SystemVerilog Assertions Handbook

… for Formal and Dynamic Verification

Published by:
VhdlCohen Publishing
P.O. 2362
Palos Verdes Peninsula CA 90274-2362
vhdlcohen@aol.com
http://www.vhdlcohen.com

Library of Congress Cataloging-in-Publication Data
A C.I.P. Catalog record for this book is available from the Library of Congress

SystemVerilog Assertions Handbook
… for Formal and Dynamic Verification
ISBN 0-9705394-7-9

Printed on acid-free paper

Printed in the United States of America

Contents

All code is available for download

ch1/
ch1/cookie.sv
ch1/reqack.sv
ch1/wire_explorer.sv

ch2/
ch2/handshake2_1_3.sv
ch2/handshake2_2_1.sv
ch2/handshake2_2_1b.sv
ch2/handshake2_3_2.sv
ch2/hburstmode2_1_6.sv
ch2/interface2_3_2.sv
ch2/packets2_3_1b.sv
ch2/pushpop2_3_1.sv

ch3/
ch3/cache3_2.sv
ch3/clocking3_2_5_1.sv
ch3/formal_3_2_3.sv
ch3/if3_2_5_3_1.sv
ch3/pdata3_2_4.sv
ch3/sample3_1_1.sv

ch4/
ch4/ended4_7_1.sv
ch4/event4_7_4.sv
ch4/firstmatch2.sv
ch4/match3ed4_7_2.sv
ch4/repetitions4_5.sv
ch4/repetitions4_5_1_2.sv
ch4/throughout4_6_6.sv
ch4/triggered4_7_3.sv
ch4/within4_6_5.sv

ch5/
ch5/mixed_vhdl_sv_section5_14/
ch5/mixed_vhdl_sv_section5_14/
bind.sv
ch5/mixed_vhdl_sv_section5_14/
makefile
ch5/mixed_vhdl_sv_section5_14/
reqack.vhd
ch5/mixed_vhdl_sv_section5_14/
reqack2_tb.vhd
ch5/mixed_vhdl_sv_section5_14/
reqack3_tb.sv
ch5/mixed_vhdl_sv_section5_14/
reqack_sva.sv
ch5/mixed_vhdl_sv_section5_14
reqack_tb.vhd
ch5/oneshot5_7.sv
ch5/psl_like5_7.sv
ch5/recursion5_6.sv
ch5/refinement5_4.sv
ch5/req2send_5_3.sv
ch5/sampled5_8.sv
ch5/section5_12.sv
ch5/section5_13.sv
ch5/section5_8.sv
ch5/section5_9.sv
ch5/unbounded5_5.sv
ch5/unexpectedA_5_2.sv
ch5/unexpectedB_5_2.sv

ch6/
ch6/fifo_assert_control.sv
ch6/fifo_if.sv
ch6/fifo_if_withQ.sv
ch6/fifo_pkg_include.sv
ch6/fifo_props.sv
ch6/fifo_rtl.sv
ch6/fifo_tb.sv
ch6/vpi_sect6_2_7.sv

ch7/
ch7/tlightfail/
ch7/tlightfail/tb_bc.sv
ch7/tlightfail/trafficlight080404.sv
ch7/tlightok/
ch7/tlightok/trafficlightok10avg.sv
ch7/tlightok/trafficlightok10csv.sv
ch7/tlightok/trafficlight_props.sv
ch7/tlightok/trafficlight_top.sv

ch8/
ch8/section8_2.sv
ch8/section8_2_5_10.sv
ch8/section8_2_5_4.sv
ch8/section8_2_5_5.sv
ch8/section8_2_5_9.sv

ch9/
ch9/section9_15.sv
ch9/section9_16.sv
ch9/testmodels/
ch9/testmodels/bus_xfr_data_inte
grity_check.sv
ch9/testmodels/d17.sv
ch9/testmodels/d1819.sv
ch9/testmodels/d1to16.sv
ch9/testmodels/d20.sv
ch9/testmodels/d21.sv
ch9/testmodels/d22.sv
ch9/testmodels/d23.sv
ch9/testmodels/d24.sv
ch9/testmodels/d25.sv
ch9/testmodels/d26.sv
ch9/testmodels/d27.sv
ch9/testmodels/d281.sv
ch9/testmodels/d282.sv
ch9/testmodels/d29.sv
ch9/testmodels/
memory_data_integrity_check.sv

appendixA/
appendixA/parkinglot.sv
appendixA/slave_handshake.sv
appendixA/slave_handshake_tb.sv

FOREWORD, Surrendra A. Dudani

The study of assertions has a range of applications in hardware design verification, including bug detection in simulation and emulation, formal proofs of design correctness, functional coverage of complex behaviors, and constraint-based random stimulus generation. Assertions offer improvements at every stage of design and verification process. To guide users of assertions, the authors have previously contributed to the subject of assertions, its importance in design verification and provided numerous examples illustrating its usage.

