



IEEE Standard for Verilog[®] Hardware Description Language

IEEE Computer Society

Sponsored by the
Design Automation Standards Committee

1364TM

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

7 April 2006

IEEE Std 1364TM-2005
(Revision of IEEE Std 1364-2001)

IEEE Standard for Verilog® Hardware Description Language

Sponsor

Design Automation Standards Committee
of the
IEEE Computer Society

Abstract: The Verilog hardware description language (HDL) is defined in this standard. Verilog HDL is a formal notation intended for use in all phases of the creation of electronic systems. Because it is both machine-readable and human-readable, it supports the development, verification, synthesis, and testing of hardware designs; the communication of hardware design data; and the maintenance, modification, and procurement of hardware. The primary audiences for this standard are the implementors of tools supporting the language and advanced users of the language.

Keywords: computer, computer languages, digital systems, electronic systems, hardware, hardware description languages, hardware design, HDL, PLI, programming language interface, Verilog, Verilog HDL, Verilog PLI

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

Copyright © 2006 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 7 April 2006. Printed in the United States of America.

IEEE is a registered trademark 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.

Print: ISBN 0-7381-4850-4 SH95395
PDF: ISBN 0-7381-4851-2 SS95395

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.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The 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 the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The 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 this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, 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 herein 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 five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, 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 this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received 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 interpretation requests except in those cases where the matter has previously received formal consideration. 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, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

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

NOTE—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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

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.

Introduction

This introduction is not a part of IEEE Std 1364-2005, IEEE Standard for Verilog[®] Hardware Description Language.

The Verilog hardware description language (HDL) became an IEEE standard in 1995 as IEEE Std 1364-1995. It was designed to be simple, intuitive, and effective at multiple levels of abstraction in a standard textual format for a variety of design tools, including verification simulation, timing analysis, test analysis, and synthesis. It is because of these rich features that Verilog has been accepted to be the language of choice by an overwhelming number of integrated circuit (IC) designers.

Verilog contains a rich set of built-in primitives, including logic gates, user-definable primitives, switches, and wired logic. It also has device pin-to-pin delays and timing checks. The mixing of abstract levels is essentially provided by the semantics of two data types: nets and variables. Continuous assignments, in which expressions of both variables and nets can continuously drive values onto nets, provide the basic structural construct. Procedural assignments, in which the results of calculations involving variable and net values can be stored into variables, provide the basic behavioral construct. A design consists of a set of modules, each of which has an input/output (I/O) interface, and a description of its function, which can be structural, behavioral, or a mix. These modules are formed into a hierarchy and are interconnected with nets.

The Verilog language is extensible via the programming language interface (PLI) and the Verilog procedural interface (VPI) routines. The PLI/VPI is a collection of routines that allows foreign functions to access information contained in a Verilog HDL description of the design and facilitates dynamic interaction with simulation. Applications of PLI/VPI include connecting to a Verilog HDL simulator with other simulation and computer-assisted design (CAD) systems, customized debugging tasks, delay calculators, and annotators.

The language that influenced Verilog HDL the most was HILO-2, which was developed at Brunel University in England under a contract to produce a test generation system for the British Ministry of Defense. HILO-2 successfully combined the gate and register transfer levels of abstraction and supported verification simulation, timing analysis, fault simulation, and test generation.

In 1990, Cadence Design Systems placed the Verilog HDL into the public domain and the independent Open Verilog International (OVI) was formed to manage and promote Verilog HDL. In 1992, the Board of Directors of OVI began an effort to establish Verilog HDL as an IEEE standard. In 1993, the first IEEE working group was formed; and after 18 months of focused efforts, Verilog became an IEEE standard as IEEE Std 1364-1995.

After the standardization process was complete, the IEEE P1364 Working Group started looking for feedback from IEEE 1364 users worldwide so the standard could be enhanced and modified accordingly. This led to a five-year effort to get a much better Verilog standard in IEEE Std 1364-2001.

With the completion of IEEE Std 1364-2001, work continued in the larger Verilog community to identify outstanding issues with the language as well as ideas for possible enhancements. As Accellera began working on standardizing SystemVerilog in 2001, additional issues were identified that could possibly have led to incompatibilities between Verilog 1364 and SystemVerilog. The IEEE P1364 Working Group was established as a subcommittee of the SystemVerilog P1800 Working Group to help ensure consistent resolution of such issues. The result of this collaborative work is this standard, IEEE Std 1364-2005.

Notice to users

Errata

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

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Participants

At the time this standard was completed, the IEEE P1364 Working Group had the following membership:

Johny Srouji, IBM, *IEEE SystemVerilog Working Group Chair*
Tom Fitzpatrick, Mentor Graphics Corporation, *Chair*
Neil Korpusik, Sun Microsystems, Inc., *Co-chair*
Stuart Sutherland, Sutherland HDL, Inc., *Editor*
Shalom Bresticker, Intel Corporation, *Editor through February 2005*

The Errata Task Force had the following membership:

Karen Pieper, Synopsys, Inc., *Chair*

Kurt Baty, WFSDB Consulting
Stefen Boyd, Boyd Technology
Shalom Bresticker, Intel Corporation
Dennis Brophy, Mentor Graphics Corporation
Cliff Cummings, Sunburst Design, Inc.
Charles Dawson, Cadence Design Systems, Inc.
Tom Fitzpatrick, Mentor Graphics Corporation
Ronald Goodstein, First Shot Logic Simulation and Design
Mark Hartoog, Synopsys, Inc.
James Markevitch, Evergreen Technology Group

Dennis Marsa, Xilinx
Francoise Martinolle, Cadence Design Systems, Inc.
Mike McNamara, Verisity, Ltd.
Don Mills, LCDM Engineering
Anders Nordstrom, Cadence Design Systems, Inc.
Karen Pieper, Synopsys, Inc.
Brad Pierce, Synopsys, Inc.
Steven Sharp, Cadence Design Systems, Inc.
Alec Stanculescu, Fintronic USA, Inc.
Stuart Sutherland, Sutherland HDL, Inc.
Gordon Vreugdenhil, Mentor Graphics Corporation
Jason Woolf, Cadence Design Systems, Inc.

The Behavioral Task Force had the following membership:

Steven Sharp, Cadence Design Systems, Inc., *Chair*

Kurt Baty, WFSDB Consulting	Jay Lawrence, Cadence Design Systems, Inc.
Stefen Boyd, Boyd Technology	Francoise Martinolle, Cadence Design Systems, Inc.
Shalom Bresticker, Intel Corporation	Kathryn McKinley, Cadence Design Systems, Inc.
Dennis Brophy, Mentor Graphics Corporation	Michael McNamara, Verisity, Ltd.
Cliff Cummings, Sunburst Design, Inc.	Don Mills, LCDM Engineering
Steven Dovich, Cadence Design Systems, Inc.	Mehdi Mohtashemi, Synopsys, Inc.
Tom Fitzpatrick, Mentor Graphics Corporation	Karen Pieper, Synopsys, Inc.
Ronald Goodstein, First Shot Logic Simulation and Design	Brad Pierce, Synopsys, Inc.
Keith Gover, Mentor Graphics Corporation	Dave Rich, Mentor Graphics Corporation
Mark Hartoog, Synopsys, Inc.	Steven Sharp, Cadence Design Systems, Inc.
Ennis Hawk, Jeda Technologies	Alec Stanculescu, Fintronic, USA
Atsushi Kasuya, Jeda Technologies	Stuart Sutherland, Sutherland HDL, Inc.
	Gordon Vreugdenhil, Mentor Graphics Corporation

