SystemVerilog For Design

Second Edition

A Guide to Using SystemVerilog
for Hardware Design and Modeling

SystemVerilog For Design
Second Edition

A Guide to Using SystemVerilog
for Hardware Design and Modeling

by

Stuart Sutherland

Simon Davidmann

Peter Flake

Foreword by Phil Moorby

Springer

Stuart Sutherland
Sutherland DHL, Inc.
22805 SW 92nd Place
Tualatin, OR 97062
USA

Simon Davidmann
The Old Vicerage
Priest End
Thame, Oxfordshire 0X9 3AB
United Kingdom

Peter Flake
Imperas, Ltd.
Imperas Buildings, North Weston
Thame, Oxfordshire 0X9 2HA
United Kingdom

SystemVerilog for Design, Second Edition
A Guide to Using SystemVerilog for Hardware Design and Modeling

Library of Congress Control Number: 2006928944

ISBN-10: 0-387-33399-1   e-ISBN-10:   0-387-36495-1
ISBN-13: 9780387333991   e-ISBN-13:   9780387364957

Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2

springer.com

Dedications

To my wonderful wife, LeeAnn, and my children, Ammon, Tamara, Hannah, Seth and Samuel — thank you for all your patience during the many long hours and late nights while writing this book.

Stuart Sutherland
Portland, Oregon

To all of the staff of Co-Design and the many EDA colleagues that worked with me over the years — thank you for helping to evolve Verilog and make its extension and evolution a reality. And to Penny, Emma and Charles — thank you for allowing me the time to indulge in language design (and in cars and guitars...).

Simon Davidmann
Santa Clara, California

To my wife Monique, for supporting me when I was not working, and when I was working too much.

Peter Flake
Thame, UK

Table of Contents

About the Authors

Stuart Sutherland provides expert instruction on using SystemVerilog and Verilog. He has been involved in defining the Verilog language since the beginning of IEEE standardization work in 1993, and is a member of both the IEEE Verilog standards committee (where he has served as the chair and co-chair of the Verilog PLI task force), and the IEEE SystemVerilog standards committee (where he has served as the editor for the SystemVerilog Language Reference Manual). Stuart has more than 20 years of experience in hardware design, and over 17 years of experience with Verilog. He is the founder of *Sutherland HDL Inc.*, which specializes in providing expert HDL training services. He holds a Bachelors degree in Computer Science, with an emphasis in Electronic Engineering Technology. He has also authored *"The Verilog PLI Handbook"* and *"Verilog-2001: A Guide to the New Features of the Verilog HDL"*.

Simon Davidmann has been involved with HDLs since 1978. He was a member of the HILO team at Brunel University in the UK. In 1984 he became an ASIC designer and embedded software developer of real time professional musical instruments for Simmons Percussion. In 1988, he became involved with Verilog as the first European employee of Gateway Design Automation. He founded Chronologic Simulation in Europe, the European office of Virtual Chips (inSilicon), and then the European operations of Ambit Design. In 1998, Mr. Davidmann co-founded Co-Design Automation, and was co-creator of SUPERLOG. As CEO of Co-Design, he was instrumental in transitioning SUPERLOG into Accellera as the beginning of SystemVerilog. Mr. Davidmann is a member of the Accellera SystemVerilog and IEEE 1364 Verilog committees. He is a consultant to, and board member of, several technology and EDA companies, and is Visiting Professor of Digital Systems at Queen Mary, University of London. In 2005 Mr. Davidmann founded Imperas, Inc where he is President & CEO.

Peter Flake was a co-founder and Chief Technical Officer at Co-Design Automation and was the main architect of the SUPERLOG language. With the acquisition of Co-Design by Synopsys in 2002, he became a Scientist at Synopsys. His EDA career spans more than 30 years: he was the language architect and project leader of the HILO development effort while at Brunel University in Uxbridge, U.K., and at GenRad. HILO was the first commercial HDL-based simulation, fault simulation and timing analysis system of the early/mid 1980s. In 2005 he became Chief Scientist at Imperas. He holds a Master of Arts degree from Cambridge University in the U.K. and has made many conference presentations on the subject of HDLs.

