

| Yes | No | Score | Req. # | Requirements: Syntax Compatability | ID |
|-----|----|-------|--------|---|----------------|
| 14 | 1 | 13 | R5b | Declarative assertions inside of modules/interfaces | SM6, RH8, AK11 |
| 14 | 1 | 13 | R7b | Common sequence syntax for procedural and declarative assertions | AK2 |
| 14 | 1 | 13 | R8 | Define boolean behavior of 4-state, multi-bit expressions | EM1 |
| | | | | Declarative assertions as well as procedural | RR14, SM6 |
| 12 | 1 | 11 | R5a | | |
| 11 | 0 | 11 | R7c | Define semantic overlap between procedural and declarative assertions | EM4 |
| 11 | 2 | 9 | R1c | Assertion and sequential expression syntax compatible in style with the rest of SystemVerilog | TF7 |
| 11 | 4 | 7 | R5c | Declarative assertions external to modules/interfaces | SD5 |
| 10 | 3 | 7 | R7a | Uniformity of expressiveness across multiple layers of behavioral specification | DG2 |
| 10 | 5 | 5 | R4b | Different syntax from PSL for different semantics | CE1, AK9 |
| 9 | 4 | 5 | R1b | Assertion and sequential expression syntax compatible with the rest of SystemVerilog | RR1, SM2 |
| 8 | 4 | 4 | R1d | Assertions must be embedded directly in-line with SystemVerilog code i.e. must be module_or_generate_item or statement_item in BNF | VG1 |
| 9 | 6 | 3 | R4a | Common syntax with PSL for common semantics | CE1, AK9 |
| 7 | 6 | 1 | R2 | Support all SystemVerilog expressions/types in assertions | SM1 |
| 7 | 6 | 1 | R3 | Support all SystemVerilog expressions/types in sequential expressions | SM1 |
| 6 | 7 | -1 | R6 | Pass/fail statement for declarative assertions | AK1 |
| 4 | 8 | -4 | R1a | Must be a superset of SystemVerilog 3.0 | DG1 |

Sequences

| | | | | | |
|----|---|----|------|--|------------------|
| 15 | 0 | 15 | R14 | Ability to specify repeat count in sequences | FV6 |
| 15 | 0 | 15 | R18a | Syntax to specify that a boolean cond shall be true within a window | RR5 |
| 15 | 0 | 15 | R20 | Ability to define/declare reusable sequences (and refer to them by name) | SM3, TF3 |
| 15 | 0 | 15 | R23 | Ability to combine sequences using logical operators | TF4 |
| 14 | 0 | 14 | R10 | Defined sample event when expressions in declarative or procedural assertions are sampled | RR9, SD11, RH2 |
| 14 | 1 | 13 | R18b | Syntax to specify that a boolean cond shall be true throughout a window | RR6, SD34 |
| 14 | 1 | 13 | R25 | Specify that boolean condition hold true throughout another sequence (isttrue) | SD4 |
| 13 | 0 | 13 | R11a | Sample clock can be any boolean expression of other signals | SD12 |
| 13 | 0 | 13 | R13 | Ability to refer to and/or synchronize between sequences with different sample clocks | SD26 |
| 13 | 2 | 11 | R16 | Syntax shortcut to specify "window" (min/max cycle count) within a sequence | RR11 |
| 13 | 2 | 11 | R31a | sequence completion to trigger other operations | RR10 |
| 13 | 2 | 11 | R9 | Ability to specify sequences with relationship between events | DG11, SD16, SD10 |
| 12 | 1 | 11 | R21 | Ability to define parameterized sequences. i.e: seq #(N=1) (a;b*[N];c) foo; foo(2) -> (a;b*[2];c); | TF2 |
| 12 | 1 | 11 | R22 | Ability to compose complex sequences out of other sequences. i.e. seq (...) foo; seq (...) bar; seq (foo;bar) foo;bar; | TF5 |
| 12 | 1 | 11 | R24 | Ability to specify that a sequence/expression doesn't occur | DG8 |
| 12 | 1 | 11 | R26 | Ability to restrict the length of a sequence | SD3 |
| 12 | 2 | 10 | R31b | sequence completion for coverage no obligation of proof | RR13 |
| 12 | 3 | 9 | R11b | Sample clock (step control) can be completion of a sequence | DG12 |
| 12 | 3 | 9 | R12 | Ability to specify sequences containing boolean conditions evaluated on different sample events | DG10, SD6 |
| 11 | 3 | 8 | R31c | Coverage supported by declarative assertions | |
| 11 | 4 | 7 | R18d | Syntax to specify that a boolean cond shall change value exactly once within a window | DG4 |
| 11 | 4 | 7 | R19a | Basic implication operator if (bool) then (seq1) [else (seq2)]; NOTE: Condition may only be boolean, no circular reasoning | SD17 |
| 10 | 3 | 7 | R19b | Basic implication operator if (seq) then (seq1); | SD17, RR10 |
| 10 | 5 | 5 | R15 | Syntax shortcut to represent repetition of 'true' | SM4 |
| 10 | 5 | 5 | R18c | Syntax to specify that a boolean cond shall hold value throughout a window | RR7 |
| 9 | 5 | 4 | R11c | support iff in step-control | SD1 |

