

IEEE Standard for SystemVerilog— Unified Hardware Design, Specification, and Verification Language

IEEE Computer Society
and the
IEEE Standards Association Corporate Advisory Group

Sponsored by the
Design Automation Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 1800™-2012
(Revision of
IEEE Std 1800-2009)

21 February 2013

IEEE Std 1800™-2012
(Revision of
IEEE Std 1800-2009)

IEEE Standard for SystemVerilog— Unified Hardware Design, Specification, and Verification Language

Sponsor

Design Automation Standards Committee
of the
IEEE Computer Society

and the
IEEE Standards Association Corporate Advisory Group

Approved 5 December 2012

IEEE-SA Standards Board

Abstract: The definition of the language syntax and semantics for SystemVerilog, which is a unified hardware design, specification, and verification language, is provided. This standard includes support for modeling hardware at the behavioral, register transfer level (RTL), and gate-level abstraction levels, and for writing testbenches using coverage, assertions, object-oriented programming, and constrained random verification. The standard also provides application programming interfaces (APIs) to foreign programming languages.

Keywords: assertions, design automation, design verification, hardware description language, HDL, HDVL, IEEE 1800™, PLI, programming language interface, SystemVerilog, Verilog®, VPI

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2013 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 21 February 2013. Printed in the United States of America.

IEEE, 802, and POSIX are registered trademarks in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

Verilog is a registered trademark of Cadence Design Systems, Inc.

PDF: ISBN 978-0-7381-8110-3 STDGT98078
Print: ISBN 978-0-7381-8111-0 STDPD98078

IEEE prohibits discrimination, harassment and bullying. For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>. No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Notice and Disclaimer of Liability Concerning the Use of IEEE Documents: IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon any IEEE Standard document.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “**AS IS**.”

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

Translations: The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official Statements: A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on Standards: Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important to ensure that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE working group at <http://standards.ieee.org/develop/wg/>.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Photocopies: Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Notice to users

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at <http://standards.ieee.org/index.html> or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit IEEE-SA Website at <http://standards.ieee.org/index.html>.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

The **SystemVerilog Language Working Group** is entity based. At the time this standard was completed, the SystemVerilog Working Group had the following membership:

Karen Pieper, Accellera Representative, Tabula, Inc., *Chair*
Neil Korpusik, Oracle Corporation, *Vice Chair, Technical Chair*
Dennis Brophy, Mentor Graphics Corporation, *Secretary*
Stuart Sutherland, Sutherland HDL, Inc., *Technical Editor*

Shalom Bresticker, Intel Corporation
Charles Dawson, Cadence Design Systems, Inc.
Josef Derner, Mentor Graphics Corporation
John Goodenough, ARM, Ltd.
Kaiming Ho, Fraunhofer IIS
Haim Kerem, Intel Corporation

Dmitry Korchemny, Intel Corporation
Dave Rich, Mentor Graphics Corporation
Neil Sensarkar, Marvell Technology Group Ltd.
Yatin Trivedi, Synopsys, Inc.
Tony Tsai, Cisco Systems, Inc.

Work on this standard was divided among primary committees.

The **Champions Committee** was responsible for ensuring consistency in the work done by each committee.

Neil Korpusik, Oracle Corporation, *Chair*
Dave Rich, Mentor Graphics Corporation, *Co-Chair*

Shalom Bresticker, Intel Corporation
Surrendra Dudani, Synopsys, Inc.
Francoise Martinolle, Cadence Design Systems, Inc.

Brad Pierce, Synopsys, Inc.
Stuart Sutherland, Sutherland HDL, Inc.

The **Basic/Design Committee (SV-BC)** was responsible for the specification of the design features of SystemVerilog.

Matt Maidment, Intel Corporation, *Chair*
Brad Pierce, Synopsys, Inc., *Co-Chair*

Tom Alsop, Intel Corporation
Shalom Bresticker, Intel Corporation
Eric Coffin, Mentor Graphics Corporation
Peter Flake, Accellera Systems Initiative
Alex Gran, Mentor Graphics Corporation
Mark Hartoog, Synopsys, Inc.
Kaiming Ho, Fraunhofer IIS

Francoise Martinolle, Cadence Design Systems, Inc.
Dave Rich, Mentor Graphics Corporation
Arnab Saha, Mentor Graphics Corporation
Daniel Schostak, ARM, Ltd.
Steven Sharp, Cadence Design Systems, Inc.
Stuart Sutherland, Sutherland HDL, Inc.
Gordon Vreugdenhil, Mentor Graphics Corporation

The **Enhancement Committee (SV-EC)** was responsible for the specification of the testbench features of SystemVerilog.

Mehdi Mohtashemi, Synopsys, Inc., *Chair*
Neil Korpusik, Oracle Corporation, *Co-Chair*

Tom Alsop, Intel Corporation	Ray Ryan, Mentor Graphics Corporation
Jonathan Bromley, Accellera Systems Initiative	Arturo Salz, Synopsys, Inc.
Dhiraj Goswami, Synopsys, Inc.	Daniel Schostak, ARM Ltd.
Alex Gran, Mentor Graphics Corporation	Nilotpal Sensarkar, Marvell Technology Group, Ltd.
Mark Hartoog, Synopsys, Inc.	Steven Sharp, Cadence Design Systems, Inc.
Scott Little, Intel Corporation	Brandon Tipp, Intel Corporation
Francoise Martinolle, Cadence Design Systems, Inc.	Tony Tsai, Cisco Systems, Inc.
Dave Rich, Mentor Graphics Corporation	Gordon Vreugdenhil, Mentor Graphics Corporation

The **Assertions Committee (SV-AC)** was responsible for the specification of the assertion features of SystemVerilog.

Dmitry Korchemny, Intel Corporation, *Chair*
Tom Thatcher, Oracle Corporation, *Co-Chair*

Ashok Bhatt, Cadence Design Systems, Inc.	Jacob Katz, Intel Corporation
Laurence Bisht, Intel Corporation	Manisha Kulshrestha, Mentor Graphics Corporation
Eduard Cerny, Synopsys, Inc.	Scott Little, Intel Corporation
Ben Cohen, Accellera Systems Initiative	Anupam Prabhakar, Mentor Graphics Corporation
Dana Fisman, Synopsys, Inc.	Erik Seligman, Intel Corporation
John Havlicek, Freescale, Inc.	Samik Sengupta, Synopsys, Inc.
Tapan Kapoor, Cadence Design Systems, Inc.	

The **C API Committee (SV-CC)** was responsible for on the specification of the DPI, the SystemVerilog Verification Procedural Interface (VPI), and the additional coverage API.

Charles Dawson, Cadence Design Systems, Inc., *Chair*
Ghassan Khoory, Synopsys, Inc., *Co-Chair*

Chuck Berking, Cadence Design Systems, Inc.	Arnab Saha, Mentor Graphics Corporation
Steve Dovich, Cadence Design Systems, Inc.	Arturo Salz, Synopsys, Inc.
Amit Kohli, Cadence Design Systems, Inc.	George Scott, Mentor Graphics Corporation
Francoise Martinolle, Cadence Design Systems, Inc.	Bassam Tabbara, Synopsys, Inc.
Abigail Moorhouse, Mentor Graphics Corporation	Jim Vellenga, Cadence Design Systems, Inc.
Michael Rohleder, Freescale, Inc.	Vitaly Yankelevich, Cadence Design Systems, Inc.

The **Discrete Committee (SV-DC)** was responsible for defining features to support modeling of analog/mixed-signal circuit components in the discrete domain.

Scott Little, Intel Corporation, *Chair*
Abhijeet Kolpekwar, Cadence Design Systems, Inc., *Co-Chair*

Shekar Chetput, Cadence Design Systems, Inc.	Francoise Martinolle, Cadence Design Systems, Inc.
Scott Cranston, Cadence Design Systems, Inc.	Arturo Salz, Synopsys, Inc.
Dave Cronauer, Synopsys, Inc.	Sundaram Sangameswaran, Texas Instruments, Inc.
Mark Hartoog, Synopsys, Inc.	Steven Sharp, Cadence Design Systems, Inc.
John Havlicek, Freescale, Inc.	Gordon Vreugdenhil, Mentor Graphics Corporation
Ghassan Khoory, Synopsys, Inc.	Ian Wilson, Accellera Systems Initiative

The following members of the entity balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Accellera Systems Initiative
Cadence Design Systems, Inc.
Fraunhofer IIS
Freescale, Inc.
Intel Corporation

Japan Electronics and Information Technology
Industries Association (JEITA)
Marvell Technology Group Ltd.
Mentor Graphics Corporation
Oracle Corporation
Synopsys, Inc.