List of Examples

This book contains a number of examples that illustrate the proper usage of SystemVerilog constructs. A summary of the major code examples is listed in this section. In addition to these examples, each chapter contains many code fragments that illustrate specific features of SystemVerilog. The source code for these full examples, as well as many of the smaller code snippets, can be downloaded from ***http://www.sutherland-hdl.com***. Navigate the links to ***"SystemVerilog Book Examples"***.

Page xxv of the Preface provides more details on the code examples in this book.

Foreword

by Phil Moorby
The creator of the Verilog language

When Verilog was created in the mid-1980s, the typical design size was of the order of five to ten thousand gates, the typical design creation method was that of using graphical schematic entry tools, and simulation was beginning to be an essential gate level verification tool. Verilog addressed the problems of the day, but also included capabilities that enabled a new generation of EDA technology to evolve, namely synthesis from RTL. Verilog thus became the mainstay language of IC designers.

Throughout the 1990s, the Verilog language continued to evolve with technology, and the IEEE ratified new extensions to the standard in 2001. Most of the new capabilities in the 2001 standard that users were eagerly waiting for were relatively minor feature refinements as found in other HDLs, such as multidimensional arrays, automatic variables and the generate statement. Today many EDA tools support these Verilog-2001 enhancements, and thus provide users with access to these new capabilities.

SystemVerilog is a significant new enhancement to Verilog and includes major extensions into abstract design, testbench, formal, and C-based APIs. SystemVerilog also defines new layers in the Verilog simulation strata. These extensions provide significant new capabilities to the designer, verification engineer and architect, allowing better teamwork and co-ordination between different project members. As was the case with the original Verilog, teams who adopt SystemVerilog based tools will be more productive and produce better quality designs in shorter periods.

A strong guiding requirement for SystemVerilog is that it should be a true superset of Verilog, and as new tools become available, I believe all Verilog users, and many users of other HDLs, will naturally adopt it.

When I developed the original Verilog LRM and simulator, I had an expectation of maybe a 10-15 year life-span, and during this time I have kept involved with its evolution. When Co-Design Automation was formed by two of the authors, Peter Flake and Simon Davidmann, to develop SUPERLOG and evolve Verilog, I was invited to join its Technical Advisory Board and, later, I joined the company and chaired its SUPERLOG Working Group. More recently, SUPERLOG was adopted by Accellera and has become the basis of SystemVerilog. I did not expect Verilog to be as successful as it has been and, with the extensions in SystemVerilog, I believe that it will now become the dominant HDL and provide significant benefits to the current and future generation of hardware designers, architects and verification engineers, as they endeavor to create smaller, better, faster, cheaper products.

If you are a designer or architect building digital systems, or a verification engineer searching for bugs in these designs, then SystemVerilog will provide you with significant benefits, and this book is a great place to start to learn SystemVerilog and the future of Hardware Design and Verification Languages.

Phil Moorby,
New England, 2003

Preface

SystemVerilog, officially the **IEEE Std 1800-2005™** standard, is a set of extensions to the **IEEE Std 1364-2005™ Verilog Standard** (commonly referred to as ***"Verilog-2005")***. These extensions provide new and powerful language constructs for modeling and verifying the behavior of designs that are ever increasing in size and complexity. The SystemVerilog extensions to Verilog can be generalized to two primary categories:

- Enhancements primarily addressing the needs of hardware modeling, both in terms of overall efficiency and abstraction levels.
- Verification enhancements and assertions for writing efficient, race-free testbenches for very large, complex designs.

Accordingly, the discussion of SystemVerilog is divided into two books. This book, ***SystemVerilog for Design***, addresses the first category, using SystemVerilog for modeling hardware designs at the RTL and system levels of abstraction. Most of the examples in this book can be realized in hardware, and are synthesizable. A companion book, ***SystemVerilog for Verification***[1], covers the second purpose of SystemVerilog, that of verifying correct functionality of large, complex designs.

Target audience

NOTE This book assumes the reader is already familiar with the Verilog Hardware Description Language.

This book is intended to help users of the Verilog language understand the capabilities of the SystemVerilog enhancements to Verilog. The book presents SystemVerilog in the context of examples, with an emphasis on correct usage of SystemVerilog constructs. These examples include a mix of standard Verilog code along with SystemVerilog the enhancements. The explanations in the book focus on these SystemVerilog enhancements, with an assumption that the reader will understand the Verilog portions of the examples.