| | | | | | |
|---|----|----|------|---|------------|
| 9 | 5 | 4 | R17 | Syntax to specify start/end events for window within a sequence | RR12 |
| 8 | 4 | 4 | R29a | Optional user-specified unique name for assertions | EM5 |
| 9 | 6 | 3 | R18e | Syntax to specify that a boolean cond shall change value one or more times within a window | RR8 |
| 7 | 5 | 2 | R29b | Mandatory user-specified unique name for assertions | RR15, AK14 |
| 5 | 7 | -2 | R28 | Ability to pause a sequence until some condition becomes true possibly infinite repetition of true | SD19 |
| 5 | 8 | -3 | R19c | Basic implication operator if (seq) then (seq1) [else (seq2)]; NOTE: Condition may only be boolean, no circular reasoning | SD17 |
| 5 | 10 | -5 | R27 | Explicitly disallow sequential expression implication mutex with 19b | SD18 |

Set/Reset of Sequences

| | | | | | |
|----|---|----|------|--|----------|
| 13 | 2 | 11 | R32 | Capability for dynamic disabling of assertions (i.e. suspend during reset, then restart) | DG3, RH1 |
| 12 | 1 | 11 | R33a | Ability to specify explicit synchronous condition to force sequence to terminate as "successful" (i.e. "set") Synchronous = condition evaluated only on sequence sample clock | SD27 |
| 10 | 2 | 8 | R34c | Default reset for assertions in a given region | EM7, SD8 |
| 10 | 3 | 7 | R33b | Ability to specify explicit synchronous condition to force sequence to terminate as "failed" (i.e. "reset") | SD28 |
| 9 | 4 | 5 | R34d | Default set for assertions in a given region | EM7, SD8 |
| 8 | 3 | 5 | R34b | set/reset of assertions from outside the assertion but within the same module | RR16 |
| 7 | 8 | -1 | R33c | Ability to specify explicit asynchronous event to force sequence to terminate as "successful" (i.e. "set") Asynchronous = condition evaluated independent of sample clock | SD29 |
| 6 | 9 | -3 | R33d | Ability to specify explicit asynchronous event to force sequence to terminate as "failed" (i.e. "reset") | SD30 |
| 2 | 6 | -4 | R35 | assertions as atomic objects | EM6 |
| 2 | 7 | -5 | R34a | set/reset of assertions from outside the assertion via hierarchical name | DG5 |

