HASE: An Environment for Hardware/Software Codesign

P.S. Coe, R.N. Ibbett, N. Rafferty & L.M. Williams

Technical Report CSG-41-98
Department of Computer Science
University of Edinburgh

March 17, 1998

Abstract: Hardware/software codesign has recently become an explicit topic in computer research circles, although it has been an implicit part of many projects for several decades. Tools are required to aid designers to follow codesign practices and methodologies, enabling software and hardware designers to work within a unified environment when creating a system. The Hierarchical Architecture design and Simulation Environment, HASE which allows rapid development and exploration of computer architectures encompassing both hardware and software, will be discussed as a tool to be used as a basis for such an environment. Example systems are also presented to highlight the features of HASE.

Introduction

Hardware/software codesign has recently become an explicit topic in computer research circles, although it has been an implicit part of many projects for several decades. Codesign is a concurrent approach to systems design and there are several key issues to be addressed, these are specification, partitioning, synthesis, simulation and design-space exploration. Most systems are designed by applying these issues separately to the design of the hardware and the software. Codesign methodologies and environments attempt to merge these design paths so that expensive mistakes are not made when assigning components to hardware and software. Due to the merging of design paths new environments and tool sets have to be designed to enable the specification, simulation, testing and synthesis of both hardware and software to be made simultaneously. This would enable designers to explore a wider range of possible tradeoffs when designing the system, which should result in better designs, more efficient implementations, better interfaces between hardware and software components and shorter design times.

This paper outlines the HASE system and some of its features that could provide a basis of a hardware software codesign environment. Some of the more recent projects carried out within HASE are discussed, emphasising many of its useful features when designing and evaluating systems. Possible developments and future extensions to the system are also presented.

Overview of HASE

The Hierarchical computer Architecture design and Simulation Environment (HASE) developed at the University of Edinburgh allows rapid development and exploration of computer architectures at multiple levels of abstraction, encompassing both hardware and software. The system was intended to be used as a tool for computer architecture designers to aid in the development and exploration of new architectures. Although other uses have been suggested, including, incorporating it into the Integrated Learning Support Environment (ILSE) which is an online teaching environment for computer architecture. HASE would enable animations of different architectural concepts, this work lead to the development of simjava [1], which allows simulations and animations to be included in web pages. The main features of the HASE environment include a Entity Description Language, EDL [2] for project data storage and libraries, a simulation engine, a hierarchical method of model representation, a visualisation mechanism and results gathering tools. Figure 1 shows the main components of the HASE system and how they interact with each other.

The EDL definitions of the architecture components provide information about architecture parameter definitions, component parameters and ports, hierarchical on-screen structure, global system parameters and component interconnect. Predefined multiprocessor topology templates are also included in EDL to aid in the rapid development and exploration of multiprocessor networks. This description when combined with the Entity Layout file containing display information completely describes the architecture to be simulated.

Once the architecture is loaded into HASE from an EDL file, the simulation executable can be generated by combining the architecture information and user defined parameters with the individual component behavioural descriptions. This simulation executable can then be executed with various input parameters specified in the architecture descriptions and any pieces of code loaded into memory components. There are two simulation engines used at present, SIM++ [3] and HASE++ [4], although additional languages can be easily

