

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

IEEE Computer Society

Developed by the
Design Automation Standards Committee

IEEE Std 1800™-2023
(Revision of IEEE Std 1800-2017)



STANDARDS

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

Developed by the

Design Automation Standards Committee
of the
IEEE Computer Society

Approved 6 December 2023

IEEE SA Standards Board

This is a Redline document produced by Accuris.

Redlines are designed to save time and improve productivity by using the latest software technology to find and highlight document changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document. This Redline is produced solely by Accuris. The publisher has not prepared this Redline and is not responsible for its accuracy.

Contact:

<https://store accuristech.com/pages/contact.html>

Redline Processing Notes:

1. ~~Red Text~~ - Red strikethrough text denotes deletions.
2. Blue Text - Blue underlined text denotes modifications and additions.

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 Std 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 © 2024 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 28 February 2024. 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 979-8-8557-0500-3 STDGT26763
Print: ISBN 979-8-8557-0501-0 STDPD26763

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.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. 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.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. IEEE Standards documents do not guarantee safety, security, health, or environmental protection, or guarantee against interference with or from other devices or networks. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. 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.

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 their 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.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

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 is 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 or inferred to be 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 the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, 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 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 subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#).¹ An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.²

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 constitute 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.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

¹ Available at: <https://development.standards.ieee.org/myproject-web/public/view.html#landing>.

² Available at: <https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html>.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted 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 these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards 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.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 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 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 [IEEE Xplore](#) or [contact IEEE](#).³ For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#).⁴ Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

³ Available at: <https://ieeexplore.ieee.org/browse/standards/collection/ieee>.

⁴ Available at: <https://standards.ieee.org/standard/index.html>.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).⁵

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 <https://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.

IMPORTANT NOTICE

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. IEEE standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users 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.

⁵ Available at: <https://standards.ieee.org/about/sasb/patcom/materials.html>.

Participants

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

Tom Fitzpatrick, Siemens Corporation, *Chair*
Don Mills, Accellera Systems Initiative, Inc., *Vice Chair*
Michiel Ligthart, Verific Design Automation, Inc., *Secretary*
Shalom Bresticker, Accellera Systems Initiative, Inc., *Editor*

<i>Organization Represented</i>	<i>Name of Representative</i>
Cadence Design System, Inc.....	Steven Sharp
Infineon Technologies.....	Thomas Kruse
Intel Corporation	Mehdi Mohatshemi
Marvell Technology, Inc.	Mark Strickland
NVIDIA Corporation	Justin Refice
Qualcomm Incorporated.....	Bob Pau
Siemens Corporation	Dave Rich
Synopsys Inc.	Dmitry Korchemny
Texas Instruments Incorporated.....	Lakshmanan Balasubramanian

The **SystemVerilog Language Working Group** gratefully acknowledges the contribution of the following participants. Without their assistance and dedication, this standard would not have been completed:

Tom Alsop	Chuck McClish
Boon Chong Ang	Rahul Mhatre
Luis Humberto Rezende Barbosa	Dillan Mills
Nayana Datta	Brad Pierce
Jonathan David	Arturo Salz
Prateek Jain	Erik Seligman
Robert Liu	Brandon Tipp
Matt Maidment	Asheque Zaidi
Francoise Martinolle	

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

Accellera Systems Initiative, Inc.	NVIDIA Corporation
Cadence Design Systems, Inc.	Samsung Electronics
Infineon Technologies	Shanghai Jiao Tong University
Institute of Biomedical Engineering	Siemens Corporation
Intel Corporation	STMicroelectronics
Marvell Technology, Inc.	Synopsys, Inc.
Microsoft Corporation	Verific Design Automation, Inc.

When the IEEE-SA Standards Board approved this standard on 6 December 2024, it had the following membership:

David J. Law, *Chair*
Ted Burse, *Vice Chair*
Gary Hoffman, *Past Chair*
Konstantinos Karachalios, *Secretary*

Sara R. Biyabani
Doug Edwards
Ramy Ahmed Fathy
Guido R. Hiertz
Yousef Kimiagar
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick

Joseph S. Levy
Howard Li
Johnny Daozhuang Lin
Gui Lin
Xiaohui Liu
Kevin W. Lu
Daleep C. Mohla
Andrew Myles

Paul Nikolic
Annette D. Reilly
Robby Robson
Lei Wang
F. Keith Waters
Karl Weber
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1800-2023, 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) [\[B4\]](#), published in 2004. 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.

Contents

Part One: Design and Verification Constructs

1.	Overview	41
1.1	Scope	41
1.2	Purpose	41
1.3	Content summary	41
1.4	Special terms	42
1.5	Conventions used in this standard	42
1.6	Syntactic description	43
1.7	Use of color in this standard	43
1.8	Contents of this standard	43
1.9	Deprecated clauses	47
1.10	Examples	47
1.11	Prerequisites	47
2.	Normative references	48
3.	Design and verification building blocks	50
3.1	General	50
3.2	Design elements	50
3.3	Modules	50
3.4	Programs	51
3.5	Interfaces	52
3.6	Checkers	53
3.7	Primitives	53
3.8	Subroutines	53
3.9	Packages	53
3.10	Configurations	54
3.11	Overview of hierarchy	54
3.12	Compilation and elaboration	55
3.13	Name spaces	57
3.14	Simulation time units and precision	58
4.	Scheduling semantics	62
4.1	General	62
4.2	Execution of a hardware model and its verification environment	62
4.3	Event simulation	62
4.4	Stratified event scheduler	63
4.5	SystemVerilog simulation reference algorithm	68
4.6	Determinism	69
4.7	Nondeterminism	69
4.8	Race conditions	69
4.9	Scheduling implication of assignments	70

4.10	PLI callback control points	71
5.	Lexical conventions	73
5.1	General	73
5.2	Lexical tokens	73
5.3	White space	73
5.4	Comments	73
5.5	Operators	73
5.6	Identifiers, keywords, and system names	74
5.7	Numbers	75
5.8	Time literals	80
5.9	String literals	80
5.10	Structure literals	83
5.11	Array literals	85
5.12	Attributes	85
5.13	Built-in methods	87
6.	Data types	88
6.1	General	88
6.2	Data types and data objects	88
6.3	Value set	88
6.4	Singular and aggregate types	89
6.5	Nets and variables	90
6.6	Net types	91
6.7	Net declarations	102
6.8	Variable declarations	105
6.9	Vector declarations	107
6.10	Implicit declarations	108
6.11	Integer data types	109
6.12	Real, shortreal, and realtime data types	110
6.13	Void data type	110
6.14	Chandle data type	111
6.15	Class	111
6.16	String data type	112
6.17	Event data type	117
6.18	User-defined types	117
6.19	Enumerations	119
6.20	Constants	124
6.21	Scope and lifetime	132
6.22	Type compatibility	134
6.23	Type operator	138
6.24	Casting	139
6.25	Parameterized data types	144
7.	Aggregate data types	146
7.1	General	146

7.2	Structures	146
7.3	Unions	148
7.4	Packed and unpacked arrays	153
7.5	Dynamic arrays	157
7.6	Array assignments.....	160
7.7	Arrays as arguments to subroutines	162
7.8	Associative arrays	163
7.9	Associative array methods	166
7.10	Queues	169
7.11	Array querying functions	173
7.12	Array manipulation methods	173
8.	Classes	179
8.1	General.....	179
8.2	Overview.....	179
8.3	Syntax	180
8.4	Objects (class instance).....	181
8.5	Object properties and object parameter data.....	183
8.6	Object methods	183
8.7	Constructors	184
8.8	Typed constructor calls	185
8.9	Static class properties.....	186
8.10	Static methods.....	186
8.11	This	187
8.12	Assignment, renaming, and copying.....	187
8.13	Inheritance and subclasses	189
8.14	Overridden members.....	190
8.15	Super	191
8.16	Casting	191
8.17	Chaining constructors	192
8.18	Data hiding and encapsulation	194
8.19	Constant class properties	195
8.20	Virtual methods.....	195
8.21	Abstract classes and pure virtual methods	198
8.22	Polymorphism: dynamic method lookup	199
8.23	Class scope resolution operator ::	200
8.24	Out-of-block declarations	202
8.25	Parameterized classes	203
8.26	Interface classes	206
8.27	Typedef class	215
8.28	Classes and structures	216
8.29	Memory management	216
8.30	Weak references.....	217

