SystemC: From the Ground Up

David C. Black · Jack Donovan
Bill Bunton · Anna Keist

SystemC: From the Ground Up

Springer

David C. Black
XtremeEDA
Austin, TX 78749
USA
dcblack@eslx.com

Jack Donovan
HighIP Design Company
Round Rock, TX 78764
USA
jack@donovanweb.org

Bill Bunton
LSI Corporation
Austin, TX 78749
USA

Anna Keist
XtremeEDA
Austin, TX 78749
USA

ISBN 978-0-387-69957-8 e-ISBN 978-0-387-69958-5
DOI 10.1007/978-0-387-69958-5
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2009933997

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

This book is dedicated to our spouses Pamela Black, Carol Donovan, Evelyn Bunton, and Rob Keist and to our children Christina, Loretta, & William Black, Chris, Karen, Jenny, & Becca Donovan, John Williams & Sylvia Headrick, Alex, Madeline, and Michael Keist

2nd Edition Preface

Why the 2nd Edition?

The reader (or prospective buyer of this book) might ask about the need for a second edition. The first edition was highly successful and progressed to a second and third life after being translated to Japanese and Korean.

There are three over-arching reasons for the second edition:

- A fast changing technical landscape
- Incorporation of additional topic suggestions from readers
- Fixing of errors and improvement of confusing text segments and chapters

To address the shifting technical landscape, we have significantly updated the chapters addressing Electronic System-Level design to reflect the refinements of ESL methodology thinking in the industry. Although this is not a complete discussion of ESL, it is an overview of the industry as currently viewed by the authors.

We have added a chapter on TLM, a standard that will enable interoperability of models and a model marketplace. Although this chapter discusses TLM 1.0, we think it imparts to the reader a basic understanding of TLM. Those of you who follow the industry will note that this is not TLM 2.0. This new standard was still emerging during the writing of this edition. But not to worry! Purchasers of this edition can download an additional chapter on TLM 2.0 when it becomes available within the next six months at www.scftgu.com.

Although SystemC is now IEEE 1666 it is not immune from the shifting technical landscape, so the authors have included material on some proposed extensions to the SystemC standard related to process control.

Readers have suggested several additional topics over the last several years and we have tried to address these with an additional chapter on the SystemC Verification (SCV) Library and an appendix on C++ fundamentals.

The chapter on the SCV library is a high level introduction and points the reader to additional resources. The authors have found that many organizations have started using the SCV library after becoming familiar with SystemC and ESL methodologies. For those readers, we have added this chapter.

The authors received several suggestions asking us to add examples and comparisons to HDL languages like Verilog and VHDL. The authors have respectfully declined, as we feel this actually impedes the reader from seeing the intended uses of SystemC. After exploring these suggestions, we have found that these readers were not entirely comfortable with C++, and because C++ is fundamental to an understanding of SystemC, this edition includes a special appendix that attempts to highlight those aspects of C++ that are important prerequisites, which is most of the language.

Writing a book of this type is very humbling, as most who have journeyed on similar endeavors will confirm. Despite our best efforts at eliminating errors from the first edition, the errata list had grown quite long. We have also received feedback that certain portions of the book were "confusing" or "not clear". After reviewing many of these sections, we had to ask: What were we thinking? (a question asked by many developers when they revisit their "code" after several years)

In some cases we were obviously "not thinking", so several chapters and sections of chapters have been significantly updated or completely rewritten. The topic of concurrency proved a more challenging concept to explain than the authors first thought. This edition effectively rewrote the chapters and sections dealing with the concept of concurrency.

The authors have been quite gratified at the acceptance of the first edition and the rapid adoption of SystemC. We hope we have played at least a small part in the resulting changes to our community. We wish you good luck with SystemC and your contributions to our community.

Jack Donovan, David Black, Bill Bunton, Anne Keist
4authors@scftgu.com

Preface

Jack Donovan, David Black, Bill Bunton, and Anne Keist

Why This Book

The first question any reader should ask is "Why this book?" We decided to write this book after learning SystemC using minimal documentation to help us through the quest to deeper understanding of SystemC. After teaching several SystemC classes, we were even more convinced that an introductory book focused on the SystemC language was needed. We decided to contribute such a book.

This book is about SystemC. It focuses on enabling the reader to master the language. The reader will be introduced to the syntax and structure of the language, and the reader will also learn a few techniques for use of SystemC as a tool to shorten the development cycle of large system designs.

