
Using PSL/Sugar for Formal and Dynamic Verification 2nd Edition

*Guide to Property Specification Language for
Assertion-Based Verification*

Ben Cohen
Srinivasan Venkataramanan
Ajeetha Kumari



VhdlCohen Publishing
Los Angeles, California
<http://www.vhdlcohen.com/>

Using PSL/Sugar for Formal and Dynamic Verification 2nd Edition

Guide to Property Specification Language for Assertion-Based 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**

**Using PSL/SUGAR for Formal and Dynamic Verification 2nd Edition
Guide to Property Specification Language for Assertion-Based Verification
ISBN 0-9705394-6-0**

Copyright © 2004 by VhdlCohen Publishing

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without the prior written permission from the author, except for the inclusion of brief quotations in a review.

Printed on acid-free paper

Printed in the United States of America

PREFACE

Using PSL/Sugar for Formal and Dynamic Verification 2nd Edition is a second edition of the book *Using PSL/Sugar with Verilog and VHDL*. This edition puts more emphasis on *PSL* as a language, on applications of *PSL* for different classes of designs, and on formal verification with *PSL*. It also provides a “dictionary” of English to *PSL* examples, along with guidelines in the use of *PSL*, and a definition of the syntax with code samples. This information facilitates the learning of *PSL* by using this book as a guide. 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), thus bringing more synergism to this edition.

One of the reasons that we, the authors, decided to write this second edition is the positive impact that Assertion-based Verification (ABV) is providing in the design process, and we wanted to expand on those attributes. ABV with the Property Specification Language (*PSL*) is changing the traditional design process because that methodology helps to formally characterize the design intent and expected operations. ABV also quickens the verification task because it provides feedback at the white-box level¹. As a formal property specification facility, *PSL* facilitates 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 of the design to requirements.

The *Accellera Property Specification Language (PSL)*² 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 designer to capture design intent and assumptions in a verifiable form,

¹ *Writing Testbenches: Functional Verification of HDL Models*, Janick Bergeron, Kluwer Academic Publishers

² <http://www.accellera.com>

http://www.eda.org/vfv/docs/psl_lrm-1.01.pdf

while enabling the verification engineer 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 also provides a standard means for hardware designers and verification engineers to rigorously document the design specification using a machine-executable format.

PSL is a specification language supported by tools to improve the quality of digital designs and to eliminate defects per the **Six Sigma methodology**³ because *PSL* assertions play an important role in a unified verification methodology ranging from requirement definitions through design and verification (see Chapter 4 for discussion on the design process with *PSL*). Assertions express functional design intent and can be used to express assumed input behavior, expected output behavior, or forbidden behavior. Assertions allow the architect or designer 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 process. Assertions, working in a unified verification methodology, reduce the verification time by detecting bugs earlier and isolating where a bug is located (by being closer to the source of error). In addition to bug detection, assertions improve the efficiency in a unified methodology by improving reuse, enhancing testbench checking, and capturing coverage information. Per Lionel Benning⁴ experience, designers created fewer initial bugs in the RTL as ABV methodology forced them to think more clearly and accurately about what the design. Also, properties are more accurate and less prone to misinterpretation than comments in the RTL.

When we were first exposed to *PSL*, we realized its strong potentials in specifying design functional specification requirements and properties in an easy manner to learn, write, and read. We particularly liked the concise syntax of *PSL*, along with 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. It is interesting to note that hardware verification language (HVL) provided help for automation of the verification environment. Some of these HVLs support assertions through their temporal language subset, however, due to the lack of a standard HVL, it is hard to create assertions/properties that can be understood by more than one vendor. In addition, HVLs are more of “programming languages” than a “specification language”, and hence the authors believe that HVL tools⁵ will in the near future understand *PSL*.

³http://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.

⁴ *Verifiable RTL Design: A Functional Coding Style Supporting Verification Processes in Verilog*, Lionel Benning and Harry Foster, Kluwer Academic Publishers

⁵ Recently Verisity announced a collaboration with 0-in Design automation to support *PSL* based assertions

Today, many companies are supporting *PSL*, and the list is growing. For example (this is by far not a complete list), Cadence supports the *Incisive™ unified simulator* that simulates VHDL and Verilog flavors of *PSL* in a unified environment. Mentor Graphics *ModelSim*⁶ supports *PSL*. @HDL⁷ provides *@Verifier* and *Assertion Studio*. Safelogic⁸ provides *Safelogic Monitor®* and *Safelogic Verifier®*. Simulation is generally the first, and sometimes the only, means for verification. Therefore, having simulation capability of *PSL* with HDLs offers a great advantage in the verification process. There are many companies that make *PSL*-aware simulators, and there are initiatives to develop tools that generate HDL code for the *PSL* properties, thus creating code that is simulator independent⁹.