SystemVerilog Assertions Handbook introduces SystemVerilog's language of assertions from the point of view of practitioners that primarily consist of design and verification engineers. SystemVerilog language evolved from Verilog hardware description language as an industry standard language to describe hardware design as well as to write verification programs supporting the enormous task of verifying the design. The capability to encapsulate design and verification in one language is invaluable to thousands of engineers who have the formidable challenge of designing and verifying present day complex hardware systems. One of the unique features of SystemVerilog is the ability to freely express the interaction of the behavior of sequences and assertions with other features of the language that direct stimulus generation and coverage.

The past few years have brought a shift in design verification methodology, as new techniques have emerged from the work of research and academic institutions to industries that promoted them as matured products providing improvements in quality, productivity, and management of hardware design projects to successful completion in a reasonable timeframe. No longer an engineer thinks only of writing tests and checking the results of simulation as the basis of verification. The art of checking the sanity of results has been formalized into assertions, expressed in concise language form that has a mathematical foundation to also allow formal proof techniques. Despite its compelling benefits, assertion based verification has not reached the desks of many engineers due to its overwhelming expressive power in the language features and a new style of specification. The need of books, guides, and training in informing and instructing engineers with these new concepts is plainly evident.

The authors of this book, Ben Cohen, Srinivasan Venkataramanan and Ajeetha Kumari, are well experienced in the real world of design verification to present a solid introduction to assertions in a practical manner to captivate many engineers who may otherwise be reluctant to take up a new subject. Although hardware engineers are well versed in parallel behaviors interlaced with timing subtleties, assertion language features bring these together in a new and concise form of expressing. The authors have illustrated these concepts in a step-by-step manner with practical examples to relate the outcome of language constructs with the results intuitively expected by the readers. Before delving into the details of the language, a chapter is devoted to acquaint the reader with assertions, its methodology of usage and benefits. Later, the chapters are organized to bring out simpler to more advanced features of expressing sequences, properties and assertions. The depiction of design process using SystemVerilog helps the reader to step back from the language semantics and view the role of assertions in various verification tasks.

The readers of this book will benefit from the clear presentation of concepts together with practical examples and appropriate usage of the language features. Design engineers will find a wealth of easy to apply assertions as checkers to improve the quality of their day-to-day design projects. On the other hand, verification engineers will learn advanced concepts to simplify writing temporal behaviors at the system level to perform system level checking. Further, the book will also assist architects of methodology in deploying advanced verification techniques using SystemVerilog Assertions. This book is a guide much needed to fully capitalize many benefits offered by SystemVerilog Assertions.

Surrendra A. Dudani
Synopsys Scientist
http://www.synopsys.com/

FOREWORD, Stuart Sutherland

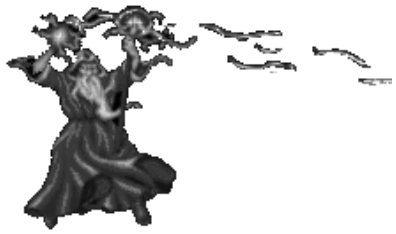
Assertion-based design and verification is an absolute necessity in today's large, complex designs. For that matter, the use of assertions applies even to the simplest of designs. *Every design engineer should be adding assertion checks to his design!* Assertions both document and check the design engineer's assumptions and expectations about the design functionality. *Every verification engineer should be taking full advantage of assertions!* Assertions can dramatically decrease the amount of effort required to define intelligent, self-checking testbenches, and, at the same time, increase the effectiveness of the testbench. As this book shows, assertions offer countless other benefits to both the design engineer and the verification engineer.

The new and exciting SystemVerilog standard adds hundreds of powerful extensions to the IEEE Verilog language standard. Prominent among these extensions is a native assertion language that is fully compatible with the existing Verilog language. SystemVerilog Assertions, abbreviated as SVA, syntactically and semantically fit into Verilog code. Engineers can directly specify assertions in their Verilog models and testbenches, without having to hide the assertions within comments, pragmas or conditional compilation directives. However, SVA is a very rich language in its own right, and is not simple to adopt. SVA has the ability to concisely describe the expected (or unexpected) results of extremely complex sequences of changes within a design. There are a number of conference papers, and even some books, that discuss SystemVerilog Assertions. These papers and books discuss the importance of SVA, and how to use an assertion based verification methodology in design projects. However, I have yet to find a paper or book that teaches how to *write* SystemVerilog Assertions.

This book fills that void. It introduces the concepts and importance of assertion-based verification, and then goes into great depth on how to write both simple and complex assertions using the SystemVerilog Assertions language. Hundreds of examples illustrate the proper usage of SVA. Many of the examples are based on real-world designs. The examples do more than just illustrate how to write an assertion. The examples serve as a cookbook of assertions that can be applied to a variety of designs.

