

Verilog 1364-200x User Survey

For each question below, circle the number to the right that best fits your opinion on including the language feature in the next generation of the IEEE 1364 Verilog standard.

0-No Opinion 1-Would Not Use 2-Might Not Use 3-Neutral 4-Might Use 5-Would Use

Primary Job Function: Design Verification CAD Support Manager Other _____

Section 1 - Data Type Extensions

1. ^{SV} 2-state data types	0	1	2	3	4	5
2. ^{SV} Data type extensions (e.g. structures, enumerations, arrays)	0	1	2	3	4	5
3. ^{SV} Object-oriented types and techniques (e.g. classes)	0	1	2	3	4	5
4. ^{SV} Explicit type casting	0	1	2	3	4	5
5. ^{SV} Dynamic memory allocation of arrays (e.g. dynamic, associative and sparse arrays)	0	1	2	3	4	5
6. Dynamic memory allocation for all variable types (eg. Cadence IEEE proposal)	0	1	2	3	4	5
7. Variable-width floating point types	0	1	2	3	4	5
8. Universal implicit data type (implicit declaration of both wire and reg types)	0	1	2	3	4	5
9. Apply extended types to nets and variables	0	1	2	3	4	5
10. Aspect-oriented types and techniques (similar to 'e')	0	1	2	3	4	5

Section 2 - Sequential/Behavioral Extensions

11. ^{SV} Add sequential statements (e.g. enhanced for loops, do-while, return, break, continue)	0	1	2	3	4	5
12. ^{SV} Add output and inout parameters to functions	0	1	2	3	4	5
13. ^{SV} Allow void function return type (so functions can act as statements)	0	1	2	3	4	5
14. ^{SV} Arguments passed-by-reference to tasks/functions (pointer vs. pass-by-copy)	0	1	2	3	4	5

Section 3 - Concurrent Statement Extensions

15. ^{SV} Specialized always blocks for different sensitivity/lint rules (e.g. always_{comb,latch,ff})	0	1	2	3	4	5
16. ^{SV} Final blocks (e.g. block that runs immediately prior to simulation exiting)	0	1	2	3	4	5
17. ^{SV} Enhanced fork/join control (e.g. fork/join_any, fork/join_none)	0	1	2	3	4	5
18. Dynamic processes (thread-like processes with independent handles and control)	0	1	2	3	4	5

Section 4 - Hierarchical Extensions

19. ^{SV} Interconnect modeling / encapsulated interfaces	0	1	2	3	4	5
20. ^{SV} Program block with special scheduling for verification	0	1	2	3	4	5
21. ^{SV} Cycle-accurate, synchronous, modeling block (e.g. SystemVerilog clocking domains)	0	1	2	3	4	5
22. ^{SV} Global declarations	0	1	2	3	4	5
23. ^{SV} Implicit port interconnect by name (e.g. .name, .*)	0	1	2	3	4	5

Section 5 - Assertions

24. ^{SV} Clocked sequential assertions	0	1	2	3	4	5
25. ^{SV} Declarative assertions (outside of an initial, always, task or function block)	0	1	2	3	4	5
26. ^{SV} Sequential assertions (contained in an initial, always, task or function block)	0	1	2	3	4	5
27. ^{SV} Assertions with multiple clocks	0	1	2	3	4	5
28. Invariant (unlocked) assertions	0	1	2	3	4	5

"SV" indicates this feature is present in the Accellera "SystemVerilog" extensions to Verilog.

Section 6 - Constraints and Randomization, Coverage

29. ^{SV} Randomization and constraint methods for objects in classes	0	1	2	3	4	5
30. Allow randomization of all variables	0	1	2	3	4	5
31. Functional coverage support, including goal specification (no specific proposal yet)	0	1	2	3	4	5

Section 7 - C-Interfaces

32. ^{SV} Direct access interface (function export/import to/from C instead of PLI registration)	0	1	2	3	4	5
33. ^{SV} VPI extensions for coverage	0	1	2	3	4	5
34. Extend VPI (PLI 2.0) for all new language features	0	1	2	3	4	5
35. Access VPI functions from HDL (Introspection)	0	1	2	3	4	5

Section 8 - Language Extension Mechanisms

36. Define standard mechanism for lexical pragmas (e.g. // synthesis translate_off)	0	1	2	3	4	5
37. Company/tool defined namespace for attributes (prevent conflicts from multiple sources)	0	1	2	3	4	5
38. Operator/type extension mechanism (e.g. VHDL-like packages or `include mechanism)	0	1	2	3	4	5
39. Increase support for separate compilation	0	1	2	3	4	5
40. Standardize support of encryption (e.g. Cadence proposal on encryption key IP protection)	0	1	2	3	4	5
41. Standard directive to identify language version (e.g. Perl-like 'require' statement)	0	1	2	3	4	5
42. Allow secondary name for modules (e.g. explicit rtl vs. gate version of same module)	0	1	2	3	4	5
43. Allow macros to be used in names (e.g. module dff size)	0	1	2	3	4	5
44. Add explicit power modeling (i.e. gating off power)	0	1	2	3	4	5
45. Enable the future (e.g. MIT Bluespec, or others)	0	1	2	3	4	5

Section 9 - Language Cleanup and Maintenance

46. Deprecate defparam (keep in-line parameter redefinition)	0	1	2	3	4	5
47. Deprecate procedural assign/deassign (keep continuous assign)	0	1	2	3	4	5
48. Clarify 'disable' ambiguity (e.g. what happens to scheduled non-blocking assignments)	0	1	2	3	4	5
49. Clarify interaction of arrays of regs and arrays of wires	0	1	2	3	4	5
50. Remove PLI 1.0 sections (TF/ACC) from future versions of the LRM (keep PLI 2.0 - VPI)	0	1	2	3	4	5
51. Create "umbrella" standard (e.g. integrate Verilog-AMS as 1364.X standard)	0	1	2	3	4	5
52. Rewrite LRM (has been incrementally enhanced for 13 years)	0	1	2	3	4	5
53. Extend VCD to support all Verilog-2001 data types and any new data types	0	1	2	3	4	5
54. Standard format for compressed VCD files	0	1	2	3	4	5
55. Define Interoperability between HDLs (VHDL, SystemC, PSL, Sugar, 'e', Vera)	0	1	2	3	4	5

"SV" indicates this feature is present in the Accellera "SystemVerilog" extensions to Verilog.

Please Write In Any Additional Suggestions For The Next Generation Of Verilog

	0	1	2	3	4	5
	0	1	2	3	4	5
	0	1	2	3	4	5
	0	1	2	3	4	5
	0	1	2	3	4	5
	0	1	2	3	4	5