

Book Reviews

Putting SystemC into practice

Grant Martin

■ **SYSTEMC IS A REALITY.** Created with contributions from many companies and volunteers, the language is becoming the de facto standard for system-level modeling. Because SystemC is a class library and simulation kernel built on top of C++, there is a high need for good training, particularly for system architects from hardware backgrounds.

There are many ways to learn a modeling and design language. Of course, there is no substitute for lots of practice in using it. There are training courses, Web-based tutorials, and paper presentations at conferences. But almost all forms of training, practice, and self-learning require written material to accompany them.

SystemC presents an extra layer of complication to those who are learning it, because it requires both knowledge of C++ and its idiosyncrasies, and an understanding of the SystemC syntax and semantics, as well as system-level modeling methodologies.

Users might start with the language reference manuals and user guides from the Open SystemC Initiative (OSCI). Members of the community have also written several books. However, we now have a better starting point: *SystemC: From the Ground Up*, by David Black and Jack Donovan of Eklektic Ally Inc., a consultancy (Kluwer Academic Publishers, 2004, ISBN 1-402-07988-5, 244 pp., \$125). This book is a welcome addition to SystemC literature. It is a thorough overview of SystemC and its capabilities and peculiarities. The book is useful for beginners and those with some experience, although expert users will probably not find much that is new except for a preview of just-released features in SystemC 2.1.

Reviewed in this issue

SystemC: From the Ground Up, by David Black and Jack Donovan (Kluwer Academic Publishers, 2004, ISBN 1402079885, 244 pp. \$125).

What sets this book apart from the rest is the combination of language review with a very strong focus on pragmatic guidelines for language usage. True to its title, the book starts with a ground-up discussion of system-level modeling's role in the design flow, and, in chapter 2, the transaction-level design approach for which SystemC is a good fit. Chapter 3 gives an overview of basic SystemC components, including a first-level description of the simulation kernel. As the book progresses, it revisits the simulation kernel and describes it in more detail in chapters 7 and 9. Chapter 4 is an introduction to SystemC data types, covering native C++ data types and the SystemC arithmetic, Boolean, multivalued and fixed-point types, along with a strong recommendation to look at the C++ standard template library for additional data types of value in modeling. Chapters 5, 6, 7, and 8 deal with the fundamental building blocks of modules, time, concurrency, and channels. The notions of *channels*—that is ports and interfaces, whether primitive or hierarchical—are vital to understanding the communications model that SystemC supports at several levels of abstraction. Thus the material on basic channels (chapter 8), evaluate-update channels (chapter 9), ports and communications (chapter 11), specialized ports (chapter 12), and custom channel modeling (chapter 13) represent the heart of the book's treatment of this topic. It is a treatment that goes a long way in explaining the intricacies of the SystemC modeling styles.

Chapter 10 deals with model structuring, and gives a good overview of the choices available to modelers in creating header and implementation files, with a discussion of the impact of these various styles on design hierarchy and model provisioning and maintenance. Discussion of advanced topics in chapter 14 include clocking and clocked threads; programmable hierarchy and configuration elaboration; debugging and signal tracing; and dynamic process creation (another feature introduced officially with SystemC 2.1). Finally, chap-

ter 15 (on odds and ends) includes notes on model performance, some of the idiosyncrasies of C++ that can cause problems, and coding styles and SystemC adoption methods.

The book is strongest in its pragmatic, example-based approach, as opposed to the theoretical one common among most system-modeling monographs. The authors base the guidelines in their book on years of practical experience as system-modeling consultants. This emphasis on practical approaches extends to the discussion of advanced topics, and modeling and language-writing conventions. The discussion of features in the upcoming SystemC 2.1 release from OSCI (just released for review as of October 2004) is especially valuable and will keep the book timely. The explanation of ports, channels, and interfaces, and the modeling uses of primitive and hierarchical channels are especially strong in this book, and are areas of SystemC where novices need special help. The authors make extensive use of example code or code fragments. Complete code samples are available online on their Web site, making it easy for readers to learn through examples. In addition, the book is highly readable, with a good flow of text, code samples, and annotated diagrams that give readers a good understanding of fundamental topics.

I DID NOT FIND MANY WEAKNESSES in the book. The major issue is due to C++; out of necessity, the discussion becomes complex at places where it needs to rely on advanced C++ concepts. The authors seem aware of the subject difficulty and encourage readers with helpful comments and recommendations to learn the most complex concepts through modeling practice. I heartily recommend this book to beginners and intermediate students of SystemC as a good way to start learning the language. Users who are intimately familiar with SystemC will likely not gain much from this book. Given the budding community of novices in SystemC, this book is a welcome addition to the literature on this important subject. ■

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>.

Let your e-mail address show your professional commitment.

An IEEE Computer Society e-mail alias forwards e-mail to you, even if you change companies or ISPs.

you@computer.org

The e-mail address of computing professionals