Additional references on SystemVerilog and Verilog are listed on page xxvii.

1. Spear, Chris *"SystemVerilog for Verification"*, Norwell, MA: Springer 2006, 0-387-27036-1.

Topics covered

This book focusses on the portion of SystemVerilog that is intended for representing hardware designs in a manner that is both simulatable and synthesizable.

Chapter 1 presents a brief overview of SystemVerilog and the key enhancements that it adds to the Verilog language.

Chapter 2 discusses the enhancements SystemVerilog provides on where design data can be declared. Packages, $unit, shared variables and other important topics regarding declarations are covered.

Chapter 3 goes into detail on the many new data types SystemVerilog adds to Verilog. The chapter covers the intended and proper usage of these new data types.

Chapter 4 presents user-defined data types, a powerful enhancement to Verilog. The topics include how to create new data type definitions using **`typedef`** and defining enumerated type variables.

Chapter 5 looks at using structures and unions in hardware models. The chapter also presents a number of enhancements to arrays, together with suggestions as to how they can be used as abstract, yet synthesizable, hardware modeling constructs.

Chapter 6 presents the specialized procedural blocks, coding blocks and enhanced task and function definitions in SystemVerilog, and how these enhancements will help create models that are correct by design.

Chapter 7 shows how to use the enhancements to Verilog operators and procedural statements to code accurate and deterministic hardware models, using fewer lines of code compared to standard Verilog.

Chapter 8 provides guidelines on how to use enumerated types and specialized procedural blocks for modeling Finite State Machine (FSM) designs. This chapter also presents a number of guidelines on modeling hardware using 2-state logic.

Chapter 9 examines the enhancements to design hierarchy that SystemVerilog provides. Significant constructs are presented, including nested module declarations and simplified module instance declarations.

Chapter 10 discusses the powerful interface construct that SystemVerilog adds to Verilog. Interfaces greatly simplify the representation of complex busses and enable the creation of more intelligent, easier to use IP (intellectual property) models.

Chapter 11 ties together the concepts from all the previous chapters by applying them to a much more extensive example. The example shows a complete model of an ATM switch design, modeled in SystemVerilog.

Chapter 12 provides another complete example of using SystemVerilog. This chapter covers the usage of SystemVerilog to represent models at a much higher level of abstraction, using transactions.

Appendix A lists the formal syntax of SystemVerilog using the Backus-Naur Form (BNF). The SystemVerilog BNF includes the full Verilog-2005 BNF, with the SystemVerilog extensions integrated into the BNF.

Appendix B lists the set of reserved keywords in the Verilog and SystemVerilog standards. The appendix also shows how to mix Verilog models and SystemVerilog models in the same design, and maintain compatibility between the different keyword lists.

Appendix C presents an informative history of hardware description languages and Verilog. It covers the development of the SUPERLOG language, which became the basis for much of the synthesizable modeling constructs in SystemVerilog.

About the examples in this book

The examples in this book are intended to illustrate specific SystemVerilog constructs in a realistic but brief context. To maintain that focus, many of the examples are relatively small, and often do not reflect the full context of a complete model. However, the examples serve to show the proper usage of SystemVerilog constructs. To show the power of SystemVerilog in a more complete context, Chapter 11 contains the full source code of a more extensive example.

The examples contained in the book use the convention of showing all Verilog and SystemVerilog keywords in bold, as illustrated below:

Example: SystemVerilog code sample

```
module uart (output logic [7:0] data,
             output logic       data_rdy,
             input              serial_in);

  enum {WAITE, LOAD, READY} State, NextState;
  logic [2:0] bit_cnt;
  logic       cntr_rst, shift_en;
  always_ff @(posedge clock, negedge resetN) begin: shifter
    if (!resetN)
      data <= 8'h0;                    //reset (active low)
    else if (shift_en)
      data <= {serial_in, data[7:1]};  //shift right
  end: shifter
endmodule
```

Longer examples in this book list the code between double horizontal lines, as shown above. There are also many shorter examples in each chapter that are embedded in the body of the text, without the use of horizontal lines to set them apart. For both styles of examples, the full source code is not always included in the book. This was done in order to focus on specific aspects of SystemVerilog constructs without excessive clutter from surrounding code.