9.	Processes	220
9.1	General	220
9.2	Structured procedures	220
9.3	Block statements	224
9.4	Procedural timing controls	230
9.5	Process execution threads	240
9.6	Process control	240
9.7	Fine-grain process control	244
10.	Assignment statements	247
10.1	General	247
10.2	Overview	247
10.3	Continuous assignments	248
10.4	Procedural assignments	251
10.5	Variable declaration assignment (variable initialization)	256
10.6	Procedural continuous assignments	256
10.7	Assignment extension and truncation	258
10.8	Assignment-like contexts	259
10.9	Assignment patterns	260
10.10	Unpacked array concatenation	264
10.11	Net aliasing	267
11.	Operators and expressions	269
11.1	General	269
11.2	Overview	269
11.3	Operators	270
11.4	Operator descriptions	275
11.5	Operands	295
11.6	Expression bit lengths	298
11.7	Signed expressions	301
11.8	Expression evaluation rules	302
11.9	Tagged union expressions and member access	303
11.10	String literal expressions	304
11.11	Minimum, typical, and maximum delay expressions	306
11.12	Let construct	307
12.	Procedural programming statements	314
12.1	General	314
12.2	Overview	314
12.3	Syntax	314
12.4	Conditional if-else statement	315
12.5	Case statement	320
12.6	Pattern matching conditional statements	325
12.7	Loop statements	329
12.8	Jump statements	334

13.	Tasks and functions (subroutines)	335
13.1	General	335
13.2	Overview	335
13.3	Tasks	335
13.4	Functions	340
13.5	Subroutine calls and argument passing	347
13.6	Import and export functions	352
13.7	Task and function names	352
13.8	Parameterized tasks and functions	352
14.	Clocking blocks	354
14.1	General	354
14.2	Overview	354
14.3	Clocking block declaration	354
14.4	Input and output skews	356
14.5	Hierarchical expressions	357
14.6	Signals in multiple clocking blocks	358
14.7	Clocking block scope and lifetime	358
14.8	Multiple clocking blocks example	358
14.9	Interfaces and clocking blocks	359
14.10	Clocking block events	360
14.11	Cycle delay: ##	361
14.12	Default clocking	361
14.13	Input sampling	363
14.14	Global clocking	363
14.15	Synchronous events	367
14.16	Synchronous drives	368
15.	Interprocess synchronization and communication	372
15.1	General	372
15.2	Overview	372
15.3	Semaphores	372
15.4	Mailboxes	374
15.5	Named events	377
16.	Assertions	383
16.1	General	383
16.2	Overview	383
16.3	Immediate assertions	383
16.4	Deferred assertions	386
16.5	Concurrent assertions overview	393
16.6	Boolean expressions	396
16.7	Sequences	398
16.8	Declaring sequences	402
16.9	Sequence operations	410

16.10	Local variables	432
16.11	Calling subroutines on match of a sequence	438
16.12	Declaring properties	439
16.13	Multiclock support	465
16.14	Concurrent assertions	475
16.15	Disable iff resolution	493
16.16	Clock resolution	495
16.17	Expect statement	500
16.18	Clocking blocks and concurrent assertions	501
17.	Checkers	503
17.1	Overview	503
17.2	Checker declaration	503
17.3	Checker instantiation	506
17.4	Context inference	509
17.5	Checker procedures	510
17.6	Covergroups in checkers	512
17.7	Checker variables	513
17.8	Functions in checkers	519
17.9	Complex checker example	520
18.	Constrained random value generation	522
18.1	General	522
18.2	Overview	522
18.3	Concepts and usage	522
18.4	Random variables	525
18.5	Constraint blocks	528
18.6	Randomization methods	550
18.7	Inline constraints—randomize() with	552
18.8	Disabling random variables with rand_mode()	554
18.9	Controlling constraints with constraint_mode()	556
18.10	Dynamic constraint modification	557
18.11	Inline random variable control	557
18.12	Randomization of scope variables—std::randomize()	558
18.13	Random number system functions and methods	560
18.14	Random stability	561
18.15	Manually seeding randomize	564
18.16	Random weighted case—randcase	564
18.17	Random sequence generation—randsequence	565
19.	Functional coverage	575
19.1	General	575
19.2	Overview	575
19.3	Defining the coverage model: covergroup	576
19.4	Using covergroups in classes	578
19.5	Defining coverage points	582

19.6	Defining cross coverage.....	597
19.7	Specifying coverage options.....	605
19.8	Predefined coverage methods	610
19.9	Predefined coverage system tasks and system functions.....	612
19.10	Organization of option and type_option members	613
19.11	Coverage computation	614
20.	Utility system tasks and system functions	619
20.1	General.....	619
20.2	Simulation control system tasks	620
20.3	Simulation time system functions.....	620
20.4	Timescale system tasks and system functions	622
20.5	Conversion functions	626
20.6	Data query functions	628
20.7	Array query functions	630
20.8	Math functions	632
20.9	Bit vector system functions.....	633
20.10	Severity system tasks	635
20.11	Assertion control system tasks.....	637
20.12	Sampled value system functions.....	644
20.13	Coverage system functions	644
20.14	Probabilistic distribution functions	644
20.15	Stochastic analysis tasks and functions	646
20.16	Programmable logic array modeling system tasks	648
20.17	Miscellaneous tasks and functions.....	652
21.	Input/output system tasks and system functions.....	654
21.1	General.....	654
21.2	Display system tasks	654
21.3	File input/output system tasks and system functions.....	665
21.4	Loading memory array data from a file	676
21.5	Writing memory array data to a file.....	679
21.6	Command line input.....	680
21.7	Value change dump (VCD) files	683
22.	Compiler directives.....	704
22.1	General.....	704
22.2	Overview	704
22.3	`resetall.....	705
22.4	`include	705
22.5	`define, `undef, and `undefineall	706
22.6	`ifdef, `else, `elsif, `endif, `ifndef	712
22.7	`timescale	716
22.8	`default_nettype	718
22.9	`unconnected_drive and `nounconnected_drive	718
22.10	`celldefine and `endcelldefine.....	718

22.11	`pragma	718
22.12	`line	719
22.13	`__FILE__ and `__LINE__	720
22.14	`begin_keywords, `end_keywords	721

Part Two: Hierarchy Constructs

23.	Modules and hierarchy	728
23.1	General	728
23.2	Module definitions	728
23.3	Module instances (hierarchy)	740
23.4	Nested modules	751
23.5	Extern modules	752
23.6	Hierarchical names	753
23.7	Member selects and hierarchical names	757
23.8	Upwards name referencing	759
23.9	Scope rules	761
23.10	Overriding module parameters	763
23.11	Binding auxiliary code to scopes or instances	771
24.	Programs	774
24.1	General	774
24.2	Overview	774
24.3	The program construct	774
24.4	Eliminating testbench races	778
24.5	Blocking tasks in cycle/event mode	778
24.6	Program-wide space and anonymous programs	779
24.7	Program control tasks	779
25.	Interfaces	780
25.1	General	780
25.2	Overview	780
25.3	Interface syntax	781
25.4	Ports in interfaces	785
25.5	Modports	786
25.6	Interfaces and specify blocks	792
25.7	Tasks and functions in interfaces	793
25.8	Parameterized interfaces	799
25.9	Virtual interfaces	801
25.10	Access to interface objects	806
26.	Packages	807
26.1	General	807
26.2	Package declarations	807
26.3	Referencing data in packages	808
26.4	Using packages in module headers	812

26.5	Search order rules	813
26.6	Exporting imported names from packages	815
26.7	The std built-in package.....	816
27.	Generate constructs.....	818
27.1	General.....	818
27.2	Overview.....	818
27.3	Generate construct syntax	818
27.4	Loop generate constructs	820
27.5	Conditional generate constructs.....	824
27.6	External names for unnamed generate blocks	827
28.	Gate-level and switch-level modeling	829
28.1	General.....	829
28.2	Overview.....	829
28.3	Gate and switch declaration syntax	829
28.4	and, nand, nor, or, xor, and xnor gates.....	835
28.5	buf and not gates.....	836
28.6	bufif1, bufif0, notif1, and notif0 gates.....	837
28.7	MOS switches	838
28.8	Bidirectional pass switches	839
28.9	CMOS switches	840
28.10	pullup and pulldown sources	841
28.11	Logic strength modeling.....	842
28.12	Strengths and values of combined signals	843
28.13	Strength reduction by nonresistive devices	855
28.14	Strength reduction by resistive devices	855
28.15	Strengths of net types.....	855
28.16	Gate and net delays	856
29.	User-defined primitives	860
29.1	General.....	860
29.2	Overview.....	860
29.3	UDP definition	860
29.4	Combinational UDPs	864
29.5	Level-sensitive sequential UDPs	865
29.6	Edge-sensitive sequential UDPs.....	865
29.7	Sequential UDP initialization	866
29.8	UDP instances.....	868
29.9	Mixing level-sensitive and edge-sensitive descriptions.....	869
29.10	Level-sensitive dominance	870
30.	Specify blocks.....	871
30.1	General.....	871
30.2	Overview.....	871