When the IEEE-SA Standards Board approved this standard on 5 December 2012, it had the following membership:

Richard H. Hulett, *Chair*
John Kulick, *Vice Chair*
Robert M. Grow, *Past Chair*
Konstantinos Karachalios, *Secretary*

Satish Aggarwal
Masayuki Ariyoshi
Peter Balma
William Bartley
Ted Burse
Clint Chaplin
Wael Diab
Jean-Philippe Faure

Alexander Gelman
Paul Houzé
Jim Hughes
Young Kyun Kim
Joseph L. Koepfinger*
David J. Law
Thomas Lee
Hung Ling

Oleg Logvinov
Ted Olsen
Gary Robinson
Jon Walter Rosdahl
Mike Seavy
Yatin Trivedi
Phil Winston
Yu Yuan

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Matthew J. Ceglia
IEEE Manager, Professional Services

Michelle Turner
IEEE Standards Program Manager, Document Development

Joan Woolery
IEEE Standards Program Manager, Technical Program Development

Introduction

This introduction is not part of IEEE Std 1800-2012, IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language.

The purpose of this standard is to provide the electronic design automation (EDA), semiconductor, and system design communities with a well-defined and official IEEE unified hardware design, specification, and verification standard language. The language is designed to coexist and enhance the hardware description and verification languages (HDVLS) presently used by designers while providing the capabilities lacking in those languages.

SystemVerilog is a unified hardware design, specification, and verification language based on the Accellera SystemVerilog 3.1a extensions to the Verilog hardware description language (HDL) [B3], published in 2004.^a Accellera is a consortium of EDA, semiconductor, and system companies. IEEE Std 1800 enables a productivity boost in design and validation and covers design, simulation, validation, and formal assertion-based verification flows.

SystemVerilog enables the use of a unified language for abstract and detailed specification of the design, specification of assertions, coverage, and testbench verification based on manual or automatic methodologies. SystemVerilog offers application programming interfaces (APIs) for coverage and assertions, and a direct programming interface (DPI) to access proprietary functionality. SystemVerilog offers methods that allow designers to continue to use present design languages when necessary to leverage existing designs and intellectual property (IP). This standardization project will provide the VLSI design engineers with a well-defined IEEE standard, which meets their requirements in design and validation, and which enables a step function increase in their productivity. This standardization project will also provide the EDA industry with a standard to which they can adhere and that they can support in order to deliver their solutions in this area.

^aThe numbers in brackets correspond to those of the bibliography in [Annex Q](#).

Contents

Part One: Design and Verification Constructs

1.	Overview	2
1.1	Scope	2
1.2	Purpose	2
1.3	Content summary	2
1.4	Special terms	3
1.5	Conventions used in this standard	3
1.6	Syntactic description	4
1.7	Use of color in this standard	5
1.8	Contents of this standard	5
1.9	Deprecated clauses	8
1.10	Examples	8
1.11	Prerequisites	8
2.	Normative references	9
3.	Design and verification building blocks	11
3.1	General	11
3.2	Design elements	11
3.3	Modules	11
3.4	Programs	12
3.5	Interfaces	13
3.6	Checkers	14
3.7	Primitives	14
3.8	Subroutines	14
3.9	Packages	14
3.10	Configurations	15
3.11	Overview of hierarchy	15
3.12	Compilation and elaboration	16
3.13	Name spaces	18
3.14	Simulation time units and precision	19
4.	Scheduling semantics	23
4.1	General	23
4.2	Execution of a hardware model and its verification environment	23
4.3	Event simulation	23
4.4	Stratified event scheduler	24
4.5	SystemVerilog simulation reference algorithm	29
4.6	Determinism	29
4.7	Nondeterminism	30
4.8	Race conditions	30
4.9	Scheduling implication of assignments	30
4.10	PLI callback control points	32
5.	Lexical conventions	33
5.1	General	33
5.2	Lexical tokens	33
5.3	White space	33

5.4	Comments	33
5.5	Operators.....	33
5.6	Identifiers, keywords, and system names	34
5.7	Numbers.....	35
5.8	Time literals	40
5.9	String literals.....	40
5.10	Structure literals.....	42
5.11	Array literals	43
5.12	Attributes	43
5.13	Built-in methods	45
6.	Data types	47
6.1	General.....	47
6.2	Data types and data objects.....	47
6.3	Value set	47
6.4	Singular and aggregate types	48
6.5	Nets and variables.....	49
6.6	Net types	50
6.7	Net declarations	61
6.8	Variable declarations	64
6.9	Vector declarations	66
6.10	Implicit declarations	67
6.11	Integer data types.....	68
6.12	Real, shortreal, and realtime data types	69
6.13	Void data type.....	69
6.14	Chandle data type.....	69
6.15	Class.....	70
6.16	String data type	70
6.17	Event data type.....	75
6.18	User-defined types	76
6.19	Enumerations	77
6.20	Constants.....	83
6.21	Scope and lifetime	90
6.22	Type compatibility	92
6.23	Type operator.....	95
6.24	Casting	96
6.25	Parameterized data types	101
7.	Aggregate data types.....	103
7.1	General.....	103
7.2	Structures.....	103
7.3	Unions.....	105
7.4	Packed and unpacked arrays	109
7.5	Dynamic arrays	113
7.6	Array assignments.....	116
7.7	Arrays as arguments to subroutines	117
7.8	Associative arrays	118
7.9	Associative array methods	121
7.10	Queues	124
7.11	Array querying functions	129
7.12	Array manipulation methods	129

8.	Classes	134
8.1	General	134
8.2	Overview	134
8.3	Syntax	135
8.4	Objects (class instance)	136
8.5	Object properties and object parameter data	137
8.6	Object methods	138
8.7	Constructors	138
8.8	Typed constructor calls	140
8.9	Static class properties	141
8.10	Static methods	141
8.11	This	141
8.12	Assignment, renaming, and copying	142
8.13	Inheritance and subclasses	144
8.14	Overridden members	144
8.15	Super	145
8.16	Casting	146
8.17	Chaining constructors	146
8.18	Data hiding and encapsulation	147
8.19	Constant class properties	147
8.20	Virtual methods	148
8.21	Abstract classes and pure virtual methods	150
8.22	Polymorphism: dynamic method lookup	150
8.23	Class scope resolution operator ::	151
8.24	Out-of-block declarations	153
8.25	Parameterized classes	154
8.26	Interface classes	157
8.27	Typedef class	167
8.28	Classes and structures	167
8.29	Memory management	168
9.	Processes	169
9.1	General	169
9.2	Structured procedures	169
9.3	Block statements	173
9.4	Procedural timing controls	179
9.5	Process execution threads	189
9.6	Process control	189
9.7	Fine-grain process control	193
10.	Assignment statements	196
10.1	General	196
10.2	Overview	196
10.3	Continuous assignments	197
10.4	Procedural assignments	200
10.5	Variable declaration assignment (variable initialization)	205
10.6	Procedural continuous assignments	205
10.7	Assignment extension and truncation	207
10.8	Assignment-like contexts	208
10.9	Assignment patterns	209
10.10	Unpacked array concatenation	213
10.11	Net aliasing	216

11.	Operators and expressions	218
11.1	General	218
11.2	Overview	218
11.3	Operators	219
11.4	Operator descriptions	223
11.5	Operands	243
11.6	Expression bit lengths	247
11.7	Signed expressions	249
11.8	Expression evaluation rules	250
11.9	Tagged union expressions and member access	251
11.10	String literal expressions	253
11.11	Operator overloading	254
11.12	Minimum, typical, and maximum delay expressions	256
11.13	Let construct	258
12.	Procedural programming statements	264
12.1	General	264
12.2	Overview	264
12.3	Syntax	264
12.4	Conditional if–else statement	265
12.5	Case statement	270
12.6	Pattern matching conditional statements	275
12.7	Loop statements	279
12.8	Jump statements	283
13.	Tasks and functions (subroutines)	285
13.1	General	285
13.2	Overview	285
13.3	Tasks	285
13.4	Functions	289
13.5	Subroutine calls and argument passing	295
13.6	Import and export functions	300
13.7	Task and function names	300
13.8	Parameterized tasks and functions	300
14.	Clocking blocks	302
14.1	General	302
14.2	Overview	302
14.3	Clocking block declaration	302
14.4	Input and output skews	304
14.5	Hierarchical expressions	305
14.6	Signals in multiple clocking blocks	306
14.7	Clocking block scope and lifetime	306
14.8	Multiple clocking blocks example	306
14.9	Interfaces and clocking blocks	307
14.10	Clocking block events	308
14.11	Cycle delay: ##	308
14.12	Default clocking	309
14.13	Input sampling	310
14.14	Global clocking	311