I have been involved with the definition of the SystemVerilog standard since its inception, and am excited to see this great book on SystemVerilog Assertions. My company, Sutherland HDL, Inc., provides expert training and consulting on Verilog and SystemVerilog. We are very active in helping companies adopt SystemVerilog in their current and upcoming design projects. This book will be a valuable tool that we will make full use of in our training workshops and consulting work.

I expect that every design and verification engineer will find that this book is an essential resource in their day-to-day work.



Stuart Sutherland,
Sutherland HDL, Inc.
http://www.sutherland-hdl.com

FOREWORD, Harry D. Foster

An industry-wide effort has been underway for the past few years to extend the capabilities of the Verilog language. The fruit of that labor is SystemVerilog. The SystemVerilog extensions include enhancements in modeling as well as new features for verification. One of the key features added to the language is the ability to specify formal properties as assertion checks and as elements of a functional coverage model.

So why is this important? With the advent of design reuse and IP-based SoC design, two verification challenges have emerged: verifying that the IP complies with its specification and verifying that the IP is interoperable with other compliant devices (that is, adheres to various interface standards). Although the act of specification is fundamental to the process of functional verification, historically, the process of specification has consisted of creating a natural language description for a set of design requirements. And unfortunately, this form of specification is both ambiguous and, in many cases, unverifiable (due to the lack of a standard machine-executable representation).

As assertion and property language standards such as SystemVerilog Assertions (SVA) gain a foothold, they address the problem of ambiguities in natural language specification and reduce the time spent in verification. And as a result, IP providers are able to adopt an assertion-based verification (ABV) methodology to provide verifiable forms of IP specification.

Adopting an ABV methodology benefits both IP producers and consumers. For the IP producer, developing an assertion-based specification for the IP has a collateral benefit—the formal specification process often uncovers misconceptions about the implementer's original intent. Thus, the time invested in developing the specification is time well spent. And generally, the benefits are realized early in the design and verification cycle, before the IP producer applies any form of verification to the IP. With the assertion-based specification in hand, the IP producer is positioned to verify IP compliance and interoperability. For the IP consumer, an assertion-based specification reduces integration time by unambiguously clarifying proper IP behavior under various configurations, while providing a way to verify the SoC's interoperability with the IP. But perhaps most notably, assertion-based specification benefits both producers and consumers by unifying the verification process with a

single form of specification that can be automatically leveraged across a diverse set of verification tools, such as formal verification, simulation, emulation, and even synthesis.

Although a pool of published data confirms the benefits of adopting an ABV methodology, few guidelines exist for coding effective assertion-based specification. Ben Cohen, Srinivasan Venkataramanan, and Ajeetha Kumari have addressed this challenge by creating an excellent source for mastering the art of assertion-based specification. *SystemVerilog Assertions Handbook* should be a part of every RTL design and verification engineer's library.

Harry D. Foster
Chief Methodologist
Jasper Design Automation

FOREWORD, Tarak Parikh

The growing adoption of the SystemVerilog Assertions language is enabling design and verification engineers to formalize what has been done informally for many years. Assertion-based verification at its essence checks that a design behaves a certain way, and has historically been done mostly in simulation, and using formal verification with various methods and languages.

Assertion-based verification using formal model checking also has been used in the industry, but the solutions have ranged from proprietary assertion languages to standard, but difficult to use languages such as CTL and LTL. While formal model checking is a very powerful method to find corner-case bugs and to completely verify the design meets specific requirements, it has not been widely adopted in the industry until recently. The major barriers with model checking have been the difficulty in writing assertions, as well hard to use tools with inadequate debugging capabilities, not to mention the lack of a standard language.

Now, the existence of a standard assertion language suitable for use with formal model checking, integrated with a design language familiar to all design and verification community makes it much easier for EDA vendors to create tools usable by a wide audience. It also gives the engineer more options, since they are not locked into any particular proprietary solution.

Although SystemVerilog Assertions simplify design verification and makes it much easier to write powerful and portable checks, it is not a panacea for all of the verification challenges. For example, it is not well suited for higher-level data flow checks, such as verifying that a packet put into a system is eventually delivered to the right receiver with the correct data. However, when combined with the ability to run these assertions in simulation, adoption of SystemVerilog Assertions creates a significant improvement in verification productivity.

SystemVerilog Assertions are good for checking that design protocols meet specification, and that critical design functions are not violated. For example, it is simple to write an assertion to check that a bus request always is acknowledged within a certain number of clock cycles. With SystemVerilog Assertions, this sort of check may be used in simulation, or proven using formal model checking tools

such as our *@Verifier* product. The SystemVerilog Assertions also allow debugging tools to provide one unified interface for the creation, verification, and debugging of assertions, regardless if they are used in simulation or formal model checking. Our patent-pending *Assertion Studio* technology provides this sort of interface.

@HDL has strived to make assertion-based verification a usable and productive verification technique. The standardization of SystemVerilog Assertions language further enables our tools to fit seamlessly into the verification flows of our customers.

