

Contents

Part One:

Design and Verification Constructs

1. Overview
 - 1.1 Scope
 - 1.2 Purpose
 - 1.3 Merger of IEEE Std 1364-2005 and IEEE Std 1800-2005
 - 1.4 Special terms
 - 1.5 Conventions used in this standard
 - 1.6 Syntactic description
 - 1.7 Use of color in this standard
 - 1.8 Contents of this standard
 - 1.9 Deprecated clauses
 - 1.10 Examples
 - 1.11 Prerequisites
2. Normative references
3. Design and verification building blocks
 - 3.1 General
 - 3.2 Design elements
 - 3.3 Modules
 - 3.4 Programs
 - 3.5 Interfaces
 - 3.6 Checkers
 - 3.7 Primitives
 - 3.8 Subroutines
 - 3.9 Packages
 - 3.10 Configurations
 - 3.11 Overview of hierarchy
 - 3.12 Compilation and elaboration
 - 3.13 Name spaces
 - 3.14 Simulation time units and precision
4. Scheduling semantics
 - 4.1 General
 - 4.2 Execution of a hardware model and its verification environment
 - 4.3 Event simulation
 - 4.4 The stratified event scheduler
 - 4.5 The SystemVerilog simulation reference algorithm
 - 4.6 Determinism
 - 4.7 Nondeterminism
 - 4.8 Race conditions
 - 4.9 Scheduling implication of assignments
 - 4.10 The PLI callback control points
5. Lexical conventions
 - 5.1 General
 - 5.2 Lexical tokens
 - 5.3 White space
 - 5.4 Comments
 - 5.5 Operators
 - 5.6 Identifiers, keywords, and system names
 - 5.7 Numbers
 - 5.8 Time literals
 - 5.9 String literals
 - 5.10 Structure literals
 - 5.11 Array literals
 - 5.12 Attributes
 - 5.13 Built-in methods

- 6. Data types
 - 6.1 General
 - 6.2 Data types and data objects
 - 6.3 Value set
 - 6.4 Singular and aggregate types
 - 6.5 Nets and variables
 - 6.6 Net types
 - 6.7 Net declarations
 - 6.8 Variable declarations
 - 6.9 Vector declarations
 - 6.10 Implicit declarations
 - 6.11 Integer data types
 - 6.12 Real, shortreal and realtime data types
 - 6.13 Void data type
 - 6.14 Chandle data type
 - 6.15 Class
 - 6.16 String data type
 - 6.17 Event data type
 - 6.18 User-defined types
 - 6.19 Enumerations
 - 6.20 Constants
 - 6.21 Scope and lifetime
 - 6.22 Type compatibility
 - 6.23 Type operator
 - 6.24 Casting
- 7. Aggregate data types
 - 7.1 General
 - 7.2 Structures
 - 7.3 Unions
 - 7.4 Packed and unpacked arrays
 - 7.5 Dynamic arrays
 - 7.6 Array assignments
 - 7.7 Arrays as arguments to subroutines
 - 7.8 Associative arrays
 - 7.9 Associative array methods
 - 7.10 Queues
 - 7.11 Array querying functions
 - 7.12 Array manipulation methods
- 8. Classes
 - 8.1 General
 - 8.2 Overview
 - 8.3 Syntax
 - 8.4 Objects (class instance)
 - 8.5 Object properties and object parameter data
 - 8.6 Object methods
 - 8.7 Constructors
 - 8.8 Static class properties
 - 8.9 Static methods
 - 8.10 This
 - 8.11 Assignment, renaming, and copying
 - 8.12 Inheritance and subclasses
 - 8.13 Overridden members
 - 8.14 Super
 - 8.15 Casting
 - 8.16 Chaining constructors
 - 8.17 Data hiding and encapsulation
 - 8.18 Constant class properties
 - 8.19 Virtual methods
 - 8.20 Abstract classes and pure virtual methods
 - 8.21 Polymorphism: dynamic method lookup