14.15	Synchronous events	315
14.16	Synchronous drives	315
15.	Interprocess synchronization and communication	320
15.1	General	320
15.2	Overview	320
15.3	Semaphores	320
15.4	Mailboxes	322
15.5	Named events	325
16.	Assertions	330
16.1	General	330
16.2	Overview	330
16.3	Immediate assertions	330
16.4	Deferred assertions	333
16.5	Concurrent assertions overview	340
16.6	Boolean expressions	343
16.7	Sequences	344
16.8	Declaring sequences	348
16.9	Sequence operations	356
16.10	Local variables	378
16.11	Calling subroutines on match of a sequence	384
16.12	Declaring properties	385
16.13	Multiclock support	411
16.14	Concurrent assertions	421
16.15	Disable iff resolution	438
16.16	Clock resolution	440
16.17	Expect statement	445
16.18	Locking blocks and concurrent assertions	446
17.	Checkers	448
17.1	Overview	448
17.2	Checker declaration	448
17.3	Checker instantiation	451
17.4	Context inference	454
17.5	Checker procedures	455
17.6	Covergroups in checkers	457
17.7	Checker variables	458
17.8	Functions in checkers	464
17.9	Complex checker example	464
18.	Constrained random value generation	467
18.1	General	467
18.2	Overview	467
18.3	Concepts and usage	467
18.4	Random variables	470
18.5	Constraint blocks	472
18.6	Randomization methods	492
18.7	In-line constraints—randomize() with	493
18.8	Disabling random variables with rand_mode()	496
18.9	Controlling constraints with constraint_mode()	497

18.10	Dynamic constraint modification.....	498
18.11	In-line random variable control	499
18.12	Randomization of scope variables—std::randomize().....	500
18.13	Random number system functions and methods	501
18.14	Random stability	503
18.15	Manually seeding randomize	505
18.16	Random weighted case—randcase	506
18.17	Random sequence generation—randsequence.....	507
19.	Functional coverage.....	517
19.1	General.....	517
19.2	Overview.....	517
19.3	Defining the coverage model: covergroup.....	518
19.4	Using covergroup in classes	520
19.5	Defining coverage points.....	522
19.6	Defining cross coverage.....	533
19.7	Specifying coverage options.....	542
19.8	Predefined coverage methods	547
19.9	Predefined coverage system tasks and system functions.....	549
19.10	Organization of option and type_option members	549
19.11	Coverage computation	550
20.	Utility system tasks and system functions	555
20.1	General.....	555
20.2	Simulation control system tasks	556
20.3	Simulation time system functions.....	556
20.4	Timescale system tasks	558
20.5	Conversion functions	561
20.6	Data query functions.....	562
20.7	Array querying functions.....	564
20.8	Math functions	566
20.9	Bit vector system functions.....	568
20.10	Severity tasks.....	569
20.11	Elaboration system tasks.....	570
20.12	Assertion control system tasks.....	571
20.13	Sampled value system functions.....	578
20.14	Coverage system functions	579
20.15	Probabilistic distribution functions.....	579
20.16	Stochastic analysis tasks and functions	581
20.17	Programmable logic array modeling system tasks	583
20.18	Miscellaneous tasks and functions.....	586
21.	Input/output system tasks and system functions.....	588
21.1	General.....	588
21.2	Display system tasks.....	588
21.3	File input/output system tasks and system functions.....	599
21.4	Loading memory array data from a file	609
21.5	Writing memory array data to a file.....	613
21.6	Command line input.....	614
21.7	Value change dump (VCD) files	617

22.	Compiler directives	638
22.1	General	638
22.2	Overview	638
22.3	<code>`resetall</code>	638
22.4	<code>`include</code>	639
22.5	<code>`define</code> , <code>`undef</code> , and <code>`undefineall</code>	639
22.6	<code>`ifdef</code> , <code>`else</code> , <code>`elsif</code> , <code>`endif</code> , <code>`ifndef</code>	645
22.7	<code>`timescale</code>	648
22.8	<code>`default_nettype</code>	649
22.9	<code>`unconnected_drive</code> and <code>`nounconnected_drive</code>	650
22.10	<code>`celldefine</code> and <code>`endcelldefine</code>	650
22.11	<code>`pragma</code>	650
22.12	<code>`line</code>	651
22.13	<code>`__FILE__</code> and <code>`__LINE__</code>	652
22.14	<code>`begin_keywords</code> , <code>`end_keywords</code>	653

Part Two: Hierarchy Constructs

23.	Modules and hierarchy	660
23.1	General	660
23.2	Module definitions	660
23.3	Module instances (hierarchy)	672
23.4	Nested modules	683
23.5	Extern modules	684
23.6	Hierarchical names	685
23.7	Member selects and hierarchical names	688
23.8	Upwards name referencing	689
23.9	Scope rules	692
23.10	Overriding module parameters	694
23.11	Binding auxiliary code to scopes or instances	701
24.	Programs	705
24.1	General	705
24.2	Overview	705
24.3	The program construct	705
24.4	Eliminating testbench races	709
24.5	Blocking tasks in cycle/event mode	709
24.6	Programwide space and anonymous programs	710
24.7	Program control tasks	710
25.	Interfaces	711
25.1	General	711
25.2	Overview	711
25.3	Interface syntax	712
25.4	Ports in interfaces	716
25.5	Modports	717
25.6	Interfaces and specify blocks	723
25.7	Tasks and functions in interfaces	724
25.8	Parameterized interfaces	730
25.9	Virtual interfaces	732
25.10	Access to interface objects	737

26.	Packages.....	738
26.1	General.....	738
26.2	Package declarations.....	738
26.3	Referencing data in packages.....	739
26.4	Using packages in module headers.....	743
26.5	Search order rules.....	744
26.6	Exporting imported names from packages.....	746
26.7	The std built-in package.....	747
27.	Generate constructs.....	749
27.1	General.....	749
27.2	Overview.....	749
27.3	Generate construct syntax.....	749
27.4	Loop generate constructs.....	751
27.5	Conditional generate constructs.....	755
27.6	External names for unnamed generate blocks.....	758
28.	Gate-level and switch-level modeling.....	760
28.1	General.....	760
28.2	Overview.....	760
28.3	Gate and switch declaration syntax.....	760
28.4	and, nand, nor, or, xor, and xnor gates.....	766
28.5	buf and not gates.....	767
28.6	bufif1, bufif0, notif1, and notif0 gates.....	768
28.7	MOS switches.....	769
28.8	Bidirectional pass switches.....	770
28.9	CMOS switches.....	771
28.10	pullup and pulldown sources.....	772
28.11	Logic strength modeling.....	772
28.12	Strengths and values of combined signals.....	774
28.13	Strength reduction by nonresistive devices.....	786
28.14	Strength reduction by resistive devices.....	786
28.15	Strengths of net types.....	786
28.16	Gate and net delays.....	787
29.	User-defined primitives.....	791
29.1	General.....	791
29.2	Overview.....	791
29.3	UDP definition.....	791
29.4	Combinational UDPs.....	795
29.5	Level-sensitive sequential UDPs.....	796
29.6	Edge-sensitive sequential UDPs.....	796
29.7	Sequential UDP initialization.....	797
29.8	UDP instances.....	799
29.9	Mixing level-sensitive and edge-sensitive descriptions.....	800
29.10	Level-sensitive dominance.....	801
30.	Specify blocks.....	802
30.1	General.....	802
30.2	Overview.....	802

30.3	Specify block declaration.....	802
30.4	Module path declarations.....	803
30.5	Assigning delays to module paths	812
30.6	Mixing module path delays and distributed delays	816
30.7	Detailed control of pulse filtering behavior.....	817
31.	Timing checks.....	826
31.1	General.....	826
31.2	Overview.....	826
31.3	Timing checks using a stability window.....	829
31.4	Timing checks for clock and control signals	836
31.5	Edge-control specifiers	845
31.6	Notifiers: user-defined responses to timing violations	846
31.7	Enabling timing checks with conditioned events	848
31.8	Vector signals in timing checks	849
31.9	Negative timing checks.....	850
32.	Backannotation using the standard delay format.....	855
32.1	General.....	855
32.2	Overview.....	855
32.3	The SDF annotator.....	855
32.4	Mapping of SDF constructs to SystemVerilog.....	855
32.5	Multiple annotations	860
32.6	Multiple SDF files	861
32.7	Pulse limit annotation	861
32.8	SDF to SystemVerilog delay value mapping.....	862
32.9	Loading timing data from an SDF file.....	863
33.	Configuring the contents of a design	865
33.1	General.....	865
33.2	Overview.....	865
33.3	Libraries	866
33.4	Configurations	868
33.5	Using libraries and configs	874
33.6	Configuration examples.....	875
33.7	Displaying library binding information	877
33.8	Library mapping examples	877
34.	Protected envelopes	880
34.1	General.....	880
34.2	Overview.....	880
34.3	Processing protected envelopes	880
34.4	Protect pragma directives.....	882
34.5	Protect pragma keywords.....	884