@HDL is pleased to have been able to help the authors develop this book. While powerful, SystemVerilog Assertions can be complicated. This book reviews the language elements and provides clear examples on what is legal and what is not. It is an excellent resource for the novice and experienced assertion-based verification engineer, and is useful for both learning the language and as a reference when developing SystemVerilog Assertions.

Tarak Parikh
Vice-President, Products
@HDL
http://www.athdl.com/

FOREWORD, Keith Rieken

Ben Cohen, Srinivasan Venkataramanan, and Ajeetha Kumari have once again provided a timely, application-oriented approach to an emerging design methodology. As design practices demand ever-increasing productivity in functional verification, the SystemVerilog language coupled with proven verification methodologies will likely be a cornerstone to improved productivity.

The 0-In Business Unit of Mentor Graphics is similarly committed to improving the overall productivity of existing functional verification processes by employing Assertion-Based Verification (ABV) methods. Having pioneered the commercialization of ABV technologies, 0-In believes SystemVerilog provides another alternative for realizing the productivity improvements associated with ABV. The standardization of a language that enables a sophisticated testbench automation solution, design construct improvements and language-based assertion specification will accelerate the adoption of ABV by the design community.

Mentor Graphics continues to support such standardization efforts through the incorporation of SystemVerilog support into its Scalable Verification solution. Through a combination of the Mentor solution and the applied approaches demonstrated in this *SystemVerilog Assertions Handbook*, teams will be able to quickly realize the benefits of a SystemVerilog ABV solution.

Keith Rieken
Director, Technical Marketing Engineering,
0-In Functional Verification Business Unit,
Mentor Graphics
http://www.mentor.com/

FOREWORD, Yu-Chin Hsu

Advanced languages and assertions are coming of age because of mounting design complexity, shrinking time-to-market, and a leveling off in productivity. The potential of SystemVerilog is not merely in its impressive host of useful modeling elements, but also lies in the benefits of a unifying coherent framework for comprehensive design, verification, and debug methodology. SVA, the assertion temporal language portion of SystemVerilog, draws on the strengths of three languages: PSL from Accellera with roots in Sugar, a language used at IBM, VERA from Synopsys, and OVA from Synopsys, also with origins stemming from the industrial setting at Sun Microsystems. The SVA syntax constructs and semantics are designed to be native to SystemVerilog to make them more easily integrated with the design and testbench, and readily approachable by designers.

Assertions can be used in a variety of roles: constraints, checkers, integration monitors, and for functional coverage. Assertions are a formal means to bridge the gap between design specification and implementation, also a way to consolidate design, verification, and debug. Assertion driven verification is perceived as the enabling methodology for early bug detection, bug source localization, and verification reuse. Given the complexity of SoC designs, and component heterogeneity challenges, Novas has developed an enhanced debug framework targeted at empowering designers to not only manage and sustain – but rather to grow – their creativity in the face of the design challenges. Assertions are a cornerstone of this development framework that revolves around automated powerful analysis and debug, and intuitive and efficient interaction with the user. We see assertions as an ideal entry, interaction, and abstraction mechanism for design and debug that boosts design productivity and understanding.

Ben Cohen and co-authors have published numerous books about design and verification of HDLs. This publication follows on the heels of the book on PSL, and contains many useful comparisons and contrasts. Readers will undoubtedly enjoy the *SystemVerilog Assertions Handbook* for it addresses the language from an application-oriented viewpoint. Ben Cohen, Srinivasan Venkataramanan, and Ajeetha Kumari also point out in this book many subtleties in the language and its underpinnings. With the development of OVA, PSL and now SVA, it is quite conceivable that any one user will likely encounter several of these assertion languages (even within the same project) each with its particular strengths, so a good understanding of the foundation of temporal languages and the particulars of each is well advised.

Yu-Chin Hsu
Vice President, Research and Development
Novas Software, Inc.
http://www.novas.com/

FOREWORD, Alain Raynaud

Fifteen years ago, schematics designers surely must have wondered what went wrong with their design practices when first explained the rules on latch inference for the synthesizable subset of Verilog. You are likely to experience the same puzzled feeling today when reading about the vacuous success of temporal properties. But today, just like fifteen years ago, it would be a huge mistake to dismiss the topic of assertion-based verification (ABV) as specialized or limited in scope.

Designers went from expressing the structure of the design through schematics capture, to writing register transfer level code where the key concept is to define everything that happens one clock cycle at a time. With its assertions, SystemVerilog is the first mainstream unified language to break that clock barrier and allow the expression of relationships across many cycles. This will have a tremendous impact on hardware design in the coming years. Verification is the obvious first methodology to face this new wave, and a large part of this book rightfully focuses on verification. Coverage-driven verification, in particular, is bound to see major improvements and hopefully will receive the attention it deserves. From where I sit, in the emulation side of the verification problem, I'd happily see more emulation cycles being dedicated to running actual interesting application code and fewer cycles spent on blind random regression testing.