30.3	Specify block declaration.....	871
30.4	Module path declarations.....	872
30.5	Assigning delays to module paths	881
30.6	Mixing module path delays and distributed delays	885
30.7	Detailed control of pulse filtering behavior.....	886
31.	Timing checks.....	895
31.1	General.....	895
31.2	Overview.....	895
31.3	Timing checks using a stability window.....	898
31.4	Timing checks for clock and control signals	905
31.5	Edge-control specifiers	914
31.6	Notifiers: user-defined responses to timing violations	915
31.7	Enabling timing checks with conditioned events	917
31.8	Vector signals in timing checks.....	918
31.9	Negative timing checks.....	919
32.	Backannotation using the standard delay format.....	924
32.1	General.....	924
32.2	Overview.....	924
32.3	The SDF annotator.....	924
32.4	Mapping of SDF constructs to SystemVerilog	924
32.5	Multiple annotations	929
32.6	Multiple SDF files	930
32.7	Pulse limit annotation	930
32.8	SDF to SystemVerilog delay value mapping.....	931
32.9	Loading timing data from an SDF file.....	932
33.	Configuring the contents of a design	934
33.1	General.....	934
33.2	Overview.....	934
33.3	Libraries	935
33.4	Configurations	937
33.5	Using libraries and configs	943
33.6	Configuration examples.....	944
33.7	Displaying library binding information	946
33.8	Library mapping examples	946
34.	Protected envelopes	949
34.1	General.....	949
34.2	Overview.....	949
34.3	Processing protected envelopes	949
34.4	Protect pragma directives.....	951
34.5	Protect pragma keywords.....	953

Part Three: Application Programming Interfaces

35.	Direct programming interface.....	970
35.1	General.....	970
35.2	Overview.....	970
35.3	Two layers of DPI.....	971
35.4	Global name space of imported and exported functions.....	972
35.5	Imported tasks and functions.....	973
35.6	Calling imported functions.....	981
35.7	Exported functions.....	982
35.8	Exported tasks.....	983
35.9	Disabling DPI tasks and functions.....	983
36.	Programming language interface (PLI/VPI) overview.....	985
36.1	General.....	985
36.2	PLI purpose and history.....	985
36.3	User-defined system task and system function names.....	986
36.4	User-defined system task and system function arguments.....	987
36.5	User-defined system task and system function types.....	987
36.6	User-supplied PLI applications.....	987
36.7	PLI include files.....	987
36.8	VPI sizetf, compiletf, and calltf routines.....	987
36.9	PLI mechanism.....	988
36.10	VPI access to SystemVerilog objects and simulation objects.....	990
36.11	List of VPI routines by functional category.....	991
36.12	VPI backwards compatibility features and limitations.....	993
37.	VPI object model diagrams.....	998
37.1	General.....	998
37.2	VPI handles.....	998
37.3	VPI object classifications.....	999
37.4	Key to data model diagrams.....	1005
37.5	Module.....	1008
37.6	Interface.....	1009
37.7	Modport.....	1009
37.8	Interface task or function declaration.....	1009
37.9	Program.....	1010
37.10	Instance.....	1011
37.11	Instance arrays.....	1013
37.12	Scope.....	1014
37.13	IO declaration.....	1015
37.14	Ports.....	1016
37.15	Reference objects.....	1017
37.16	Nets.....	1019
37.17	Variables.....	1025
37.18	Packed array variables.....	1029

37.19	Variable select	1030
37.20	Memory.....	1031
37.21	Variable drivers and loads	1031
37.22	Object range.....	1032
37.23	Nettype declaration	1032
37.24	Generic interconnect	1033
37.25	Typespec	1034
37.26	Structures and unions	1036
37.27	Named events	1037
37.28	Parameter, spec param, def param, param assign	1038
37.29	Virtual interface	1039
37.30	Interface typespec	1041
37.31	Class definition	1042
37.32	Class typespec	1043
37.33	Class variables and class objects	1045
37.34	Constraint, constraint ordering, distribution	1047
37.35	Primitive, prim term.....	1048
37.36	UDP	1049
37.37	Intermodule path.....	1049
37.38	Constraint expression	1050
37.39	Module path, path term	1051
37.40	Timing check	1052
37.41	Task and function declaration	1053
37.42	Task and function call	1054
37.43	Frames	1056
37.44	Threads	1057
37.45	Delay terminals	1057
37.46	Net drivers and loads	1058
37.47	Continuous assignment	1059
37.48	Clocking block	1060
37.49	Assertion	1061
37.50	Concurrent assertion	1062
37.51	Property declaration	1063
37.52	Property specification	1064
37.53	Sequence declaration	1065
37.54	Sequence expression	1066
37.55	Immediate assertions	1067
37.56	Multiclock sequence expression	1068
37.57	Let	1068
37.58	Simple expressions	1069
37.59	Expressions	1070
37.60	Atomic statement	1073
37.61	Dynamic prefixing	1074
37.62	Event statement	1075
37.63	Process	1075
37.64	Assignment	1076
37.65	Event control	1076
37.66	While, repeat.....	1077
37.67	Waits	1077
37.68	Delay control.....	1077

37.69	Repeat control	1078
37.70	Forever	1078
37.71	If, if-else	1078
37.72	Case, pattern	1079
37.73	Expect	1080
37.74	For	1080
37.75	Do-while, foreach	1080
37.76	Alias statement	1081
37.77	Disables	1081
37.78	Return statement	1081
37.79	Assign statement, deassign, force, release	1082
37.80	Callback	1082
37.81	Time queue	1083
37.82	Active time format	1083
37.83	Attribute	1084
37.84	Iterator	1085
37.85	Generates	1086
38.	VPI routine definitions	1088
38.1	General	1088
38.2	vpi_chk_error()	1088
38.3	vpi_compare_objects()	1089
38.4	vpi_control()	1091
38.5	vpi_flush()	1092
38.6	vpi_get()	1092
38.7	vpi_get64()	1093
38.8	vpi_get_cb_info()	1093
38.9	vpi_get_data()	1094
38.10	vpi_get_delays()	1095
38.11	vpi_get_str()	1097
38.12	vpi_get_systf_info()	1098
38.13	vpi_get_time()	1099
38.14	vpi_get_userdata()	1100
38.15	vpi_get_value()	1100
38.16	vpi_get_value_array()	1106
38.17	vpi_get_vlog_info()	1110
38.18	vpi_handle()	1111
38.19	vpi_handle_by_index()	1112
38.20	vpi_handle_by_multi_index()	1112
38.21	vpi_handle_by_name()	1113
38.22	vpi_handle_multi()	1114
38.23	vpi_iterate()	1114
38.24	vpi_mcd_close()	1115
38.25	vpi_mcd_flush()	1116
38.26	vpi_mcd_name()	1116
38.27	vpi_mcd_open()	1117
38.28	vpi_mcd_printf()	1118
38.29	vpi_mcd_vprintf()	1119

38.30	vpi_printf()	1119
38.31	vpi_put_data()	1120
38.32	vpi_put_delays()	1122
38.33	vpi_put_userdata()	1125
38.34	vpi_put_value()	1125
38.35	vpi_put_value_array()	1128
38.36	vpi_register_cb()	1132
38.37	vpi_register_systf()	1140
38.38	vpi_release_handle()	1144
38.39	vpi_remove_cb()	1144
38.40	vpi_scan()	1145
38.41	vpi_vprintf()	1146
39.	Assertion API	1147
39.1	General	1147
39.2	Overview	1147
39.3	Static information	1147
39.4	Dynamic information	1148
39.5	Control functions	1152
40.	Code coverage control and API	1156
40.1	General	1156
40.2	Overview	1156
40.3	SystemVerilog real-time coverage access	1157
40.4	FSM recognition	1162
40.5	VPI coverage extensions	1165
41.	Data read API	1170