NOTE The examples do not distinguish standard Verilog constructs and keywords from SystemVerilog constructs and keywords. It is expected that the reader is already familiar with the Verilog HDL, and will recognize standard Verilog versus the new constructs and keywords added with SystemVerilog.

Obtaining copies of the examples

The complete code for all the examples listed in this book are available for personal, non-commercial use. They can be downloaded from ***http://www.sutherland-hdl.com***. Navigate the links to ***"SystemVerilog Book Examples"***.

Example testing

Most examples in this book have been tested using the *Synopsys* **VCS**® simulator, version 2005.06-SP1, and the *Mentor Graphics* **Questa**™ simulator, version 6.2. Most models in this book are synthesizable, and have been tested using the *Synopsys* **DC Compiler**™ synthesis compiler, version 2005.12.[1]

1. All company names and product names mentioned in this book are the trademark or registered trademark names of their respective companies.

Other sources of information

This book only explains the SystemVerilog enhancements for modeling hardware designs. The book does not go into detail on the SystemVerilog enhancements for verification, and does not cover the Verilog standard. Some other resources which can serve as excellent companions to this book are:

SystemVerilog for Verification—A Guide to Learning the Testbench Language Features by Chris Spear.

> Copyright 2006, Springer, Norwalk, Massachusetts. ISBN 0-387-27036-1.
>
> A companion to this book, with a focus on verification methodology using the SystemVerilog assertion and testbench enhancements to Verilog. This book presents the numerous verification constructs in SystemVerilog, which are not covered in this book. Together, the two books provide a comprehensive look at the extensive set of extensions that SystemVerilog adds to the Verilog language. For more information, refer to the publisher's web site: *www.springer.com/sgw/cda/frontpage/0,11855,4-40109-22-107949012-0,00.html.*

IEEE Std 1800-2005, SystemVerilog Language Reference Manual LRM)—IEEE Standard for SystemVerilog: Unified Hardware Design, Specification and Verification Language.

> Copyright 2005, IEEE, Inc., New York, NY. ISBN 0-7381-4811-3. Electronic PDF form, (also available in soft cover).
>
> This is the official SystemVerilog standard. The book is a syntax and semantics reference, not a tutorial for learning SystemVerilog. For information on ordering, visit the web site: *http://shop.ieee.org/store* and search for SystemVerilog.

IEEE Std 1364-2005, Verilog Language Reference Manual LRM)—IEEE Standard for Verilog Hardware Description Language.

> Copyright 2005, IEEE, Inc., New York, NY. ISBN 0-7381-4851-2. Electronic PDF form, (also available in soft cover).
>
> This is the official Verilog HDL and PLI standard. The book is a syntax and semantics reference, not a tutorial for learning Verilog. For information on ordering, visit the web site: *http://shop.ieee.org/store* and search for Verilog.

1364.1-2002 IEEE Standard for Verilog Register Transfer Level Synthesis 2002—Standard syntax and semantics for Verilog HDL-based RTL synthesis.

Copyright 2002, IEEE, Inc., New York, NY. ISBN 0-7381-3501-1. Softcover, 106 pages (also available as a downloadable PDF file).

This is the official synthesizable subset of the Verilog language. For information on ordering, visit the web site: *http://shop.ieee.org/store* and search for Verilog.

Writing Testbenches Using SystemVerilog by Janick Bergeron

Copyright 2006, Springer, Norwell Massachusetts.
ISBN: 0-387-29221-7. Hardcover, 412 pages.

Provides an explanation of the many testbench extensions that SystemVerilog adds for verification, and how to use those extensions for efficient verification. For more information, refer to the publisher's web site: *www.springer.com/sgw/cda/frontpage/0,11855,4-40109-22-104242164-0,00.html.*

The Verification Methodology Manual for SystemVerilog (VMM) by Janick Bergeron, Eduard Cerny, Alan Hunter, Andrew Nightingale

Copyright 2005, Springer, Norwell Massachusetts.
ISBN: 0-387-25538-9. Hardcover, 510 pages.

A methodology book on how to use SystemVerilog for advanced verification techniques. This is an advanced-level book; It is not a tutorial for learning SystemVerilog. For more information, refer to the publisher's web site: *www.springer.com/sgw/cda/frontpage/0,11855,4-40109-22-52495600-0,00.html.*

A Practical Guide for SystemVerilog Assertions, by Srikanth Vijayaraghavan, and Meyyappan Ramanathan

Copyright 2005, Springer, Norwell Massachusetts.
ISBN: 0-387-26049-8. Hardcover, 334 pages.

