ASSERTION-BASED DESIGN

SECOND EDITION

ASSERTION-BASED DESIGN

SECOND EDITION

Harry Foster
Jasper Design Automation, Inc.

Adam Krolnik
LSI Logic Corporation.

David Lacey
Hewlett-Packard Company

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 1-4020-8028-X
Print ISBN: 1-4020-8027-1

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.kluweronline.com
and the Springer Global Website Online at: http://www.springeronline.com

Dedicated to:

Jeanne—the most wonderful person in my life.
And to my children—Elliott, Lance, and Hannah.
Always remember, when I was your age I used to have to walk across the room to change the channel.

-Harry

Cindy, Seth, Nicholas, Sarah and Jesus the Christ.

-Adam

To my loving wife, Deborah, for her patience and support while this book was written.

-David

TABLE OF CONTENTS

FOREWORD

There is much excitement in the design and verification community about assertion-based design. The question is, who should study assertion-based design? The emphatic answer is, both design and verification engineers.

What may be unintuitive to many design engineers is that adding assertions to RTL code will actually reduce design time, while better documenting design intent.

Every design engineer should read this book! Design engineers that add assertions to their design will not only reduce the time needed to complete a design, they will also reduce the number of interruptions from verification engineers to answer questions about design intent and to address verification suite mistakes. With design assertions in place, the majority of the interruptions from verification engineers will be related to actual design problems and the error feedback provided will be more useful to help identify design flaws. A design engineer who does not add assertions to the RTL code will spend more time with verification engineers explaining the design functionality and intended interface requirements, knowledge that is needed by the verification engineer to complete the job of testing the design.

Every verification engineer should read this book! The smart verification engineer will assist the design engineer to add assertions to the RTL-design code because the sooner a design engineer understands the usage and benefits of inserting assertions into the design, the more valuable that design engineer will be to the verification effort. A smart verification engineer is someone who can help a designer to catch the vision and understand the ease and value of assertion-based design. This is the first book to comprehensively address and explain HDL assertion-based design.

My colleague Harry Foster is the best-known name in the Verilog verification and assertion-based methodology community. Along with Lionel Bening, Harry pioneered the Verilog Open Verification Library (OVL), a freely available set of verification-focused Verilog modules that have been used in advanced design and verification environments ever since they were introduced.

My colleague Adam Krolnik was the verification champion of the Verilog-2001 Standards Group. I counted on Adam to promote

and propose verification enhancements to the IEEE Verilog language.

David Lacey, Harry and Adam are key participants on the Accellera SystemVerilog Standards Group. Their practical verification experience has contributed to the value of the assertion enhancements added to the SystemVerilog standard.

These three verification specialists have written a book that will endow the reader with an understanding of the fundamental and important topics needed to comprehend and implement assertion-based design.

Included in Chapter 7 of this book is a valuable set of commonly used assertion examples to help the reader become familiar with the capabilities of assertion-based design. This book is a must for all design and verification engineers.

Clifford E. Cummings

Verilog Guru & President, Sunburst Design, Inc.
Member IEEE 1364-1995 Verilog Standards Group
Member IEEE 1364-2001 Verilog Standards Group
Member IEEE 1364-2002 Verilog RTL Synthesis Standards Group
Member Accellera SystemVerilog 3.0 Standards Group
Member Accellera SystemVerilog 3.1 Standards Group

PREFACE

You may have heard that this is a book about verification and now you're wondering why it's called *Assertion-Based Design,* and not *Assertion-Based Verification.* The answer to that is one of the driving forces in this book: Verification doesn't happen in a vacuum. Specification has to occur before any form of verification, and as you know, specification occurs very early in the design cycle. Thus, our contention is that assertion specification is one of the integral pieces of a contemporary design cycle.

Within that context then, the focus of this book is three-fold:

- How to specify assertions
- How to create and adopt a methodology that supports assertion-based design (predominately for RTL design)
- What to do with the assertions and methodology once you have them

To support these three over-arching goals, we showcase multiple forms of assertion specification: Accellera Open Verification Library (OVL), Accellera Property Specification Langauge (PSL), and Accellera SystemVerilog.