Part Four: Annexes

Annex A (normative) Formal syntax	1172
A.1 Source text	1172
A.2 Declarations	1181
A.3 Primitive instances	1193
A.4 Instantiations	1194
A.5 UDP declaration and instantiation	1196
A.6 Behavioral statements	1197
A.7 Specify section	1204
A.8 Expressions	1208
A.9 General	1213
A.10 BNF clarifications	1216
Annex B (normative) Keywords	1219

Annex C (normative) Deprecation.....	1221
C.1 General.....	1221
C.2 Constructs that have been deprecated	1221
C.3 Accellera SystemVerilog 3.1a-compatible access to packed data	1222
C.4 Constructs identified for deprecation.....	1222
Annex D (informative) Optional system tasks and system functions.....	1225
D.1 General.....	1225
D.2 \$countdrivers	1225
D.3 \$getpattern	1226
D.4 \$input	1227
D.5 \$key and \$nokey	1227
D.6 \$list.....	1227
D.7 \$log and \$nolog	1227
D.8 \$reset, \$reset_count, and \$reset_value	1228
D.9 \$save, \$restart, and \$incsave.....	1229
D.10 \$scale	1230
D.11 \$scope	1230
D.12 \$showscopes	1230
D.13 \$showvars	1230
D.14 \$readmemb and \$readmemh.....	1230
Annex E (informative) Optional compiler directives	1232
E.1 General.....	1232
E.2 `default_decay_time.....	1232
E.3 `default_trireg_strength	1232
E.4 `delay_mode_distributed	1233
E.5 `delay_mode_path.....	1233
E.6 `delay_mode_unit	1233
E.7 `delay_mode_zero.....	1233
Annex F (normative) Formal semantics of concurrent assertions	1234
F.1 General.....	1234
F.2 Overview.....	1234
F.3 Abstract syntax	1235
F.4 Rewriting algorithms	1241
F.5 Semantics	1245
F.6 Extended expressions.....	1255
F.7 Recursive properties	1255
Annex G (normative) Std package.....	1257
G.1 General.....	1257
G.2 Overview.....	1257
G.3 Semaphore	1257
G.4 Mailbox	1257
G.5 Randomize	1258
G.6 Process	1258
G.7 Weak reference	1258

Annex H (normative) DPI C layer	1259
H.1 General.....	1259
H.2 Overview.....	1259
H.3 Naming conventions	1260
H.4 Portability.....	1260
H.5 svdpi.h include file.....	1260
H.6 Semantic constraints	1261
H.7 Data types	1263
H.8 Argument passing modes.....	1267
H.9 Context tasks and functions	1270
H.10 Include files.....	1274
H.11 Arrays.....	1277
H.12 Open arrays	1280
H.13 Time and timescale	1286
H.14 SV3.1a-compatible access to packed data (deprecated functionality).....	1286
Annex I (normative) svdpi.h	1292
I.1 General.....	1292
I.2 Overview.....	1292
I.3 Source code.....	1292
Annex J (normative) Inclusion of foreign language code.....	1302
J.1 General.....	1302
J.2 Overview.....	1302
J.3 Location independence	1303
J.4 Object code inclusion.....	1303
Annex K (normative) vpi_user.h	1306
K.1 General.....	1306
K.2 Source code.....	1306
Annex L (normative) vpi_compatibility.h	1323
L.1 General.....	1323
L.2 Source code.....	1323
Annex M (normative) sv_vpi_user.h	1326
M.1 General.....	1326
M.2 Source code.....	1326
Annex N (normative) Algorithm for probabilistic distribution functions	1337
N.1 General.....	1337
N.2 Source code.....	1337
Annex O (informative) Encryption/decryption flow	1345
O.1 General.....	1345
O.2 Overview.....	1345

O.3 Tool vendor secret key encryption system 1345

O.4 IP author secret key encryption system 1346

O.5 Digital envelopes 1347

Annex P (informative) Glossary 1349

Annex Q (informative) Bibliography 1352

List of figures

Figure 4-1—Event scheduling regions	67
Figure 6-1—Simulation values of a trireg and its driver	94
Figure 6-2—Simulation results of a capacitive network	95
Figure 6-3—Simulation results of charge sharing	96
Figure 7-1—Data_u type with soft packed qualifier	150
Figure 7-2—VInt type with packed qualifier	152
Figure 7-3—Instr type with packed qualifier	152
Figure 9-1—Intra-assignment repeat event control utilizing a clock edge.....	239
Figure 14-1—Sample and drive times including skew with respect to the positive edge of the clock	357
Figure 16-1—Sampling a variable in a simulation time step	395
Figure 16-2—Concatenation of sequences	401
Figure 16-3—Value change expressions	416
Figure 16-4—Future value change	420
Figure 16-5—ANDing (and) two sequences	422
Figure 16-6—ANDing (and) two sequences, including a time range	423
Figure 16-7—ANDing (and) two Boolean expressions	423
Figure 16-8—Intersecting two sequences.....	424
Figure 16-9—ORing (or) two Boolean expressions	425
Figure 16-10—ORing (or) two sequences.....	426
Figure 16-11—ORing (or) two sequences, including a time range.....	427
Figure 16-12—Match with throughout restriction fails.....	429
Figure 16-13—Match with throughout restriction succeeds	430
Figure 16-14—Conditional sequence matching	446
Figure 16-15—Conditional sequences.....	447
Figure 16-16—Results without the condition.....	448
Figure 16-17—Clocking blocks and concurrent assertion	502
Figure 17-1—Nondeterministic free checker variable	514
Figure 18-1—Example of randc	528
Figure 18-2—Global constraints	539
Figure 18-3—Truth tables for conjunction, disjunction, and negation rules.....	545
Figure 21-1—Creating the 4-state VCD file.....	683
Figure 21-2—Creating the extended VCD file.....	693
Figure 23-1—Hierarchy in a model.....	755
Figure 23-2—Scopes available to upward name referencing	762
Figure 28-1—Schematic diagram of interconnections in array of instances.....	835
Figure 28-2—Scale of strengths	843
Figure 28-3—Combining unequal strengths.....	844
Figure 28-4—Combination of signals of equal strength and opposite values	844
Figure 28-5—Weak x signal strength.....	844

Figure 28-6—Bufifs with control inputs of x	845
Figure 28-7—Strong H range of values.....	845
Figure 28-8—Strong L range of values	845
Figure 28-9—Combined signals of ambiguous strength	846
Figure 28-10—Range of strengths for an unknown signal.....	846
Figure 28-11—Ambiguous strengths from switch networks.....	846
Figure 28-12—Range of two strengths of a defined value	847
Figure 28-13—Range of three strengths of a defined value	847
Figure 28-14—Unknown value with a range of strengths.....	847
Figure 28-15—Strong X range	848
Figure 28-16—Ambiguous strength from gates	848
Figure 28-17—Ambiguous strength signal from a gate	848
Figure 28-18—Weak 0	849
Figure 28-19—Ambiguous strength in combined gate signals	849
Figure 28-20—Elimination of strength levels	850
Figure 28-21—Result showing a range and the elimination of strength levels of two values	851
Figure 28-22—Result showing a range and the elimination of strength levels of one value	852
Figure 28-23—A range of both values	852
Figure 28-24—Wired logic with unambiguous strength signals	853
Figure 28-25—Wired logic and ambiguous strengths.....	854
Figure 28-26—Trireg net with capacitance	859
Figure 29-1—Module schematic and simulation times of initial value propagation	868
Figure 30-1—Module path delays	873
Figure 30-2—Difference between parallel and full connection paths.....	879
Figure 30-3—Module path delays longer than distributed delays.....	886
Figure 30-4—Module path delays shorter than distributed delays.....	886
Figure 30-5—Example of pulse filtering.....	887
Figure 30-6—On-detect versus on-event.....	889
Figure 30-7—Current event cancellation problem and correction	891
Figure 30-8—NAND gate with nearly simultaneous input switching where one event is scheduled prior to another that has not matured	892
Figure 30-9—NAND gate with nearly simultaneous input switching with output event scheduled at same time	893
Figure 31-1—Sample \$timeskew	907
Figure 31-2—Sample \$timeskew with remain_active_flag set.....	908
Figure 31-3—Sample \$fullskew	910
Figure 31-4—Data constraint interval, positive setup/hold.....	919
Figure 31-5—Data constraint interval, negative setup/hold	920
Figure 31-6—Timing check violation windows.....	923
Figure 37-1—Example of object relationships diagram.....	1000
Figure 37-2—Accessing a class of objects using tags	1001