Specifically covers the SystemVerilog Assertions portion of the SystemVerilog standard. For more information, refer to the publisher's web site: *www.springer.com/sgw/cda/frontpage/0,11855,4-40109-22-50493024-0,00.html.*

SystemVerilog Assertions Handbook, Ben Cohen, Srinivasan Venkataramanan, Ajeetha Kumari

Copyright 2004, VhdlCohen, Palos Verdes Peninsula, California.
ISBN: 0-9705394-7-9. Softcover, 330 pages.

Presents Assertion-Based Verification techniques using the SystemVerilog Assertions portion of the SystemVerilog standard. For more information, refer to the publisher's web site: *www.abv-sva.org/#svah.*

Assertions-Based Design, Second Edition, Harry Foster, Adam Krolnik, and David Lacey

Copyright 2004, Springer, Norwell Massachusetts.
ISBN: 1-4020-8027-1. Hardcover, 414 pages.

Presents how assertions are used in the design and verification process, and illustrates the usage of OVL, PSL and SystemVerilog assertions. For more information, refer to the publisher's web site: *www.springer.com/sgw/cda/frontpage/0,11855,4-102-22-33837980-0,00.html.*

The Verilog Hardware Description Language, 5th Edition by Donald E. Thomas and Philip R. Moorby.

Copyright 2002, Kluwer Academic Publishers, Norwell MA.
ISBN: 1-4020-7089-6. Hardcover, 408 pages.

A complete book on Verilog, covering RTL modeling, behavioral modeling and gate level modeling. The book has more detail on the gate, switch and strength level aspects of Verilog than many other books. For more information, refer to the web site *www.wkap.nl/prod/b/1-4020-7089-6*.

Verilog Quickstart, A Practical Guide to Simulation and Synthesis, 3rd Edition by James M. Lee.

Copyright 2002, Kluwer Academic Publishers, Norwell MA.
ISBN: 0-7923-7672-2. Hardcover, 384 pages.

An excellent book for learning the Verilog HDL. The book teaches the basics of Verilog modeling, without getting bogged down with the more obscure aspects of the Verilog language. For more information, refer to the web site *www.wkap.nl/prod/b/0-7923-7672-2*.

Verilog 2001: A Guide to the New Features of the Verilog Hardware Description Language by Stuart Sutherland.

Copyright 2002, Kluwer Academic Publishers, Norwell MA.
ISBN: 0-7923-7568-8. Hardcover, 136 pages.

An overview of the many enhancements added as part of the IEEE 1364-2001 standard. For more information, refer to the web site *www.wkap.nl/book.htm/0-7923-7568-8*.

Acknowledgements

The authors would like to express their gratitude to all those who have helped with this book. A number of SystemVerilog experts have taken the time to review all or part of the text and examples, and provided invaluable feedback on how to make the book useful and accurate.

We would like to specifically thank those that provided invaluable feedback by reviewing this book. These reviewers of the first edition include (listed alphabetically) **Clifford E. Cummings**., **Tom Fitzpatrick**, **Dave Kelf**, **James Kenney**, **Matthew Hall**, **Monique L'Huillier**, **Phil Moorby**, **Lee Moore**, **Karen L. Pieper**, **Dave Rich**, **LeeAnn Sutherland** and **David W. Smith**. The updates made for the second edition were reviewed by **Shalom Bresticker** and **LeeAnn Sutherland**.

We also want to acknowledge the significant contribution of **Lee Moore**, who converted the *Verification Guild* ATM model shown in Chapter 11 from behavioral Verilog into synthesizable SystemVerilog. The authors also express their appreciation to **Janick Bergeron**, moderator of the *Verification Guild* on-line newsletter, for granting permission to use this ATM switch example.