Part Three: Application Programming Interfaces

35.	Direct programming interface.....	901
35.1	General.....	901
35.2	Overview.....	901

35.3	Two layers of DPI.....	902
35.4	Global name space of imported and exported functions.....	903
35.5	Imported tasks and functions	904
35.6	Calling imported functions	911
35.7	Exported functions.....	913
35.8	Exported tasks.....	914
35.9	Disabling DPI tasks and functions.....	914
36.	Programming language interface (PLI/VPI) overview	916
36.1	General.....	916
36.2	PLI purpose and history.....	916
36.3	User-defined system task and system function names.....	917
36.4	User-defined system task and system function arguments	918
36.5	User-defined system task and system function types	918
36.6	User-supplied PLI applications.....	918
36.7	PLI include files.....	918
36.8	VPI sizetf, compiletf, and calltf routines	918
36.9	PLI mechanism	919
36.10	VPI access to SystemVerilog objects and simulation objects	921
36.11	List of VPI routines by functional category.....	922
36.12	VPI backwards compatibility features and limitations	924
37.	VPI object model diagrams.....	929
37.1	General.....	929
37.2	VPI Handles.....	929
37.3	VPI object classifications.....	930
37.4	Key to data model diagrams	936
37.5	Module	939
37.6	Interface	940
37.7	Modport	940
37.8	Interface task or function declaration	940
37.9	Program	941
37.10	Instance	942
37.11	Instance arrays	944
37.12	Scope	945
37.13	IO declaration	946
37.14	Ports	947
37.15	Reference objects	948
37.16	Nets	950
37.17	Variables	954
37.18	Packed array variables	957
37.19	Variable select	958
37.20	Memory.....	959
37.21	Variable drivers and loads	959
37.22	Object Range.....	960
37.23	Typespec	961
37.24	Structures and unions.....	963
37.25	Named events	964
37.26	Parameter, spec param, def param, param assign	965
37.27	Virtual interface	966
37.28	Interface typespec	968
37.29	Class definition	969

37.30	Class typespec	970
37.31	Class variables and class objects	972
37.32	Constraint, constraint ordering, distribution	974
37.33	Primitive, prim term.....	975
37.34	UDP	976
37.35	Intermodule path	976
37.36	Constraint expression	977
37.37	Module path, path term	978
37.38	Timing check	979
37.39	Task and function declaration	980
37.40	Task and function call	981
37.41	Frames	983
37.42	Threads	984
37.43	Delay terminals	984
37.44	Net drivers and loads	985
37.45	Continuous assignment	986
37.46	Clocking block	987
37.47	Assertion	988
37.48	Concurrent assertions	989
37.49	Property declaration	990
37.50	Property specification	991
37.51	Sequence declaration	992
37.52	Sequence expression	993
37.53	Immediate assertions	994
37.54	Multiclock sequence expression	995
37.55	Let	995
37.56	Simple expressions	996
37.57	Expressions	997
37.58	Atomic statement	1000
37.59	Dynamic prefixing	1001
37.60	Event statement	1002
37.61	Process	1002
37.62	Assignment	1003
37.63	Event control	1003
37.64	While, repeat.....	1004
37.65	Waits	1004
37.66	Delay control.....	1004
37.67	Repeat control.....	1005
37.68	Forever	1005
37.69	If, if-else	1005
37.70	Case, pattern	1006
37.71	Expect	1007
37.72	For	1007
37.73	Do-while, foreach	1007
37.74	Alias statement	1008
37.75	Disables.....	1008
37.76	Return statement	1008
37.77	Assign statement, deassign, force, release	1009
37.78	Callback	1009
37.79	Time queue	1010
37.80	Active time format	1010
37.81	Attribute	1011
37.82	Iterator.....	1012
37.83	Generates	1013

38.	VPI routine definitions.....	1015
38.1	General.....	1015
38.2	vpi_chk_error().....	1015
38.3	vpi_compare_objects().....	1016
38.4	vpi_control().....	1018
38.5	vpi_flush().....	1019
38.6	vpi_get().....	1019
38.7	vpi_get64().....	1020
38.8	vpi_get_cb_info().....	1020
38.9	vpi_get_data().....	1021
38.10	vpi_get_delays().....	1022
38.11	vpi_get_str().....	1024
38.12	vpi_get_systf_info().....	1025
38.13	vpi_get_time().....	1026
38.14	vpi_get_userdata().....	1027
38.15	vpi_get_value().....	1027
38.16	vpi_get_value_array().....	1033
38.17	vpi_get_vlog_info().....	1037
38.18	vpi_handle().....	1038
38.19	vpi_handle_by_index().....	1039
38.20	vpi_handle_by_multi_index().....	1039
38.21	vpi_handle_by_name().....	1040
38.22	vpi_handle_multi().....	1041
38.23	vpi_iterate().....	1041
38.24	vpi_mcd_close().....	1042
38.25	vpi_mcd_flush().....	1043
38.26	vpi_mcd_name().....	1043
38.27	vpi_mcd_open().....	1044
38.28	vpi_mcd_printf().....	1045
38.29	vpi_mcd_vprintf().....	1046
38.30	vpi_printf().....	1046
38.31	vpi_put_data().....	1047
38.32	vpi_put_delays().....	1049
38.33	vpi_put_userdata().....	1052
38.34	vpi_put_value().....	1052
38.35	vpi_put_value_array().....	1055
38.36	vpi_register_cb().....	1059
38.37	vpi_register_systf().....	1067
38.38	vpi_release_handle().....	1071
38.39	vpi_remove_cb().....	1071
38.40	vpi_scan().....	1072
38.41	vpi_vprintf().....	1073
39.	Assertion API.....	1074
39.1	General.....	1074
39.2	Overview.....	1074
39.3	Static information.....	1074
39.4	Dynamic information.....	1075
39.5	Control functions.....	1079

40.	Code coverage control and API	1083
40.1	General	1083
40.2	Overview	1083
40.3	SystemVerilog real-time coverage access	1084
40.4	FSM recognition	1089
40.5	VPI coverage extensions.....	1092
41.	Data read API.....	1097

Part Four: Annexes

Annex A (normative) Formal syntax	1099
A.1 Source text	1099
A.2 Declarations	1108
A.3 Primitive instances.....	1119
A.4 Instantiations	1120
A.5 UDP declaration and instantiation	1122
A.6 Behavioral statements	1123
A.7 Specify section	1130
A.8 Expressions	1134
A.9 General.....	1139
A.10 Footnotes (normative).....	1142
Annex B (normative) Keywords.....	1145
Annex C (normative) Deprecation.....	1147
C.1 General.....	1147
C.2 Constructs that have been deprecated.....	1147
C.3 Accellera SystemVerilog 3.1a-compatible access to packed data.....	1148
C.4 Constructs identified for deprecation.....	1148
Annex D (informative) Optional system tasks and system functions.....	1151
D.1 General.....	1151
D.2 \$countdrivers	1151
D.3 \$getpattern	1152
D.4 \$input	1153
D.5 \$key and \$nokey	1153
D.6 \$list.....	1153
D.7 \$log and \$nolog	1153
D.8 \$reset, \$reset_count, and \$reset_value	1154
D.9 \$save, \$restart, and \$incsave.....	1155
D.10 \$scale	1156
D.11 \$scope	1156
D.12 \$showscopes	1156
D.13 \$showvars	1156
D.14 \$sreadmemb and \$sreadmemh.....	1156

