

# Verilog Test suites

Verilog Test suite is a combination of three different test suites namely SystemVerilog 3.0, Verilog-AMS 2.0 and Verilog-2001. Verilog test cases only for IEEE-1364 are not present. Each is categorized separately.

This is an ideal set of test cases for people who want to upgrade their tool-set to higher versions of Verilog. This rich set of test cases ensures that you achieve 100% coverage.

Leading EDA vendors and design firms have integrated these suites in their testing environment.

## Highlights

The test suite covers SystemVerilog 3.0 (approved by Accellera, 2002), Verilog-AMS 2.0 (Approved by Open Verilog International, 2000), IEEE-1364 and Verilog-2001.

All test cases are arranged according to the section number as provided in the LRM. Also each test case clearly states the feature that is being tested.

All the related test cases are grouped together under the same category.

Inside a group every test case is stored under a separate directory.

Each synthesizable test case has a bench file. Designs which are not synthesizable are self testing.

There is a large set of miscellaneous designs which combines different SystemVerilog features and also features of Verilog-95/Verilog-2001.

There is rich set of negative test cases. All negative test cases have a 'neg\_' prefix.



# Verilog Test suites

SystemVerilog 3.0 features tested in the test suite.

- Unsized literal using apostrophe (‘).
- Specifying time units like ps, ns etc to time values.
- Special string characters like \v (vertical tab), \f (form feed), \a (bell) etc.
- Array initialization using concat or multiconcat.
- Structure and union initialization using concat and multiconcat.
- New data types viz. char, int, shortint, longint, byte, bit, logic which is integer type and shortreal which is equivalent to C float.
- New keywords viz. timeunit and timeprecision.
- User defined data type declaration using typedef.
- New constructs like structure, union and enum.
- Complex data type declaration using structure, union and enum.
- Packed structure and unions.
- Packed and unpacked arrays and their use.
- Array querying system functions like \$left, \$right etc.
- Casting using the cast operator (^).
- Type conversion using PLI functions like \$itor, \$rtoi, \$bitstoreal, \$realtobits etc.
- Constant declaration using the keyword const.
- Static and automatic variable declaration.
- Assignment operators and special bitwise assignment operators, such as +=, -=, \*=, /=, %=, &=, |=, ^=, <<=, >>=, <<<=, >>>=, and incrementor decrementor operators ++ and --.
- Selection statements using keywords priority and unique.
- New loop statement using the do-while construct.
- New jump statements like break, continue or return.
- Named block with matching name at the end of the block.
- Declaration in unnamed block.
- New event control constructs using iff and new keyword like changed.



# Verilog Test suites

- New process constructs using `always_comb`, `always_latch`, `always_ff`. Process keyword is used for dynamic processes.
- Task and function with input, output and inout ports.
- Functions returning void.
- Function and task returning control using the keyword `return`.
- Automatic task or functions with static variables.
- Procedural assertion.
- System function like `$asserton`, `$assertoff` etc with assertion.
- The use of `$root` as the top level scope.
- Use of constructs like declaration, instantiation and procedural statements in the `$root` scope.
- Access of variables declared in `$root` scope using `$root` as the first name in a hierarchical name.
- Nested module declaration.
- Module instantiation using `(.name)` connection, using `(.*)` connection.
- Validation of connection in terms of size and data type.
- Interfaces and `modport` definition.
- Interface instantiation, generic interface and import export of functions.
- Parameter declaration using the keyword `type`.
- Instantiating modules with type parameters.
- Macro text with double back tick, to allow identifiers to be constructed from arguments.



# Verilog Test suites

Verilog-2001 features tested in the test suite.

- Configuration and liblist.
- Generate (if, for, case).
- Use of genvar. Positive as well as negative examples.
- localparam declaration and examples to check that they cannot be overridden.
- Use of new compiler directives 'ifndef, 'elsif, 'undef, 'line.
- Examples on attributes. These attributes are just to test whether the parser allows them.
- New operators >>>, <<<. \*\*.
- Constant functions and expressions.
- Multidimensional arrays of reg, integer, wire etc.
- signed object declaration and use of signed constants, objects.
- Indexed vector part select both in LHS and RHS.
- Examples to check whether sign bit is extended when the value is 'x' or 'z'.
- Use of \$signed and \$unsigned system function calls.
- Automatic function and task calls.
- Different kinds of sensitivity lists (comma separated, wildcard).
- Examples to check default vector net declaration and disable default net declaration ('default\_nettype none).
- Explicit parameter overriding (named association).
- Combined port data type declaration.
- reg initialization with declaration.
- ANSI style module parameter and port declaration list.
- Function and task declaration in ANSI style.
- Enhanced file I/O, including build in file I/O task like \$fgetc, \$fscanf.
- Distribution PLI task functions (dist\_normal, dist\_poisson etc).
- Enhanced invocation option tests (\$value\$plusargs).



# Verilog Test suites

Verilog-AMS features tested in the test suite.

- Analog block
- Analog sequential block
- Analog branch contribution
- Analog indirect branch assignment
- Analog conditional statement
- Analog for statement
- Analog case statement
- While, repeat and for statement must not include analog operators.
- Analog event controlled statement
- Initial step event - Occurs only within analog block
- Final step event - Occurs only within analog block
- Cross event
- Above event
- Timer event
- Procedural assignment in analog block
- Generate statement
- Operators in analog block - The operands of the operators can be integer or real.
- Standard mathematical functions, trigonometric and hyperbolic functions.
- Time derivative function ddt and time integral functions idt and idtmod
- Branch declaration
- Analog function declaration
- Real number - Unit like 'T', 'G' etc. can be used in representing real constant.
- Optional range specification to designate permissible values of a parameter in parameter declaration.
- Dynamic parameter declaration
- Nature declaration
- Discipline declaration
- Net-discipline declaration



# Verilog Test suites

- Use ground declaration to specify an already declared net of continuous discipline.
- wreal net declaration
- Simulator functions like discontinuity, bound\_step, last\_crossing, abstime, realtime, temperature, vt.
- Access functions - The access function names are defined by the access attributes specified for the discipline's natures.
- Port Access operator (<>)
- Hierarchical referencing operator can be used to access the attributes for a node or branch
- Functions to examine driver properties like driver\_count, driver\_state, driver\_strength, driver\_update
- Analysis dependent functions like analysis, ac\_stim, white\_noise, flicker\_noise, noise\_table
- Analog operators like idt, ddt etc.
- Connect module
- Connect specification - direction overridden and discipline overridden
- Compiler directive like 'default\_transition, 'default\_discipline etc.

