Guest Editors’ Introduction

Computer Hardware Description Languages: The Bridge Between Software and Hardware

Mario R. Barbacci
Carnegie-Mellon University

Takao Uehara
Fujitsu Laboratories, Ltd.

We are very happy to have the opportunity to continue a tradition of special issues on computer hardware description languages. This is the third in a series that spans a decade, and although they can’t provide an exhaustive coverage, the issues present a clear picture of the evolution of the field.

“Why do we need computer hardware description languages?” was the question posed by Yaohan Chu, the guest editor for the first special issue on HDLs, published in December 1974.1 Ten experts answered the question, discussing the similarities and differences between programming and hardware design and the necessity for a high-level language in which designers could express asynchronous operations, parallel control, and the structure of a hardware system. At the time, there were already in existence a number of HDLs—the special issue included articles describing AHPL, CDL, DDL, ISP, and PMS and surveys on hardware description languages used around the world.

G. Jack Lipovski warned that “we seem to be confused in our Tower of Babel; everybody is talking a different language but nobody is communicating” in the second special issue on HDLs, published in June 1977.2 Stephen Su, the guest editor, suggested possible directions for future research, in particular, the use of a common language for hardware designers and a design language that reflects a common modeling philosophy for software and hardware. The articles in the issue revealed a growing interest in applications such as synthesis and test sequence generation.

This third special issue on HDLs includes results of efforts to develop a common language as well as several new languages with special applications, such as formal verification of hardware designs. These new languages help to bridge the gap between software and hardware. We see this bridging effort as a healthy trend, since many of the problems faced by hardware and software designers are of similar nature.

A standard hardware description language has been the goal of system designers for three decades. This issue of Computer provides an update of work done around the world, identifying languages that now bring totally automated design closer to realization.
Current problems include correctness and verification, separate compilation facilities, access to program libraries, version control, and finally configuration management. Many good ideas have been developed in programming environments to enhance the productivity of programmers and the correctness of the program development process. We hope to see similarly rich environments at the service of hardware designers.

The chronicle of hardware description languages can be also traced through the series of international symposia on computer hardware description languages and their applications. Following a modest beginning with meetings at Rutgers University (1973) and at the Technical University of Darmstadt (1974), the IEEE Computer Society sponsored full-fledged conferences in New York (1975) and Palo Alto (1979). In 1979, the conferences came under the aegis of the International Federation for Information Processing, which sponsored meetings in Kaiserslautern (1981) and Pittsburgh (1983). The seventh meeting is scheduled for Tokyo later this year.

There has been a great deal of synergy between these special issues and the symposia. The guest editors have all been instrumental in the organization of the symposia, and authors of several of the articles appearing in this issue of Computer have presented their results in Kaiserslautern and Pittsburgh: Moszkowski, Maruyama and Fujita, Lam and Mostow present revised versions of papers read at the sixth symposium,3 while Suzuki's article is based on his invited lecture and appears here for the first time.

A trend towards more formal applications of HDLs has been observed over the years (for instance, one-third of the papers presented at the sixth symposium dealt with formal verification issues). Formal aspects of HDLs are represented in this issue by two articles. The article by Moszkowski describes a logical notation for reasoning about digital circuits, with a temporal predicate calculus as the formal core of the notation. The article by Maruyama and Fujita describes two formal verification techniques for hardware design. Like Moszkowski, the authors also make use of a temporal logic notation.

The article by Suzuki is an example of the use of a programming language, Concurrent Prolog, for describing hardware. It uses a pipe-line control mechanism for a high performance personal computer as an example. The same language is used to write both functional specifications and requirement specifications (I/O assertions) and to check the consistency between specifications by executing the Prolog programs.

Synthesis of hardware from formal machine descriptions has been a longstanding interest in the field. The article by Lam and Mostow presents a notation and a technique for the design of systolic logic arrays. The authors describe Sys, a design program that accepts a software algorithm, along with some advice, and applies a series of transformations to produce a functional-level circuit description of a systolic design. The article by German and Lieberherr introduces Zeus, a language that provides facilities for describing circuits by recursive and iterative methods. These methods permit changing a few elements in the description and recomposing it for specification (and functional verification) of parameterized families of designs.

The article by Dasgupta surveys an area of synthesis different from that of the previous two articles. Rather than addressing the problem of transforming a high-level description into a physical entity, Dasgupta deals with the construction of machine-independent microprogramming systems, that is, systems that can be used to generate efficient, correct microcode for different host micro-architectures.

The last two articles in this issue are the results of efforts towards standardization on a common language. Piloty and Borrione describe the current state of the Conlan project. The Consensus Language that provided the project name was defined by an international committee charged with the responsibility of defining a common, kernel language together with a set of extension mechanisms. In the Conlan approach, a family of different hardware description languages can be derived for special applications, all sharing a common semantic base, base consensus language.

The article by Shahdad, Lipsett, Marschner, Sheehan, Cohen, Waxman, and Ackley describes, for the first time, the proposed hardware description language sponsored by the Department of Defense. Although the initial application is to support the DoD VHSIC program, it is clear that the potential impact of VHDL extends far beyond this initial application.

We would like to express our sincere appreciation to the referees for their invaluable help in selecting the papers. Bruce Shriver, formerly with the University of Southwestern Louisiana and now with the IBM Thomas J. Watson Research Center, first suggested in April 1983 the idea of this special
issue. Stephen Yau, the former editor-in-chief of Computer, was very helpful, providing much needed guidance to the guest editors. We also appreciate the smooth transition to the current editor-in-chief, Michael Mulder. Finally, our sincere thanks to the authors for doing such a fine job within rather stringent deadlines.

References


Mario R. Barbacci is the associate director for project engineering at the Software Engineering Institute and a senior research scientist with the Department of Computer Science, Carnegie-Mellon University. He received the BSEE and Engineer degrees from the Universidad Nacional de Ingenieria, Lima, Peru, in 1966 and 1968, respectively, and a doctorate in computer science from CMU in 1974. He has worked in the formal specification of computer architectures for simulation, evaluation, and synthesis, but his professional interests also include automatic programming and programming environments. Prior to assuming his current position at the Software Engineering Institute, he was a member of CMU's Spice Project, responsible for the design and implementation of the Ada+ programming environment.

Takao Uehara is deputy manager of the software laboratory, Fujitsu Laboratories Ltd. He has been engaged in research on CAD, computer architecture, software engineering, and artificial intelligence since he joined Fujitsu in 1970. Uehara received his BS, MS, and Dr. Eng. degrees from Waseda University, Tokyo, in 1965, and 1967, and 1970, respectively. He was a visiting scholar at Stanford University in 1977. He is a member of IFIP Working Group on Computer Descriptions and Tools, chaired CHDL 83, and is the local arrangement chairman of CHDL 85.

Questions concerning the general context of this special issue can be addressed to Barbacci, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.