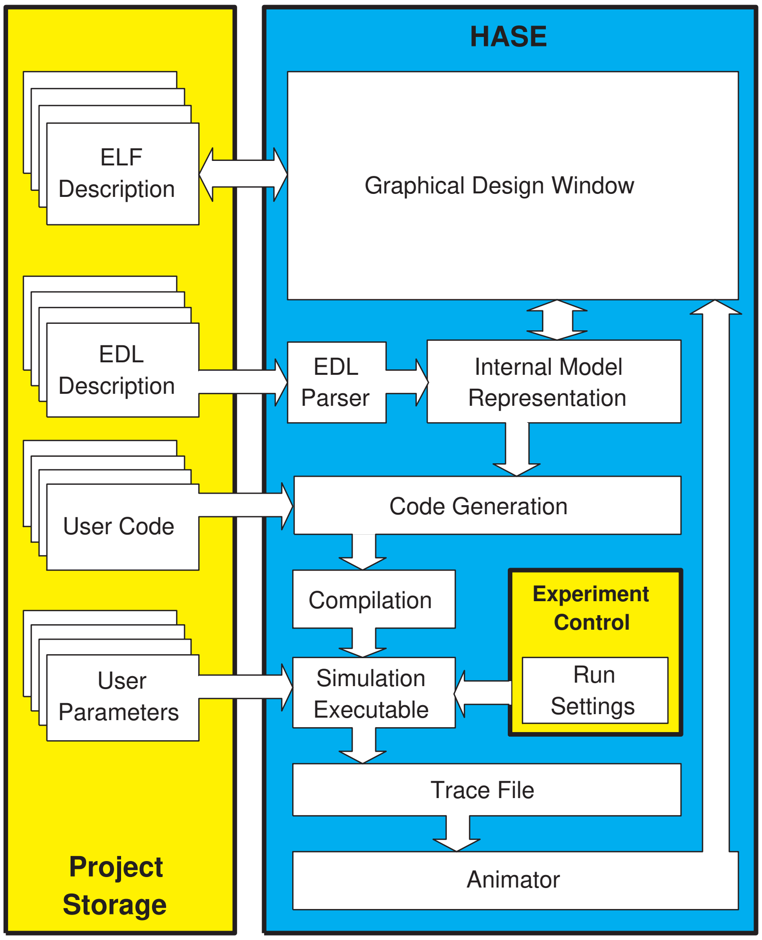

Figure 1: HASE Software Architecture

incorporated, for example, VHDL is a likely candidate for some future project.

The results gathering tools enable automatic, repeated execution of the simulation using different input parameter values and memory files and allow graphs of the components' output values over single or repeated simulation executions to be drawn.

The visualisation mechanism allows the graphical display of an architectural model to be animated by reading in trace files generated by the simulation execution, thus enabling the designer to inspect the model for correct operation. The hierarchical nature of HASE controls the displayed complexity of the simulation, according to the areas of the model being concentrated on. It is also possible to animate the simulation as it progresses, removing the need for a trace file. Although this slows down the simulation as it has to wait for the animation for each stage to complete before progressing, it does provide the designer with the ability to adjust architectural parameters while the simulation is in progress. Enabling the immediate effects of the parameter change to be monitored without the need to regenerate the entire trace file. Possible examples of interactive adjustments could include interactively removing nodes in a network to monitor fault tolerance, changing parameters of individual components, for example, cache speed to study the effects on performance and modifying contents of registers and memory components to create program breakpoints.

There are five modes of operation in HASE (1) Design (2) Model Validation (3) Build Simulation (4) Simulate System and (5) Experiment with system. Each mode provides the designer with a different set of menu options allowing different operations to be performed. The current active mode also determines the functionality of the menus attached to components, for example, when in Simulate System mode, memory components have an option to load new code. Figure 2 shows the main HASE window, containing a multiprocessor network shown at multiple levels of abstraction.

HASE and Hardware/Software Codesign

There are several features of HASE which make it a suitable system to be used for the exploration of a variety of different trade-offs within a codesign methodology.

Cosimulation, the ability to include hardware and software components of a system design in a simulation, is an important feature of HASE. The software components are usually included by loading the required code into a memory component, and then executing the code through a processor component. The generation of template machine description files for the GNU C/C++ compiler is currently being investigated. This would partially automate the process of software generation for experimental hardware, removing the need for the designer to continually write code in assembly language native to the hardware. The level of detail at which the processor and memory components are simulated affects the balance between the accuracy and speed of the simulation, as well as the complexity of the code. The ability of HASE to deal with these different levels of component abstraction is another useful feature of the environment.