- 8.22 Class scope resolution operator ::
- 8.23 Out-of-block declarations
- 8.24 Parameterized classes
- 8.25 Typedef class
- 8.26 Classes and structures
- 8.27 Memory management
- 9. Processes
 - 9.1 General
 - 9.2 Structured procedures
 - 9.3 Block statements
 - 9.4 Procedural timing controls
 - 9.5 Process execution threads
 - 9.6 Process control
 - 9.7 Fine-grain process control
- 10. Assignment statements
 - 10.1 General
 - 10.2 Overview
 - 10.3 Continuous assignments
 - 10.4 Procedural assignments
 - 10.5 Variable declaration assignment (variable initialization)
 - 10.6 Procedural continuous assignments
 - 10.7 Assignment extension and truncation
 - 10.8 Assignment-like contexts
 - 10.9 Assignment patterns
 - 10.10 Unpacked array concatenation
 - 10.11 Net aliasing
- 11. Operators and expressions
 - 11.1 General
 - 11.2 Overview
 - 11.3 Operators
 - 11.4 Operator descriptions
 - 11.5 Operands
 - 11.6 Expression bit lengths
 - 11.7 Signed expressions
 - 11.8 Expression evaluation rules
 - 11.9 Tagged union expressions and member access
 - 11.10 String literal expressions
 - 11.11 Operator overloading
 - 11.12 Minimum, typical, and maximum delay expressions
 - 11.13 Let construct
- 12. Procedural programming statements
 - 12.1 General
 - 12.2 Overview
 - 12.3 Syntax
 - 12.4 Conditional if-else statement
 - 12.5 Case statement
 - 12.6 Pattern matching conditional statements
 - 12.7 Loop statements
 - 12.8 Jump statements
- 13. Tasks and functions (subroutines)
 - 13.1 General
 - 13.2 Overview
 - 13.3 Tasks
 - 13.4 Functions
 - 13.5 Subroutine calls and argument passing
 - 13.6 Import and export functions
 - 13.7 Task and function names
- 14. Clocking blocks
 - 14.1 General
 - 14.2 Overview

- 14.3 Clocking block declaration
- 14.4 Input and output skews
- 14.5 Hierarchical expressions
- 14.6 Signals in multiple clocking blocks
- 14.7 Clocking block scope and lifetime
- 14.8 Multiple clocking blocks example
- 14.9 Interfaces and clocking blocks
- 14.10 Clocking block events
- 14.11 Cycle delay: ##
- 14.12 Default clocking
- 14.13 Input sampling
- 14.14 Global clocking
- 14.15 Synchronous events
- 14.16 Synchronous drives
- 15. Interprocess synchronization and communication
 - 15.1 General
 - 15.2 Overview
 - 15.3 Semaphores
 - 15.4 Mailboxes
 - 15.5 Named events
- 16. Assertions
 - 16.1 General
 - 16.2 Overview
 - 16.3 Immediate assertions
 - 16.4 Deferred assertions
 - 16.5 Concurrent assertions overview
 - 16.6 Boolean expressions
 - 16.7 Sequences
 - 16.8 Declaring sequences
 - 16.9 Sequence operations
 - 16.10 Local variables
 - 16.11 Calling subroutines on match of a sequence
 - 16.12 System functions
 - 16.13 Declaring properties
 - 16.14 Multiclock support
 - 16.15 Concurrent assertions
 - 16.16 Disable iff resolution
 - 16.17 Clock resolution
 - 16.18 Expect statement
 - 16.19 Clocking blocks and concurrent assertions
- 17. Checkers
 - 17.1 Overview
 - 17.2 Checker declaration
 - 17.3 Checker instantiation
 - 17.4 Context inference
 - 17.5 Checker procedures
 - 17.6 Covergroups in checkers
 - 17.7 Checker variables
 - 17.8 Functions in checkers
 - 17.9 Complex checker example
- 18. Constrained random value generation
 - 18.1 General
 - 18.2 Overview
 - 18.3 Concepts and usage
 - 18.4 Random variables
 - 18.5 Constraint blocks
 - 18.6 Randomization methods
 - 18.7 In-line constraints—randomize() with
 - 18.8 Disabling random variables with rand_mode()
 - 18.9 Controlling constraints with constraint_mode()

