Creating Assertion-Based IP

Series on Integrated Circuits and Systems

Series Editor: Anantha Chandrakasan
Massachusetts Institute of Technology
Cambridge, Massachusetts

Creating Assertion-Based IP
Harry D. Foster and Adam C. Krolnik
ISBN 978-0-387-36641-8

Design for Manufacturability and Statistical Design: A Constructive Approach
Michael Orshansky, Sani R. Nassif, and Duane Boning
ISBN 978-0-387-30928-6

Low Power Methodology Manual: For System-on-Chip Design
Michael Keating, David Flynn, Rob Aitken, Alan Gibbons, and Kaijian Shi
ISBN 978-0-387-71818-7

Modern Circuit Placement: Best Practices and Results
Gi-Joon Nam and Jason Cong
ISBN 978-0-387-36837-5

CMOS Biotechnology
Hakho Lee, Donhee Ham and Robert M. Westervelt
ISBN 978-0-387-36836-8

SAT-Based Scalable Formal Verification Solutions
Malay Ganai and Aarti Gupta
ISBN 978-0-387-69166-4, 2007

Ultra-Low Voltage Nano-Scale Memories
Kiyoo Itoh, Masashi Horiguchi and Hitoshi Tanaka
ISBN 978-0-387-33398-4, 2007

Routing Congestion in VLSI Circuits: Estimation and Optimization
Prashant Saxena, Rupesh S. Shelar, Sachin Sapatnekar
ISBN 978-0-387-30037-5, 2007

Ultra-Low Power Wireless Technologies for Sensor Networks
Brian Otis and Jan Rabaey
ISBN 978-0-387-30930-9, 2007

Sub-Threshold Design for Ultra Low-Power Systems
Alice Wang, Benton H. Calhoun and Anantha Chandrakasan
ISBN 978-0-387-33515-5, 2006

High Performance Energy Efficient Microprocessor Design
Vojin Oklibdzija and Ram Krishnamurthy (Eds.)
ISBN 978-0-387-28594-8, 2006

Abstraction Refinement for Large Scale Model Checking
Chao Wang, Gary D. Hachtel, and Fabio Somenzi
ISBN 978-0-387-28594-2, 2006

A Practical Introduction to PSL
Cindy Eisner and Dana Fisman
ISBN 978-0-387-35313-5, 2006

Thermal and Power Management of Integrated Systems
Arman Vassighi and Manoj Sachdev
ISBN 978-0-387-25762-4, 2006

Continued after index

Harry D. Foster • Adam C. Krolnik

Creating Assertion-Based IP

Springer

Harry D. Foster
Mentor Graphics
Plano, TX
USA

Adam C. Krolnik
LSI Logic Corporation
Allen, TX
USA

Library of Congress Control Number: 2007937035

ISBN 978-0-387-36641-8 e-ISBN 978-0-387-68398-0

Printed on acid-free paper.

9 8 7 6 5 4 3 2 1

springer.com

Dedications

Dedicated to the most wonderfully magical person in my life—Jeanne. And to Elliott & Stephanie, Lance, and Hannah. Always remember, in this world of automation, there is no substitute for thinking.

-Harry

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

-Adam

TABLE OF CONTENTS

FOREWORD

The design of systems in the 21st century is necessarily one of decomposition of product requirements into functional subsystems. Often these subsystems are implemented with a mix of commercial and in-house design IP. In order to fit into a modern design flow, each design IP should be accompanied by a verification IP (VIP) and a verification plan IP (VPIP). If an electronic system level (ESL) design flow is employed, the VIP and VPIP are employed twice in the flow: during post-partitioning verification and implementation verification. In post-partitioning verification, the behavior of the abstract hardware and software models is demonstrated to conform to their functional specifications. During implementation verification, the behavior of the RTL and embedded production software is demonstrated to conform with the behavior of their corresponding abstract models, as well as with implementation requirements captured in design specifications. Hence, the role of VIP and VPIP is becoming increasingly important, requiring rigorous development processes.