Figure 38-1—s_vpi_error_info structure definition	1089
Figure 38-2—s_cb_data structure definition	1094
Figure 38-3—s_vpi_delay structure definition.....	1095
Figure 38-4—s_vpi_time structure definition	1095
Figure 38-5—s_vpi_systf_data structure definition	1098
Figure 38-6—s_vpi_time structure definition	1099
Figure 38-7—s_vpi_value structure definition.....	1101
Figure 38-8—s_vpi_vecval structure definition	1101
Figure 38-9—s_vpi_strengthval structure definition.....	1101
Figure 38-10—s_vpi_vlog_info structure definition.....	1110
Figure 38-11—s_vpi_delay structure definition.....	1123
Figure 38-12—s_vpi_time structure definition	1123
Figure 38-13—s_vpi_value structure definition.....	1127
Figure 38-14—s_vpi_time structure definition	1127
Figure 38-15—s_vpi_vecval structure definition	1128
Figure 38-16—s_vpi_strengthval structure definition.....	1128
Figure 38-17—s_cb_data structure definition	1132
Figure 38-18—s_vpi_systf_data structure definition	1141
Figure 39-1—Assertions with global clocking future sampled value functions	1152
Figure 40-1—Hierarchical instance example	1160
Figure 40-2—FSM specified with pragmas.....	1165

List of tables

Table 3-1—Time unit strings.....	58
Table 4-1—PLI callbacks	72
Table 5-1—Specifying special characters in string literals	83
Table 6-1—Built-in net types	92
Table 6-2—Truth table for wire and tri nets	92
Table 6-3—Truth table for wand and triand nets	93
Table 6-4—Truth table for wor and trior nets	93
Table 6-5—Truth table for tri0 net	97
Table 6-6—Truth table for tri1 net	97
Table 6-7—Default variable initial values.....	107
Table 6-8—Integer data types.....	109
Table 6-9—String operators	113
Table 6-10—Enumeration element ranges	121
Table 6-11—Differences between specparams and parameters	129
Table 7-1—Value read from a nonexistent array entry	156
Table 8-1—Comparison of pointer and handle types	182
Table 9-1—fork-join control options.....	226
Table 9-2—Detecting posedge and negedge	232
Table 9-3—Intra-assignment timing control equivalence	238
Table 10-1—Legal left-hand forms in assignment statements	247
Table 11-1—Operators and data types	271
Table 11-2—Operator precedence and associativity	272
Table 11-3—Arithmetic operators defined	275
Table 11-4—Integral power operator rules.....	276
Table 11-5—Examples of modulus and power operators	276
Table 11-6—Unary operators defined	277
Table 11-7—Data type interpretation by arithmetic operators.....	277
Table 11-8—Definitions of relational operators	278
Table 11-9—Definitions of equality operators.....	279
Table 11-10—Wildcard equality and wildcard inequality operators.....	280
Table 11-11—Bitwise binary AND operator.....	282
Table 11-12—Bitwise binary OR operator.....	282
Table 11-13—Bitwise binary exclusive OR operator.....	282
Table 11-14—Bitwise binary exclusive NOR operator.....	282
Table 11-15—Bitwise unary negation operator.....	283
Table 11-16—Reduction unary AND operator	283
Table 11-17—Reduction unary OR operator.....	283
Table 11-18—Reduction unary exclusive OR operator	284
Table 11-19—Results of unary reduction operations	284

Table 11-20—Ambiguous condition results for conditional operator	285
Table 11-21—Bit lengths resulting from self-determined expressions	299
Table 16-1—Operator precedence and associativity	410
Table 16-2—Global clocking future sampled value functions	420
Table 16-3—Sequence and property operator precedence and associativity	442
Table 18-1—Unordered constraint c legal value probability	540
Table 18-2—Ordered constraint c legal value probability	540
Table 18-3—rand_mode argument	555
Table 18-4—constraint_mode argument	556
Table 19-1—Instance-specific coverage options	606
Table 19-2—Coverage options per syntactic level	608
Table 19-3—Coverage group type (static) options	608
Table 19-4—Coverage type options	610
Table 19-5—Predefined coverage methods	610
Table 20-1—Diagnostics for \$finish	620
Table 20-2—Time unit and precision number values	623
Table 20-3—\$timeformat default values for arguments	625
Table 20-4—SystemVerilog to C real math function cross-listing	633
Table 20-5—Values for control_type for assertion control tasks	638
Table 20-6—Values for assertion_type for assertion control tasks	639
Table 20-7—Values for directive_type for assertion control tasks	639
Table 20-8—VPI callbacks for assertion control tasks	643
Table 20-9—Types of queues of \$q_type values	647
Table 20-10—Argument values for \$q_exam system task	648
Table 20-11—Status code values	648
Table 20-12—PLA modeling system tasks	649
Table 21-1—Escape sequences for format specifications	656
Table 21-2—Format specifications for real numbers	658
Table 21-3—Logic value component of strength format	660
Table 21-4—Mnemonics for strength levels	661
Table 21-5—Explanation of strength formats	662
Table 21-6—Types for file descriptors	666
Table 21-7—\$fscanf input field characters	671
Table 21-8—Rules for left-extending vector values	689
Table 21-9—How the VCD can shorten values	689
Table 21-10—Keyword commands	690
Table 21-11—VCD type mapping	702
Table 22-1—IEEE Std 1364-1995 reserved keywords	723
Table 22-2—IEEE Std 1364-2001 additional reserved keywords	724
Table 22-3—IEEE Std 1364-2005 additional reserved keywords	724

Table 22-4—IEEE Std 1800-2005 additional reserved keywords	725
Table 22-5—IEEE Std 1800-2009 additional reserved keywords	725
Table 22-6—IEEE Std 1800-2012 additional reserved keywords	726
Table 23-1—Net types resulting from dissimilar port connections.....	750
Table 26-1—Scoping rules for package importation.....	814
Table 28-1—Built-in gates and switches.....	831
Table 28-2—Valid gate types for strength specifications	831
Table 28-3—Truth tables for multiple input logic gates	836
Table 28-4—Truth tables for multiple output logic gates	837
Table 28-5—Truth tables for three-state logic gates	838
Table 28-6—Truth tables for MOS switches	839
Table 28-7—Strength levels for scalar net signal values	842
Table 28-8—Strength reduction rules.....	855
Table 28-9—Rules for propagation delays	856
Table 29-1—UDP table symbols	863
Table 29-2—Initial statements in UDPs and modules.....	866
Table 29-3—Mixing of level-sensitive and edge-sensitive entries	870
Table 30-1—List of valid operators in state-dependent path delay expression.....	876
Table 30-2—Associating path delay expressions with transitions	883
Table 30-3—Calculating delays for x transitions	884
Table 31-1—\$setup arguments	898
Table 31-2—\$hold arguments	899
Table 31-3—\$setuphold arguments	900
Table 31-4—\$removal arguments	902
Table 31-5—\$recovery arguments	903
Table 31-6—\$recrem arguments	904
Table 31-7—\$skew arguments	906
Table 31-8—\$timeskew arguments.....	907
Table 31-9—\$fullskew arguments.....	909
Table 31-10—\$width arguments	911
Table 31-11—\$period arguments	912
Table 31-12—\$nochange arguments	913
Table 31-13—Notifier value responses to timing violations	915
Table 32-1—Mapping of SDF delay constructs to SystemVerilog declarations.....	925
Table 32-2—Mapping of SDF timing check constructs to SystemVerilog.....	926
Table 32-3—SDF annotation of interconnect delays	928
Table 32-4—SDF to SystemVerilog delay value mapping	931
Table 32-5—mtm_spec argument	932
Table 32-6—scale_type argument	933
Table 34-1—protect pragma keywords	952