Assertions

| | | | | | |
|----|----|----|------|---|----------------------|
| 14 | 0 | 14 | R36a | Ability to specify different sample clock for each assertion | SD25 |
| 14 | 0 | 14 | R36b | Ability to specify default clock for all assertions within a scope | SD14, AK13 |
| 13 | 2 | 11 | R45a | Ability to specify that assertion must be checked | RH6, EM8, RR17, SD22 |
| 13 | 2 | 11 | R45b | Ability to specify that assertion is to be a constraint | RH6, EM8, RR17, SD23 |
| 11 | 1 | 10 | R43 | Automatically prevent false-firing in combinational blocks | AK7 |
| 10 | 1 | 9 | R44c | Optional specifier for assume/check | |
| 11 | 3 | 8 | R36c | Ability to specify default clock for all assertions globally | SD15, AK13 |
| 11 | 3 | 8 | R39a | Operator to match some other sequence | SD20 |
| 11 | 3 | 8 | R39b | Operator to match some other sequence in a different clock domain | SD21 |
| 10 | 2 | 8 | R36d | Ability to specify default clock for groups of assertions | SD7 |
| 10 | 5 | 5 | R45c | Ability to specify that assertion can be either assumed or proved | RH6, EM8, RR17, SD24 |
| 7 | 6 | 1 | R37a | Specify rules for sample clock to be extracted/inferred from context for procedural assertions (similar to synthesis) | SD13 |
| 6 | 6 | 0 | R38 | Restrict sequential assertions from being used in combinational always blocks (to avoid false firing) | SD38 |
| 6 | 9 | -3 | R41c | Automatically extract/infer lhs from usage context | SD50 |
| 6 | 9 | -3 | R41d | Automatically extract/infer rhs from usage context | SD51 |
| 6 | 9 | -3 | R42 | Ability to refer to inferred information from within template | SD52 |
| 5 | 9 | -4 | R40 | Inference from context for variable values | RH4 |
| 5 | 10 | -5 | R41a | Automatically extract/infer reset condition from usage context | SD39 |
| 5 | 10 | -5 | R41b | Automatically extract/infer enable condition from usage context | SD40 |
| 3 | 8 | -5 | R37b | Specify rules for sample clock to be extracted/inferred from context for declarative assertions | |
| 2 | 8 | -6 | R44b | Mandatory specifier for assume/check | DG6 |
| 2 | 9 | -7 | R44a | No ability to specify assume/check | DG7 |

Semantics

| | | | | | |
|----|---|----|------|---|----------------|
| 15 | 0 | 15 | R46 | Formally defined semantics for assertions | AK8, CE2, SD37 |
| 14 | 1 | 13 | R50a | Assertion evaluation shall have no side effects | CE5 |
| 13 | 0 | 13 | R48 | Semantics independent of implementation | CE4 |
| 12 | 1 | 11 | R47 | Define subset and/or coding guidelines for which semantics will be consistent between event-based and cycle-based (i.e. formal) tools | AK9, CE3, TF1 |
| 12 | 2 | 10 | R54 | Automatic variables, localized to a particular execution of a given assertion | SD55 |
| 10 | 2 | 8 | R56 | Expressions in assertions evaluated at end of timestep | AK6 |
| 11 | 4 | 7 | R53 | Recognition of multiple concurrent (overlapping) data-dependent instances of the same behavior described by a single property | FV7 |

| | | | | | |
|---|---|---|------|--|-----------|
| 9 | 4 | 5 | R50b | Assertion statement may not cause side effects in the design NOTE: Incompatible with SV3.0 | RH9 |
| 8 | 3 | 5 | R51 | Fifo semantics for handshake protocol of two expressions | AK10 |
| 7 | 3 | 4 | R49 | Ability to disable non-simulation properties | AK12 |
| 7 | 3 | 4 | R55 | Sample expressions in declarative or procedural assertions at predefined point in a timestep | AK6 |
| 9 | 6 | 3 | R52 | Allow assignment/reference to variables within an assertion | EM3, SD53 |