Verification IP is available across a matrix of choices, ranging from dynamic to static verification IP, to language-specific and language-neutral verification components, to application-specification property sets. Dynamic verification IP runs in a simulation environment that exercises the design under verification (DUV), checks that its response conforms with the functional and design specifications, and measures verification progress against metrics defined in its associated verification plan. Static verification IP foregoes the simulation environment in favor of using formal analysis to demonstrate that a design does not violate any of its declared properties. Language-specific verification components—such as Cadence ***e***VCs—and language-neutral verification components—such as Mentor 0-In CheckerWare—bundle the stimulus generation,

checking, and coverage measurement aspects into an application-specific verification environment. An application-specific property set, such as Synopsys AHB AIP, includes all of the properties that any implementation must satisfy and is usually suitable for both formal analysis as well as simulation.

This book by Harry Foster and Adam Krolnik reduces to process the creation of one of the most valuable kinds of VIP: assertion-based VIP. Assertion-based VIP is trailblazing a path toward property-based design, where a property set is written and assembled from a functional specification, partitioned abstract models are synthesized from the properties, and RTL and software components are synthesized from the abstract models. Today, as an essential element of the verification environment, assertions succinctly capture design requirements without the need for error-prone reference models and comparison logic. The authors address the process of designing and implementing simulation- and assertion-based VIP by first defining their terminology in chapter two and then introducing the steps to be followed for this VIP development in chapter three. In each of the subsequent chapters, the influence of a particular kind of design IP on its VIP counterpart is addressed: bus-based designs in chapter four, interfaces in chapter five, arbiters in chapter six, controllers in chapter seven, and finally, datapaths in chapter eight. The Open Verification Methodology (OVM) is employed throughout, enabling VIP structured in this fashion to interoperate with all other OVM VIP and to properly run on all SystemVerilog simulators.

I am honored to be given the pleasure of introducing you, the reader, to this long-awaited book. Harry, Adam, and I shared the burden of verifying a MIMD minisupercomputer in the last century. Out of that experience and many others, you will benefit from the authors' years of experience in the application of assertions for design verification. This book will serve as a valuable reference for years to come.

Andrew Piziali, Sr. Design Verification Engineer
Co-Author, *ESL Design and Verification: A Prescription for Electronic System Level Methodology*
Author, *Functional Verification Coverage Measurement and Analysis*

PREFACE

Within the last decade, we have seen significant interest in assertion-based techniques, which have moved beyond the bounds of academic discussions into the realm of industry application. With the recent standardization of assertion and property languages, such as the IEEE Property Specification Language (PSL) and SystemVerilog Assertions (SVA), we have also seen the development of new assertion-based design and verification technologies, which have opened new EDA markets by providing automated solutions to many verification challenges (for example, debugging, coverage specification, and intent validation).

As interest in assertion-based techniques has grown, a myriad of books have emerged that focus predominately on teaching syntax and semantics for these new assertion language standards. Often, the examples provided in these books focus on implementation-level assertions. That is not to say it is worthless to add implementation-level assertions to a design. On the contrary, plentiful industry-published data and project statistics tout their benefits. Yet, one of the criticisms we received after completing our *Assertion-Based Design* [Foster et al., 2004] book, was that our assertion patterns and cookbook examples tended to focus predominately on implementation-level assertion examples. Although this was helpful for the design engineer (it provided an aid to learning how to write embedded RTL assertions), it was of less value to the verification engineer who was interested in validating higher-level or black-box behaviors. Hence, the focus of this book is to bring the assertion discussion up to a higher level and introduce a

process for creating effective, reusable, assertion-based IP, which easily integrates with the user's existing verification environment (that is, testbench infrastructure).

We believe that assertion-based IP is much more than a comprehensive set of related assertions. It is a full-fledged reusable and configurable verification component, which is used to detect both interesting and incorrect behaviors. Upon detecting interesting or incorrect behavior, the assertion-based IP alerts other verification components within a simulation environment, which are responsible for taking appropriate action.