The recommendations and claims we make in this book are based on our combined actual experiences in applying an assertion-based methodology to real design and verification as well as our work in developing industry assertion standards.

Real-world experience. In *Assertion-Based Design*, we have pooled our combined experiences to share our understanding and provide a reality-based picture of our chosen topic. The following is a summary of our background related to this topic:

- Harry Foster—Chairs the Accellera Formal Verification Technical Committee; which is developing the PSL standard; created the Open Verification Library; member of the SystemVerilog Assertion Committee; and previously developed assertion-based methodologies at Hewlett-Packard Company.

- Adam Krolnik—Accellera SystemVerilog Assertion Committee committee member and major contributor in the development of the SystemVerilog assertion constructs; created assertion-based methodologies at Cyrix and LSI Logic Corporation.
- David Lacey—Chairs the Accellera Open Verification Library committee; member of the SystemVerilog Assertion Committee; created functional coverage and assertion-based methodologies at Hewlett-Packard Company.

Fundamentals. Property specification is fundamental to an assertion-based verification platform (that is, assertions, constraints, and functional coverage). Once specified, properties enable the following components, which may be included in your assertion-based verification platform:

- *verifiable testplans* through property specification (for example, executable *functional coverage models,* which help answer the question *"what functionality has not been exercised?"*)
- *exhaustive* and *semi-exhaustive* formal property checking technology (for example, model checking and bounded-model checking)
- *dynamic property checking* technology (for example, monitoring assertions in simulation) for improved observability to reduce the time involved in debug
- *hardware verification languages* (HVLs) for testbench generation that leverage property specification to define expected input (constraints) and output (assertions) behavior
- *constraint-driven stimulus generation* based on interface properties targeting block-level designs
- *assertion property synthesis* to address silicon observability challenges during chip bring-up in the lab, as well as operational error detection required for high availability (HA) class systems

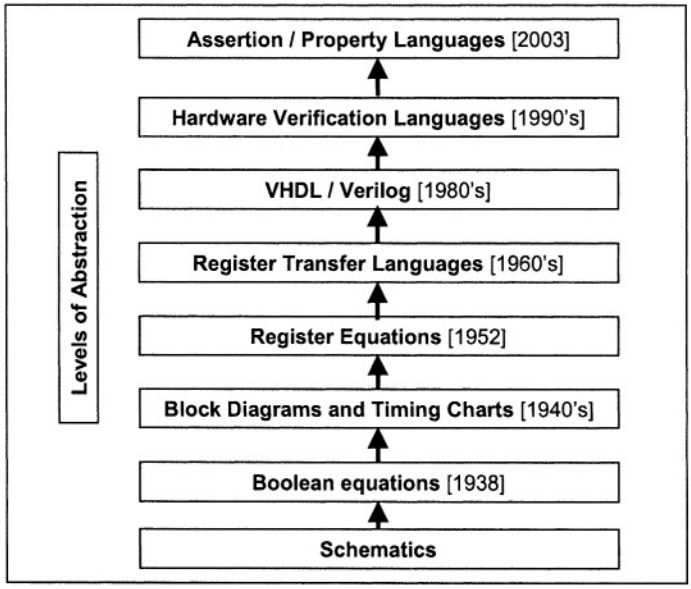
In this book, we discuss the important role that property specification plays in an assertion-based verification flow.

Evolution in levels of abstraction. The following figure shows an evolution in levels of design notation and specification abstraction. Each time we move up a level of abstraction, we expand possibilities, increase productivity, and improve communication of design intent. Perhaps most importantly, is the growth our field has experienced in conceptualizing new forms of specification, developing new technologies based on these new forms of specification, and then developing standards, which in turn opens new markets. For example, the development of Register Transfer Languages in the mid-1960's lead to the development of synthesis. However, it was the standardization of

VHDL/Verilog in the early 1990's that opened new markets and helped drive synthesis adoption.