Chapter 1
Introduction to SystemVerilog

This chapter provides an overview of SystemVerilog. The topics presented in this chapter include:

- The origins of SystemVerilog
- Technical donations that went into SystemVerilog
- Highlights of key SystemVerilog features

1.1 SystemVerilog origins

SystemVerilog extends Verilog

SystemVerilog is a standard set of extensions to the **IEEE 1364-2005 Verilog Standard** (commonly referred to as ***"Verilog-2005"***). The SystemVerilog extensions to the Verilog HDL that are described in this book are targeted at design and writing synthesizable models. These extensions integrate many of the features of the SUPERLOG and C languages. SystemVerilog also contains many extensions for the verification of large designs, integrating features from the SUPERLOG, VERA C, C++, and VHDL languages, along with OVA and PSL assertions. These verification assertions are in a companion book, *SystemVerilog for Verification*[1].

1. Spear, Chris *"SystemVerilog for Verification"*, Norwell, MA: Springer 2006, 0-387-27036-1.

This integrated whole created by SystemVerilog greatly exceeds the sum of its individual components, creating a new type of engineering language, a **H**ardware **D**escription *and* **V**erification **L**anguage or **HDVL**. Using a single, unified language enables engineers to model large, complex designs, and verify that these designs are functionally correct.

The Accellera standards organization

SystemVerilog started as an Accellera standard

The specification of the SystemVerilog enhancements to Verilog began with a standards group under the auspices of the ***Accellera Standards Organization***, rather than directly by the IEEE. Accellera is a non-profit organization with the goal of supporting the development and use of Electronic Design Automation (EDA) languages. Accellera is the combined VHDL International and Open Verilog International organizations. Accellera helps sponsor the IEEE 1076 VHDL and IEEE 1364 Verilog standards groups. In addition, Accellera sponsors a number of committees doing research on future languages. SystemVerilog is the result of one of those Accellera committees. Accellera itself receives its funding from member companies. These companies comprise several major EDA software vendors and several major electronic design corporations. More information on Accellera, its members, and its current projects can be found at *www.accellera.org*.

SystemVerilog is based on proven technology

Accellera based the SystemVerilog enhancements to Verilog on proven technologies. Various companies have donated technology to Accellera, which has then been carefully reviewed and integrated into SystemVerilog. A major benefit of using donations of technologies is that the SystemVerilog enhancements have already been proven to work and accomplish the objective of modeling and verifying much larger designs.

1.1.1 Generations of the SystemVerilog standard

Accellera SystemVerilog 3.0 extended modeling capability

A major portion of SystemVerilog was released as an Accellera standard in June of 2002 under the title of ***SystemVerilog 3.0***. This initial release of the SystemVerilog standard allowed EDA companies to begin adding the SystemVerilog extensions to existing simulators, synthesis compilers and other engineering tools. The focus of this first release of the SystemVerilog standard was to extend the synthesizable constructs of Verilog, and to enable modeling hardware at a higher level of abstraction. These are the constructs that are addressed in this book.

SystemVerilog is the third generation of Verilog

SystemVerilog began with a version number of 3.0 to show that SystemVerilog is the third major generation of the Verilog language. Verilog-1995 is the first generation, which represents the standardization of the original Verilog language defined by Phil Moorby in the early 1980s. Verilog-2001 is the second major generation of Verilog, and SystemVerilog is the third major generation. Appendix C of this book contains more details on the history of hardware descriptions languages, and the evolution of Verilog that led up to SystemVerilog.

Accellera SystemVerilog 3.1 extends verification capability

A major update to the SystemVerilog set of extensions was released in May of 2003. This release was referred to as ***SystemVerilog 3.1***, and added a substantial number of verification capabilities to SystemVerilog. These testbench enhancements are covered in the companion book, *SystemVerilog for Verification*[1].

Accellera SystemVerilog 3.1a was donated to the IEEE

Accellera continued to refine the SystemVerilog 3.1 standard by working closely with major Electronic Design Automation (EDA) companies to ensure that the SystemVerilog specification could be implemented as intended. A few additional modeling and verification constructs were also defined. In May of 2004, a final Accellera SystemVerilog draft was ratified by Accellera, and called ***SystemVerilog 3.1a***.

SystemVerilog 3.1a was donated to the IEEE

In June of 2004, right after SystemVerilog 3.1a was ratified, Accellera donated the SystemVerilog standard to the IEEE Standards Association (IEEE-SA), which oversees the Verilog 1364 standard. Accellera worked with the IEEE to form a new standards request, to review and standardize the SystemVerilog extensions to Verilog. The project number assigned to SystemVerilog was P1800 (the "P" in IEEE standards numbers stands for "proposed", and is dropped once the IEEE has officially approved of the standard).