The PLI Task Force had the following membership:

Charles Dawson, Cadence Design Systems, Inc., *Chair*

Ghassan Khoory, Synopsys, Inc., *Co-chair*

Tapati Basu, Synopsys, Inc.	Michael Rohleder, Freescale Semiconductor, Inc.
Steven Dovich, Cadence Design Systems, Inc.	Rob Slater, Freescale Semiconductor, Inc.
Ralph Duncan, Mentor Graphics Corporation	John Stickley, Mentor Graphics Corporation
Jim Garnett, Mentor Graphics Corporation	Stuart Sutherland, Sutherland HDL, Inc.
Joao Geada, CLK Design Automation	Bassam Tabbara, Novas Software, Inc.
Andrzej Litwiniuk, Synopsys, Inc.	Jim Vellenga, Cadence Design Systems, Inc.
Francoise Martinolle, Cadence Design Systems, Inc.	Doug Warmke, Mentor Graphics Corporation
Sachchidananda Patel, Synopsys, Inc.	

In addition, the working group wishes to recognize the substantial efforts of past contributors:

Michael McNamara, Cadence Design Systems, Inc.,

1364 Working Group past chair (through September 2004)

Alec Stanculescu, Fintronic USA, *1364 Working Group past vice-chair (through June 2004)*

Stefen Boyd, Boyd Technology, *ETF past co-chair (through November 2004)*

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

Accellera	Intel Corporation
Bluespec, Inc.	Mentor Graphics Corporation
Cadence Design Systems, Inc.	Sun Microsystems, Inc.
Fintronic U.S.A.	Sunburst Design, Inc.
IBM	Sutherland HDL, Inc.
Infineon Technologies	Synopsys, Inc.

When the IEEE-SA Standards Board approved this standard on 8 November 2005, it had the following membership:

Steve M. Mills, *Chair*
Richard H. Hulett, *Vice Chair*
Don Wright, *Past Chair*
Judith Gorman, *Secretary*

Mark D. Bowman
Dennis B. Brophy
Joseph Bruder
Richard Cox
Bob Davis
Julian Forster*
Joanna N. Guenin
Mark S. Halpin
Raymond Hapeman

William B. Hopf
Lowell G. Johnson
Herman Koch
Joseph L. Koepfinger*
David J. Law
Daleep C. Mohla
Paul Nikolich

T. W. Olsen
Glenn Parsons
Ronald C. Petersen
Gary S. Robinson
Frank Stone
Malcolm V. Thaden
Richard L. Townsend
Joe D. Watson
Howard L. Wolfman

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan H. Cookson, *NIST Representative*

Michelle D. Turner
IEEE Standards Project Editor

Contents

1.	Overview.....	1
1.1	Scope.....	1
1.2	Conventions used in this standard	1
1.3	Syntactic description.....	2
1.4	Use of color in this standard	3
1.5	Contents of this standard.....	3
1.6	Deprecated clauses.....	5
1.7	Header file listings	5
1.8	Examples.....	5
1.9	Prerequisites.....	5
2.	Normative references	6
3.	Lexical conventions	8
3.1	Lexical tokens	8
3.2	White space.....	8
3.3	Comments	8
3.4	Operators.....	8
3.5	Numbers.....	9
3.5.1	Integer constants	10
3.5.2	Real constants	12
3.5.3	Conversion	12
3.6	Strings.....	12
3.6.1	String variable declaration	13
3.6.2	String manipulation.....	13
3.6.3	Special characters in strings.....	13
3.7	Identifiers, keywords, and system names	14
3.7.1	Escaped identifiers	14
3.7.2	Keywords	15
3.7.3	System tasks and functions	15
3.7.4	Compiler directives.....	15
3.8	Attributes	16
3.8.1	Examples.....	16
3.8.2	Syntax	18
4.	Data types	21
4.1	Value set	21
4.2	Nets and variables.....	21
4.2.1	Net declarations	21
4.2.2	Variable declarations	23
4.3	Vectors.....	24
4.3.1	Specifying vectors.....	24
4.3.2	Vector net accessibility.....	24
4.4	Strengths	25
4.4.1	Charge strength.....	25
4.4.2	Drive strength	25
4.5	Implicit declarations	25
4.6	Net types	26
4.6.1	Wire and tri nets.....	26
4.6.2	Wired nets	27
4.6.3	Trireg net.....	28

4.6.4	Tri0 and tri1 nets	31
4.6.5	Unresolved nets	31
4.6.6	Supply nets	32
4.7	Regs	32
4.8	Integers, reals, times, and realtimes	32
4.8.1	Operators and real numbers	33
4.8.2	Conversion	33
4.9	Arrays	34
4.9.1	Net arrays	34
4.9.2	reg and variable arrays	34
4.9.3	Memories	35
4.10	Parameters	35
4.10.1	Module parameters	36
4.10.2	Local parameters (localparam)	37
4.10.3	Specify parameters	38
4.11	Name spaces	39
5.	Expressions	41
5.1	Operators	41
5.1.1	Operators with real operands	42
5.1.2	Operator precedence	43
5.1.3	Using integer numbers in expressions	44
5.1.4	Expression evaluation order	45
5.1.5	Arithmetic operators	45
5.1.6	Arithmetic expressions with regs and integers	47
5.1.7	Relational operators	48
5.1.8	Equality operators	49
5.1.9	Logical operators	49
5.1.10	Bitwise operators	50
5.1.11	Reduction operators	51
5.1.12	Shift operators	53
5.1.13	Conditional operator	53
5.1.14	Concatenations	54
5.2	Operands	55
5.2.1	Vector bit-select and part-select addressing	56
5.2.2	Array and memory addressing	57
5.2.3	Strings	58
5.3	Minimum, typical, and maximum delay expressions	61
5.4	Expression bit lengths	62
5.4.1	Rules for expression bit lengths	62
5.4.2	Example of expression bit-length problem	63
5.4.3	Example of self-determined expressions	64
5.5	Signed expressions	64
5.5.1	Rules for expression types	65
5.5.2	Steps for evaluating an expression	65
5.5.3	Steps for evaluating an assignment	66
5.5.4	Handling X and Z in signed expressions	66
5.6	Assignments and truncation	66
6.	Assignments	68
6.1	Continuous assignments	68
6.1.1	The net declaration assignment	69
6.1.2	The continuous assignment statement	69
6.1.3	Delays	71