The way design and verification has traditionally been performed is changing. In the future, we predict that design and verification will become property-based. Through the standardization of assertion and property languages that are occurring at the time of this publication, we foresee new and exciting EDA markets emerge, once again opening the door for improved productivity. All made possible through assertion-based design practices.

Design notation and specification levels of abstraction



Book organization

Assertion-Based Design is organized into chapters that can be read sequentially for a full understanding of our topic. Or you may wish to focus on a particular area of interest and visit the remaining chapters to supplement your understanding. We have allowed repetitions to support the readers who prefer to browse for information as well as readers who intend to read cover-to-cover and then revisit sections of particular personal relevance. Highlights of the contents follow.

Chapter 1, "Introduction"

Chapter 1 introduces property checking and modern verification techniques. It then introduces assertions and their use in industry, and statistics of their effectiveness. It discusses the benefits of assertion-based verification methodologies and dispels common misconceptions about assertion use within an RTL design flow.

Chapter 2, "Assertion Methodology" Chapter 2 discusses details for effectively creating and managing an assertion-based methodology, which includes validating assertions using simulation and formal verification. It shows how to best apply assertion-based methodologies to both new and existing designs. It explores how an assertion-based methodology works together with, not separate from, a general design methodology. Finally, it introduces several key learnings that must be accepted by the project team in order to be successful with an assertion-based methodology.

Chapter 3, "Specifying RTL Properties" Chapter 3 discusses basic property and assertion specification concepts and definitions. It introduces various forms of RTL assertion specifications, which include: the Accellera Open Verification Library (OVL), Accellera PSL formal property language, and Accellera SystemVerilog assertion constructs. Examples are given throughout the chapter to demonstrate how each property concept is used in concert with one of the assertion specification forms to create assertions. Many of these assertion examples can be used today, as they utilize the OVL. Other examples can be used as the EDA vendors move to support PSL and SystemVerilog. Finally, the chapter closes with a PCI property specification example.

Chapter 4, "PLI-Based Assertions" Chapter 4 discusses PLI-based assertion and procedural assertion techniques. It provides actual code that can be used as an assertion specification form with today's Verilog simulators that support the PLI. It also addresses ways to prevent false firing of procedural assertions.

Chapter 5, "Functional Coverage" Chapter 5 introduces the verification coverage models and how this topic fits in a book about assertions. It discusses the concepts of black-box and white-box testing, as well as assertion techniques that improve functional coverage. It explores the ideas of controllability and observability and their role in verification. The chapter focuses primarily on functional coverage models, but introduces other traditional coverage metrics (such as programming code coverage) and their role in the overall coverage efforts. Real world examples of successes using functional coverage is provides in this chapter as well as a discussion of the benefits of using functional coverage in your verification efforts. It provides an outline for an effective functional coverage methodology, including guidelines for achieving correct functional coverage density within your design. Finally, it provides a description of several forms of functional coverage specification before concluding with examples of both low-level RTL implementation and high-level architectural functional coverage specification.

Chapter 6, "Assertion Patterns" Chapter 6 introduces the concept of assertion patterns as a convenient method to document and communicate commonly occurring assertions that are found in today's RTL designs. This provides an easy reference for broad classes of assertions that

allows the reader to more easily apply the concepts to their specific designs. A number of different assertion patterns are presented, including signal, set, conditional, past and future event, window, and sequence. Each assertion pattern is described in great detail and provides examples of applicable assertions and descriptive waveforms for additional clarity, as needed.

Chapter 7, "Assertion Cookbook" Chapter 7 provides concrete examples of the common assertions and functional coverage points for components, interfaces, and general logic, including queues, stacks, finite state machines, encoders, decoders, multiplexers, state table structures, memory, and arbiters. It includes examples that use each form described in Chapter 2 (OVL, SystemVerilog, and PSL). While the list of components is not exhaustive, the chapter gives a broad coverage of components that enables readers to extend the concepts to their specific applications.