- 18.10 Dynamic constraint modification
- 18.11 In-line random variable control
- 18.12 Randomization of scope variables—std::randomize()
- 18.13 Random number system functions and methods
- 18.14 Random stability
- 18.15 Manually seeding randomize
- 18.16 Random weighted case—randcase
- 18.17 Random sequence generation—randsequence
- 19. Functional coverage
 - 19.1 General
 - 19.2 Overview
 - 19.3 Defining the coverage model: covergroup
 - 19.4 Using covergroup in classes
 - 19.5 Defining coverage points
 - 19.6 Defining cross coverage
 - 19.7 Specifying coverage options
 - 19.8 Predefined coverage methods
 - 19.9 Predefined coverage system tasks and system functions
 - 19.10 Organization of option and type_option members
 - 19.11 Coverage computation
- 20. Utility system tasks and system functions
 - 20.1 General
 - 20.2 Simulation control system tasks
 - 20.3 Simulation time system functions
 - 20.4 Timescale system tasks
 - 20.5 Conversion functions
 - 20.6 Data query functions
 - 20.7 Array querying functions
 - 20.8 Math functions
 - 20.9 Severity tasks
 - 20.10 Elaboration system tasks
 - 20.11 Assertion control system tasks
 - 20.12 Assertion action control system tasks
 - 20.13 Assertion system functions
 - 20.14 Coverage system functions
 - 20.15 Probabilistic distribution functions
 - 20.16 Stochastic analysis tasks and functions
 - 20.17 Programmable logic array (PLA) modeling system tasks
 - 20.18 Miscellaneous tasks and functions
- 21. I/O system tasks and system functions
 - 21.1 General
 - 21.2 Display system tasks
 - 21.3 File input-output system tasks and system functions
 - 21.4 Loading memory array data from a file
 - 21.5 Writing memory array data to a file
 - 21.6 Command line input
 - 21.7 Value change dump (VCD) files
- 22. Compiler directives
 - 22.1 General
 - 22.2 Overview
 - 22.3 `resetall
 - 22.4 `include
 - 22.5 `define, `undef and `undefineall
 - 22.6 `ifdef, `else, `elsif, `endif, `ifndef
 - 22.7 `timescale
 - 22.8 `default_nettype
 - 22.9 `unconnected\_drive and `nounconnected_drive
 - 22.10 `celldefine and `endcelldefine
 - 22.11 `pragma
 - 22.12 `line

- 22.13 `\_\_FILE\_\_` and `\_\_LINE\_\_`
- 22.14 `begin\_keywords`, `end\_keywords`
- Part Two:
- Hierarchy Constructs
- 23. Modules and hierarchy
 - 23.1 General
 - 23.2 Module definitions
 - 23.3 Module instances (hierarchy)
 - 23.4 Nested modules
 - 23.5 Extern modules
 - 23.6 Hierarchical names
 - 23.7 Member selects and hierarchical names
 - 23.8 Upwards name referencing
 - 23.9 Scope rules
 - 23.10 Overriding module parameters
 - 23.11 Binding auxiliary code to scopes or instances
- 24. Programs
 - 24.1 General
 - 24.2 Overview
 - 24.3 The program construct
 - 24.4 Eliminating testbench races
 - 24.5 Blocking tasks in cycle/event mode
 - 24.6 Programwide space and anonymous programs
 - 24.7 Program control tasks
- 25. Interfaces
 - 25.1 General
 - 25.2 Overview
 - 25.3 Interface syntax
 - 25.4 Ports in interfaces
 - 25.5 Modports
 - 25.6 Interfaces and specify blocks
 - 25.7 Tasks and functions in interfaces
 - 25.8 Parameterized interfaces
 - 25.9 Virtual interfaces
 - 25.10 Access to interface objects
- 26. Packages
 - 26.1 General
 - 26.2 Package declarations
 - 26.3 Referencing data in packages
 - 26.4 Using packages in module headers
 - 26.5 Search order rules
 - 26.6 Exporting imported names from packages
 - 26.7 The std built-in package
- 27. Generate constructs
 - 27.1 General
 - 27.2 Overview
 - 27.3 Generate construct syntax
 - 27.4 Loop generate constructs
 - 27.5 Conditional generate constructs
 - 27.6 External names for unnamed generate blocks
- 28. Gate-level and switch-level modeling
 - 28.1 General
 - 28.2 Overview
 - 28.3 Gate and switch declaration syntax
 - 28.4 and, nand, nor, or, xor, and xnor gates
 - 28.5 buf and not gates
 - 28.6 bufif1, bufif0, notif1, and notif0 gates
 - 28.7 MOS switches
 - 28.8 Bidirectional pass switches
 - 28.9 CMOS switches