6.1.4	Strength.....	71
6.2	Procedural assignments.....	72
6.2.1	Variable declaration assignment.....	72
6.2.2	Variable declaration syntax.....	73
7.	Gate- and switch-level modeling.....	74
7.1	Gate and switch declaration syntax.....	74
7.1.1	The gate type specification.....	76
7.1.2	The drive strength specification.....	76
7.1.3	The delay specification.....	77
7.1.4	The primitive instance identifier.....	77
7.1.5	The range specification.....	77
7.1.6	Primitive instance connection list.....	78
7.2	and, nand, nor, or, xor, and xnor gates.....	80
7.3	buf and not gates.....	81
7.4	bufif1, bufif0, notif1, and notif0 gates.....	82
7.5	MOS switches.....	83
7.6	Bidirectional pass switches.....	84
7.7	CMOS switches.....	85
7.8	pullup and pulldown sources.....	86
7.9	Logic strength modeling.....	86
7.10	Strengths and values of combined signals.....	88
7.10.1	Combined signals of unambiguous strength.....	88
7.10.2	Ambiguous strengths: sources and combinations.....	89
7.10.3	Ambiguous strength signals and unambiguous signals.....	94
7.10.4	Wired logic net types.....	98
7.11	Strength reduction by nonresistive devices.....	100
7.12	Strength reduction by resistive devices.....	100
7.13	Strengths of net types.....	100
7.13.1	tri0 and tri1 net strengths.....	100
7.13.2	trireg strength.....	100
7.13.3	supply0 and supply1 net strengths.....	101
7.14	Gate and net delays.....	101
7.14.1	min:typ:max delays.....	102
7.14.2	trireg net charge decay.....	103
8.	User-defined primitives (UDPs).....	105
8.1	UDP definition.....	105
8.1.1	UDP header.....	107
8.1.2	UDP port declarations.....	107
8.1.3	Sequential UDP initial statement.....	107
8.1.4	UDP state table.....	107
8.1.5	Z values in UDP.....	108
8.1.6	Summary of symbols.....	108
8.2	Combinational UDPs.....	109
8.3	Level-sensitive sequential UDPs.....	110
8.4	Edge-sensitive sequential UDPs.....	110
8.5	Sequential UDP initialization.....	111
8.6	UDP instances.....	113
8.7	Mixing level-sensitive and edge-sensitive descriptions.....	114
8.8	Level-sensitive dominance.....	115
9.	Behavioral modeling.....	116
9.1	Behavioral model overview.....	116

9.2	Procedural assignments.....	117
9.2.1	Blocking procedural assignments	117
9.2.2	The nonblocking procedural assignment	118
9.3	Procedural continuous assignments	122
9.3.1	The assign and deassign procedural statements.....	123
9.3.2	The force and release procedural statements	124
9.4	Conditional statement	125
9.4.1	If-else-if construct.....	126
9.5	Case statement	127
9.5.1	Case statement with do-not-cares	128
9.5.2	Constant expression in case statement.....	129
9.6	Looping statements.....	130
9.7	Procedural timing controls.....	131
9.7.1	Delay control.....	132
9.7.2	Event control.....	132
9.7.3	Named events.....	133
9.7.4	Event or operator	134
9.7.5	Implicit event_expression list	134
9.7.6	Level-sensitive event control	136
9.7.7	Intra-assignment timing controls	136
9.8	Block statements.....	139
9.8.1	Sequential blocks	140
9.8.2	Parallel blocks.....	141
9.8.3	Block names.....	141
9.8.4	Start and finish times	142
9.9	Structured procedures	143
9.9.1	Initial construct	143
9.9.2	Always construct.....	144
10.	Tasks and functions	145
10.1	Distinctions between tasks and functions	145
10.2	Tasks and task enabling	145
10.2.1	Task declarations	146
10.2.2	Task enabling and argument passing.....	147
10.2.3	Task memory usage and concurrent activation.....	149
10.3	Disabling of named blocks and tasks.....	150
10.4	Functions and function calling.....	152
10.4.1	Function declarations.....	152
10.4.2	Returning a value from a function	154
10.4.3	Calling a function.....	155
10.4.4	Function rules	155
10.4.5	Use of constant functions.....	156
11.	Scheduling semantics.....	158
11.1	Execution of a model	158
11.2	Event simulation	158
11.3	The stratified event queue.....	158
11.4	Verilog simulation reference model	159
11.4.1	Determinism.....	160
11.4.2	Nondeterminism.....	160
11.5	Race conditions.....	160
11.6	Scheduling implication of assignments	161
11.6.1	Continuous assignment.....	161
11.6.2	Procedural continuous assignment.....	161

11.6.3	Blocking assignment.....	161
11.6.4	Nonblocking assignment.....	161
11.6.5	Switch (transistor) processing.....	161
11.6.6	Port connections.....	162
11.6.7	Functions and tasks.....	162
12.	Hierarchical structures	163
12.1	Modules	163
12.1.1	Top-level modules	165
12.1.2	Module instantiation	165
12.2	Overriding module parameter values.....	167
12.2.1	defparam statement.....	168
12.2.2	Module instance parameter value assignment	170
12.2.3	Parameter dependence	173
12.3	Ports	173
12.3.1	Port definition	173
12.3.2	List of ports.....	174
12.3.3	Port declarations	174
12.3.4	List of ports declarations.....	176
12.3.5	Connecting module instance ports by ordered list.....	176
12.3.6	Connecting module instance ports by name	177
12.3.7	Real numbers in port connections.....	178
12.3.8	Connecting dissimilar ports	178
12.3.9	Port connection rules	179
12.3.10	Net types resulting from dissimilar port connections	179
12.3.11	Connecting signed values via ports	181
12.4	Generate constructs.....	181
12.4.1	Loop generate constructs	183
12.4.2	Conditional generate constructs.....	186
12.4.3	External names for unnamed generate blocks	190
12.5	Hierarchical names	191
12.6	Upwards name referencing	193
12.7	Scope rules	195
12.8	Elaboration.....	197
12.8.1	Order of elaboration.....	197
12.8.2	Early resolution of hierarchical names	197
13.	Configuring the contents of a design	199
13.1	Introduction.....	199
13.1.1	Library notation	199
13.1.2	Basic configuration elements.....	200
13.2	Libraries.....	200
13.2.1	Specifying libraries—the library map file	200
13.2.2	Using multiple library map files	202
13.2.3	Mapping source files to libraries	202
13.3	Configurations	202
13.3.1	Basic configuration syntax.....	202
13.3.2	Hierarchical configurations.....	205
13.4	Using libraries and configs	205
13.4.1	Precompiling in a single-pass use model.....	205
13.4.2	Elaboration-time compiling in a single-pass use model	206
13.4.3	Precompiling using a separate compilation tool	206
13.4.4	Command line considerations.....	206
13.5	Configuration examples.....	206