We allude to system design techniques and methods by way of examples throughout the book. Several books that discuss system-level design methodology are available, and we believe that SystemC is ideally suited to implement many of these methods. After reading this resource, the reader should not only be adept at using SystemC constructs, but also should have an appreciation of how the constructs can be used to create high performance simulation models.

We believe there is enough necessary information about SystemC to learn the language that a stand-alone book is justified. We hope you agree. We also believe that there is enough material for a second book that focuses on using SystemC to implement these system-level design methods. With reader encouragement, the authors have started on a second book that delves deeper into the application of the language.

Prerequisites for This Book

As with every technical book, the authors must write the content assuming a basic level of understanding; this assumption avoids repeating most of an engineering undergraduate curriculum. For this book, we assumed that the reader has a working knowledge of C++ and minimal knowledge of hardware design.

For C++ skills, we do not assume that the reader is a "wizard". Instead, we assumed that you have a good knowledge of the syntax, the object-oriented

features, and the methods of using C++. The authors have found that this level of C++ knowledge is universal to current or recent graduates with a computer science or engineering degree from a four-year university.

Interestingly, the authors have also found that this level of knowledge is lacking for most ASIC designers with 10 or more years of experience. For those readers, assimilating this content will be quite a challenge but not an impossible one.

As an aid to understanding the C++ basics, this edition includes an appendix on C++. Those who have been exposed to C++ in the past are encouraged to quickly review this appendix. For a few novices, this appendix may also work as a quick introduction to the topics, but it is unlikely to be completely sufficient.

For readers who are C++ novices or for those who may be rusty, we recommend finding a good C++ class at a community college, taking advantage of many of the online tutorials, or finding a good consulting group offering an Intro to C++ class. For a list of sources, see Appendix A. We find (from our own experience) that those who have learned several procedural languages (like FORTRAN or PL/I) greatly underestimate the difficulty of learning a modern object-oriented language.

To understand the examples completely, the reader will need minimal understanding of digital electronics.

Book Conventions

Throughout this book, we include many syntax and code examples. We've chosen to use an example-based approach because most engineers have an easier time understanding examples rather than strict Backus-Naur form[1] (BNF) syntax or precise library declarations. Syntax examples illustrate the code in the manner it may be seen in real use with placeholders for user-specified items. For more complete library information, we refer the reader to the SystemC Language Reference IEEE 1666-2005, which you can obtain for free via www.systemc.org.

Code will appear in monotype Courier font. Keywords for both C/C++ and SystemC will be shown in Courier **bold**. User-selectable syntax items are in *italics* for emphasis. Repeated items may be indicated with an ellipsis (…) or subscripts. The following is an example:

```
wait(name.posedge_event()|event_i…);
if (name.posedge()) { //previous delta-cycle
  //ACTIONS…
```

Fig. 1 Example of sample code

[1]John Backus and Peter Naur first introduced BNF as a formal notation to describe the syntax of a given language. Naur, P. (1960, May). Revised report on the algorithmic language ALGOL 60. *Communications of the ACM*, *3*(5), 299-314.

When referring to a class within the text it will appear as **class_name** or **sc_class_name**. When referring to a templated class within the text, it will appear as **template_class_name<*T*>**. When referring to a member function or method from the text it will appear as **member_name(***args***)** or **sc_member_name**(*args*). Occasionally, we will refer to a member function or method without the arguments. It will appear in the text as **member_name**() or **sc_member_name**().