The guiding principles we are promoting in this book when creating an assertion-based IP monitor are:

- *modularity*—assertion-based IP should have a clear separation between detection and action
- *clarity*—assertion-based IP should be written initially focusing on capturing intent (versus optimizations)

No doubt, our application of these principles will seem fairly radical and even foreign to most people familiar with assertion techniques. Yet, our belief is that modularity facilitates reusable verification components and simplifies maintenance. We are promoting a clear separation of detection from action when creating assertion-based IP so we do not restrict its use. This enables support for many different types of verification components and testbench architectural features. For example, by clearly separating detection from action, the testbench can be constructed to support error injection without having to modify the assertion-based IP. When the assertion-based IP detects an error, it can alert other verification components within the environment, which can then take an appropriate action. This arrangement facilitates reuse.

The concepts presented in this book are drawn from the authors' experience developing assertion-based IP, as well as general assertion-based techniques. We hope this text will foster discussion and debate, and ultimately contribute to the growing body of work related to new assertion-based techniques and verification IP.

Open Verification Methodology

The Open Verification Methodology (OVM) is the first, true, system-level-to-RTL verification methodology that allows you to apply leading-edge verification technologies to designs at multiple levels of abstraction, using multiple languages. The OVM provides libraries of base classes and modules in open-source form and uses TLM interfaces as the communication mechanism between verification components. Although the focus of this book is not specifically on the details of the OVM, throughout the text we do demonstrate how modern assertion-based IP can be constructed to communicate with other verification components within a contemporary testbench. Specifically, our assertion-based IP communication mechanism is created using the Advanced Verification Methodology (AVM) [Glasser *et al*., 2007], which is a subset of the newly formed OVM. As the new OVM begins to mature, naming conventions might vary slightly from the examples we provide. However, you will see that we actually need very few OVM base-class library function calls to establish communication between our assertion-based IP and other verification components, and AVM backwards compatibility is currently a goal of the new OVM.

Acknowledgements

The idea for this book actually started after the completion of our *Assertion-Based Design* book, and has taken on many different forms (or different books) during the course of its evolution. The authors wish to thank Cindy Eisner, Dana Fisman, and Erich Marschner, who made many recommendations on a very different version of this book, which they would likely not even recognize today. In addition, the authors wish to thank Kenneth Larsen, Joe Richards, and Vibarajan Viswanathan, who reviewed various versions of the manuscript and provided invaluable feedback. The authors especially thank Andrew Piziali for not only writing the Foreword, but also providing a very detailed review of the manuscript, and participating in many enlightening lunch-time and E-mail discussions on its

evolution. Finally, we wish to thank the Mentor Graphics VTECH team for providing an indepth understanding of transaction-level modeling and contemporary testbenches. Many of the architect symbols, definitions, and testbench concepts presented in Chapter 2 are derived from the Advanced Verification Methodology Cookbook developed by the VTECH team.

Inspiration never happens in a vacuum. Over the course of our careers there have been many people who directly or indirectly influenced our thoughts related to the topic of assertion-based IP. Hence, we wish to thank the following people for their inspiration over the years: Johan Alfredsson, Yann Antonioli, Roy Armoni, Brian Bailey, Lionel Bening, Janick Bergeron, Eduard Cerny, Edmund Clarke, Claudionor Coelho, Abhishek Datta, Rich Edelman, Cindy Eisner, E. Allen Emerson, Tom Fitzpatrick, Dana Fisman, Mark Glasser, John Havlicek, Alan Hu, Norris Ip, Jan Johnson, Kenneth Larsen, Lawrence Loh, Erich Marschner, Johan Mårtensson, Carl Pixley, Andrew Piziali, Duaine Pryor, Dave Rich, Joe Richards, Adam Rose, Vigyan Singhal, Sundaram Subramanian, Mike Turpin, Moshe Vardi, Vibarajan Viswanathan, and Ping Yeung.

Finally, a very special thanks to Jeanne Foster for providing high-quality editing suggestions throughout this project.