Chapter 8, "Specifying Correct Behavior" Chapter 8 present a set of clarifying ideas that we have found effective when attempting to specify various aspects of a design, at multiple levels of abstraction. We first present a set of common ambiguities that arise when interpreting a natural language specification. We then discuss the limitation of temporal property languages, and the need for additional modeling to overcome these limitations when attempting to specify higher level requirements. Finally, we conclude by presenting a construction guide we have found useful in our own work with assertion and functional coverage specification.

Appendix A, "Open Verification Library" This appendix provides a detailed discussion of the most commonly used monitors in the Open Verification Library.

Appendix B, "PSL Property Specification Language" This appendix provides a detailed discussion of the most commonly used keywords and operators in the PSL property specification language.

Appendix C, "SystemVerilog Assertions" This appendix provides a detailed discussion of the most commonly used keywords and operators in the proposed SystemVerilog standard.

New in Second Edition

Differences between the first edition and the second edition include:

- Updates to the manuscript based on newer versions of standards
- Corrections to errata identified during reviewer feedback
- New material that presents techniques on how to avoid common ambiguity errors
- New material that discusses high-level requirements modeling for specification

Since the first edition was published, subtle changes occurred in a few of the standards we originally presented. In this edition, we have updated the manuscript to be in line with the Accellera SystemVerilog 3.1a and the Accellera PSL 1.1 proposed standards.

In addition, we have compiled feedback from multiple reviewers, and made the appropriate corrections. And we simplified the coding of a number of the examples and fixed known errors.

Finally, we added new material, Chapter 8, "Specifying Correct Behavior". This chapter presents a set of clarifying ideas (that is, tips) about a set of common ambiguities that arise when interpreting a natural language specification and suggest techniques to avoid these common errors. In addition, this chapter discusses the limitations of temporal property languages and the need for additional modeling to overcome these limitations when attempting to specify higher-level requirements.

Acknowledgements

The authors wish to thank the following people who participated in discussions, made suggestions and other contributions to our *Assertion-Based Design* project:

Salim Ahmed, Johan Alfredsson, Tom Anderson, Yann Antonioli, Roy Armoni, Brian Bailey, Lionel Bening, Janick Bergeron, Dino Caporossi, Michael Chang, KC Chen, Carina Chiang, Ashwini Choudhary, Edmond Clarke, Claudionor Coelho, Ben Cohen, Cliff Cummings, Bernard Deadman, Kashyap Doerah, Surrendra Dudani, Cindy Eisner, Jeffrey Elbert, E. Allen Emerson, Limor Fix, Tom Fitzpatrick, Dana Fisman, Peter Flake, Jeanne Foster, Gary Gostin, Faisal Haque, John Havlicek, Bert Hill, Richard Ho,

Ramin Hojati, Alan Hu, Alan Hunter, C. Norris Ip, Tony Jones, Yaron Kashai, Kathryn Kranen, Avner Landver, James Lee, Amir Lehavot, Andy Lin, Lawrence Loh, Joseph Lu, Adriana Maggiore, Erich Marschner, Johan Mårtensson, David Matt, Anthony McIsaac, Steve Meier, Hillel Miller, Prakash Narain, Avigail Orni, Doug Perry, Gary Pimentel, Carl Pixley, Andy Piziali, David Price, Jeff Quigley, Bahman Rabii, Rajeev Ranjan, Joe Richards, Sitvanit Ruah, Vigyan Singhal, Sean Smith, Michal Siwinski, Sandeep Shukla, Bassam Tabbara, Sean Torsney, Andy Tsay, Mike Turpin, David Van Campenhout, Gal Vardi, Moshe Vardi, Paul Vogel, Tony Wilcox, Yaron Wolfsthal, Howard Wong-Toi.

Special thanks to Cliff Cummings, Lionel Bening, Avner Landver, Erich Marschner, Gary Pimentel, Andy Piziali, Joe Richards, Paul Vogel, Tony Wilcox.

Finally, a very special thanks to Jeanne Foster for providing high quality editing advice and services throughout this project.