Past/Future Values

| | | | | | |
|----|----|----|------|--|-----------|
| 15 | 0 | 15 | R58b | refer to past value of signal constant number of cycles in the past | AK5 |
| 15 | 0 | 15 | R59a | Function/keyword to evaluate if signal was false on previous cycle and is now true (i.e. "posedge") | AK3, RR4 |
| 15 | 0 | 15 | R59b | Function/keyword to evaluate if signal was true on previous cycle and is now false (i.e. "negedge") | AK3, RR4 |
| 14 | 1 | 13 | R58a | refer to past value of signal one cycle in the past | AK5, RR3 |
| 11 | 2 | 9 | R59c | Function/keyword to evaluate if signal has the same value it had on the previous cycle (i.e. "stable") | AK3 |
| 10 | 4 | 6 | R57a | refer to future values of signal one cycle in the future | RR2, SD54 |
| 7 | 5 | 2 | R58c | refer to past value of signal constant number of cycles in the past with explicit enable | AK5 |
| 6 | 6 | 0 | R57b | refer to future values of signal constant number of cycles in the future | |
| 4 | 10 | -6 | R58d | refer to past value of signal variable number of cycles in the past | |
| 2 | 11 | -9 | R57c | refer to future values of signal variable number of cycles in the future | |

System Functions

| | | | | | |
|----|---|----|------|--|------|
| 11 | 3 | 8 | R61a | Ability to operate on compile-time constant sets | SD35 |
| 7 | 4 | 3 | R60 | Extend 3.0 system functions, \$onehot, \$onecold, \$isunknown to accept multiple arguments (i.e. make the "{}" optional) | AK4 |
| 5 | 8 | -3 | R61b | Ability to operate on dynamic-sized sets | SD36 |

Formal

| | | | | | |
|----|---|----|-----|--|----------|
| 14 | 1 | 13 | R62 | Recognition of first occurrence of an event or completion of sequence | FV1, SD9 |
| 13 | 2 | 11 | R63 | Recognition of all occurrences of an event | FV2, SM5 |
| 13 | 2 | 11 | R64 | Support Safety and Liveness properties | FV3 |
| 11 | 3 | 8 | R67 | Support for Weak and Strong clocks | FV9 |
| 11 | 3 | 8 | R68 | Level-sensitive clocks | FV10 |
| 11 | 3 | 8 | R70 | Allow composition of formulae with Temporal operators (i.e. before, until, etc) | DG9 |
| 10 | 4 | 6 | R65 | Closed under negation of a formula, not sequences | FV5 |
| 9 | 3 | 6 | R66 | Vacuity Check | FV8 |
| 6 | 6 | 0 | R69 | Require Finite Time Reasoning (i.e. Explicitly disallow Infinite Time Reasoning) | FV4 |
| 6 | 6 | 0 | R71 | Easy way to specify non-overlapping suffix implications | EM2 |

Usage

| | | | | | |
|----|---|---|------|--|-----------|
| 12 | 3 | 9 | R77 | Assertion encapsulation as element/template with arguments | RH7, SD31 |
| 11 | 4 | 7 | R78 | Ability to instantiate template | SD32 |
| 10 | 3 | 7 | R72 | Easy to use | RR18, SM8 |
| 10 | 5 | 5 | R75a | The assertion language should be demonstrably verifiable through simulation by sketching an algorithm for checking the assertions in reasonable complexity | RA1 |
| 9 | 4 | 5 | R73 | Ability to set severity of assertions | RH5 |
| 8 | 5 | 3 | R74 | Define recommended method to separate design code from verification code Tool/methodology requirement rather than a language requirements | PN1 |
| 8 | 5 | 3 | R76a | The assertion language should be demonstrably verifiable through model checking by sketching an algorithm for checking the assertions in reasonable complexity | RA3 |
| 8 | 6 | 2 | R75b | The assertion language should be demonstrably verifiable through simulation by showing a working tool that implements checking the assertions in reasonable complexity | RA2, VG2 |
| 8 | 6 | 2 | R79 | Ability to instantiate template without adding hierarchy | SD33, TF6 |
| 6 | 6 | 0 | R76b | The assertion language should be demonstrably verifiable through model checking by showing a working tool that implements checking the assertions in reasonable complexity | RA4, VG2 |

Status Characterization

| | | | | | |
|----|---|----|------|--|------|
| 13 | 1 | 12 | R80a | Language shall define the status of specified named assertion(s) | SD56 |
| 13 | 1 | 12 | R80b | Language shall define the status of specified named sequence(s) | SD57 |