13.5.1	Default configuration from library map file	207
13.5.2	Using default clause	207
13.5.3	Using cell clause	207
13.5.4	Using instance clause	208
13.5.5	Using hierarchical config	208
13.6	Displaying library binding information	208
13.7	Library mapping examples	209
13.7.1	Using the command line to control library searching	209
13.7.2	File path specification examples	209
13.7.3	Resolving multiple path specifications	209
14.	Specify blocks	211
14.1	Specify block declaration	211
14.2	Module path declarations	212
14.2.1	Module path restrictions	212
14.2.2	Simple module paths	213
14.2.3	Edge-sensitive paths	214
14.2.4	State-dependent paths	215
14.2.5	Full connection and parallel connection paths	219
14.2.6	Declaring multiple module paths in a single statement	220
14.2.7	Module path polarity	220
14.3	Assigning delays to module paths	222
14.3.1	Specifying transition delays on module paths	222
14.3.2	Specifying x transition delays	224
14.3.3	Delay selection	225
14.4	Mixing module path delays and distributed delays	225
14.5	Driving wired logic	226
14.6	Detailed control of pulse filtering behavior	228
14.6.1	Specify block control of pulse limit values	229
14.6.2	Global control of pulse limit values	230
14.6.3	SDF annotation of pulse limit values	230
14.6.4	Detailed pulse control capabilities	230
15.	Timing checks	237
15.1	Overview	237
15.2	Timing checks using a stability window	240
15.2.1	\$setup	241
15.2.2	\$hold	242
15.2.3	\$setuphold	243
15.2.4	\$removal	245
15.2.5	\$recovery	246
15.2.6	\$crem	247
15.3	Timing checks for clock and control signals	248
15.3.1	\$skew	249
15.3.2	\$timeskew	250
15.3.3	\$fullskew	252
15.3.4	\$width	255
15.3.5	\$period	256
15.3.6	\$nochange	257
15.4	Edge-control specifiers	258
15.5	Notifiers: user-defined responses to timing violations	259
15.5.1	Requirements for accurate simulation	261
15.5.2	Conditions in negative timing checks	263
15.5.3	Notifiers in negative timing checks	264

15.5.4	Option behavior	264
15.6	Enabling timing checks with conditioned events	265
15.7	Vector signals in timing checks	266
15.8	Negative timing checks	266
16.	Backannotation using the standard delay format (SDF)	269
16.1	The SDF annotator	269
16.2	Mapping of SDF constructs to Verilog	269
16.2.1	Mapping of SDF delay constructs to Verilog declarations	269
16.2.2	Mapping of SDF timing check constructs to Verilog	271
16.2.3	SDF annotation of specparams	272
16.2.4	SDF annotation of interconnect delays	273
16.3	Multiple annotations	274
16.4	Multiple SDF files	275
16.5	Pulse limit annotation	275
16.6	SDF to Verilog delay value mapping	276
17.	System tasks and functions	277
17.1	Display system tasks	278
17.1.1	The display and write tasks	278
17.1.2	Strobed monitoring	285
17.1.3	Continuous monitoring	286
17.2	File input-output system tasks and functions	286
17.2.1	Opening and closing files	287
17.2.2	File output system tasks	288
17.2.3	Formatting data to a string	289
17.2.4	Reading data from a file	290
17.2.5	File positioning	294
17.2.6	Flushing output	295
17.2.7	I/O error status	295
17.2.8	Detecting EOF	295
17.2.9	Loading memory data from a file	296
17.2.10	Loading timing data from an SDF file	297
17.3	Timescale system tasks	298
17.3.1	\$prnttimescale	299
17.3.2	\$timeformat	300
17.4	Simulation control system tasks	302
17.4.1	\$finish	302
17.4.2	\$stop	302
17.5	Programmable logic array (PLA) modeling system tasks	303
17.5.1	Array types	303
17.5.2	Array logic types	304
17.5.3	Logic array personality declaration and loading	304
17.5.4	Logic array personality formats	304
17.6	Stochastic analysis tasks	307
17.6.1	\$q_initialize	307
17.6.2	\$q_add	307
17.6.3	\$q_remove	307
17.6.4	\$q_full	308
17.6.5	\$q_exam	308
17.6.6	Status codes	308
17.7	Simulation time system functions	309
17.7.1	\$time	309
17.7.2	\$stime	309

17.7.3	\$realtime	310
17.8	Conversion functions	310
17.9	Probabilistic distribution functions	311
17.9.1	\$random function	311
17.9.2	\$dist_ functions	312
17.9.3	Algorithm for probabilistic distribution functions	313
17.10	Command line input	320
17.10.1	\$test\$plusargs (string)	320
17.10.2	\$value\$plusargs (user_string, variable)	321
17.11	Math functions	323
17.11.1	Integer math functions	323
17.11.2	Real math functions	323
18.	Value change dump (VCD) files	325
18.1	Creating four-state VCD file	325
18.1.1	Specifying name of dump file (\$dumpfile)	325
18.1.2	Specifying variables to be dumped (\$dumpvars)	326
18.1.3	Stopping and resuming the dump (\$dumpoff/\$dumpon)	327
18.1.4	Generating a checkpoint (\$dumpall)	328
18.1.5	Limiting size of dump file (\$dumplimit)	328
18.1.6	Reading dump file during simulation (\$dumpflush)	328
18.2	Format of four-state VCD file	329
18.2.1	Syntax of four-state VCD file	330
18.2.2	Formats of variable values	331
18.2.3	Description of keyword commands	332
18.2.4	Four-state VCD file format example	337
18.3	Creating extended VCD file	338
18.3.1	Specifying dump file name and ports to be dumped (\$dumpports)	338
18.3.2	Stopping and resuming the dump (\$dumpportsoff/\$dumpportson)	339
18.3.3	Generating a checkpoint (\$dumpportsall)	340
18.3.4	Limiting size of dump file (\$dumpportslimit)	340
18.3.5	Reading dump file during simulation (\$dumpportsflush)	341
18.3.6	Description of keyword commands	341
18.3.7	General rules for extended VCD system tasks	341
18.4	Format of extended VCD file	342
18.4.1	Syntax of extended VCD file	342
18.4.2	Extended VCD node information	344
18.4.3	Value changes	346
18.4.4	Extended VCD file format example	347
19.	Compiler directives	349
19.1	`celldefine and `endcelldefine	349
19.2	`default_nettype	349
19.3	`define and `undef	350
19.3.1	`define	350
19.3.2	`undef	352
19.4	`ifdef, `else, `elsif, `endif, `ifndef	352
19.5	`include	356
19.6	`resetall	356
19.7	`line	357
19.8	`timescale	358
19.9	`unconnected_drive and `nounconnected_drive	360
19.10	`pragma	360
19.10.1	Standard pragmas	361

19.11	`begin_keywords, `end_keywords	361
20.	Programming language interface (PLI) overview	366
20.1	PLI purpose and history	366
20.2	User-defined system task/function names	367
20.3	User-defined system task/function types	367
20.4	Overriding built-in system task/function names	367
20.5	User-supplied PLI applications	367
20.6	PLI mechanism	368
20.7	User-defined system task/function arguments	368
20.8	PLI include files	368
21.	PLI TF and ACC interface mechanism (deprecated)	369
22.	Using ACC routines (deprecated)	370
23.	ACC routine definitions (deprecated)	371
24.	Using TF routines (deprecated)	372
25.	TF routine definitions (deprecated)	373
26.	Using Verilog procedural interface (VPI) routines	374
26.1	VPI system tasks and functions	374
26.1.1	sizetf VPI application routine	374
26.1.2	compiletf VPI application routine	374
26.1.3	calltf VPI application routine	375
26.1.4	Arguments to sizetf, compiletf, and calltf application routines	375
26.2	VPI mechanism	375
26.2.1	VPI callbacks	375
26.2.2	VPI access to Verilog HDL objects and simulation objects	376
26.2.3	Error handling	376
26.2.4	Function availability	376
26.2.5	Traversing expressions	377
26.3	VPI object classifications	377
26.3.1	Accessing object relationships and properties	378
26.3.2	Object type properties	379
26.3.3	Object file and line properties	380
26.3.4	Delays and values	380
26.3.5	Object protection properties	381
26.4	List of VPI routines by functional category	381
26.5	Key to data model diagrams	383
26.5.1	Diagram key for objects and classes	384
26.5.2	Diagram key for accessing properties	384
26.5.3	Diagram key for traversing relationships	385
26.6	Object data model diagrams	386
26.6.1	Module	387
26.6.2	Instance arrays	388
26.6.3	Scope	389
26.6.4	IO declaration	389
26.6.5	Ports	390
26.6.6	Nets and net arrays	391
26.6.7	Regs and reg arrays	393
26.6.8	Variables	395