Annex E (informative) Optional compiler directives	1158
E.1 General	1158
E.2 <code>`default_decay_time</code>	1158
E.3 <code>`default_trireg_strength</code>	1158
E.4 <code>`delay_mode_distributed</code>	1159
E.5 <code>`delay_mode_path</code>	1159
E.6 <code>`delay_mode_unit</code>	1159
E.7 <code>`delay_mode_zero</code>	1159
Annex F (normative) Formal semantics of concurrent assertions	1160
F.1 General	1160
F.2 Overview	1160
F.3 Abstract syntax	1161
F.4 Rewriting algorithms	1167
F.5 Semantics	1171
F.6 Extended expressions	1180
F.7 Recursive properties	1180
Annex G (normative) Std package	1182
G.1 General	1182
G.2 Overview	1182
G.3 Semaphore	1182
G.4 Mailbox	1182
G.5 Randomize	1183
G.6 Process	1183
Annex H (normative) DPI C layer	1184
H.1 General	1184
H.2 Overview	1184
H.3 Naming conventions	1185
H.4 Portability	1185
H.5 <code>svdpi.h</code> include file	1185
H.6 Semantic constraints	1186
H.7 Data types	1188
H.8 Argument passing modes	1192
H.9 Context tasks and functions	1195
H.10 Include files	1199
H.11 Arrays	1202
H.12 Open arrays	1205
H.13 SV3.1a-compatible access to packed data (deprecated functionality)	1211
Annex I (normative) <code>svdpi.h</code>	1217
I.1 General	1217
I.2 Overview	1217
I.3 Source code	1217

Annex J (normative) Inclusion of foreign language code.....	1226
J.1 General.....	1226
J.2 Overview.....	1226
J.3 Location independence.....	1227
J.4 Object code inclusion.....	1227
Annex K (normative) vpi_user.h	1230
K.1 General.....	1230
K.2 Source code.....	1230
Annex L (normative) vpi_compatibility.h	1247
L.1 General.....	1247
L.2 Source code.....	1247
Annex M (normative) sv_vpi_user.h	1250
M.1 General.....	1250
M.2 Source code.....	1250
Annex N (normative) Algorithm for probabilistic distribution functions	1260
N.1 General.....	1260
N.2 Source code.....	1260
Annex O (informative) Encryption/decryption flow	1268
O.1 General.....	1268
O.2 Overview.....	1268
O.3 Tool vendor secret key encryption system	1268
O.4 IP author secret key encryption system	1269
O.5 Digital envelopes	1270
Annex P (informative) Glossary	1272
Annex Q (informative) Bibliography	1275

List of figures

Figure 4-1—Event scheduling regions	28
Figure 6-1—Simulation values of a trireg and its driver	53
Figure 6-2—Simulation results of a capacitive network	54
Figure 6-3—Simulation results of charge sharing	55
Figure 7-1—VInt type with packed qualifier	108
Figure 7-2—Instr type with packed qualifier	108
Figure 9-1—Intra-assignment repeat event control utilizing a clock edge	188
Figure 14-1—Sample and drive times including skew with respect to the positive edge of the clock	305
Figure 16-1—Sampling a variable in a simulation time step	342
Figure 16-2—Concatenation of sequences	347
Figure 16-3—Value change expressions	362
Figure 16-4—Future value change	366
Figure 16-5—ANDing (and) two sequences	368
Figure 16-6—ANDing (and) two sequences, including a time range	369
Figure 16-7—ANDing (and) two Boolean expressions	369
Figure 16-8—Intersecting two sequences	370
Figure 16-9—ORing (or) two Boolean expressions	371
Figure 16-10—ORing (or) two sequences	372
Figure 16-11—ORing (or) two sequences, including a time range	373
Figure 16-12—Match with throughout restriction fails	375
Figure 16-13—Match with throughout restriction succeeds	376
Figure 16-14—Conditional sequence matching	392
Figure 16-15—Conditional sequences	393
Figure 16-16—Results without the condition	393
Figure 16-17—Clocking blocks and concurrent assertion	447
Figure 17-1—Nondeterministic free checker variable	459
Figure 18-1—Example of randc	472
Figure 18-2—Global constraints	481
Figure 18-3—Truth tables for conjunction, disjunction, and negation rules	487
Figure 21-1—Creating the 4-state VCD file	617
Figure 21-2—Creating the extended VCD file	627
Figure 23-1—Hierarchy in a model	687
Figure 23-1—Scopes available to upward name referencing	693
Figure 28-1—Schematic diagram of interconnections in array of instances	766
Figure 28-2—Scale of strengths	774
Figure 28-3—Combining unequal strengths	774
Figure 28-4—Combination of signals of equal strength and opposite values	775
Figure 28-5—Weak x signal strength	775
Figure 28-6—Bufifs with control inputs of x	776

Figure 28-7—Strong H range of values	776
Figure 28-8—Strong L range of values	776
Figure 28-9—Combined signals of ambiguous strength	777
Figure 28-10—Range of strengths for an unknown signal	777
Figure 28-11—Ambiguous strengths from switch networks	777
Figure 28-12—Range of two strengths of a defined value	778
Figure 28-13—Range of three strengths of a defined value	778
Figure 28-14—Unknown value with a range of strengths	778
Figure 28-15—Strong X range	779
Figure 28-16—Ambiguous strength from gates	779
Figure 28-17—Ambiguous strength signal from a gate	779
Figure 28-18—Weak 0	780
Figure 28-19—Ambiguous strength in combined gate signals	780
Figure 28-20—Elimination of strength levels	781
Figure 28-21—Result showing a range and the elimination of strength levels of two values	782
Figure 28-22—Result showing a range and the elimination of strength levels of one value	783
Figure 28-23—A range of both values	783
Figure 28-24—Wired logic with unambiguous strength signals	784
Figure 28-25—Wired logic and ambiguous strengths	785
Figure 28-26—Trireg net with capacitance	790
Figure 29-1—Module schematic and simulation times of initial value propagation	799
Figure 30-1—Module path delays	804
Figure 30-2—Difference between parallel and full connection paths	810
Figure 30-3—Module path delays longer than distributed delays	816
Figure 30-4—Module path delays shorter than distributed delays	817
Figure 30-5—Example of pulse filtering	817
Figure 30-6—On-detect versus on-event	820
Figure 30-7—Current event cancellation problem and correction	822
Figure 30-8—NAND gate with nearly simultaneous input switching where one event is scheduled prior to another that has not matured	823
Figure 30-9—NAND gate with nearly simultaneous input switching with output event scheduled at same time	824
Figure 31-1—Sample \$timeskew	838
Figure 31-2—Sample \$timeskew with remain_active_flag set	839
Figure 31-3—Sample \$fullskew	841
Figure 31-4—Data constraint interval, positive setup/hold	850
Figure 31-5—Data constraint interval, negative setup/hold	851
Figure 31-6—Timing check violation windows	854
Figure 37-1—Example of object relationships diagram	931
Figure 37-2—Accessing a class of objects using tags	932
Figure 38-1—s_vpi_error_info structure definition	1016

Figure 38-2—s_cb_data structure definition	1021
Figure 38-3—s_vpi_delay structure definition	1022
Figure 38-4—s_vpi_time structure definition	1022
Figure 38-5—s_vpi_systf_data structure definition	1025
Figure 38-6—s_vpi_time structure definition	1026
Figure 38-7—s_vpi_value structure definition	1028
Figure 38-8—s_vpi_vecval structure definition	1028
Figure 38-9—s_vpi_strengthval structure definition	1028
Figure 38-10—s_vpi_vlog_info structure definition	1037
Figure 38-11—s_vpi_delay structure definition	1050
Figure 38-12—s_vpi_time structure definition	1050
Figure 38-13—s_vpi_value structure definition	1054
Figure 38-14—s_vpi_time structure definition	1054
Figure 38-15—s_vpi_vecval structure definition	1055
Figure 38-16—s_vpi_strengthval structure definition	1055
Figure 38-17—s_cb_data structure definition	1059
Figure 38-18—s_vpi_systf_data structure definition	1068
Figure 39-1—Assertions with global clocking future sampled value functions	1079
Figure 40-1—Hierarchical instance example	1087
Figure 40-2—FSM specified with pragmas	1092