The ability of HASE to simulate at a high level of design abstraction, i.e., before the partitioning stage, is a another useful feature for a codesign environment, as it allows a general system design to be created and simulated before deciding on particular implementations for individual components. The EDL provides a mechanism for specify components of a system, what parameters they possess and how they are connected together without assuming any implementation details. Once constructed, high level behavioural descriptions can be written for each of the system components in HASE++, again without assuming any implementation details. The animation facilities and results gathering tools aid the designer in identifying major design flaws and difficulties before continuing with the design

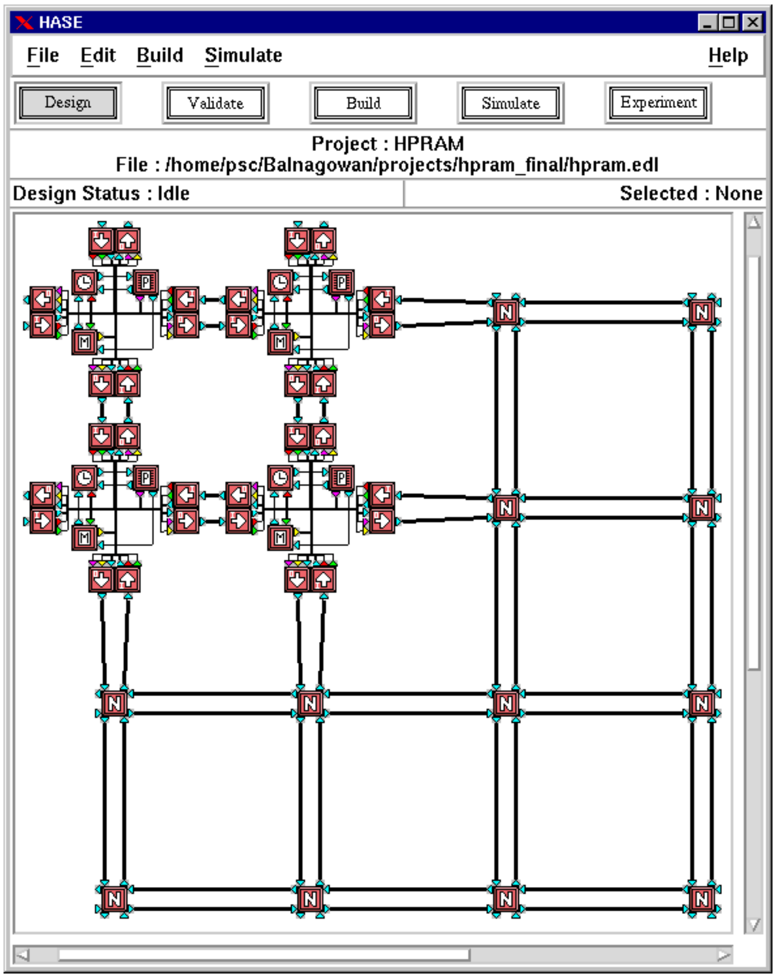

Figure 2: Main HASE Window

process.

The use of EDL as the system specification language allows external algorithms to be applied to the system specification, for example, model validation and hardware/software partitioning. There are many partitioning algorithms appearing [5, 6, 7], these could be evaluated by applying them to the EDL and them simulating the resulting system implementation in HASE, providing relevant hardware and software implementations of all the components available.

The large design space for a particular system can be explored through the use of parameterised components and component libraries. These parameterised components enable changes to the architecture design and/or hardware/software partition to be made quickly and easily. The ability to change and experiment with these parameters and monitor the effects whilst the simulation is running, provides the designer with instant feedback on these adjustments without the need to rerun the entire system simulation. There are features of the results gathering tools that also allow the design space to be explored quickly. In particular the ability to execute repeated simulation runs automatically, using a different input parameter value in each run and to automatically plot graphs of output parameters as functions of the varying inputs.

Applications of HASE

There is a wide range of applications for which HASE is a suitable design tool, e.g. individual components of a microprocessor, memory hierarchies and complete systems and networks that contain one or more processors.

Motorola M68HC08

A fully functional simulation of the Motorola M68HC08 micro controller unit has also been produced [8]. This can be used as a basis for demonstrating possible hardware/software trade-offs within a micro controller. Figure 3 shows the HASE architectural representation of two of these micro controllers connected together. The simulation addressed the following areas of the micro controller design (a) memory configuration (b) interrupt processing (c) instruction set execution and (d) asynchronous I/O processing. The simulation has been developed in modules starting with a basic CPU/Memory model and extending it to include the system integration module, I/O and external interrupts.

The architecture representation shown in Figure 3 illustrates several of the features of the HASE system. The two micro controllers are displayed at different hierarchical levels, one showing the connections between memory, system integration module and CPU and the other a more detailed view of the CPU components and parameters. During animation of the system, information packets are shown passing between components, parameters are changed and memory contents are updated.