Design practices will also evolve gradually as EDA tools start making use of the full power of assertions. We are not that far off from ABD: Assertion-Based Design; and this book touches on that subject as well, making it a must read for every designer who wants to stay ahead of the curve. When those EDA vendors come visit you a year from now to pitch their revolutionary new design tools, you'll already understand the underlying technology and you'll be able to separate the wheat from the chaff.

Ben Cohen, Srinivasan Venkataramanan and Ajeetha Kumari have written a book that will teach you more than you ever wanted to know about SystemVerilog Assertions. It can be put to practical use today and will give you an edge for tomorrow. All the right ingredients and topics are covered: from coding style to reference, from coverage to formal proof. This book not only should be part of every verification engineer's library, but designers ought to be on-board as well.

Alain Raynaud
Technical Director
EVE USA, inc.
http://www.eve-team.com/

PREFACE

The Book

SystemVerilog Assertions Handbook is a follow-up book to *Using PSL/Sugar for Formal and Dynamic Verification 2nd Edition*. It focuses on the assertions aspect of SystemVerilog, along with an explanation of the language concepts along with many examples to demonstrate how SystemVerilog Assertions (SVA) can be effectively used in an Assertion-Based Verification methodology to verify designs written in HDLs like SystemVerilog, Verilog, or VHDL. The integration of assertions in SystemVerilog proves very beneficial for the definition of a verification environment because SystemVerilog is a modern language with powerful and advanced constructs like interfaces, queues, associative array, semaphores, system functions, classes, methods, packages, safe pointers, etc.

This book presents different classes of designs, and demonstrates how SystemVerilog Assertions are used in the design process from requirements document, verification plan, design and verification using simulation and formal verification. Many of the examples use the advanced features of SystemVerilog including packages, interfaces, types, and binding. In addition, synthesizable RTL SystemVerilog code examples were synthesized to demonstrated feasibility. Other features provided in this book are a "dictionary" of English to SystemVerilog Assertions examples, guidelines in the use of SystemVerilog Assertions, and a quick reference guide of the SystemVerilog Assertions syntax. This book represents the collaboration of three authors who are experts in system engineering, architecture, and design and verification with hardware description languages (HDLs) and hardware verification languages (HVLs), along with experience in authoring books, thus bringing more synergism to this *SystemVerilog Assertions Handbook*.

The Intent

One of the reasons that we, the authors, decided to write this handbook on SystemVerilog Assertions is the positive impact that Assertion-based Verification (ABV) is providing, and we believe that SystemVerilog is setting up a new viable and effective standard in the design and verification processes. We also felt that the "assertions" aspect of SystemVerilog needed special emphasis. Thus we decided to maintain the focus of this book on SystemVerilog Assertions, with usage of many of the new features that SystemVerilog provides. We are assuming that the users are familiar with SystemVerilog, and have access to the SystemVerilog LRM and to books that address SystemVerilog language.[1]

---

[1] * *SystemVerilog Language Reference Manual* http://www.systemverilog.org/
* *SystemVerilog For Verification*, Tom Fitzpatrick, Dave Rich, Aturo Salz and Stuart Sutherland, 2005, Springer Springeronline.com
* *SystemVerilog For Design A Guide to Using SystemVerilog for Hardware Design and Modeling* Stuart Sutherland, Simon Davidmann, Peter Flake, KAP, June 2003, ISBN 1-4020-7530-8

Assertion-Based Verification is changing the traditional design process because that methodology helps to formally characterize the design intent and expected operations.[2] ABV also quickens the verification task because it provides feedback at the white-box level.[3] As a formal property specification language, SystemVerilog Assertions facilitate automation of common verification tasks that can be exploited across various verification methodologies.

As designers and consultants/trainers, we experienced many designs that were weakly specified. The RTL modeling lacked information about properties and design characteristics, and that led to difficulties and/or ambiguities in the maintenance and verification processes. A design specification is helpful in defining requirements. However, specifications are generally defined in an informal language, like English. They lack a standard machine-executable representation and cannot be dynamically simulated and/or statically processed by a formal verification tool to ensure compliance to requirements.

Assertion-Based Verification with SystemVerilog Assertions

In a manner similar to *Accellera Property Specification Language* (*PSL*)[4], the assertion aspect of SystemVerilog was developed to address these shortcomings. It gives the design architects a standard means of specifying design properties using a concise syntax with clearly defined formal semantics. Similarly, it enables the RTL designers to capture design intent and assumptions in a verifiable form, while enabling the verification engineers to validate that the implementation satisfies its specification through dynamic (i.e., simulation) and formal verification options. Furthermore, it provides a means to measure the quality of the verification process through the creation of functional coverage models built on formally specified properties. It provides a standard means for hardware designers and verification engineers to rigorously document the design specifications using a machine-executable format.