Table 34-2—Encoding algorithm identifiers	956
Table 34-3—Encryption algorithm identifiers	958
Table 34-4—Message digest algorithm identifiers.....	963
Table 36-1—VPI routines for simulation-related callbacks	991
Table 36-2—VPI routines for system task or system function callbacks.....	992
Table 36-3—VPI routines for traversing SystemVerilog hierarchy	992
Table 36-4—VPI routines for accessing properties of objects	992
Table 36-5—VPI routines for accessing objects from properties.....	992
Table 36-6—VPI routines for delay processing	992
Table 36-7—VPI routines for logic and strength value processing.....	992
Table 36-8—VPI routines for simulation time processing.....	993
Table 36-9—VPI routines for miscellaneous utilities	993
Table 36-10—Summary of VPI incompatibilities across versions	994
Table 37-1—Part-select parent expressions	1072
Table 38-1—Return error constants for vpi_chk_error().....	1089
Table 38-2—Size of the s_vpi_delay->da array	1096
Table 38-3—Return value field of the s_vpi_value structure union	1102
Table 38-4—Size of the s_vpi_delay->da array	1123
Table 38-5—Value format field of cb_data_p->value->format	1134
Table 38-6—cbStmt callbacks.....	1136
Table 40-1—Coverage control return values.....	1158
Table 40-2—Instance coverage permutations	1159
Table 40-3—Assertion coverage results.....	1167
Table B.1—Reserved keywords	1219
Table D.1—Argument return value for \$countdriver function.....	1226
Table H.1—Mapping data types.....	1264
Table N.1—SystemVerilog to C function cross-listing.....	1337

List of syntax excerpts

Syntax 5-1—Syntax for system tasks and system functions (excerpt from Annex A).....	75
Syntax 5-2—Syntax for integer and real numbers (excerpt from Annex A).....	76
Syntax 5-3—Syntax for string literals (excerpt from Annex A)	80
Syntax 5-4—Syntax for attributes (excerpt from Annex A).....	85
Syntax 6-1—Syntax for net type declarations (excerpt from Annex A)	97
Syntax 6-2—Syntax for net declarations (excerpt from Annex A)	102
Syntax 6-3—Syntax for variable declarations (excerpt from Annex A)	106
Syntax 6-4—User-defined types (excerpt from Annex A)	117
Syntax 6-5—Enumerated types (excerpt from Annex A).....	119
Syntax 6-6—Parameter declaration syntax (excerpt from Annex A).....	125
Syntax 6-7—Casting (excerpt from Annex A)	139
Syntax 7-1—Structure declaration syntax (excerpt from Annex A)	146
Syntax 7-2—Union declaration syntax (excerpt from Annex A)	149
Syntax 7-3—Dynamic array new constructor syntax (excerpt from Annex A)	158
Syntax 7-4—Declaration of queue dimension (excerpt from Annex A)	169
Syntax 7-5—Array method call syntax (not in Annex A)	173
Syntax 8-1—Class syntax (excerpt from Annex A)	181
Syntax 8-2—Calling a constructor (excerpt from Annex A).....	185
Syntax 8-3—Interface class syntax (excerpt from Annex A).....	208
Syntax 9-1—Syntax for structured procedures (excerpt from Annex A)	220
Syntax 9-2—Syntax for sequential block (excerpt from Annex A)	225
Syntax 9-3—Syntax for parallel block (excerpt from Annex A).....	226
Syntax 9-4—Delay and event control syntax (excerpt from Annex A).....	231
Syntax 9-5—Syntax for wait statement (excerpt from Annex A)	236
Syntax 9-6—Syntax for intra-assignment delay and event control (excerpt from Annex A)	237
Syntax 9-7—Syntax for process control statements (excerpt from Annex A)	240
Syntax 10-1—Syntax for continuous assignment (excerpt from Annex A).....	248
Syntax 10-2—Blocking assignment syntax (excerpt from Annex A)	252
Syntax 10-3—Nonblocking assignment syntax (excerpt from Annex A).....	253
Syntax 10-4—Syntax for procedural continuous assignments (excerpt from Annex A)	256
Syntax 10-5—Assignment patterns syntax (excerpt from Annex A)	261
Syntax 10-6—Syntax for net aliasing (excerpt from Annex A)	267
Syntax 11-1—Operator syntax (excerpt from Annex A).....	271
Syntax 11-2—Conditional operator syntax (excerpt from Annex A).....	285
Syntax 11-3—Inside expression syntax (excerpt from Annex A).....	289
Syntax 11-4—Streaming concatenation syntax (excerpt from Annex A)	290
Syntax 11-5—With expression syntax (excerpt from Annex A).....	293
Syntax 11-6—Tagged union syntax (excerpt from Annex A).....	303
Syntax 11-7—Syntax for min:typ:max expression (excerpt from Annex A).....	307

Syntax 11-8—Let syntax (excerpt from Annex A)	308
Syntax 12-1—Procedural statement syntax (excerpt from Annex A)	315
Syntax 12-2—Syntax for if-else statement (excerpt from Annex A)	315
Syntax 12-3—Syntax for case statements (excerpt from Annex A)	320
Syntax 12-4—Pattern syntax (excerpt from Annex A)	325
Syntax 12-5—Loop statement syntax (excerpt from Annex A)	329
Syntax 12-6—Jump statement syntax (excerpt from Annex A)	334
Syntax 13-1—Task syntax (excerpt from Annex A)	336
Syntax 13-2—Function syntax (excerpt from Annex A)	341
Syntax 13-3—Task or function call syntax (excerpt from Annex A)	347
Syntax 14-1—Clocking block syntax (excerpt from Annex A)	355
Syntax 14-2—Cycle delay syntax (excerpt from Annex A)	361
Syntax 14-3—Default clocking syntax (excerpt from Annex A)	362
Syntax 14-4—Global clocking syntax (excerpt from Annex A)	364
Syntax 14-5—Synchronous drive syntax (excerpt from Annex A)	368
Syntax 15-1—Event trigger syntax (excerpt from Annex A)	378
Syntax 15-2—Wait_order event sequencing syntax (excerpt from Annex A)	380
Syntax 16-1—Immediate assertion syntax (excerpt from Annex A)	384
Syntax 16-2—Deferred immediate assertion syntax (excerpt from Annex A)	386
Syntax 16-3—Sequence syntax (excerpt from Annex A)	399
Syntax 16-4—Sequence concatenation syntax (excerpt from Annex A)	400
Syntax 16-5—Sequence declaration syntax (excerpt from Annex A)	402
Syntax 16-6—Sequence repetition syntax (excerpt from Annex A)	410
Syntax 16-7—And operator syntax (excerpt from Annex A)	421
Syntax 16-8—Intersect operator syntax (excerpt from Annex A)	424
Syntax 16-9—Or operator syntax (excerpt from Annex A)	425
Syntax 16-10—First_match operator syntax (excerpt from Annex A)	427
Syntax 16-11—Throughout construct syntax (excerpt from Annex A)	429
Syntax 16-12—Within construct syntax (excerpt from Annex A)	430
Syntax 16-13—Assertion variable declaration syntax (excerpt from Annex A)	433
Syntax 16-14—Variable assignment syntax (excerpt from Annex A)	433
Syntax 16-15—Subroutine call in sequence syntax (excerpt from Annex A)	438
Syntax 16-16—Property construct syntax (excerpt from Annex A)	441
Syntax 16-17—Property statement case syntax (excerpt from Annex A)	458
Syntax 16-18—Concurrent assertion construct syntax (excerpt from Annex A)	476
Syntax 16-19—Default disable syntax (excerpt from Annex A)	493
Syntax 16-20—Expect statement syntax (excerpt from Annex A)	500
Syntax 17-1—Checker declaration syntax (excerpt from Annex A)	504
Syntax 17-2—Checker instantiation syntax (excerpt from Annex A)	507
Syntax 18-1—Random variable declaration syntax (excerpt from Annex A)	525