A small set of simple experiments has been performed on this model including (i) an analysis of how the performance of the memory effects the execution times of simple programs, (ii) simple code optimisations by examining program execution and adjusting the operation of the micro controller accordingly, and (iii) instruction usage measurements.

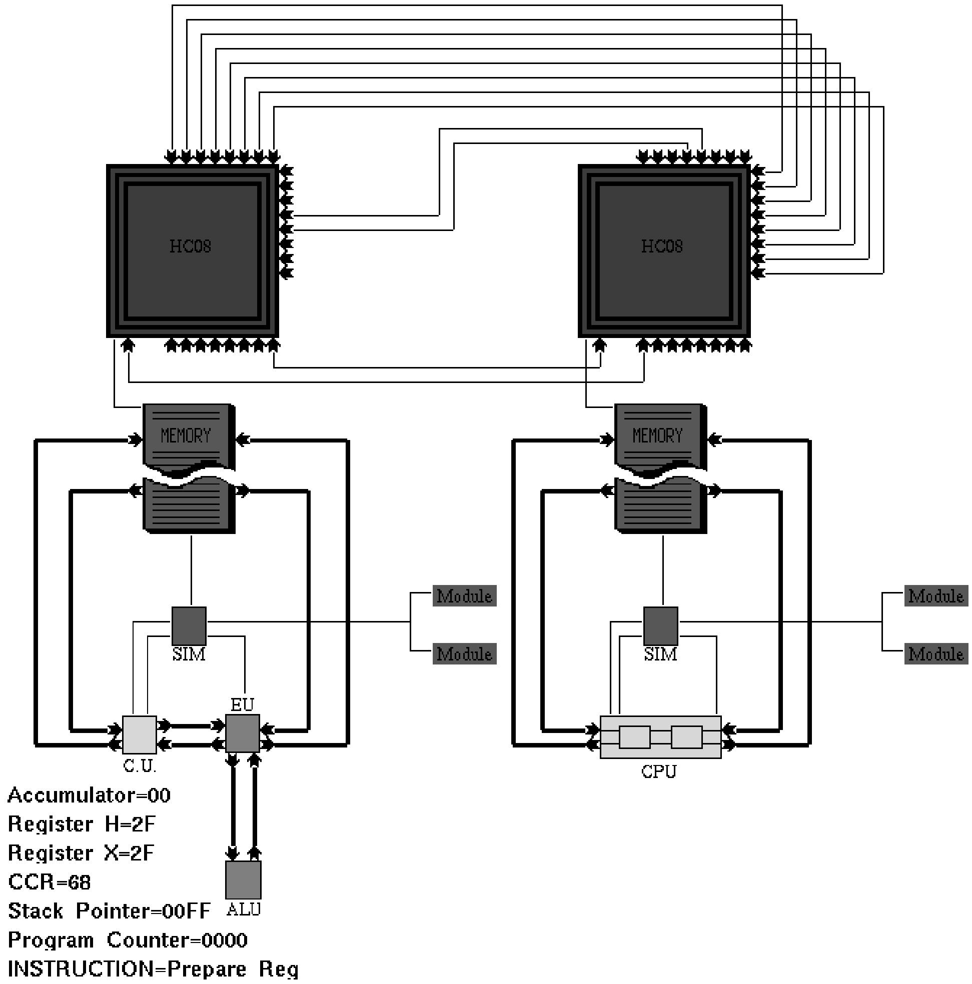

Figure 3: Two M68HC08 micro controllers

Other Architectures

Some of the other systems simulated recently include the Stanford DASH architecture [9], the Hierarchical PRAM [10] model of parallel computation and a PC memory hierarchy system.

The Stanford DASH architecture was designed to be a scalable high performance machine with multiple coherent caches and a single address space. The HASE simulation concentrated on implementing and illustrating the cache coherency protocols.

The Hierarchical PRAM model of parallel computation was simulated on a 2D mesh. The goals of this project were to investigate the scalability and efficiency with which this model of computation might be implemented on realistic parallel architectures. The investigation involved simulating the H-PRAM on a 2D mesh of processors and the parallel programming language H-FORK (a derivative of FORK) which allowed complete programs to be studied.