SystemVerilog with assertions improves the quality of digital designs and helps eliminate defects per the **Six Sigma methodology**[5] because assertions play an important role in a unified verification methodology ranging from requirement definitions through design and verification (see Chapter 6 for discussion on the design process with SystemVerilog Assertions). Assertions express functional design intent and can be used to express assumed input behavior, expected output behavior, or forbidden behavior. Assertions allow the architects or designers to capture the design intent and assumptions in a manner that can be verified in the implementation. Assertions are captured during the development process and are continuously verified throughout the design and verification process. Working in a unified verification methodology, assertions reduce the verification time by detecting bugs earlier, and by isolating where a bug is located (by being closer to the source of error). In addition to detection of property violations, assertions improve the efficiency in a unified methodology by improving reuse, enhancing testbench checking, and capturing coverage information. Per Lionel

---

[2] *Assertion-Based Design, Second Edition*, Harry D. Foster, Adam C. Krolnik, David J. Lacey
June 2004, ISBN 1-4020-8027-1,
The SystemVerilog Verification Methodology Manual (VMM), 2005 Springeronline.com

[3] *Writing Testbenches: Functional Verification of HDL Models*, Janick Bergeron, Kluwer Academic Publishers

[4] http://www.accellera.com
http://www.eda.org/vfv/docs/psl_lrm-1.01.pdf

[5] http://www.isixsigma.com/sixsigma/six_sigma.asp
Six Sigma is a disciplined, data-driven approach and methodology for eliminating defects (driving towards six standard deviations between the mean and the nearest specification limit) in any process -- from manufacturing to transactional, and from product to service.

Benning's[6] experience, designers created fewer initial bugs in the RTL as an ABV methodology forced them to think more clearly and accurately about what to design. Also, properties are more accurate and less prone to misinterpretation than comments in the RTL.

When we were first exposed to SystemVerilog Assertions, we realized its strong potentials in specifying design functional specification requirements and properties in a manner easy to learn, write, and read. We particularly liked the concise syntax of the assertions, which are tightly integrated with SystemVerilog. We also appreciated the rigorously well-defined formal semantics, and expressive power of the language, permitting the formal specification (and documentation) for a large class of real world design properties. Expressing the same functionality with HDLs would require extensive coding with explicit FSM machines.

SystemVerilog Tool Support
Today, many companies are supporting SystemVerilog with assertions, and the list is growing. A list of vendors supporting SystemVerilog is shown at the site shown in the footnote.[7] During the process of writing this book, we had access to tools from Synopsys and @HDL. The intent of this book is to present the general concepts of using SystemVerilog with assertions for dynamic and formal verification in a tool independent manner.

Why ABV with SystemVerilog Assertions?
Assertion-Based Verification moved the traditional design process from an informal RTL coding approach with typically poor documentation to a process that provides the following benefits: 1) addresses and documents design decisions; 2) documents design properties and assumptions; 3) addresses solutions (e.g., interfaces, implied FSMs) to requirements prior to any RTL coding; 4) addresses verification of assertions, which items to watch out for during the design of RTL and testbench; 5) facilitates functional coverage to ensure that simulation addresses complex timing-based corner cases; 6) provides excellent basis for design and verification reviews; 7) simplifies design of testbench reference models or scoreboards, which verifies the correctness of results; 8) guides testbench vectors for conditions to be addressed.

It is important to note that SystemVerilog Assertions define the properties, and are implementation independent. It presents a different viewpoint of the design. The property definitions may imply FSMs in the implementation. However, those properties do not necessarily show any design optimizations, such as the use of don't-care conditions. As the design matures, it may be necessary to revisit the assertions, as they may be too restrictive. In addition, it may also be necessary to add assertions at the functional level. But this experience of tuning the assertions and the design is healthy because it forces users to delve into the requirements and implementation.

Our experience with the usage of SystemVerilog Assertions for front-end design definitions demonstrated that SystemVerilog Assertions are very powerful in the process of delving into design requirements, design architecture, and definition of restrictions imposed by the architecture. We found the property and assertion definitions more expressive and precise than the use of a natural language, e.g., English. The RTL design and verification tasks were greatly simplified as a result of using this assertion-based methodology because it alleviated the need to write a thorough testbench reference model prior to debugging the model. During simulation the assertions immediately alerted us of

---

[6] *Verifiable RTL Design: A Functional Coding Style Supporting Verification Processes in Verilog*, Lionel Benning and Harry Foster, Kluwer Academic Publishers

[7] http://www.synopsys.com/partners/systemverilog/systemverilog_partners.html

design and testbench errors. In fact, we strongly recommend the use of ABV with SystemVerilog on design projects. ABV is a very viable methodology for the definition and verification of designs. We must admit though that at times assertions are very frustrating because they (correctly) insisted that our designs were in error when we believed that we had all the necessary fixes!!!