Syntax 18-2—Constraint syntax (excerpt from Annex A)	529
Syntax 18-3—Constraint distribution syntax (excerpt from Annex A)	532
Syntax 18-4—Uniqueness constraint syntax (excerpt from Annex A)	534
Syntax 18-5—Constraint implication syntax (excerpt from Annex A)	535
Syntax 18-6—If-else constraint syntax (excerpt from Annex A)	536
Syntax 18-7—Foreach iterative constraint syntax (excerpt from Annex A)	537
Syntax 18-8—Solve...before constraint ordering syntax (excerpt from Annex A)	541
Syntax 18-9—Static constraint syntax (excerpt from Annex A)	542
Syntax 18-10—Inline constraint syntax (excerpt from Annex A)	552
Syntax 18-11—Scope randomize function syntax (not in Annex A)	558
Syntax 18-12—Randcase syntax (excerpt from Annex A)	564
Syntax 18-13—Randsequence syntax (excerpt from Annex A)	566
Syntax 18-14—Random production weights syntax (excerpt from Annex A)	567
Syntax 18-15—If-else conditional random production syntax (excerpt from Annex A)	568
Syntax 18-16—Case random production syntax (excerpt from Annex A)	568
Syntax 18-17—Repeat random production syntax (excerpt from Annex A)	569
Syntax 18-18—Rand join random production syntax (excerpt from Annex A)	569
Syntax 18-19—Random production syntax (excerpt from Annex A)	571
Syntax 19-1—Covergroup syntax (excerpt from Annex A)	577
Syntax 19-2—Coverage point syntax (excerpt from Annex A)	582
Syntax 19-3—Transition bin syntax (excerpt from Annex A)	591
Syntax 19-4—Cross coverage syntax (excerpt from Annex A)	598
Syntax 20-1—Syntax for simulation control tasks (not in Annex A)	620
Syntax 20-2—Syntax for time system functions (not in Annex A)	620
Syntax 20-3—Syntax for \$timeunit and \$timeprecision (not in Annex A)	622
Syntax 20-4—Syntax for \$printtimescale (not in Annex A)	624
Syntax 20-5—Syntax for \$timeformat (not in Annex A)	625
Syntax 20-6—Type name function syntax (not in Annex A)	628
Syntax 20-7—Size function syntax (not in Annex A)	629
Syntax 20-8—Range function syntax (not in Annex A)	630
Syntax 20-9—Array querying function syntax (not in Annex A)	630
Syntax 20-10—Bit vector system function syntax (not in Annex A)	634
Syntax 20-11—Severity system task syntax (excerpt from Annex A)	635
Syntax 20-12—Assertion control syntax (not in Annex A)	637
Syntax 20-13—Sampled value system function syntax (not in Annex A)	644
Syntax 20-14—Syntax for \$random (not in Annex A)	645
Syntax 20-15—Syntax for probabilistic distribution functions (not in Annex A)	646
Syntax 20-16—Syntax for PLA modeling system task (not in Annex A)	649
Syntax 20-17—\$system function syntax (not in Annex A)	652
Syntax 20-18—\$stacktrace function syntax (not in Annex A)	653

Syntax 21-1—Syntax for \$display and \$write system tasks (not in Annex A)	655
Syntax 21-2—Syntax for \$strobe system tasks (not in Annex A)	663
Syntax 21-3—Syntax for \$monitor system tasks (not in Annex A)	665
Syntax 21-4—Syntax for \$fopen and \$fclose system tasks (not in Annex A)	666
Syntax 21-5—Syntax for file output system tasks (not in Annex A)	667
Syntax 21-6—Syntax for formatting data tasks (not in Annex A)	668
Syntax 21-7—Syntax for file read system functions (not in Annex A)	669
Syntax 21-8—Syntax for file positioning system functions (not in Annex A)	674
Syntax 21-9—Syntax for file flush system task (not in Annex A)	675
Syntax 21-10—Syntax for file I/O error detection system function (not in Annex A)	675
Syntax 21-11—Syntax for end-of-file file detection system function (not in Annex A)	675
Syntax 21-12—Syntax for memory load system tasks (not in Annex A)	676
Syntax 21-13—\$writemem system task syntax (not in Annex A)	679
Syntax 21-14—Syntax for \$dumpfile task (not in Annex A)	684
Syntax 21-15—Syntax for \$dumpvars task (not in Annex A)	684
Syntax 21-16—Syntax for \$dumpoff and \$dumpon tasks (not in Annex A)	685
Syntax 21-17—Syntax for \$dumpall task (not in Annex A)	686
Syntax 21-18—Syntax for \$dumplimit task (not in Annex A)	686
Syntax 21-19—Syntax for \$dumpflush task (not in Annex A)	686
Syntax 21-20—Syntax for output 4-state VCD file (not in Annex A)	688
Syntax 21-21—Syntax for \$dumpports task (not in Annex A)	694
Syntax 21-22—Syntax for \$dumpportsoff and \$dumpportson system tasks (not in Annex A)	694
Syntax 21-23—Syntax for \$dumpportsall system task (not in Annex A)	695
Syntax 21-24—Syntax for \$dumpportslimit system task (not in Annex A)	695
Syntax 21-25—Syntax for \$dumpportsflush system task (not in Annex A)	696
Syntax 21-26—Syntax for \$vcdclose keyword (not in Annex A)	696
Syntax 21-27—Syntax for output extended VCD file (not in Annex A)	698
Syntax 21-28—Syntax for extended VCD node information (not in Annex A)	698
Syntax 21-29—Syntax for value change section (not in Annex A)	700
Syntax 22-1—Syntax for include compiler directive (not in Annex A)	705
Syntax 22-2—Syntax for text macro definition (not in Annex A)	706
Syntax 22-3—Syntax for text macro usage (not in Annex A)	708
Syntax 22-4—Syntax for undef compiler directive (not in Annex A)	712
Syntax 22-5—Syntax for conditional compilation directives (not in Annex A)	713
Syntax 22-6—Syntax for timescale compiler directive (not in Annex A)	716
Syntax 22-7—Syntax for default_nettype compiler directive (not in Annex A)	718
Syntax 22-8—Syntax for pragma compiler directive (not in Annex A)	719
Syntax 22-9—Syntax for line compiler directive (not in Annex A)	720
Syntax 22-10—Syntax for begin_keywords and end_keywords compiler directives (not in Annex A)	721
Syntax 23-1—Module declaration syntax (excerpt from Annex A)	729

Syntax 23-2—Non-ANSI style module header declaration syntax (excerpt from Annex A)	730
Syntax 23-3—Non-ANSI style port declaration syntax (excerpt from Annex A).....	731
Syntax 23-4—ANSI style list_of_port_declarations syntax (excerpt from Annex A).....	734
Syntax 23-5—Module item syntax (excerpt from Annex A)	740
Syntax 23-6—Module instance syntax (excerpt from Annex A)	741
Syntax 23-7—Syntax for hierarchical path names (excerpt from Annex A).....	754
Syntax 23-8—Syntax for upward name referencing (not in Annex A)	759
Syntax 23-9—Bind construct syntax (excerpt from Annex A)	771
Syntax 24-1—Program declaration syntax (excerpt from Annex A)	775
Syntax 25-1—Interface syntax (excerpt from Annex A).....	782
Syntax 25-2—Modport clocking declaration syntax (excerpt from Annex A)	791
Syntax 25-3—Virtual interface declaration syntax (excerpt from Annex A).....	801
Syntax 26-1—Package declaration syntax (excerpt from Annex A).....	808
Syntax 26-2—Package import syntax (excerpt from Annex A).....	809
Syntax 26-3—Package import in header syntax (excerpt from Annex A)	813
Syntax 26-4—Package export syntax (excerpt from Annex A)	815
Syntax 26-5—Std package import syntax (not in Annex A).....	817
Syntax 27-1—Syntax for generate constructs (excerpt from Annex A).....	820
Syntax 28-1—Syntax for gate instantiation (excerpt from Annex A)	830
Syntax 29-1—Syntax for UDPs (excerpt from Annex A).....	861
Syntax 29-2—Syntax for UDP instances (excerpt from Annex A).....	868
Syntax 30-1—Syntax for specify block (excerpt from Annex A)	871
Syntax 30-2—Syntax for module path declaration (excerpt from Annex A).....	872
Syntax 30-3—Syntax for simple module path (excerpt from Annex A)	873
Syntax 30-4—Syntax for edge-sensitive path declaration (excerpt from Annex A).....	874
Syntax 30-5—Syntax for state-dependent paths (excerpt from Annex A)	875
Syntax 30-6—Syntax for path delay value (excerpt from Annex A)	882
Syntax 30-7—Syntax for PATHPULSE\$ pulse control (excerpt from Annex A)	887
Syntax 30-8—Syntax for pulse style declarations (excerpt from Annex A).....	889
Syntax 30-9—Syntax for showcanceled declarations (excerpt from Annex A).....	890
Syntax 31-1—Syntax for system timing checks (excerpt from Annex A)	896
Syntax 31-2—Syntax for time check conditions and timing check events (excerpt from Annex A).....	897
Syntax 31-3—Syntax for \$setup (excerpt from Annex A)	898
Syntax 31-4—Syntax for \$hold (excerpt from Annex A)	899
Syntax 31-5—Syntax for \$setuphold (excerpt from Annex A)	900
Syntax 31-6—Syntax for \$removal (excerpt from Annex A)	902
Syntax 31-7—Syntax for \$recovery (excerpt from Annex A).....	902
Syntax 31-8—Syntax for \$recrem (excerpt from Annex A)	903
Syntax 31-9—Syntax for \$skew (excerpt from Annex A)	905
Syntax 31-10—Syntax for \$timeskew (excerpt from Annex A)	906

Syntax 31-11—Syntax for \$fullskew (excerpt from Annex A).....	909
Syntax 31-12—Syntax for \$width (excerpt from Annex A)	911
Syntax 31-13—Syntax for \$period (excerpt from Annex A)	912
Syntax 31-14—Syntax for \$nochange (excerpt from Annex A)	913
Syntax 31-15—Syntax for edge-control specifier (excerpt from Annex A).....	914
Syntax 31-16—Syntax for controlled timing check events (excerpt from Annex A)	917
Syntax 32-1—Syntax for \$sdf_annotate system task (not in Annex A).....	932
Syntax 33-1—Syntax for cell (excerpt from Annex A).....	935
Syntax 33-2—Syntax for declaring library in library map file (excerpt from Annex A).....	936
Syntax 33-3—Syntax for include command (excerpt from Annex A).....	937
Syntax 33-4—Syntax for configurations (excerpt from Annex A)	938
Syntax 35-1—DPI import declaration syntax (excerpt from Annex A).....	977
Syntax 35-2—DPI export declaration syntax (excerpt from Annex A)	982

Part One:

Design and Verification Constructs

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

1. Overview

1.1 Scope

This standard provides the definition of the language syntax and semantics for the IEEE [Std](#) 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 [Std](#) 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-[2017](#).⁶ ~~2012~~.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:

^{6,1}Information on references can be found in Clause 2

- The formal syntax and semantics of all SystemVerilog constructs
- Simulation system tasks and system functions, such as text output display commands
- Compiler directives, such as text substitution macros and simulation time scaling
- The Programming Language Interface (PLI) mechanism
- The formal syntax and semantics of the SystemVerilog Verification Procedural Interface (VPI)
- An Application Programming Interface (API) for coverage access not included in VPI
- Direct programming interface (DPI) for interoperability with the C programming language
- VPI, API, and DPI header files
- Concurrent assertion formal semantics
- The formal syntax and semantics of standard delay format (SDF) constructs
- Informative usage examples

NOTE—An earlier standard, IEEE Std 1800-2009, represented a merger of two previous standards: IEEE Std 1364™-2005 and IEEE Std 1800-2005. In these previous standards, Verilog® was the base language and defined a completely self-contained standard. SystemVerilog defined a number of significant extensions to Verilog, but IEEE Std 1800-2005 was not a self-contained standard; IEEE Std 1800-2005 referred to, and relied on, IEEE Std 1364-2005. These two standards were designed to be used as one language. Merging the base Verilog language into the SystemVerilog standard enabled users to have all information regarding syntax and semantics in a single document. [7.8.2-3](#)

[7.2](#) Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

[8.3](#) Verilog is a registered trademark of Cadence Design Systems, Inc.

1.4 Special terms

Throughout this standard, the following terms apply:

- *SystemVerilog 3.1a* refers to the Accellera *SystemVerilog 3.1a Language Reference Manual* [B4], a precursor to IEEE Std 1800-2005.^{2,4}
- *Verilog* refers to IEEE Std 1364-2005 for the Verilog hardware description language (HDL).
- *Language Reference Manual (LRM)* refers to the document describing a Verilog or SystemVerilog standard.
- *Tool* refers to a software implementation that reads SystemVerilog source code, such as a logic simulator.

NOTE—In IEEE Std 1800-2005, *SystemVerilog* referred to just the extensions to the IEEE [Std](#) 1364-2005 Verilog language and did not include the Verilog base language.

1.5 Conventions used in this standard

This standard is organized into clauses, each of which focuses on a specific area of the language. There are subclauses within each clause to discuss individual constructs and concepts. The discussion begins with an introduction and an optional rationale for the construct or the concept, followed by syntax and semantic descriptions, followed by examples and notes.

The terminology conventions used throughout this standard are as follows:

^{2,4}The numbers in [square](#) brackets correspond to those of the bibliography in Annex Q.

- The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).^{10,11}
- The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).
- The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).
- The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

1.6 Syntactic description

The main text uses the following conventions:

- *Italicized font* [for syntactic categories \(see Annex A\) or](#) when a term is being defined
- Constant-width font for examples, file names, and references to constants, especially 0, 1, x, and z values
- **Boldface constant-width** font for SystemVerilog keywords, when referring to the actual keyword

The formal syntax of SystemVerilog is described using Backus-Naur Form (BNF). The following conventions are used:

¹⁰ [The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.](#)

¹¹ [The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.](#)

- Lowercase words, some containing embedded underscores, denote syntactic categories. For example:

module_declaration

- **Boldface-red** characters denote reserved keywords, operators, and punctuation marks as a required part of the syntax. For example:

module => ;

- A vertical bar (|) that is not in boldface-red separates alternative items. For example:

unary_operator ::= + | - | ! | ~ | & | ~& | | | ~ | ^ | ~^ | ^~

- Square brackets ([]) that are not in boldface-red enclose optional items. For example:

net_port_header ::= [port_direction] net_port_type

function_declaration ::= function [lifetime] function_body_declaration

- Braces ({ }) that are not in boldface-red enclose a repeated item. The item may appear zero or more times; the repetitions occur from left to right as with an equivalent left-recursive rule. Thus, the following two rules are equivalent:

list_of_param_assignments ::= param_assignment { , param_assignment }

list_of_param_assignments ::=

param_assignment

| list_of_param_assignments, param_assignment

~~A qualified term in the syntax is a term such as `array_identifier` for which the “array” portion represents some semantic intent and the “identifier” term indicates that the qualified term reduces to the “identifier” term in the syntax. The syntax does not completely define the semantics of such qualified terms; for example while an identifier that would qualify semantically as an `array_identifier` is created by a declaration, such declaration forms are not explicitly described using `array_identifier` in the syntax.~~

1.7 Use of color in this standard

This standard uses a minimal amount of color to enhance readability. The coloring is not essential and does not affect the accuracy of this standard when viewed in pure black and white. The places where color is used are the following:

- Cross references that are hyperlinked to other portions of this standard are shown in underlined-blue text (hyperlinking works when this standard is viewed interactively as a PDF file).
- Syntactic keywords and tokens in the formal language definitions are shown in **boldface-red text**.
- Some figures use a minimal amount of color to enhance readability.

1.8 Contents of this standard

A synopsis of the clauses and annexes is presented as a quick reference. All clauses and several of the annexes are normative parts of this standard. Some annexes are included for informative purposes only.

Part One: Design and Verification Constructs

Clause 1 describes the contents of this standard and the conventions used in this standard.

Clause 2 lists references to other standards that are required in order to implement this standard.

Clause 3 introduces the major building blocks that make up a SystemVerilog design and verification environment: modules, programs, interfaces, checkers, packages, and configurations. This clause also discusses primitives, name spaces, the **\$unit** compilation space, and the concept of simulation time.

Clause 4 describes the SystemVerilog simulation scheduling semantics.

Clause 5 describes the lexical tokens used in SystemVerilog source text and their conventions.

Clause 6 describes SystemVerilog data objects and types, including nets and variables, their declaration syntax and usage rules, and charge strength of the values on nets. This clause also discusses strings and string methods, enumerated types, user-defined types, constants, data scope and lifetime, and type compatibility._

Clause 7 describes SystemVerilog compound data types: structures, unions, arrays, including packed and unpacked arrays, dynamic arrays, associative arrays, and queues. This clause also describes various array methods.

Clause 8 describes the object-oriented programming capabilities in SystemVerilog. Topics include defining classes, interface classes, dynamically constructing objects, inheritance and subclasses, data hiding and encapsulation, polymorphism, and parameterized classes.

Clause 9 describes the SystemVerilog procedural blocks: **initial**, **always**, **always_comb**, **always_ff**, **always_latch**, and **final**. Sequential and parallel statement grouping, block names, statement labels, and process control are also described.

Clause 10 describes continuous assignments, blocking and nonblocking procedural assignments, and procedural continuous assignments.

Clause 11 describes the operators and operands that can be used in expressions.

Clause 12 describes SystemVerilog procedural programming statements, such as decision statements and looping constructs.