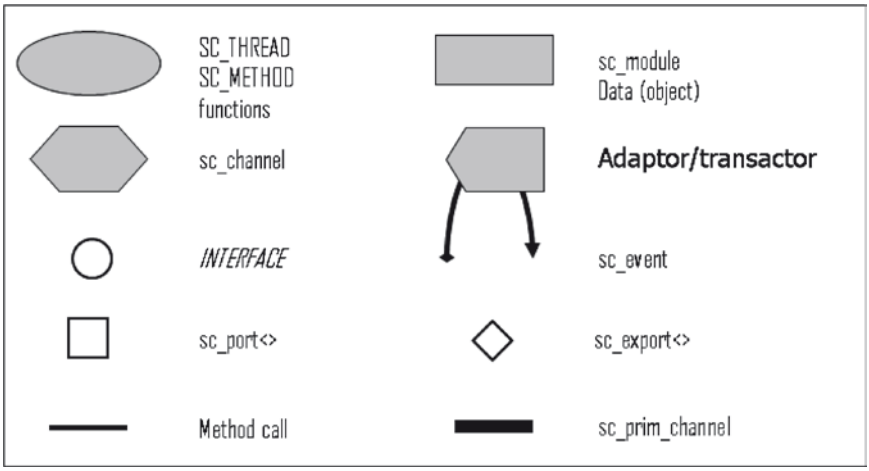

Fig. 2 Standard graphical notations

In addition, we have adopted standard graphical notations as shown in Fig 2. The terminology will be presented as the book progresses. Readers of the first edition will note that we changed the depiction of an **sc_export** from a dotted circle to a diamond. This change was the result of comments that the dotted circle was too hard to make out in some cases. We also removed arrows since in most cases, the meaning is not always clear[2].

SystemC uses a naming convention where most SystemC-specific identifiers are prefixed with **sc_** or **SC_**. This convention is reserved for the SystemC library, and you should not use it in end-user code (your code).

About the Examples

To introduce the syntax of SystemC and demonstrate its usage, we have filled this book with examples. Most examples are not real-world examples. Real examples become too cluttered too fast. The goal of these examples is to communicate

[2]Does an arrow convey calling direction (i.e., C++ function call) or direction of data flow? Since many interfaces contain a mixture of calls, some input and some output, showing data flow direction is not very useful.

concepts clearly; we hope that the reader can extend them into the real world. For the most part, we used a common theme of an automobile for the examples.

By convention, we show syntax examples stylistically as if SystemC is a special language, which it is not. We hope that this presentation style will help you apply SystemC on your first coding efforts. If you are looking for the C++ declarations, please browse the Language Reference Manual (LRM) or look directly into the SystemC Open SystemC Initiative reference source code (www.systemc.org).

It should also be noted that due to space limitations and to reduce clutter, we have omitted showing standard includes (i.e., **#include**) and standard namespace prefixes in most of the examples. You may assume something such as the following is implied in most of the examples:

```
#include <iostream>
#include <systemc>
#include <scv.h>
using namespace std;
using namespace sc_core;
using namespace sc_dt;
```

Fig. 3 Assumed code in examples

Please note that it is considered bad C++ form to include the standard namespace in header files (i.e., do not include "**using namespace** std;" in a header). We believe making the examples clear and brief warrants ignoring this common wisdom.

How to Use This Book

The authors designed this book primarily for the student or engineer new to SystemC. This book's structure is best appreciated by reading sequentially from beginning to end. A reader already familiar with SystemC will find this content to be a great reference.

Chapters 1 through 3 provide introductions and overviews of the language and its usage. Methodology is briefly discussed.

For the student or engineer new to SystemC, the authors present the language from the bottom up and explain the topics from a context of C++ with ties to hardware concepts. We provide exercises at the end of Chapters 4 through 16 to reinforce the concepts presented in the text. Chapters 16 through 18 strengthen the basic language concepts. In these chapters, readers will find discussions of areas to watch and understand when designing, writing, or using SystemC in a production environment.

For the student or engineer already familiar with SystemC, Chapters 4 through 13 will provide some interesting background and insights into the language. You can either skip or read these early chapters lightly to pick up more nuances of the language. The content here is not meant to be a complete description of the language.

For a thorough description, the reader is referred to the SystemC LRM. Chapters 14 through 18 provide intermediate to advanced material.

For the instructor, this book may provide part of an advanced undergraduate class on simulation or augment a class on systems design.

In most of the examples presented in the book, the authors show code fragments only so as to emphasize the points being made. Examples are designed to illustrate specific concepts, and as such are toy examples to simplify learning. Complete source code for all examples and exercises is available for download from www.scftgu.com as a compressed archive. You will need this book to make best use of these files.

SystemC Background

SystemC is a system design language based on C++. As with most design languages, SystemC has evolved. Many times a brief overview of the history of language will help answer the question "Why do it that way?" We include a brief history of SystemC and the Open SystemC Initiative to help answer these questions.

The Evolution of SystemC

SystemC is the result of the evolution of many concepts in the research and commercial EDA communities. Many research groups and EDA companies have contributed to the language. A timeline of SystemC is included in Table 1.