More about the Book

SystemVerilog Assertions Handbook addresses the practical aspects of understanding and using assertions with SystemVerilog. This is accomplished by first defining the language, in a non-LRM manner with many examples to explain the various syntax and nuances of the language. This is then followed by explaining how SystemVerilog assertions are used in the design process through all phases of the design including system level definition, architectural and verification plans, RTL and testbench designs, dynamic and static verification. This is done by example, using a simple synchronous FIFO as a project model. Formal verification concepts and application of formal verification with SystemVerilog Assertions are then presented, along with an example of a traffic light controller to demonstrate the application of tools, and types of results typically presented by such tools. A set of language and application guidelines emanating from our experience with SystemVerilog Assertions is presented. A "dictionary" of examples demonstrates how various English requirements can be translated into SystemVerilog properties.

Book Organization

Chapter 1 provides an introduction to Assertion-Based Verification. **Chapter 2** serves as an introduction to SystemVerilog Assertions (SVA) concepts with emphasis on properties and assertions. It prepares the readers for Chapters 3, 4, and 5, which represent the "core" of SystemVerilog Assertions. **Chapter 3** delves into understanding properties. **Chapter 4** delves into the understanding and application of sequences that represent the real potential of SystemVerilog Assertions. **Chapter 5** provides a deeper appreciation of SystemVerilog Assertions by addressing advanced topics for properties and sequences. **Chapter 6** addresses the methodologies in using properties / sequences / assertions during the requirement and verification planning phases at the requirement and verification planning levels, in addition to the RTL and testbench levels. It first explains the process, and then demonstrates an application of assertions in the requirements specification and verification plan using a synchronous First-In First-Out (FIFO) as an IP (Intellectual Property) block. SystemVerilog packages, interfaces, modules, and bindings are also demonstrated. **Chapter 7** addresses the formal verification aspects of SystemVerilog Assertions. It focuses on Formal Verification (FV) methodologies for functional verification of RTL designs. It provides a case study using a traffic light controller model. **Chapter 8** provides a summary a rich set of guidelines in using SystemVerilog Assertions. These guidelines emerged from experience with usage of Assertion-Based Verification with Accellera's PSL, vendor's recommendations, code reviews, and LRM documentation. **Chapter 9** represents a "dictionary" of classes of application examples that translate English descriptions of properties to SystemVerilog properties**. Appendix A** provides the answers to the exercises asked at the end of some chapters. **Appendix B i**s a summary of terms and definitions used within this book. **Appendix C** is a SystemVerilog Assertions quick reference guide of the syntax and examples. **Appendix D** represents the SystemVerilog Assertions syntax. The book **Index** provides a page lookup for information available in this book.

Acknowledgements

SystemVerilog Assertions Handbook could not have been written without the support of Synopsys who provided us with SystemVerilog tools including *vcs* and *Magellan* and helped us in the review process.

We thank *Accellera* for granting us permission to reproduce some material from the *Accelera's* SystemVerilog 3.1a Language Reference Manual, the document that defines the rules of SystemVerilog and SystemVerilog Assertions.

Several SystemVerilog experts participated in the review process of this book. The review is a necessary step to iron out areas of disagreements, and to provide a piece of work that meets user's requirements in the use of SystemVerilog Assertions. In that endeavor, we sincerely thank the following people and organizations:

- **Stuart Sutherland**, Stuart Sutherland of Sutherland HDL, Inc., an expert Verilog and SystemVerilog instructor and consultant -- for sharing his SystemVerilog expertise, and providing valuable and constructive comments in the review of our book, and for writing a foreword.[8]
- **Surrendra Dudani,** Synopsys Scientist, for reviewing our book and writing a foreword.
- **Eduard Cerny,** Synopsys Principal Engineer, R&D, VG, for his in-depth review especially into the language part and providing feedback.
- **Badri Gopalan**, Synopsys – for being very supportive and encouraging throughout the process of book writing and for his feedbacks from a experienced Assertions user perspective.
- **Alessandro Fasan**, Synopsys, Product Marketing Manager, functional verification methodology and RTL formal verification, for reviewing the book and for providing constructive comments and suggestions.
- **Bruce Greene**, Synopsys technical marketing manager, for sharing his knowledge and for testing some of our models in Magellan.

[8] **Stuart Sutherland,** Sutherland HDL, Inc., Training Engineers to be Verilog and SystemVerilog Wizards.
http://www.sutherland-hdl.com

- **Cliff Cummings**, Sunburst Design, for providing technical feedback on our book.[9]
- **Harry Foster,** Chief Methodologist , Jasper Design Automation, for reviewing our book and for writing a foreword.[10]
- **Tarak Parikh,** Vice-President, Products**,** @HDL, for reviewing our book, executing some of our SystemVerilog Assertions models with @HDL *@Verifier* and *@Designer* formal verification tools, and for writing a foreword.[11]
- **Keith Rieken**, Director, Technical Marketing Engineering, 0-In Functional Verification Business Unit, Mentor Graphics, for reviewing our book and for writing a foreword.
- **Yu-Chin Hsu**, Vice President, Research and Development, Novas Software, Inc., for providing technical comments in the review of our book and for writing a foreword.[i]
- **Dr. Bassam Tabbara,** Novas Software, Inc., Architect, R&D, for providing technical comments in the review of our book
- **Alain Raynaud**, EVE USA, inc., Technical Director, for reviewing our book and for writing a foreword.[ii]

We also thank *Forte Design Systems*[12] for granting us a license of *TimingDesigner* as a tool to draw timing diagrams for use in this book.

