The past several years have seen a growth in what we might call “processor-centric design” in embedded systems and complex system-on-chip devices. In this design style, more and more of the product function is migrated away from dedicated hardware blocks and onto software running on fixed-instruction-set-architecture (ISA) processors, and application-specific instruction-set processors (ASIPs). Fixed-ISA processors—control processors and digital signal processors (DSPs)—are suitable for some tasks. However, the more interesting trend is the rise of ASIPs, which are customized for data-plane-intensive processing by configuring processor structure and the instruction set of the processor to optimize the processor’s performance and energy consumption for specific application tasks. ASIPs can thus offer a combination of the flexibility and programmability of a fixed-ISA processor and the performance and power characteristics (or close to it) of dedicated hardware. This makes them increasingly suitable for more and more of the embedded SoC architecture.

But if we had to design ASIPs using the same methods as for most fixed-ISA processors—that is, primarily manual design, with some use of RTL synthesis and automated layout mixed with regular block compilation and semiautomated assembly—as well as write the complete processor software tool chain by hand, it would take forever to design each ASIP. The rise of ASIPs would have been impossible without the development of automated tools that could take in a description of a specific processor and quickly and reliably produce much or all of the deliverables for a processor—hardware, testbenches, and the software tool chain (compiler, assembler, linkers, loaders, instruction set simulator, debugger, and associated libraries to assist with RTOS and middleware porting). And the most powerful way to create an automated ASIP creation tool is to base it on processor descriptions captured in an architectural description language (ADL).

With this in mind, in Processor Description Languages Prabhat Mishra and Nikil Dutt decided to gather into one volume descriptions of most of the significant ADLs used for designing processors in the past 20 to 30 years or so. This provides the valuable service of giving readers a comprehensive overview of various ADLs and their associated tools, which might help them in choosing one of the approaches for their own design and research work, or to come up with a unique approach that can complement the state of the art in both industry and academia.

Processor Description Languages offers in its 14 chapters detailed descriptions of 11 industrial and academic ADLs used for ASIP creation, each written by some of the key developers of each ADL. In addition, it summarizes the salient features of eight more approaches in a concluding chapter. It thus can serve as the standard comparative reference work for this approach to processor-centric design for several years to come. (For full disclosure, I will mention here that Chapter 8 deals with Tensilica’s TIE language and was written by two colleagues of mine, so I will not comment on this chapter further in this review.)

The book’s first two chapters introduce the ADL concept and the various design flows that are possible using ADLs for ASIP—or partial—generation.
Sometimes ADLs are used just to generate specific tools, such as instruction set simulators or compilers, rather than generate all the deliverables for a configured ASIP. These were written or cowritten by at least one of the book’s editors. The next six chapters describe several processor ADLs that were introduced at least a decade ago—three from the research community and three used in a commercial context, although two of these started as university research projects. These include one of the oldest ADL projects—Mimola, which began at the University of Kiel in Germany in 1976 and moved to the University of Dortmund in 1990. nML started at TU Berlin in the early 1990s, and has been used and evolved by Target Compiler Technologies for commercial purposes. Lisa began at RWTH Aachen University, became the basis for spinoff company LisaTEK, and was then acquired by CoWare. The Expression ADL from the University of California (UC) Irvine, and ASIPMeister from Osaka University, are academic projects that have been used for many years and whose results have been widely reported.

Because these first ADL examples either have had commercial use, or are the results of long-standing academic research projects of considerable influence, I found the chapters describing them to be of the most interest. The tools built around the various ADLs in the first half of the book are arguably the most mature and have been applied to a wider variety of applications over time.

The next five chapters of the book deal with academic ADLs. These include MADL from Princeton, ADL++ from Michigan Technological University, ArchC from University of Campinas, Brazil, MAML from the University of Erlangen-Nuremberg, and GNR (NISC), also from UC Irvine. The final chapter, as mentioned, is a survey of eight other ADL approaches, some dating back to the early 1990s. Of particular interest in the chapters on MAML, ArchC, and GNR was the discussion on applying these ADLs for designing a variety of multiprocessor architectures as well as the individual ASIPs used in these MPSoC approaches.

The first chapter of the book established a framework for comparing ADLs: a taxonomy of structural, behavioral, and mixed approaches, and an orthogonal classification into synthesis-, validation-, compilation-, and simulation-oriented goals. One thing that would have been useful in Processor Description Languages would have been a tabular comparison of all the ADLs presented, written in a concluding chapter that would sum up and compare the major ADL approaches just covered. This would have helped reinforce the ADL taxonomy developed in Chapter 1 and assisted the reader in placing the various methods onto this conceptual map after reading all the details contained in the various chapters.

Certainly the rich set of industrial and research approaches covered in this book indicate a field that is still active and growing. Continued development of existing ADLs and tools is very likely as more designers embrace the processor-centric approach, and no doubt new research projects will arise, especially in carrying the ADL concept further into the multiprocessor world.

This book is likely to become the main resource for anyone wanting to understand the ADL area in detail. If the field continues to grow, and new contributions are made, the authors might want to consider a second edition in a few years containing information about new ADLs and new features added to the current ones. It was a pleasure to read and will be a well-thumbed addition to my bookshelf. If the design of complex heterogeneous multiprocessor systems can be likened to a processor “stew,” this book contains recipes for many tasty varieties.

Direct questions and comments about this department to Grant Martin, 2424 Raven Road, Pleasanton, CA 94566; gmartin@ieee.org.

For further information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.