26.6.9	Memory.....	396
26.6.10	Object range.....	396
26.6.11	Named event.....	397
26.6.12	Parameter, specparam.....	398
26.6.13	Primitive, prim term.....	399
26.6.14	UDP.....	400
26.6.15	Module path, path term.....	401
26.6.16	Intermodule path.....	401
26.6.17	Timing check.....	402
26.6.18	Task, function declaration.....	402
26.6.19	Task/function call.....	403
26.6.20	Frames.....	404
26.6.21	Delay terminals.....	405
26.6.22	Net drivers and loads.....	405
26.6.23	Reg drivers and loads.....	406
26.6.24	Continuous assignment.....	406
26.6.25	Simple expressions.....	407
26.6.26	Expressions.....	408
26.6.27	Process, block, statement, event statement.....	409
26.6.28	Assignment.....	410
26.6.29	Delay control.....	410
26.6.30	Event control.....	410
26.6.31	Repeat control.....	411
26.6.32	While, repeat, wait.....	411
26.6.33	For.....	411
26.6.34	Forever.....	411
26.6.35	If, if-else.....	412
26.6.36	Case.....	412
26.6.37	Assign statement, deassign, force, release.....	413
26.6.38	Disable.....	413
26.6.39	Callback.....	414
26.6.40	Time queue.....	414
26.6.41	Active time format.....	414
26.6.42	Attributes.....	415
26.6.43	Iterator.....	416
26.6.44	Generates.....	417
27.	VPI routine definitions.....	418
27.1	vpi_chk_error().....	418
27.2	vpi_compare_objects().....	420
27.3	vpi_control().....	420
27.4	vpi_flush().....	421
27.5	vpi_free_object().....	421
27.6	vpi_get().....	422
27.7	vpi_get_cb_info().....	422
27.8	vpi_get_data().....	423
27.9	vpi_get_delays().....	424
27.10	vpi_get_str().....	426
27.11	vpi_get_systf_info().....	427
27.12	vpi_get_time().....	428
27.13	vpi_get_userdata().....	429
27.14	vpi_get_value().....	429
27.15	vpi_get_vlog_info().....	435
27.16	vpi_handle().....	436

27.17	vpi_handle_by_index()	437
27.18	vpi_handle_by_multi_index()	438
27.19	vpi_handle_by_name()	438
27.20	vpi_handle_multi()	439
27.21	vpi_iterate()	439
27.22	vpi_mcd_close()	440
27.23	vpi_mcd_flush()	441
27.24	vpi_mcd_name()	441
27.25	vpi_mcd_open()	442
27.26	vpi_mcd_printf()	443
27.27	vpi_mcd_vprintf()	444
27.28	vpi_printf()	444
27.29	vpi_put_data()	445
27.30	vpi_put_delays()	447
27.31	vpi_put_userdata()	450
27.32	vpi_put_value()	450
27.33	vpi_register_cb()	453
27.33.1	Simulation event callbacks	454
27.33.2	Simulation time callbacks	458
27.33.3	Simulator action or feature callbacks	460
27.34	vpi_register_systf()	461
27.34.1	System task/function callbacks	462
27.34.2	Initializing VPI system task/function callbacks	463
27.34.3	Registering multiple system tasks and functions	464
27.35	vpi_remove_cb()	465
27.36	vpi_scan()	465
27.37	vpi_vprintf()	466
28.	Protected envelopes	467
28.1	General	467
28.2	Processing protected envelopes	467
28.2.1	Encryption	468
28.2.2	Decryption	469
28.3	Protect pragma directives	469
28.4	Protect pragma keywords	471
28.4.1	begin	471
28.4.2	end	471
28.4.3	begin_protected	471
28.4.4	end_protected	472
28.4.5	author	472
28.4.6	author_info	473
28.4.7	encrypt_agent	473
28.4.8	encrypt_agent_info	473
28.4.9	encoding	474
28.4.10	data_keyowner	475
28.4.11	data_method	475
28.4.12	data_keyname	476
28.4.13	data_public_key	477
28.4.14	data_decrypt_key	477
28.4.15	data_block	478
28.4.16	digest_keyowner	478
28.4.17	digest_key_method	478
28.4.18	digest_keyname	479
28.4.19	digest_public_key	479

28.4.20	digest_decrypt_key	480
28.4.21	digest_method	480
28.4.22	digest_block	481
28.4.23	key_keyowner	482
28.4.24	key_method	482
28.4.25	key_keyname	482
28.4.26	key_public_key	483
28.4.27	key_block	483
28.4.28	decrypt_license	484
28.4.29	runtime_license	484
28.4.30	comment	485
28.4.31	reset	485
28.4.32	viewport	486
Annex A	(normative) Formal syntax definition	487
A.1	Source text	487
A.1.1	Library source text	487
A.1.2	Verilog source text	487
A.1.3	Module parameters and ports	487
A.1.4	Module items	488
A.1.5	Configuration source text	489
A.2	Declarations	489
A.2.1	Declaration types	489
A.2.2	Declaration data types	490
A.2.3	Declaration lists	491
A.2.4	Declaration assignments	491
A.2.5	Declaration ranges	492
A.2.6	Function declarations	492
A.2.7	Task declarations	492
A.2.8	Block item declarations	493
A.3	Primitive instances	493
A.3.1	Primitive instantiation and instances	493
A.3.2	Primitive strengths	494
A.3.3	Primitive terminals	494
A.3.4	Primitive gate and switch types	494
A.4	Module instantiation and generate construct	495
A.4.1	Module instantiation	495
A.4.2	Generate construct	495
A.5	UDP declaration and instantiation	496
A.5.1	UDP declaration	496
A.5.2	UDP ports	496
A.5.3	UDP body	496
A.5.4	UDP instantiation	497
A.6	Behavioral statements	497
A.6.1	Continuous assignment statements	497
A.6.2	Procedural blocks and assignments	497
A.6.3	Parallel and sequential blocks	497
A.6.4	Statements	498
A.6.5	Timing control statements	498
A.6.6	Conditional statements	499
A.6.7	Case statements	499
A.6.8	Looping statements	499
A.6.9	Task enable statements	499
A.7	Specify section	500