IEEE 1800-2005 is the official SystemVerilog standard

The IEEE-SA formed a P1800 Working Group to review the SystemVerilog 3.1a documentation and prepare it for full IEEE standardization. The working group formed several focused committees, which met on a very aggressive schedule for the next several months. The P1800 Working Group completed its work in

1. Spear, Chris *"SystemVerilog for Verification"*, Norwell, MA: Springer 2006, 0-387-27036-1.

March of 2005, and released a ballot draft of the P1800 standard for voting on by corporate members of the IEEE-SA. The balloting and final IEEE approval process were completed in October 2005, and, in November of 2005, the official IEEE 1800-2005 standard was released to the public. See page xxvii of the Preface for information on obtaining the IEEE 1800-2005 SystemVerilog Reference Manual (LRM).

IEEE 1364-2005 is the base language for SystemVerilog 1800-2005

Prior to the donation of SystemVerilog 3.1a to the IEEE, the IEEE-SA had already begun work on the next revision of the IEEE 1364 Verilog standard. At the encouragement of Accellera, the IEEE-SA organization decided not to immediately add the SystemVerilog extensions to work already in progress for extending Verilog 1364. Instead, it was decided to keep the SystemVerilog extensions as a separate document. To ensure that the reference manual for the base Verilog language and the reference manual for the SystemVerilog extensions to Verilog remained synchronized, the IEEE-SA dissolved the 1364 Working Group and made the 1364 Verilog reference manual part of the responsibility of the 1800 SystemVerilog Working Group. The 1800 Working Group formed a subcommittee to update the 1364 Verilog standard in parallel with the specification of the P1800 SystemVerilog reference manual. For the most part, the work done on the 1364 revisions was limited to errata corrections and clarifications. Most extensions to Verilog were specified in the P1800 standard. The 1800 SystemVerilog Working Group released a ballot draft for an updated Verilog P1364 standard at the same time as the ballot draft for the new P1800 SystemVerilog standard. Both standards were approved at the same time. The 1364-2005 Verilog Language Reference Manual is the official base language for SystemVerilog 1800-2005.

1.1.2 Donations to SystemVerilog

The primary technology donations that make up SystemVerilog include:

SystemVerilog comes from several donations

- The SUPERLOG Extended Synthesizable Subset (SUPERLOG ESS), from Co-Design Automation
- The OpenVERA™ verification language from Synopsys
- PSL assertions (which began as a donation of Sugar assertions from IBM)
- OpenVERA Assertions (OVA) from Synopsys

- The DirectC and coverage Application Programming Interfaces (APIs) from Synopsys
- Separate compilation and $readmem extensions from Mentor Graphics
- Tagged unions and high-level language features from BlueSpec

SUPERLOG was donated by Co-Design

In 2001, Co-Design Automation (which was acquired by Synopsys in 2002) donated to Accellera the SUPERLOG Extended Synthesizable Subset in June of 2001. This donation makes up the majority of the hardware modeling enhancements in SystemVerilog. Accellera then organized the Verilog++ committee, which was later renamed the SystemVerilog committee, to review this donation, and create a standard set of enhancements for the Verilog HDL. Appendix C contains a more complete history of the SUPERLOG language.

OpenVERA and DirectC were donated by Synopsys

In 2002, Synopsys donated OpenVERA testbench, OpenVERA Assertions (OVA), and DirectC to Accellera, as a complement to the SUPERLOG ESS donation. These donations significantly extend the verification capabilities of the Verilog language.

The Accellera SystemVerilog committee also specified additional design and verification enhancements to the Verilog language that were not part of these core donations.

SystemVerilog is backward compatible with Verilog

Two major goals of the SystemVerilog committee within Accellera were to maintain full backward compatibility with the existing Verilog HDL, and to maintain the general look and feel of the Verilog HDL.

1.2 Key SystemVerilog enhancements for hardware design

The following list highlights some of the more significant enhancements SystemVerilog adds to the Verilog HDL for the design and verification of hardware: This list is not intended to be all inclusive of every enhancement to Verilog that is in SystemVerilog. This list just highlights a few key features that aid in writing synthesizable hardware models.

- Interfaces to encapsulate communication and protocol checking within a design