

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.

© 2008 Springer Science+Business Media, LLC

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden. The use in this publication of trade names, trademarks, service marks and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

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

Chapter 1 Introduction	1
1.1 Assertion-Based IP	1
1.1.1 Stakeholders	5
1.1.2 Levels of abstraction	7
1.1.3 Reuse	8
1.2 Properties and assertions	9
1.2.1 Languages	10
1.2.2 Libraries	11
1.2.3 Simulation vs. formal verification	13
1.3 Who should read this book?	14
1.4 Book organization	15
1.5 Summary	17
 Chapter 2 Definitions and Terminology	 19
2.1 Notation	20
2.1.1 Components	20
2.1.2 Interfaces	21
2.1.3 Interconnect	22
2.1.4 Channels	23
2.1.5 Analysis ports	24
2.2 Verification component description	25
2.3 Verification component organization	26
2.4 Definitions	31
2.5 Acronyms	35
2.6 Summary	36
 Chapter 3 The Process	 37
3.1 Guiding principles	38
3.2 Process steps	39
3.3 Assertion-based IP architecture	41
3.3.1 Module-based assertion IP	44
3.3.2 SystemVerilog interface	45

3.3.3 Analysis ports	50
3.3.4 Interface-based assertion IP	53
3.4 Guidelines and conventions	57
3.5 Summary	57
Chapter 4 Bus-Based Design Example	59
4.1 Bus-based design overview	60
4.2 Summary	61
Chapter 5 Interfaces	63
5.1 Simple generic serial bus interface	64
5.1.1 Block diagram interface description	64
5.1.2 Overview description	65
5.1.3 Natural language properties	67
5.1.4 Assertions	68
5.1.5 Encapsulate properties	75
5.2 Simple generic nonpipelined bus interface	75
5.2.1 Block diagram interface description	76
5.2.2 Overview description	77
5.2.3 Natural language properties	81
5.2.4 Assertions	81
5.2.5 Encapsulate properties	89
5.3 Simple generic pipelined bus interface	89
5.3.1 Block diagram interface definition	90
5.3.2 Overview description	92
5.3.3 Natural language properties	96
5.3.4 Assertions	98
5.3.5 Encapsulate properties	108
5.4 Interface monitor coverage example	108
5.5 Summary	112
Chapter 6 Arbiters	113
6.1 Arbitrations schemes	114
6.1.1 Fair arbitration	115
6.1.2 Specific arbiter properties	117
6.1.3 Fixed priority	122
6.1.4 Credit-based weighted priority	125
6.1.5 Dynamic priority	128
6.1.6 Interleaving request	132
6.2 Creating an arbiter assertion-based IP	134
6.2.1 Block diagram interface description	134
6.2.2 Overview description	135
6.2.3 Natural language properties	136
6.2.4 Assertions	136

6.2.5 Encapsulate properties	141
6.3 Summary	142
Chapter 7 Controllers	145
7.1 Simple generic memory controller	146
7.1.1 Block diagram interface description	147
7.1.2 Overview description	148
7.1.3 Natural language properties	158
7.1.4 Assertions	161
7.1.5 Encapsulate properties	173
7.2 Summary	174
Chapter 8 Datapath	177
8.1 Multiport register file	179
8.1.1 Block diagram interface description	179
8.1.2 Overview description	181
8.1.3 Natural language properties	183
8.1.4 Assertions	184
8.1.5 Encapsulate properties	198
8.2 Data queue	198
8.2.1 Block diagram interface description	199
8.2.2 Overview description	200
8.2.3 Natural language properties	203
8.2.4 Assertions	204
8.2.5 Encapsulate properties	213
8.3 Data error correction	214
8.3.1 Block diagram interface description	216
8.3.2 Overview description	216
8.3.3 Natural language properties	218
8.3.4 Assertions	218
8.3.5 Encapsulate properties	225
8.4 Data compression	225
8.4.1 Block diagram interface description	226
8.4.2 Overview description	227
8.4.3 Natural language properties	228
8.4.4 Assertions	230
8.4.5 Encapsulate properties	238
8.5 Data decompression	239
8.5.1 Block diagram interface description	240
8.5.2 Overview description	240
8.5.3 Natural language properties	242
8.5.4 Assertions	243
8.5.5 Encapsulate properties	251
8.6 Summary	251

Appendix A Quick Tutorial For SVA	253
A.1 SVA fundamentals	253
A.1.1 Immediate assertions	254
A.1.2 Concurrent assertions	255
A.1.3 Resets	256
A.1.4 Action blocks	257
A.2 SystemVerilog sequences	258
A.2.1 Consecutive repetition	260
A.2.2 Goto repetition	261
A.2.3 Nonconsecutive repetition	261
A.2.4 Declared named sequences	262
A.3 Property declarations	263
A.4 Sequence and property operators	264
A.4.1 AND	264
A.4.2 Sequence intersection	265
A.4.3 OR	265
A.4.4 Boolean throughout	266
A.4.5 Sequence within	266
A.4.6 Sequence ended	267
A.4.7 Sequence first_match	267
A.4.8 Implication	268
A.5 SVA system functions and task	268
A.5.1 Sampled value functions	269
A.5.2 Useful functions	270
A.5.3 System tasks	271
A.6 Dynamic data within sequences	272
A.7 SVA directives	273
A.8 Useful named property examples	274
 Appendix B Complete	
OVM/AVM Testbench Example	275
B.1 OVM/AVM Example Source Code	276
B.1.1 top module	278
B.1.2 tb_clock_reset module	278
B.1.3 pins_if interface	279
B.1.4 tb_monitor module	280
B.1.5 tb_env class	284
B.1.6 tb_tr_pkg package	287
B.1.7 tb_transaction class	287
B.1.8 tb_status class	290
B.1.9 tb_transactor package	292
B.1.10 tb_stimulus class	292
B.1.11 tb_driver class	293
B.1.12 tb_responder class	297

B.1.13 tb_coverage class 299

B.2 OVM/AVM high-level reference guide 301

Bibliography.....305

Index.....309

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 (VIP). If an electronic system level (ESL) design flow is employed, the VIP and VIP 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 VIP 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 *eVCs*—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.