I, Ben Cohen, especially thank my wife, Gloria Jean, for supporting again me in this endeavor.

---

9 **Cliff Cummings,** Sunburst Design, Expert Verilog, SystemVerilog, Synthesis & Verification Training http://www.sunburst-design.com/

10 **Harry D. Foster,** Chair of Accellera Formal Verification Technical Committee and Chief Methodologist at Jasper Design Automation,. Author of *Principles of Verifiable RTL Design,* Kluwer Academic Publishers, Second Edition, 2001, *Assertion-Based Design*, Kluwer Academic Publishers, 2003, and contributor to *Advanced Formal Verification,* Springer (http://www.springeronline.com/)

11 **Tarak Parikh**, VP of Product Engineering, @HDL, http://www.athdl.com/

12 *TimingDesigner* is a flexible, interactive timing analysis and diagram tool. http://www.timingdesigner.com/ http://www.forteds.com/

Sculpture Created by my Wife Gloria to
Express my Long Hours with a Laptop in the Creation of HDL Books

About the Authors

Ben Cohen is currently an HDL and Property languages (PSL, SystemVerilog Assertions) trainer and consultant. He has technical experience in digital and analog hardware design, computer architecture, ASIC design, synthesis, and use of hardware description languages for modeling of statistical simulations, instruction set descriptions, and hardware models. He applied VHDL since 1990 to model various bus functional models of computer interfaces. He authored *VHDL Coding Styles and Methodologies*, first and second editions, and *VHDL Answers to Frequently Asked Questions*, first and second editions, *Component Design by Example*, *Real Chip Design and Verification Using Verilog and VHDL*, and *Using PSL/SUGAR with Verilog and VHDL, Guide to Property Specification Language for ABV (1st Edition*, also translated to Japanese by Cadence*)*, and *Using PSL/Sugar for Formal and Dynamic Verification, 2nd Edition.* He was one of the pilot team members of the VHDL Synthesis Interoperability Working Group of the Design Automation Standards Committee who authored the *IEEE P1076.6 Standard for VHDL Register Transfer Level Synthesis.* He is currently a member of the *VHDL* and *Verilog Synthesis Interoperability Working Group of the Design Automation Standards Committees*, and *Accellera OVL and PSL* standardization working groups. He taught several VHDL and PSL training classes, and provided VHDL consulting services on several tasks.

VhdlCohen Publishing
Email: VhdlCohen@aol.com ben@abv-sva.org
Web: http://www.vhdlcohen.com/

Srinivasan Venkataramanan is currently employed as a Verification Solutions Engineer with Synopsys, India Private Ltd., Bangalore – India. His areas of interest are the emerging verification solutions and methodologies, such as SystemVerilog, Assertion-Based Verification, formal verification, Coverage Driven Verification (CDV), etc. He was actively involved in the verification of leading edge high-speed, multi-million gates ASIC designs. He successfully developed complex verification environments using advanced methodologies, such CDV using Verisity's *Specman*, ABV etc. Prior to joining Synopsys, he worked at Intel, Philips Semiconductors, and RealChip communications in the areas of front-end design and verification of ASICs with several HDLs and HVLs, including VHDL, Verilog, *Specman*, and *Vera*. Srini was active in several technical discussion forums, working committees of OVL, SystemVerilog, VHDL-200X Testbench & Verification, etc. Srini holds a Masters Degree from the prestigious Indian Institute of Technology (IIT), Delhi in VLSI Design, and Bachelors degree in Electrical engineering from TCE, Madurai. Srini has co-authored the book *Using PSL/Sugar for Formal and Dynamic Verification, 2nd Edition.*

Web: http://www.noveldv.com/
Email: srini@abv-sva.org

Ajeetha Kumari is currently an independent consultant in the area of front-end design and verification based in Bangalore, India. She co-authored *Using PSL/Sugar for Formal and Dynamic Verification, 2nd Edition.* Her interests include front-end design and verification methodologies, and she is actively involved in exploring new techniques and tools in this arena. She has also been involved in EDA tool evaluations. She has experience with several HDLs and HVLs including Verilog, VHDL, SystemVerilog, PSL, SystemVerilog Assertions and Vera. She currently maintains a Verification centric web site. Her interests also include mixed-signal design methodologies, which she developed as a follow-up of her research work during her Masters degree. She received her M.S. in Electrical engineering from the prestigious Indian Institute of Technology (IIT), Madras and Bachelors from TCE, Madurai.

NOVEL Design Verification
Email: ajeetha@noveldv.com ajeetha@abv-sva.org
Web : http://www.noveldv.com/