LCS/TR-516, MIT Laboratory for Computer Science, September 1991

[12] Helen Davis, Stephen R. Goldschmidt, and John Hennessy. *"Multiprocessor Simulation and Tracing Using Tango"*, Proc. 1991 ICPP, August 1991

[13] S.K. Reinhardt, M.D. Hill et al. *"The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers"*, Proc. 1993 ACM SIGMETRICS Conference, May 1993