- 28.10 pullup and pulldown sources
- 28.11 Logic strength modeling
- 28.12 Strengths and values of combined signals
- 28.13 Strength reduction by nonresistive devices
- 28.14 Strength reduction by resistive devices
- 28.15 Strengths of net types
- 28.16 Gate and net delays
- 29. User defined primitives (UDPs)
 - 29.1 General
 - 29.2 Overview
 - 29.3 UDP definition
 - 29.4 Combinational UDPs
 - 29.5 Level-sensitive sequential UDPs
 - 29.6 Edge-sensitive sequential UDPs
 - 29.7 Sequential UDP initialization
 - 29.8 UDP instances
 - 29.9 Mixing level-sensitive and edge-sensitive descriptions
 - 29.10 Level-sensitive dominance
- 30. Specify blocks
 - 30.1 General
 - 30.2 Overview
 - 30.3 Specify block declaration
 - 30.4 Module path declarations
 - 30.5 Assigning delays to module paths
 - 30.6 Mixing module path delays and distributed delays
 - 30.7 Detailed control of pulse filtering behavior
- 31. Timing checks
 - 31.1 General
 - 31.2 Overview
 - 31.3 Timing checks using a stability window
 - 31.4 Timing checks for clock and control signals
 - 31.5 Edge-control specifiers
 - 31.6 Notifiers: user-defined responses to timing violations
 - 31.7 Enabling timing checks with conditioned events
 - 31.8 Vector signals in timing checks
 - 31.9 Negative timing checks
- 32. Backannotation using the standard delay format (SDF)
 - 32.1 General
 - 32.2 Overview
 - 32.3 The SDF annotator
 - 32.4 Mapping of SDF constructs to SystemVerilog
 - 32.5 Multiple annotations
 - 32.6 Multiple SDF files
 - 32.7 Pulse limit annotation
 - 32.8 SDF to SystemVerilog delay value mapping
 - 32.9 Loading timing data from an SDF file
- 33. Configuring the contents of a design
 - 33.1 General
 - 33.2 Overview
 - 33.3 Libraries
 - 33.4 Configurations
 - 33.5 Using libraries and configs
 - 33.6 Configuration examples
 - 33.7 Displaying library binding information
 - 33.8 Library mapping examples
- 34. Protected envelopes
 - 34.1 General
 - 34.2 Overview
 - 34.3 Processing protected envelopes
 - 34.4 Protect pragma directives

34.5 Protect pragma keywords

Part Three:

Application Programming Interfaces

35. Direct programming interface (DPI)

35.1 General

35.2 Overview

35.3 Two layers of the DPI

35.4 Global name space of imported and exported functions

35.5 Imported tasks and functions

35.6 Calling imported functions

35.7 Exported functions

35.8 Exported tasks

35.9 Disabling DPI tasks and functions

36. Programming language interface (PLI/VPI) overview

36.1 General

36.2 PLI purpose and history

36.3 User-defined system task and system function names

36.4 User-defined system task and system function arguments

36.5 User-defined system task and system function types

36.6 User-supplied PLI applications

36.7 PLI include files

36.8 VPI sizetf, compiletf and calltf routines

36.9 PLI mechanism

36.10 VPI access to SystemVerilog objects and simulation objects

36.11 List of VPI routines by functional category

36.12 VPI backwards compatibility features and limitations

37. VPI object model diagrams

37.1 General

37.2 VPI Handles

37.3 VPI object classifications

37.4 Key to data model diagrams

37.5 Module

37.6 Interface

37.7 Modport

37.8 Interface task or function declaration

37.9 Program

37.10 Instance

37.11 Instance arrays

37.12 Scope

37.13 IO declaration

37.14 Ports

37.15 Reference objects

37.16 Nets

37.17 Variables

37.18 Packed array variables

37.19 Variable select

37.20 Memory

37.21 Variable drivers and loads

37.22 Object Range

37.23 Typespec

37.24 Structures and unions

37.25 Named events

37.26 Parameter, spec param, def param, param assign

37.27 Class definition

37.28 Class typespec

37.29 Class variables and class objects

37.30 Constraint, constraint ordering, distribution

37.31 Primitive, prim term

37.32 UDP

37.33 Intermodule path

- 37.34 Constraint expression
- 37.35 Module path, path term
- 37.36 Timing check
- 37.37 Task and function declaration
- 37.38 Task and function call
- 37.39 Frames
- 37.40 Threads
- 37.41 Delay terminals
- 37.42 Net drivers and loads
- 37.43 Continuous assignment
- 37.44 Clocking block
- 37.45 Assertion
- 37.46 Concurrent assertions
- 37.47 Property declaration
- 37.48 Property specification
- 37.49 Sequence declaration
- 37.50 Sequence expression
- 37.51 Immediate assertions
- 37.52 Multiclock sequence expression
- 37.53 Let
- 37.54 Simple expressions
- 37.55 Expressions
- 37.56 Atomic statement
- 37.57 Event statement
- 37.58 Process
- 37.59 Assignment
- 37.60 Event control
- 37.61 While, repeat
- 37.62 Waits
- 37.63 Delay control
- 37.64 Repeat control
- 37.65 Forever
- 37.66 If, if-else
- 37.67 Case, pattern
- 37.68 Expect
- 37.69 For
- 37.70 Do-while, foreach
- 37.71 Alias statement
- 37.72 Disables
- 37.73 Return statement
- 37.74 Assign statement, deassign, force, release
- 37.75 Callback
- 37.76 Time queue
- 37.77 Active time format
- 37.78 Attribute
- 37.79 Iterator
- 37.80 Generates
- 38. VPI routine definitions
 - 38.1 General
 - 38.2 vpi_chk_error()
 - 38.3 vpi_compare_objects()
 - 38.4 vpi_control()
 - 38.5 vpi_flush()
 - 38.6 vpi_get()
 - 38.7 vpi_get64()
 - 38.8 vpi_get_cb_info()
 - 38.9 vpi_get_data()
 - 38.10 vpi_get_delays()
 - 38.11 vpi_get_str()
 - 38.12 vpi_get_systf_info()

- 38.13 vpi_get_time()
- 38.14 vpi_get_userdata()
- 38.15 vpi_get_value()
- 38.16 vpi_get_value_array()
- 38.17 vpi_get_vlog_info()
- 38.18 vpi_handle()
- 38.19 vpi_handle_by_index()
- 38.20 vpi_handle_by_multi_index()
- 38.21 vpi_handle_by_name()
- 38.22 vpi_handle_multi()
- 38.23 vpi_iterate()
- 38.24 vpi_mcd_close()
- 38.25 vpi_mcd_flush()
- 38.26 vpi_mcd_name()
- 38.27 vpi_mcd_open()
- 38.28 vpi_mcd_printf()
- 38.29 vpi_mcd_vprintf()
- 38.30 vpi_printf()
- 38.31 vpi_put_data()
- 38.32 vpi_put_delays()
- 38.33 vpi_put_userdata()
- 38.34 vpi_put_value()
- 38.35 vpi_put_value_array()
- 38.36 vpi_register_cb()
- 38.37 vpi_register_systf()
- 38.38 vpi_release_handle()
- 38.39 vpi_remove_cb()
- 38.40 vpi_scan()
- 38.41 vpi_vprintf()
- 39. Assertion API
 - 39.1 General
 - 39.2 Overview
 - 39.3 Static information
 - 39.4 Dynamic information
 - 39.5 Control functions
- 40. Code coverage control and API
 - 40.1 General
 - 40.2 Overview
 - 40.3 SystemVerilog real-time coverage access
 - 40.4 FSM recognition
 - 40.5 VPI coverage extensions
- 41. Data read API

Part Four:

Annexes

Annex A (normative) Formal syntax

- A.1 Source text
- A.2 Declarations
- A.3 Primitive instances
- A.4 Instantiations
- A.5 UDP declaration and instantiation
- A.6 Behavioral statements
- A.7 Specify section
- A.8 Expressions
- A.9 General
- A.10 Footnotes (normative)

Annex B (normative) Keywords

Annex C (normative) Deprecation

- C.1 General
- C.2 Constructs that have been deprecated
- C.3 Accellera SystemVerilog 3.1a-compatible access to packed data

C.4 Constructs identified for deprecation

Annex D (informative) Optional system tasks and system functions

- D.1 General
- D.2 \$countdrivers
- D.3 \$getpattern
- D.4 \$input
- D.5 \$key and \$nokey
- D.6 \$list
- D.7 \$log and \$nolog
- D.8 \$reset, \$reset_count, and \$reset_value
- D.9 \$save, \$restart, and \$incsave
- D.10 \$scale
- D.11 \$scope
- D.12 \$showscopes
- D.13 \$showvars
- D.14 \$sreadmemb and \$sreadmemh

Annex E (informative) Optional compiler directives

- E.1 General
- E.2 `default_decay_time
- E.3 `default_trireg_strength
- E.4 `delay_mode_distributed
- E.5 `delay_mode_path
- E.6 `delay_mode_unit
- E.7 `delay_mode_zero

Annex F (normative) Formal semantics of concurrent assertions

- F.1 General
- F.2 Overview
- F.3 Abstract syntax
- F.4 Rewriting algorithms
- F.5 Semantics
- F.6 Extended expressions
- F.7 Recursive properties

Annex G (normative) Std package

- G.1 General
- G.2 Overview
- G.3 Semaphore
- G.4 Mailbox
- G.5 Randomize
- G.6 Process

Annex H (normative) DPI C layer

- H.1 General
- H.2 Overview
- H.3 Naming conventions
- H.4 Portability
- H.5 svdpi.h include file
- H.6 Semantic constraints
- H.7 Data types
- H.2 Argument passing modes
- H.3 Context tasks and functions
- H.4 Include files
- H.5 Arrays
- H.6 Open arrays
- H.7 SV3.1a-compatible access to packed data (deprecated functionality)

Annex I (normative) svdpi.h

- I.1 General
- I.2 Overview
- I.3 Source code

Annex J (normative) Inclusion of foreign language code

- J.1 General
- J.2 Overview

- J.3 Location independence
- J.4 Object code inclusion
- Annex K (normative) vpi_user.h
 - K.1 General
 - K.2 Source code
- Annex L (normative) vpi_compatibility.h
 - L.1 General
 - L.2 Source code
- Annex M (normative) sv_vpi_user.h
 - M.1 General
 - M.2 Source code
- Annex N (normative) Algorithm for probabilistic distribution functions
 - N.1 General
 - N.2 Source code
- Annex O (informative) Encryption/decryption flow
 - O.1 General
 - O.2 Overview
 - O.3 Tool vendor secret key encryption system
 - O.4 IP author secret key encryption system
 - O.5 Digital envelopes
- Annex P (informative) Glossary
- Annex Q (informative) Mapping of IEEE Std 1364-2005 and IEEE Std 1800-2005 clauses into IEEE Std 1800-2009
- Annex R (informative) Bibliography