SystemC started out as a very restrictive cycle-based simulator and "yet another" RTL language. The language has evolved (and is evolving) to a true system design language that includes both software and hardware concepts. Although SystemC

Table 1 Timeline of development of SystemC

Date	Version	Notes
Sept 1999	0.9	First version; cycle-based
Feb 2000	0.91	Bug fixes
Mar2000	1.0	Widely accessed major release
Oct 2000	1.0.1	Bug fixes
Feb 2001	1.2	Various improvements
Aug 2002	2.0	Add channels & events; cleaner syntax
Apr 2002	2.0.1	Bug fixes; widely used
June 2003	2.0.1	LRM in review
Spring 2004	2.1	LRM submitted for IEEE standard
Dec 2005	2.1v1	IEEE 1666-2005 ratified
July 2006	2.2	Bug fixes to more closely implement IEEE 1666-2005

does not specifically support analog hardware or mechanical components, there is no reason why these aspects of a system cannot be modeled with SystemC constructs or with co-simulation techniques.

Open SystemC Initiative

Several of the organizations that contributed heavily to the language development efforts realized very early that any new design language must be open to the community and not be proprietary. As a result, the Open SystemC Initiative (OSCI) was formed in 1999. OSCI was formed to:

- Evolve and standardize the language
- Facilitate communication among the language users and tool vendors
- Enable adoption
- Provide the mechanics for open source development and maintenance

The SystemC Verification Library

As you will learn while reading this book, SystemC consists of the language and potential methodology-specific libraries. The authors view the SystemC Verification (SCV) library as the most significant of these libraries. This library adds support for modern high-level verification language concepts such as constrained randomization, introspection, and transaction recording. The first release of the SCV library occurred in December of 2003 after over a year of Beta testing. This edition includes a chapter devoted to the SCV from a syntactic point of view.

Current Activities with OSCI

At present, the OSCI has completed the SystemC LRM that has been ratified as a standard (IEEE 1666-2005) by the Institute of Electrical and Electronics Engineers (IEEE). Additionally, sub-committees are studying such topics as synthesis subsets and formalizing terminology concerning levels of abstraction for transaction-level modeling (TLM). This edition includes a chapter devoted to TLM and current activities.

Acknowledgments

- Our inspiration was provided by:
  Mike Baird, President of Willamette HDL, who provided
  the basic knowledge to get us started on our SystemC journey.
- Technical information was provided by:
  IEEE-1666-2005 Standard
  OSCI Proof-of-Concept Library associated information on systemc.org
  Andy Goodrich of Forte Design Systems, who provided technical insights.
- Our reviewers provided feedback that helped keep us on track:
  Chris Donovan, Cisco Systems Incorporated
  Ronald Goodstein, First Shot Logic Simulation and Design
  Mark Johnson, Rockwell-Collins Corporation
  Rob Keist, Freescale Corporation
  Miriam Leeser, Northeastern University
  Chris Macionski, Synopsys Inc.
  Nasib Naser, Synopsys Inc.
  Suhas Pai, Qualcomm Incorporated
  Charles Wilson, XtremeEDA Corporation
  Claude Cloutier, XtremeEDA Corporation
  David Jones, XtremeEDA Corporation
- The team who translated the first edition of the book into Japanese and asked us many thought provoking questions that have hopefully been answered in this edition:
  Masamichi Kawarabayashi (Kaba), NEC Electronics Corporation
  Sanae Nakanishi, NEC Electronics Corporation
  Takashi Hasegawa, Fujitsu Corporation
  Masaru Kakimoto, Sony Corporation
- The translator of the Korean version of the first edition who caught many detailed errors. We hope that we have corrected them all in this edition:
  Goodkook, Anslab Corporation
- Our Graphic Artist
  Felix Castillo
- Our Technical Editors helped us say what we meant to say:
  Kyle Smith, Smith Editing
  Richard Whitfield

- Our Readers from the First Edition:
  David Jones, Junyee Lee, Soheil Samii, Kazunari Sekigawa, Ando Ki, Jeff Clark, Russell Fredrickson, Mike Campin, Marek Tomczyk, Luke Lee, Adamoin Harald Devos, Massimo Iaculo, and many others who reported errata in the first edition.

Most important of all, we acknowledge our spouses, Pamela Black, Carol Donovan, Rob Keist, and Evelyn Bunton. These wonderful life partners (despite misgivings about four wild-eyed engineers) supported us cheerfully as we spent many hours researching, typing, discussing, and talking to ourselves while pacing around the house as we struggled to write this book over the past year.

We also acknowledge our parents who gave us the foundation for both our family and professional life.

Contents