List of tables

Table 3-1—Time unit strings.....	19
Table 4-1—PLI callbacks.....	32
Table 5-1—Specifying special characters in string literals.....	41
Table 6-1—Built-in net types.....	51
Table 6-2—Truth table for wire and tri nets.....	51
Table 6-3—Truth table for wand and triand nets.....	52
Table 6-4—Truth table for wor and trior nets.....	52
Table 6-5—Truth table for tri0 net.....	56
Table 6-6—Truth table for tri1 net.....	56
Table 6-7—Default values.....	66
Table 6-8—Integer data types.....	68
Table 6-9—String operators.....	72
Table 6-10—Enumeration element ranges.....	80
Table 6-11—Differences between specparams and parameters.....	89
Table 7-1—Value read from a nonexistent array entry.....	112
Table 8-1—Comparison of pointer and handle types.....	137
Table 9-1—fork-join control options.....	175
Table 9-2—Detecting posedge and negedge.....	181
Table 9-3—Intra-assignment timing control equivalence.....	187
Table 10-1—Legal left-hand forms in assignment statements.....	196
Table 11-1—Operators and data types.....	220
Table 11-2—Operator precedence and associativity.....	221
Table 11-3—Arithmetic operators defined.....	224
Table 11-4—Power operator rules.....	225
Table 11-5—Unary operators defined.....	225
Table 11-6—Examples of modulus and power operators.....	225
Table 11-7—Data type interpretation by arithmetic operators.....	226
Table 11-8—Definitions of relational operators.....	227
Table 11-9—Definitions of equality operators.....	228
Table 11-10—Wildcard equality and wildcard inequality operators.....	228
Table 11-11—Bitwise binary AND operator.....	230
Table 11-12—Bitwise binary OR operator.....	230
Table 11-13—Bitwise binary exclusive OR operator.....	231
Table 11-14—Bitwise binary exclusive NOR operator.....	231
Table 11-15—Bitwise unary negation operator.....	231
Table 11-16—Reduction unary AND operator.....	232
Table 11-17—Reduction unary OR operator.....	232
Table 11-18—Reduction unary exclusive OR operator.....	232
Table 11-19—Results of unary reduction operations.....	233

Table 11-20—Ambiguous condition results for conditional operator	234
Table 11-21—Bit lengths resulting from self-determined expressions	248
Table 16-1—Operator precedence and associativity	356
Table 16-2—Global clocking future sampled value functions	366
Table 16-3—Sequence and property operator precedence and associativity	388
Table 18-1—Unordered constraint c legal value probability	482
Table 18-2—Ordered constraint c legal value probability	483
Table 18-3—rand_mode argument	496
Table 18-4—constraint_mode argument	498
Table 19-1—Instance-specific coverage options	542
Table 19-2—Coverage options per syntactic level	544
Table 19-3—Coverage group type (static) options	545
Table 19-4—Coverage type options	546
Table 19-5—Predefined coverage methods	547
Table 20-1—Diagnostics for \$finish	556
Table 20-2—\$timeformat units_number arguments	559
Table 20-3—\$timeformat default value for arguments	560
Table 20-4—SystemVerilog to C real math function cross-listing	567
Table 20-5—Values for control_type for assertion control tasks	573
Table 20-6—Values for assertion_type for assertion control tasks	573
Table 20-7—Values for directive_type for assertion control tasks	573
Table 20-8—VPI callbacks for assertion control tasks	577
Table 20-9—Types of queues of \$q_type values	581
Table 20-10—Argument values for \$q_exam system task	582
Table 20-11—Status code values	582
Table 20-12—PLA modeling system tasks	583
Table 21-1—Escape sequences for printing special characters	589
Table 21-2—Escape sequences for format specifications	590
Table 21-3—Format specifications for real numbers	592
Table 21-4—Logic value component of strength format	595
Table 21-5—Mnemonics for strength levels	595
Table 21-6—Explanation of strength formats	596
Table 21-7—Types for file descriptors	600
Table 21-8—\$fscanf input field characters	605
Table 21-9—Rules for left-extending vector values	623
Table 21-10—How the VCD can shorten values	623
Table 21-11—Keyword commands	624
Table 21-12—VCD type mapping	636
Table 22-1—IEEE 1364-1995 reserved keywords	655
Table 22-2—IEEE 1364-2001 additional reserved keywords	656

Table 22-3—IEEE 1364-2005 additional reserved keywords	656
Table 22-4—IEEE 1800-2005 additional reserved keywords	657
Table 22-5—IEEE 1800-2009 additional reserved keywords	657
Table 22-6—IEEE 1800-2012 additional reserved keywords	658
Table 23-1—Net types resulting from dissimilar port connections.....	682
Table 26-1—Scoping rules for package importation.....	745
Table 28-1—Built-in gates and switches.....	762
Table 28-2—Valid gate types for strength specifications	762
Table 28-3—Truth tables for multiple input logic gates	767
Table 28-4—Truth tables for multiple output logic gates	768
Table 28-5—Truth tables for three-state logic gates	769
Table 28-6—Truth tables for MOS switches	770
Table 28-7—Strength levels for scalar net signal values	773
Table 28-8—Strength reduction rules.....	786
Table 28-9—Rules for propagation delays.....	787
Table 29-1—UDP table symbols.....	794
Table 29-2—Initial statements in UDPs and modules.....	797
Table 29-3—Mixing of level-sensitive and edge-sensitive entries	801
Table 30-1—List of valid operators in state-dependent path delay expression.....	807
Table 30-2—Associating path delay expressions with transitions	813
Table 30-3—Calculating delays for x transitions	815
Table 31-1—\$setup arguments	829
Table 31-2—\$hold arguments	830
Table 31-3—\$setuphold arguments	831
Table 31-4—\$removal arguments	833
Table 31-5—\$recovery arguments	834
Table 31-6—\$recrem arguments	835
Table 31-7—\$skew arguments	837
Table 31-8—\$timeskew arguments.....	838
Table 31-9—\$fullskew arguments.....	840
Table 31-10—\$width arguments	842
Table 31-11—\$period arguments	843
Table 31-12—\$nochange arguments.....	844
Table 31-13—Notifier value responses to timing violations	846
Table 32-1—Mapping of SDF delay constructs to SystemVerilog declarations.....	856
Table 32-2—Mapping of SDF timing check constructs to SystemVerilog.....	857
Table 32-3—SDF annotation of interconnect delays	859
Table 32-4—SDF to SystemVerilog delay value mapping	862
Table 32-5—mtm_spec argument	863
Table 32-6—scale_type argument.....	864

Table 34-1—protect pragma keywords	883
Table 34-2—Encoding algorithm identifiers	887
Table 34-3—Encryption algorithm identifiers	889
Table 34-4—Message digest algorithm identifiers.....	894
Table 36-1—VPI routines for simulation-related callbacks	922
Table 36-2—VPI routines for system task or system function callbacks.....	923
Table 36-3—VPI routines for traversing SystemVerilog hierarchy	923
Table 36-4—VPI routines for accessing properties of objects	923
Table 36-5—VPI routines for accessing objects from properties.....	923
Table 36-6—VPI routines for delay processing	923
Table 36-7—VPI routines for logic and strength value processing.....	923
Table 36-8—VPI routines for simulation time processing.....	924
Table 36-9—VPI routines for miscellaneous utilities	924
Table 36-10—Summary of VPI incompatibilities across versions	925
Table 37-1—Part-select parent expressions	999
Table 38-1—Return error constants for vpi_chk_error().....	1016
Table 38-2—Size of the s_vpi_delay->da array	1023
Table 38-3—Return value field of the s_vpi_value structure union	1029
Table 38-4—Size of the s_vpi_delay->da array	1050
Table 38-5—Value format field of cb_data_p->value->format	1061
Table 38-6—cbStmt callbacks	1063
Table 40-1—Coverage control return values.....	1085
Table 40-2—Instance coverage permutations	1086
Table 40-3—Assertion coverage results.....	1094
Table B.1—Reserved keywords	1145
Table D.1—Argument return value for \$countdriver function.....	1152
Table H.1—Mapping data types.....	1189
Table N.1—SystemVerilog to C function cross-listing.....	1260