A.7.1	Specify block declaration	500
A.7.2	Specify path declarations	500
A.7.3	Specify block terminals	500
A.7.4	Specify path delays.....	500
A.7.5	System timing checks.....	502
A.8	Expressions	504
A.8.1	Concatenations	504
A.8.2	Function calls	504
A.8.3	Expressions.....	504
A.8.4	Primaries.....	505
A.8.5	Expression left-side values.....	506
A.8.6	Operators	506
A.8.7	Numbers	506
A.8.8	Strings.....	507
A.9	General.....	507
A.9.1	Attributes.....	507
A.9.2	Comments.....	508
A.9.3	Identifiers	508
A.9.4	White space	509
Annex B (normative) List of keywords		510
Annex C (informative) System tasks and functions		511
C.1	\$countdrivers	511
C.2	\$getpattern	512
C.3	\$input	513
C.4	\$key and \$nokey	513
C.5	\$list.....	513
C.6	\$log and \$nolog	514
C.7	\$reset, \$reset_count, and \$reset_value	514
C.8	\$save, \$restart, and \$incsave.....	515
C.9	\$scale	516
C.10	\$scope	516
C.11	\$showscopes	516
C.12	\$showvars	516
C.13	\$readmemb and \$readmemh.....	517
Annex D (informative) Compiler directives.....		518
D.1	`default_decay_time.....	518
D.2	`default_trireg_strength	518
D.3	`delay_mode_distributed	519
D.4	`delay_mode_path.....	519
D.5	`delay_mode_unit	519
D.6	`delay_mode_zero.....	519
Annex E (normative) acc_user.h (deprecated).....		520
Annex F (normative) veriusers.h (deprecated).....		521
Annex G (normative) vpi_user.h		522
Annex H (informative) Encryption/decryption flow		537
H.1	Tool vendor secret key encryption system	537
H.1.1	Encryption input.....	537

H.1.2	Encryption output	538
H.2	IP author secret key encryption system	538
H.2.1	Encryption input	538
H.2.2	Encryption output	539
H.3	Digital envelopes	539
H.3.1	Encryption input	540
H.3.2	Encryption output	541
Annex I (informative) Bibliography		542
Index		543

List of Figures

Figure 4-1—Simulation values of a trireg and its driver	28
Figure 4-2—Simulation results of a capacitive network	29
Figure 4-3—Simulation results of charge sharing	30
Figure 7-1—Schematic diagram of interconnections in array of instances	80
Figure 7-2—Scale of strengths	88
Figure 7-3—Combining unequal strengths	89
Figure 7-4—Combination of signals of equal strength and opposite values	89
Figure 7-5—Weak x signal strength	89
Figure 7-6—Bufifs with control inputs of x	90
Figure 7-7—Strong H range of values	90
Figure 7-8—Strong L range of values	90
Figure 7-9—Combined signals of ambiguous strength	91
Figure 7-10—Range of strengths for an unknown signal	91
Figure 7-11—Ambiguous strengths from switch networks	91
Figure 7-12—Range of two strengths of a defined value	92
Figure 7-13—Range of three strengths of a defined value	92
Figure 7-14—Unknown value with a range of strengths	92
Figure 7-15—Strong X range	93
Figure 7-16—Ambiguous strength from gates	93
Figure 7-17—Ambiguous strength signal from a gate	93
Figure 7-18—Weak 0	94
Figure 7-19—Ambiguous strength in combined gate signals	94
Figure 7-20—Elimination of strength levels	95
Figure 7-21—Result showing a range and the elimination of strength levels of two values	95
Figure 7-22—Result showing a range and the elimination of strength levels of one value	96
Figure 7-23—A range of both values	97
Figure 7-24—Wired logic with unambiguous strength signals	98
Figure 7-25—Wired logic and ambiguous strengths	99
Figure 7-26—Triage net with capacitance	104
Figure 8-1—Module schematic and simulation times of initial value propagation	113
Figure 9-1—Repeat event control utilizing a clock edge	139
Figure 12-1—Hierarchy in a model	193
Figure 12-2—Hierarchical path names in a model	193
Figure 12-3—Scopes available to upward name referencing	196
Figure 14-1—Module path delays	212
Figure 14-2—Difference between parallel and full connection paths	219
Figure 14-3—Module path delays longer than distributed delays	226
Figure 14-4—Module path delays shorter than distributed delays	226
Figure 14-5—Legal and illegal module paths	226
Figure 14-6—Illegal module paths	227
Figure 14-7—Legal module paths	227
Figure 14-8—Example of pulse filtering	228
Figure 14-9—On-detect versus on-event	231
Figure 14-10—Current event cancellation problem and correction	233
Figure 14-11—NAND gate with nearly simultaneous input switching where one event is scheduled prior to another that has not matured	234
Figure 14-12—NAND gate with nearly simultaneous input switching with output event scheduled at same time	235
Figure 15-1—Sample \$timeskew	251
Figure 15-2—Sample \$timeskew with remain_active_flag set	252
Figure 15-3—Sample \$fullskew	254

Figure 15-4—Timing check violation windows	264
Figure 15-5—Data constraint interval, positive setup/hold	267
Figure 15-6—Data constraint interval, negative setup/hold	268
Figure 18-1—Creating the four-state VCD file	325
Figure 18-2—Creating the extended VCD file	338
Figure 26-1—Example of object relationships diagram	378
Figure 26-2—Accessing a class of objects using tags	379
Figure 27-1—s_vpi_error_info structure definition	419
Figure 27-2—s_cb_data structure definition	423
Figure 27-3—s_vpi_delay structure definition	424
Figure 27-4—s_vpi_time structure definition	424
Figure 27-5—s_vpi_systf_data structure definition	427
Figure 27-6—s_vpi_time structure definition	428
Figure 27-7—s_vpi_value structure definition	430
Figure 27-8—s_vpi_vecval structure definition	430
Figure 27-9—s_vpi_strengthval structure definition	430
Figure 27-10—s_vpi_vlog_info structure definition	436
Figure 27-11—s_vpi_delay structure definition	448
Figure 27-12—s_vpi_time structure definition	448
Figure 27-13—s_vpi_value structure definition	452
Figure 27-14—s_vpi_time structure definition	453
Figure 27-15—s_vpi_vecval structure definition	453
Figure 27-16—s_vpi_strengthval structure definition	453
Figure 27-17—s_cb_data structure definition	454
Figure 27-18—s_vpi_systf_data structure definition	462