As mentioned previously, *PSL* is not restricted to simulation only. Many companies are now supporting *PSL* for formal verification. For the development of this book, we were provided access to Cadence's *Incisive™ Static Verifier*¹⁰ and @HDL's *@Verifier*¹¹. However, the list of companies supporting formal verification with *PSL* is growing, and users need to do comparative studies for features and capabilities. The intent of this book is to present the general concepts of using *PSL* for dynamic and formal verification in a tool independent manner.

ABV with *PSL* 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 assertions, which guide 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 model, which verifies the correctness of results; 8) guides testbench vectors for conditions to be addressed.

It is important to note that *PSL* defines the properties, and is implementation independent. *PSL* presents a different viewpoint of the design. *PSL* may imply FSMs in the implementation. *PSL* does 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 *PSL* assertions, as they may be too restrictive. In addition, it may also be necessary to add assertions defined 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.

⁶ <http://www.model.com/products/assertions.asp>

⁷ <http://www.athdl.com>

⁸ <http://www.safelogic.se>

⁹ FoCs is a tool from IBM that reads in Sugar code and generates equivalent HDL code.

¹⁰ <http://www.cadence.com>

¹¹ <http://www.athdl.com>

Our experience with the usage of *PSL* for front-end design definitions demonstrated that *PSL* is very powerful in the process of delving into design requirements, design architecture, and definition of restrictions imposed by the architecture. We found *PSL* more expressive and precise than English for these tasks. The RTL design and verification tasks were greatly simplified as a result of using this assertion-based methodology because *PSL* alleviated the need to write a thorough testbench reference model prior to debugging the model. During simulation *PSL* immediately alerted us of design and testbench errors. In fact, we strongly recommend the use of ABV with *PSL* on design projects, and are now recommending it when we provide HDL training. ABV is a very viable methodology for the definition and verification of designs. We must admit though that at times *PSL* was very frustrating because it (correctly) insisted that our designs were in error when we believed that we had all the necessary fixes!!!

Using PSL/Sugar for Formal and Dynamic Verification addresses the practical aspects of understanding and using *PSL*. 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 *PSL* is 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. Several simple design examples are presented in both flavors of HDL, but some are in Verilog (or VHDL) only. A FIFO and a handshake protocol models are presented to demonstrate the methodologies. Formal verification concepts and application of FV with *PSL* is 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. An AMBA AHB memory slave interface design integrates many of the concepts presented in the previous chapters for the use of *PSL* in the design and verification of a more complex application. Chapter 9 provides a set of *PSL* language and application guidelines, which resulted from our experience with *PSL*. The language BNF and examples is available for reference. In addition, cards are supplied for quick review of the language. A “dictionary” of examples is present to demonstrate how various English requirements can be translated into *PSL* properties.

Most examples were simulated with Cadence *Incisive*TM *unified simulator* using the *PSL*-based implementation of ABV (Dynamic ABV), supported in the Cadence *Incisive*TM verification platform. Many other examples were also statically evaluated using Cadence’s *Incisive*TM *Static Verifier* and *@HDL @verifier*. Synthesizable designs were synthesized with Synplicity’s *Synplify Pro*[®].

**... ABV is to verification
as RTL is to synthesis ...**

All code is available for download

Acknowledgements

Using PSL/SUGAR with Verilog and VHDL could not have been written without the support of Cadence and @HDL who supported with their tools. We also received very valuable comments and answers to technical questions from Cadence, @HDL, Safelogic, Mentor Graphics, and Harry Foster. Their support made a difference.

We thank Cadence Design Systems for granting us licenses of *Incisive™ unified simulator*¹², and for their excellent language support in the practical definition and application of *PSL*. Access to those tools for simulation and formal verification, which support both VHDL and Verilog HDLs, significantly helped us in ensuring accuracy. Cadence Design Systems is a recognized leader in the EDA tool industry. We particularly want to thank Axel Scherer and Thomas A Murray for all their comments and reviews. We also want to thank other Cadence verification team members, including Dave Allen, and Erich Marschner. The depth of their knowledge and willingness to share this information was extremely useful. Cadence was very supportive in our endeavor, and we thank them for that support and for the writing the foreword.