List of syntax excerpts

Syntax 5-1—Syntax for system tasks and system functions (excerpt from Annex A)	35
Syntax 5-2—Syntax for integer and real numbers (excerpt from Annex A)	36
Syntax 5-3—Syntax for attributes (excerpt from Annex A)	44
Syntax 6-1—Syntax for net type declarations (excerpt from Annex A)	56
Syntax 6-2—Syntax for net declarations (excerpt from Annex A)	61
Syntax 6-3—Syntax for variable declarations (excerpt from Annex A)	65
Syntax 6-4—User-defined types (excerpt from Annex A)	76
Syntax 6-5—Enumerated types (excerpt from Annex A)	78
Syntax 6-6—Parameter declaration syntax (excerpt from Annex A)	84
Syntax 6-7—Casting (excerpt from Annex A)	96
Syntax 7-1—Structure declaration syntax (excerpt from Annex A)	103
Syntax 7-2—Union declaration syntax (excerpt from Annex A)	106
Syntax 7-3—Dynamic array new constructor syntax (excerpt from Annex A)	114
Syntax 7-4—Declaration of queue dimension (excerpt from Annex A)	125
Syntax 7-5—Array method call syntax (not in Annex A)	129
Syntax 8-1—Class syntax (excerpt from Annex A)	136
Syntax 8-2—Calling a constructor (excerpt from Annex A)	140
Syntax 8-3—Class syntax (excerpt from Annex A)	159
Syntax 9-1—Syntax for structured procedures (excerpt from Annex A)	169
Syntax 9-2—Syntax for sequential block (excerpt from Annex A)	174
Syntax 9-3—Syntax for parallel block (excerpt from Annex A)	175
Syntax 9-4—Delay and event control syntax (excerpt from Annex A)	180
Syntax 9-5—Syntax for wait statement (excerpt from Annex A)	185
Syntax 9-6—Syntax for intra-assignment delay and event control (excerpt from Annex A)	186
Syntax 9-7—Syntax for process control statements (excerpt from Annex A)	189
Syntax 10-1—Syntax for continuous assignment (excerpt from Annex A)	197
Syntax 10-2—Blocking assignment syntax (excerpt from Annex A)	201
Syntax 10-3—Nonblocking assignment syntax (excerpt from Annex A)	202
Syntax 10-4—Syntax for procedural continuous assignments (excerpt from Annex A)	205
Syntax 10-5—Assignment patterns syntax (excerpt from Annex A)	210
Syntax 10-6—Syntax for net aliasing (excerpt from Annex A)	216
Syntax 11-1—Operator syntax (excerpt from Annex A)	219
Syntax 11-2—Conditional operator syntax (excerpt from Annex A)	234
Syntax 11-3—Inside expression syntax (excerpt from Annex A)	237
Syntax 11-4—Streaming concatenation syntax (excerpt from Annex A)	239
Syntax 11-5—With expression syntax (excerpt from Annex A)	241
Syntax 11-6—Tagged union syntax (excerpt from Annex A)	251
Syntax 11-7—Operator overloading syntax (excerpt from Annex A)	255
Syntax 11-8—Syntax for min:typ:max expression (excerpt from Annex A)	257

Syntax 11-9—Let syntax (excerpt from Annex A)	258
Syntax 12-1—Procedural statement syntax (excerpt from Annex A)	265
Syntax 12-2—Syntax for if-else statement (excerpt from Annex A)	265
Syntax 12-3—Syntax for case statements (excerpt from Annex A)	270
Syntax 12-4—Pattern syntax (excerpt from Annex A)	275
Syntax 12-5—Loop statement syntax (excerpt from Annex A)	279
Syntax 12-6—Jump statement syntax (excerpt from Annex A)	283
Syntax 13-1—Task syntax (excerpt from Annex A)	286
Syntax 13-2—Function syntax (excerpt from Annex A)	290
Syntax 13-3—Task or function call syntax (excerpt from Annex A)	296
Syntax 14-1—Clocking block syntax (excerpt from Annex A)	303
Syntax 14-2—Cycle delay syntax (excerpt from Annex A)	309
Syntax 14-3—Default clocking syntax (excerpt from Annex A)	310
Syntax 14-4—Global clocking syntax (excerpt from Annex A)	311
Syntax 14-5—Synchronous drive syntax (excerpt from Annex A)	316
Syntax 15-1—Event trigger syntax (excerpt from Annex A)	326
Syntax 15-2—Wait_order event sequencing syntax (excerpt from Annex A)	327
Syntax 16-1—Immediate assertion syntax (excerpt from Annex A)	331
Syntax 16-2—Deferred immediate assertion syntax (excerpt from Annex A)	334
Syntax 16-3—Sequence syntax (excerpt from Annex A)	345
Syntax 16-4—Sequence concatenation syntax (excerpt from Annex A)	346
Syntax 16-5—Declaring sequence syntax (excerpt from Annex A)	349
Syntax 16-6—Sequence repetition syntax (excerpt from Annex A)	357
Syntax 16-7—And operator syntax (excerpt from Annex A)	367
Syntax 16-8—Intersect operator syntax (excerpt from Annex A)	370
Syntax 16-9—Or operator syntax (excerpt from Annex A)	371
Syntax 16-10—First_match operator syntax (excerpt from Annex A)	373
Syntax 16-11—Throughout construct syntax (excerpt from Annex A)	375
Syntax 16-12—Within construct syntax (excerpt from Annex A)	376
Syntax 16-13—Assertion variable declaration syntax (excerpt from Annex A)	378
Syntax 16-14—Variable assignment syntax (excerpt from Annex A)	379
Syntax 16-15—Subroutine call in sequence syntax (excerpt from Annex A)	384
Syntax 16-16—Property construct syntax (excerpt from Annex A)	387
Syntax 16-17—Implication syntax (excerpt from Annex A)	390
Syntax 16-18—Followed-by syntax (excerpt from Annex A)	395
Syntax 16-19—Property statement case syntax (excerpt from Annex A)	404
Syntax 16-20—Concurrent assert construct syntax (excerpt from Annex A)	422
Syntax 16-21—Default clocking and default disable syntax (excerpt from Annex A)	438
Syntax 16-22—Expect statement syntax (excerpt from Annex A)	445
Syntax 17-1—Checker declaration syntax (excerpt from Annex A)	449

Syntax 17-2—Checker instantiation syntax (excerpt from Annex A).....	452
Syntax 18-1—Random variable declaration syntax (excerpt from Annex A).....	470
Syntax 18-2—Constraint syntax (excerpt from Annex A)	473
Syntax 18-3—Constraint distribution syntax (excerpt from Annex A)	476
Syntax 18-4—Uniqueness constraint syntax (excerpt from Annex A)	477
Syntax 18-5—Constraint implication syntax (excerpt from Annex A)	478
Syntax 18-6—If-else constraint syntax (excerpt from Annex A)	478
Syntax 18-7—Foreach iterative constraint syntax (excerpt from Annex A)	479
Syntax 18-8—Solve...before constraint ordering syntax (excerpt from Annex A).....	483
Syntax 18-9—Static constraint syntax (excerpt from Annex A)	484
Syntax 18-10—Inline constraint syntax (excerpt from Annex A)	493
Syntax 18-11—Scope randomize function syntax (not in Annex A)	500
Syntax 18-12—Randcase syntax (excerpt from Annex A).....	506
Syntax 18-13—Randsequence syntax (excerpt from Annex A).....	508
Syntax 18-14—Random production weights syntax (excerpt from Annex A)	509
Syntax 18-15—If-else conditional random production syntax (excerpt from Annex A)	509
Syntax 18-16—Case random production syntax (excerpt from Annex A)	510
Syntax 18-17—Repeat random production syntax (excerpt from Annex A)	510
Syntax 18-18—Rand join random production syntax (excerpt from Annex A)	511
Syntax 18-19—Random production syntax (excerpt from Annex A).....	513
Syntax 19-1—Covergroup syntax (excerpt from Annex A)	518
Syntax 19-2—Coverage point syntax (excerpt from Annex A)	523
Syntax 19-3—Transition bin syntax (excerpt from Annex A)	527
Syntax 19-4—Cross coverage syntax (excerpt from Annex A)	534
Syntax 20-1—Syntax for simulation control tasks (not in Annex A)	556
Syntax 20-2—Syntax for time system functions (not in Annex A)	556
Syntax 20-3—Syntax for \$printtimescale (not in Annex A)	558
Syntax 20-4—Syntax for \$timeformat (not in Annex A).....	559
Syntax 20-5—Type name function syntax (not in Annex A)	562
Syntax 20-6—Size function syntax (not in Annex A)	563
Syntax 20-7—Range function syntax (not in Annex A)	564
Syntax 20-8—Array querying function syntax (not in Annex A)	564
Syntax 20-9—Bit vector system function syntax (not in Annex A)	568
Syntax 20-10—Severity system task syntax (not in Annex A)	569
Syntax 20-11—Elaboration system task syntax (excerpt from Annex A)	570
Syntax 20-12—Assertion control syntax (not in Annex A)	572
Syntax 20-13—Sampled value system function syntax (not in Annex A)	578
Syntax 20-14—Syntax for \$random (not in Annex A)	579
Syntax 20-15—Syntax for probabilistic distribution functions (not in Annex A)	580
Syntax 20-16—Syntax for PLA modeling system task (not in Annex A)	583

Syntax 20-17—\$system function syntax (not in Annex A).....	587
Syntax 21-1—Syntax for \$display and \$write system tasks (not in Annex A)	589
Syntax 21-2—Syntax for \$strobe system tasks (not in Annex A)	598
Syntax 21-3—Syntax for \$monitor system tasks (not in Annex A)	598
Syntax 21-4—Syntax for \$fopen and \$fclose system tasks (not in Annex A)	599
Syntax 21-5—Syntax for file output system tasks (not in Annex A)	601
Syntax 21-6—Syntax for formatting data tasks (not in Annex A)	602
Syntax 21-7—Syntax for file read system functions (not in Annex A).....	603
Syntax 21-8—Syntax for file positioning system functions (not in Annex A)	607
Syntax 21-9—Syntax for file flush system task (not in Annex A)	608
Syntax 21-10—Syntax for file I/O error detection system function (not in Annex A)	609
Syntax 21-11—Syntax for end-of-file file detection system function (not in Annex A)	609
Syntax 21-12—Syntax for memory load system tasks (not in Annex A)	609
Syntax 21-13—\$writemem system task syntax (not in Annex A)	613
Syntax 21-14—Syntax for \$dumpfile task (not in Annex A)	617
Syntax 21-15—Syntax for \$dumpvars task (not in Annex A).....	618
Syntax 21-16—Syntax for \$dumpoff and \$dumpon tasks (not in Annex A)	619
Syntax 21-17—Syntax for \$dumpall task (not in Annex A)	619
Syntax 21-18—Syntax for \$dumplimit task (not in Annex A)	620
Syntax 21-19—Syntax for \$dumpflush task (not in Annex A)	620
Syntax 21-20—Syntax for output 4-state VCD file (not in Annex A)	622
Syntax 21-21—Syntax for \$dumpports task (not in Annex A)	627
Syntax 21-22—Syntax for \$dumpportsoff and \$dumpportson system tasks (not in Annex A)	628
Syntax 21-23—Syntax for \$dumpportsall system task (not in Annex A)	629
Syntax 21-24—Syntax for \$dumpportslimit system task (not in Annex A)	629
Syntax 21-25—Syntax for \$dumpportsflush system task (not in Annex A).....	629
Syntax 21-26—Syntax for \$vcdclose keyword (not in Annex A)	630
Syntax 21-27—Syntax for output extended VCD file (not in Annex A)	631
Syntax 21-28—Syntax for extended VCD node information (not in Annex A)	632
Syntax 21-29—Syntax for value change section (not in Annex A)	633
Syntax 22-1—Syntax for include compiler directive (not in Annex A)	639
Syntax 22-2—Syntax for text macro definition (not in Annex A)	640
Syntax 22-3—Syntax for text macro usage (not in Annex A)	641
Syntax 22-4—Syntax for undef compiler directive (not in Annex A)	645
Syntax 22-5—Syntax for conditional compilation directives (not in Annex A).....	645
Syntax 22-6—Syntax for timescale compiler directive (not in Annex A)	648
Syntax 22-7—Syntax for default_nettype compiler directive (not in Annex A)	650
Syntax 22-8—Syntax for pragma compiler directive (not in Annex A)	651
Syntax 22-9—Syntax for line compiler directive (not in Annex A)	652
Syntax 22-10—Syntax for begin_keywords and end_keywords compiler directives (not in Annex A)	653

Syntax 23-1—Module declaration syntax (excerpt from Annex A)	661
Syntax 23-2—Non-ANSI style module header declaration syntax (excerpt from Annex A)	662
Syntax 23-3—Non-ANSI style port declaration syntax (excerpt from Annex A)	663
Syntax 23-4—ANSI style list_of_port_declarations syntax (excerpt from Annex A)	666
Syntax 23-5—Module item syntax (excerpt from Annex A)	672
Syntax 23-6—Module instance syntax (excerpt from Annex A)	673
Syntax 23-7—Syntax for hierarchical path names (excerpt from Annex A)	686
Syntax 23-8—Syntax for upward name referencing (not in Annex A)	690
Syntax 23-9—Bind construct syntax (excerpt from Annex A)	702
Syntax 24-1—Program declaration syntax (excerpt from Annex A)	706
Syntax 25-1—Interface syntax (excerpt from Annex A)	712
Syntax 25-2—Modport clocking declaration syntax (excerpt from Annex A)	722
Syntax 25-3—Virtual interface declaration syntax (excerpt from Annex A)	732
Syntax 26-1—Package declaration syntax (excerpt from Annex A)	739
Syntax 26-2—Package import syntax (excerpt from Annex A)	740
Syntax 26-3—Package import in header syntax (excerpt from Annex A)	744
Syntax 26-4—Package export syntax (excerpt from Annex A)	746
Syntax 26-5—Std package import syntax (not in Annex A)	748
Syntax 27-1—Syntax for generate constructs (excerpt from Annex A)	751
Syntax 28-1—Syntax for gate instantiation (excerpt from Annex A)	761
Syntax 29-1—Syntax for UDPs (excerpt from Annex A)	792
Syntax 29-2—Syntax for UDP instances (excerpt from Annex A)	799
Syntax 30-1—Syntax for specify block (excerpt from Annex A)	802
Syntax 30-2—Syntax for module path declaration (excerpt from Annex A)	803
Syntax 30-3—Syntax for simple module path (excerpt from Annex A)	804
Syntax 30-4—Syntax for edge-sensitive path declaration (excerpt from Annex A)	805
Syntax 30-5—Syntax for state-dependent paths (excerpt from Annex A)	806
Syntax 30-6—Syntax for path delay value (excerpt from Annex A)	813
Syntax 30-7—Syntax for PATHPULSE\$ pulse control (excerpt from Annex A)	818
Syntax 30-8—Syntax for pulse style declarations (excerpt from Annex A)	820
Syntax 30-9—Syntax for showcanceled declarations (excerpt from Annex A)	821
Syntax 31-1—Syntax for system timing checks (excerpt from Annex A)	827
Syntax 31-2—Syntax for time check conditions and timing check events (excerpt from Annex A)	828
Syntax 31-3—Syntax for \$setup (excerpt from Annex A)	829
Syntax 31-4—Syntax for \$hold (excerpt from Annex A)	830
Syntax 31-5—Syntax for \$setuphold (excerpt from Annex A)	831
Syntax 31-6—Syntax for \$removal (excerpt from Annex A)	833
Syntax 31-7—Syntax for \$recovery (excerpt from Annex A)	833
Syntax 31-8—Syntax for \$crem (excerpt from Annex A)	834
Syntax 31-9—Syntax for \$skew (excerpt from Annex A)	836

Syntax 31-10—Syntax for \$timeskew (excerpt from Annex A)	837
Syntax 31-11—Syntax for \$fullskew (excerpt from Annex A)	840
Syntax 31-12—Syntax for \$width (excerpt from Annex A)	842
Syntax 31-13—Syntax for \$period (excerpt from Annex A)	843
Syntax 31-14—Syntax for \$nochange (excerpt from Annex A)	844
Syntax 31-15—Syntax for edge-control specifier (excerpt from Annex A)	845
Syntax 31-16—Syntax for controlled timing check events (excerpt from Annex A)	848
Syntax 32-1—Syntax for \$sdf_annotate system task (not in Annex A)	863
Syntax 33-1—Syntax for cell (excerpt from Annex A)	866
Syntax 33-2—Syntax for declaring library in library map file (excerpt from Annex A)	867
Syntax 33-3—Syntax for include command (excerpt from Annex A)	868
Syntax 33-4—Syntax for configurations (excerpt from Annex A)	869
Syntax 35-1—DPI import declaration syntax (excerpt from Annex A)	908
Syntax 35-2—DPI export declaration syntax (excerpt from Annex A)	913

Part One: Design and Verification Constructs

IEEE Standard for SystemVerilog— Unified Hardware Design, Specification, and Verification Language

IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

1.1 Scope

This standard provides the definition of the language syntax and semantics for the IEEE 1800™ SystemVerilog language, which is a unified hardware design, specification, and verification language. The standard includes support for behavioral, register transfer level (RTL), and gate-level hardware descriptions; testbench, coverage, assertion, object-oriented, and constrained random constructs; and also provides application programming interfaces (APIs) to foreign programming languages.

1.2 Purpose

This standard develops the IEEE 1800 SystemVerilog language in order to meet the increasing usage of the language in specification, design, and verification of hardware. This revision corrects errors and clarifies aspects of the language definition in IEEE Std 1800-2009.¹ This revision also provides enhanced features that ease design, improve verification, and enhance cross-language interactions.

1.3 Content summary

This standard serves as a complete specification of the SystemVerilog language. This standard contains the following:

- The formal syntax and semantics of all SystemVerilog constructs

¹Information on references can be found in [Clause 2](#).