List of Tables

Table 3-1—Specifying special characters in string	14
Table 4-1—Net types	26
Table 4-2—Truth table for wire and tri nets	27
Table 4-3—Truth table for wand and triand nets	27
Table 4-4—Truth table for wor and trior nets	27
Table 4-5—Truth table for tri0 net	31
Table 4-6—Truth table for tri1 net	31
Table 4-7—Differences between specparams and parameters	38
Table 5-1—Operators in Verilog HDL	42
Table 5-2—Legal operators for use in real expressions	43
Table 5-3—Operators not allowed for real expressions	43
Table 5-4—Precedence rules for operators	44
Table 5-5—Arithmetic operators defined	45
Table 5-6—Power operator rule examples	46
Table 5-7—Unary operators defined	46
Table 5-8—Examples of modulus and power operators	46
Table 5-9—Data type interpretation by arithmetic operators	47
Table 5-10—Definitions of relational operators	48
Table 5-11—Definitions of equality operators	49
Table 5-12—Bitwise binary and operator	50
Table 5-13—Bitwise binary or operator	50
Table 5-14—Bitwise binary exclusive or operator	51
Table 5-15—Bitwise binary exclusive nor operator	51
Table 5-16—Bitwise unary negation operator	51
Table 5-17—Reduction unary and operator	52
Table 5-18—Reduction unary or operator	52
Table 5-19—Reduction unary exclusive or operator	52
Table 5-20—Results of unary reduction operations	52
Table 5-21—Ambiguous condition results for conditional operator	54
Table 5-22—Bit lengths resulting from self-determined expressions	63
Table 6-1—Legal left-hand forms in assignment statements	68
Table 7-1—Built-in gates and switches	76
Table 7-2—Valid gate types for strength specifications	76
Table 7-3—Truth tables for multiple input logic gates	81
Table 7-4—Truth tables for multiple output logic gates	82
Table 7-5—Truth tables for three-state logic gates	83
Table 7-6—Truth tables for MOS switches	84
Table 7-7—Strength levels for scalar net signal values	87
Table 7-8—Strength reduction rules	100
Table 7-9—Rules for propagation delays	101
Table 8-1—UDP table symbols	108
Table 8-2—Initial statements in UDPs and modules	111
Table 8-3—Mixing of level-sensitive and edge-sensitive entries	115
Table 9-1—Detecting posedge and negedge	133
Table 9-2—Intra-assignment timing control equivalence	138
Table 12-1—Net types resulting from dissimilar port connections	180
Table 14-1—List of valid operators in state-dependent path delay expression	215
Table 14-2—Associating path delay expressions with transitions	223
Table 14-3—Calculating delays for x transitions	224
Table 15-1—\$setup arguments	241
Table 15-2—\$hold arguments	242

Table 15-3—\$setuphold arguments	243
Table 15-4—\$removal arguments	245
Table 15-5—\$recovery arguments	246
Table 15-6—\$recrem arguments	247
Table 15-7—\$skew arguments	249
Table 15-8—\$timeskew arguments	250
Table 15-9—\$fullskew arguments	253
Table 15-10—\$width arguments	255
Table 15-11—\$period arguments	256
Table 15-12—\$nochange arguments	257
Table 15-13—Notifier value responses to timing violations	260
Table 16-1—Mapping of SDF delay constructs to Verilog declarations	269
Table 16-2—Mapping of SDF timing check constructs to Verilog	271
Table 16-3—SDF annotation of interconnect delays	273
Table 16-4—SDF to Verilog delay value mapping	276
Table 17-1—Escape sequences for printing special characters	279
Table 17-2—Escape sequences for format specifications	279
Table 17-3—Format specifications for real numbers	281
Table 17-4—Logic value component of strength format	283
Table 17-5—Mnemonics for strength levels	284
Table 17-6—Explanation of strength formats	285
Table 17-7—Types for file descriptors	287
Table 17-8—mtm spec argument	298
Table 17-9—scale type argument	298
Table 17-10—\$timeformat units_number arguments	300
Table 17-11—\$timeformat default value for arguments	301
Table 17-12—Diagnostics for \$finish	302
Table 17-13—PLA modeling system tasks	303
Table 17-14—Types of queues of \$q_type values	307
Table 17-15—Argument values for \$q_exam system task	308
Table 17-16—Status code values	308
Table 17-17—Verilog to C function cross-listing	313
Table 17-18—Verilog to C real math function cross-listing	324
Table 18-1—Rules for left-extending vector values	331
Table 18-2—How the VCD can shorten values	331
Table 18-3—Keyword commands	332
Table 19-1—Arguments of time_precision	359
Table 19-2—IEEE 1364-1995 reserved keywords	362
Table 19-3—IEEE 1364-2001 reserved keywords	363
Table 19-4—IEEE 1364-2005 reserved keywords	364
Table 26-1—VPI routines for simulation-related callbacks	381
Table 26-2—VPI routines for system task/function callbacks	381
Table 26-3—VPI routines for traversing Verilog HDL hierarchy	382
Table 26-4—VPI routines for accessing properties of objects	382
Table 26-5—VPI routines for accessing objects from properties	382
Table 26-6—VPI routines for delay processing	382
Table 26-7—VPI routines for logic and strength value processing	382
Table 26-8—VPI routines for simulation time processing	382
Table 26-9—VPI routines for miscellaneous utilities	383
Table 27-1—Return error constants for vpi_chk_error()	419
Table 27-2—Size of the s_vpi_delay->da array	425
Table 27-3—Return value field of the s_vpi_value structure union	431
Table 27-4—Size of the s_vpi_delay->da array	449
Table 27-5—Value format field of cb_data_p->value->format	455

Table 27-6—cbStmt callbacks	457
Table 28-1—protect pragma keywords	469
Table 28-2—Encoding algorithm identifiers	474
Table 28-3—Encryption algorithm identifiers	476
Table 28-4—Message digest algorithm identifiers	481
Table C.1—Argument return value for \$countdriver function	512

List of Syntax Boxes

Syntax 3-1—Syntax for integer and real numbers.....	9
Syntax 3-2—Syntax for system tasks and functions	15
Syntax 3-3—Syntax for attributes	16
Syntax 3-4—Syntax for module declaration attributes.....	18
Syntax 3-5—Syntax for port declaration attributes	18
Syntax 3-6—Syntax for module item attributes	19
Syntax 3-7—Syntax for function port, task, and block attributes	19
Syntax 3-8—Syntax for port connection attributes	20
Syntax 3-9—Syntax for udp attributes	20
Syntax 4-1—Syntax for net declaration.....	22
Syntax 4-2—Syntax for variable declaration.....	23
Syntax 4-3—Syntax for integer, time, real, and realtime declarations.....	32
Syntax 4-4—Syntax for module parameter declaration	36
Syntax 4-5—Syntax for specparam declaration	38
Syntax 5-1—Syntax for conditional operator	53
Syntax 5-2—Syntax for mintypmax expression.....	61
Syntax 6-1—Syntax for continuous assignment.....	69
Syntax 6-2—Syntax for variable declaration assignment.....	73
Syntax 7-1—Syntax for gate instantiation.....	75
Syntax 8-1—Syntax for UDPs.....	106
Syntax 8-2—Syntax for UDP instances.....	113
Syntax 9-1—Syntax for blocking assignments.....	118
Syntax 9-2—Syntax for nonblocking assignments.....	119
Syntax 9-3—Syntax for procedural continuous assignments.....	123
Syntax 9-4—Syntax for if statement	125
Syntax 9-5—Syntax for if-else-if construct.....	126
Syntax 9-6—Syntax for case statement	127
Syntax 9-7—Syntax for looping statements	130
Syntax 9-8—Syntax for procedural timing control	132
Syntax 9-9—Syntax for event declaration.....	133
Syntax 9-10—Syntax for event trigger	134
Syntax 9-11—Syntax for wait statement	136
Syntax 9-12—Syntax for intra-assignment delay and event control	137
Syntax 9-13—Syntax for sequential block	140
Syntax 9-14—Syntax for parallel block	141
Syntax 9-15—Syntax for initial construct	143
Syntax 9-16—Syntax for always construct	144
Syntax 10-1—Syntax for task declaration	146
Syntax 10-2—Syntax for task-enabling statement	147
Syntax 10-3—Syntax for disable statement.....	150
Syntax 10-4—Syntax for function declaration	153
Syntax 10-5—Syntax for function call	155
Syntax 12-1—Syntax for module	164
Syntax 12-2—Syntax for module instantiation	165
Syntax 12-3—Syntax for port.....	173
Syntax 12-4—Syntax for port declarations	174
Syntax 12-5—Syntax for generate constructs	182
Syntax 12-6—Syntax for hierarchical path names	192
Syntax 12-7—Syntax for upward name referencing	194
Syntax 13-1—Syntax for cell	199
Syntax 13-2—Syntax for declaring library in library map file.....	201