The PC simulation project used HASE to simulate aspects of a PC architecture. The two aspects concentrated on were the cache and hard disk systems. The project investigated the performance of the architecture for different cache and hard disk parameters.

System Evaluation

Studies of some real systems are currently being started, in which the evaluation of hardware/software trade-offs will be performed. These systems will contain many different types of components with a wide variety of possible algorithms and implementations. Example applications could include printer and disk controllers, graphics/image processing hardware or components of larger systems and networks, for example, bridges and routers. To this end a fully parameterised, generic memory hierarchy, with optional optimisations and configurable bus connections is being developed, as almost all systems that are to be investigated will contain some form of memory hierarchy.

Future Work

One of the main tasks to be carried out in the future is the development of more systems to illustrate hardware/software codesign techniques and methodologies, and to show that the application of these approaches to system design can result in improved performance, cost, reliability and power consumption.

The addition of other simulation languages, especially one capable of simulating VHDL, would prove a useful asset if HASE is to be used as a basis for a codesign environment. This would enable hardware synthesis and layout tools to be used to gain important information about the cost and power consumption of hardware components, as this information is, in most cases, vital to system design.

There are two aspects related to the hierarchical nature of HASE. The first, controlling on-screen simulation complexity, has already been included and utilised in many projects. The second area, hierarchical simulation, currently requires designers to write code at different levels of abstraction and select an appropriate level from within the HASE environment. The problems with providing environmental support for code abstraction are currently the subject of a PhD project.

The areas being investigated by this project include:

- The provision of entity linkage abstractions. By providing an abstract specification of how entities in a model are linked together we hope to provide a 'plug and play' framework for simulation model construction. Such a system would allow components in a simulation to be swapped for alternate simulation entities with the minimum of programming overhead. This would prove especially useful in CoDesign related simulation as it would facilitate the rapid interchange of entities representing both hardware and software based solutions.
- The creation of specific abstractions for common architectural components (memories, processors, switches etc).
- The generation of heuristics governing the use of abstraction in different architectural contexts. This work is also being used to create model validation techniques to check that components are interfaced together correctly.

References

[1] R. McNab and F. Howell, "Using Java for Discrete Event Simulation," in *Twelfth UK Computer and Telecommunications Performance Engineering Workshop (UKPEW)*, (University of Edinburgh), pp. 219–228, 1996.

[2] P. Coe, *Entity Description Language Manual.* Computer Systems Group, Department of Computer Science, University of Edinburgh, Kings Buildings, Edinburgh, March 1997.

[3] JADE Simulations International Corporation, *SIM++. A Discrete Event Simulation Language*, 1991.

[4] F. Howell, "HASE++. A Discrete Event Simulation Library for C++." Version 0.1, 1996.

[5] A. Kalavade and E. Lee, "A Global Critically/Local Phase Driven Algorithm for Constrained Hardware/Software Partitioing," in *3rd Internation Workshop on Hardware/Software Codesign (Grenoble, France)*, pp. 42–48, IEEE Computing Society Press, Los Alamitos, CA, USA, September 1994.

[6] R. Gupta and G. DeMicheli, "Hardware-Software Cosynthesis for Digital Systems," *IEEE Design and Test of Computers*, vol. 10, pp. 29–41, September 1993.

[7] R. Ernst, J. Henkel, and T. Benner, "Hardware-Software Cosynthesis for Micro Controllers," *IEEE Design and Test of Computers*, pp. 64–75, December 1993.

[8] N. Rafferty, "A Software Simulator for the Motorola M68HC08 Micro-Controller Unit." 4th Year Project Report, University of Edinburgh, June 1997.

[9] L. Williams and R. Ibbett, "Simulating the DASH Architecture in HASE," in *The 29th Annual Simulation Symposium (New Orleans, Lousiana)*, (IEEE Computing Society Press, Los Alamitos, CA, USA), pp. 137–146, 1996 1996.

[10] G. Chochia, M. Cole, and T. Heywood, "Implementing the Hierarchical PRAM on the 2D Mesh: Analyses and Experiments," in *7th IEEE Symposium on Parallel and Distributed Processing (San Antonio)*, 1995.