We thank @HDL¹³, specifically Rich Newton, Director of Application Support, and Tarak Parikh, VP of Product Engineering, for their technical support and in granting us licenses of their *@Verifier* and *@Designer* products. Supporting both automatic assertion extraction and user-written *PSL* assertions, these products deliver significant RTL verification productivity improvement, through system-level design analysis and debugging, automatic formal model checking, and tight integration with existing simulation environments. Incorporating their patent-pending *Assertion Studio* technology for developing *PSL* assertions, *@Verifier* can deliver a highly effective ABV product suite supporting assertion development, verification, analysis and debugging. @HDL was very supportive in our endeavor, and we thank them for that support and for the writing the foreword.

¹² Cadence *Incisive unified simulator* supports Verilog®, VHDL, SystemC, SystemC verification (SCV) standard, PSL/Sugar assertions, algorithm development, and analog/mixed-signal verification.

¹³ www.athdl.com

We thank Harry D. Foster for providing valuable comments on the manuscript and for writing the foreword. Harry is the author of several recognizable books in the field of verification, including *Principles of Verifiable RTL Design*, Kluwer Academic Publishers, Second Edition, 2001, *Assertion-Based Design*, Kluwer Academic Publishers, 2003, and contributor to *Advanced Formal Verification*

We thank Safelogic for valuable input to the manuscript and for writing the foreword. We particularly want to thank Øystein Kolsrud and Johan Alfredsson for their comments and reviews. Safelogic provides tools for improved simulation, analysis and verification of

HDL designs. *Safelogic Monitor* is a plug-in to standard simulators enabling property-based verification using *PSL*. *Safelogic Verifier* is a formal property checker that verifies that RTL designs meet requirements formulated as *PSL* properties without using any test vectors. *Safelogic ASG* is an automatic stimuli generator that produces simulation stimuli constrained by *PSL* properties.

We thank Stephen Bailey and Mentor Graphics for supporting in our endeavor, for providing very useful comments, and for the writing the foreword.

We thank *Accellera* for granting us permission to reproduce some material from the *PSL* language reference manual.

We thank *Synplicity* for granting a *Synplify-Pro* license to verify the synthesizability of some models. We also thank *Forte Design Systems*¹⁴ for granting us a license of *TimingDesigner* as a tool to draw timing diagrams for use in this book.

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

We (Aji & Srini) would like to sincerely thank Ben for providing us an opportunity to co-author such a wonderful book as it demands extreme patience on such an experienced professional to include some of the new breeds into his second edition. It has been a real pleasure to work with Ben and understand what makes him such a wonderful author.

I (Srini) would like to thank Tarun Gupta & Tapas Datta of Intel India for helping with the necessary legal formalities required from Intel side to work on this book.

I (Ben Cohen) thank both Aji and Srini for all their efforts and contributions in the writing and editing of this book, and for bringing synergy and a fresh look to this project.

¹⁴ <http://www.cynapps.com/>

About the Authors

Ben Cohen is currently an HDL and *PSL* language 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)*. 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 training classes, and provided VHDL consulting services on several tasks.

VhdlCohen Publishing

Email: VhdlCohen@aol.com

Web: <http://www.vhdlcohen.com/>

Srinivasan Venkataramanan is currently working as Senior Member Technical Staff with Intel India Technology Center, Bangalore – India and has been involved in Verification of leading edge ASICs. He has successfully developed complex verification environments using emerging methodologies such as CDV using Verisity's *Specman*, *ABV* etc. He has worked with Philips Semiconductors and RealChip communications prior to joining Intel. His area of interest has been front-end design and Verification of ASICs and methodologies. Srinu has also been active in several technical discussion forums, working committees of *OVL*, *VHDL-TBV* etc. Srinu holds a Masters degree from prestigious Indian Institute of Technology (IIT), Delhi in VLSI Design and Bachelors degree in Electrical engineering from TCE, Madurai.

Email: sri@noveldv.com

Web: <http://www.noveldv.com/>

Ajeetha Kumari is currently an independent consultant in the area of front-end design and verification based in Bangalore, India. Her interests include front-end design and Verification, and she is actively involved in exploring new methodologies and tools in this arena. She has also been involved in EDA tool evaluations. She currently maintains a Verification centric web site. Her interests also include mixed signal design, 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

Web : <http://www.noveldv.com/>