Syntax 13-3—Syntax for include command.....	202
Syntax 13-4—Syntax for configuration.....	203
Syntax 13-5—Syntax for default clause.....	203
Syntax 13-6—Syntax for instance clause.....	203
Syntax 13-7—Syntax for cell clause.....	204
Syntax 13-8—Syntax for liblist clause.....	204
Syntax 13-9—Syntax for use clause.....	204
Syntax 14-1—Syntax for specify block.....	211
Syntax 14-2—Syntax for module path declaration.....	212
Syntax 14-3—Syntax for simple module path.....	213
Syntax 14-4—Syntax for edge-sensitive path declaration.....	214
Syntax 14-5—Syntax for state-dependent paths.....	215
Syntax 14-6—Syntax for path delay value.....	222
Syntax 14-7—Syntax for PATHPULSE\$ pulse control.....	229
Syntax 14-8—Syntax for pulse style declarations.....	231
Syntax 14-9—Syntax for showcanceled declarations.....	232
Syntax 15-1—Syntax for system timing checks.....	238
Syntax 15-2—Syntax for check time conditions and timing check events.....	239
Syntax 15-3—Syntax for \$setup.....	241
Syntax 15-4—Syntax for \$hold.....	242
Syntax 15-5—Syntax for \$setuphold.....	243
Syntax 15-6—Syntax for \$removal.....	245
Syntax 15-7—Syntax for \$recovery.....	246
Syntax 15-8—Syntax for \$recrem.....	247
Syntax 15-9—Syntax for \$skew.....	249
Syntax 15-10—Syntax for \$timeskew.....	250
Syntax 15-11—Syntax for \$fullskew.....	252
Syntax 15-12—Syntax for \$width.....	255
Syntax 15-13—Syntax for \$period.....	256
Syntax 15-14—Syntax for \$nochange.....	257
Syntax 15-15—Syntax for edge-control specifier.....	258
Syntax 15-16—Syntax for controlled timing check event.....	265
Syntax 17-1—Syntax for \$display and \$write system tasks.....	278
Syntax 17-2—Syntax for \$strobe system tasks.....	285
Syntax 17-3—Syntax for \$monitor system tasks.....	286
Syntax 17-4—Syntax for \$fopen and \$fclose system tasks.....	287
Syntax 17-5—Syntax for file output system tasks.....	288
Syntax 17-6—Syntax for formatting data tasks.....	289
Syntax 17-7—Syntax for memory load system tasks.....	296
Syntax 17-8—Syntax for \$sdf_annotate system task.....	297
Syntax 17-9—Syntax for \$printrtimescale.....	299
Syntax 17-10—Syntax for \$timeformat.....	300
Syntax 17-11—Syntax for \$finish.....	302
Syntax 17-12—Syntax for \$stop.....	302
Syntax 17-13—Syntax for PLA modeling system task.....	303
Syntax 17-14—Syntax for \$time.....	309
Syntax 17-15—Syntax for \$stime.....	309
Syntax 17-16—Syntax for \$realtime.....	310
Syntax 17-17—Syntax for \$random.....	311
Syntax 17-18—Syntax for probabilistic distribution functions.....	312
Syntax 18-1—Syntax for \$dumpfile task.....	325
Syntax 18-2—Syntax for filename.....	326
Syntax 18-3—Syntax for \$dumpvars task.....	326
Syntax 18-4—Syntax for \$dumpoff and \$dumpon tasks.....	327

Syntax 18-5—Syntax for \$dumpall task.....	328
Syntax 18-6—Syntax for \$dumplimit task	328
Syntax 18-7—Syntax for \$dumpflush task.....	328
Syntax 18-8—Syntax for output four-state VCD file.....	330
Syntax 18-9—Syntax for \$comment section	332
Syntax 18-10—Syntax for \$date section	332
Syntax 18-11—Syntax for \$enddefinitions section.....	333
Syntax 18-12—Syntax for \$scope section.....	333
Syntax 18-13—Syntax for \$timescale	334
Syntax 18-14—Syntax for \$upscope section.....	334
Syntax 18-15—Syntax for \$var section.....	334
Syntax 18-16—Syntax for \$version section	335
Syntax 18-17—Syntax for \$dumpall keyword	335
Syntax 18-18—Syntax for \$dumpoff keyword	336
Syntax 18-19—Syntax for \$dumpon keyword	336
Syntax 18-20—Syntax for \$dumpvars keyword	336
Syntax 18-21—Syntax for \$dumpports task.....	338
Syntax 18-22—Syntax for \$dumpportsoff and \$dumpportson system tasks	339
Syntax 18-23—Syntax for \$dumpportsall system task.....	340
Syntax 18-24—Syntax for \$dumpportslimit system task	340
Syntax 18-25—Syntax for \$dumpportsflush system task.....	341
Syntax 18-26—Syntax for \$vcdclose keyword	341
Syntax 18-27—Syntax for output extended VCD file.....	343
Syntax 18-28—Syntax for extended VCD node information.....	344
Syntax 18-29—Syntax for value change section.....	346
Syntax 19-1—Syntax for default_nettype compiler directive	350
Syntax 19-2—Syntax for text macro definition.....	350
Syntax 19-3—Syntax for text macro usage	351
Syntax 19-4—Syntax for undef compiler directive.....	352
Syntax 19-5—Syntax for conditional compilation directives.....	353
Syntax 19-6—Syntax for include compiler directive	356
Syntax 19-7—Syntax for line compiler directive.....	357
Syntax 19-8—Syntax for timescale compiler directive.....	358
Syntax 19-9—Syntax for pragma compiler directive	360
Syntax 19-10—Syntax for begin keywords and end keywords compiler directives.....	361

IEEE Standard for Verilog[®] Hardware Description Language

1. Overview

1.1 Scope

Verilog is a hardware description language (HDL) that was standardized as IEEE Std 1364[™]-1995 and first revised as IEEE Std 1364-2001. This revision corrects and clarifies features ambiguously described in the 1995 and 2001 editions. It also resolves incompatibilities and inconsistencies of IEEE 1364-2001 with IEEE Std 1800[™]-2005.

The intent of this standard is to serve as a complete specification of the Verilog HDL. This standard contains the following:

- The formal syntax and semantics of all Verilog HDL constructs
- The formal syntax and semantics of standard delay format (SDF) constructs
- Simulation system tasks and functions, such as text output display commands
- Compiler directives, such as text substitution macros and simulation time scaling
- The programming language interface (PLI) binding mechanism
- The formal syntax and semantics of the Verilog procedural interface (VPI)
- Informative usage examples
- Informative delay model for SDF
- The VPI header file

1.2 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 some examples and notes.

The term *shall* is used throughout this standard to indicate mandatory requirements, whereas the term *may* is used to indicate optional features. These terms denote different meanings to different readers of this standard: