

Working Document status wrt. final SV-AV requirements ballot

Total Requirements:	128	95 Met in Draft 5
Minimum 15 pts	8	
Minimum 14 pts	11	
Minimum 13 pts	22	
Minimum 12 pts	22	
Minimum 11 pts	39	
Minimum 10 pts	43	
Minimum 9 pts	49	
Minimum 8 pts	60	
Minimum 7 pts	69	
6pts or below	59	

Key:	Key:
Green = match SV-AC guidance	Green = mandate >7 total points
Orange = planned to match SV-AC guidance	Orange = possible requirement 0-6 points
Red = DWG did not match SV-AC guidance	Red = mandate against <0 points

Working Document Support	Syntax Example/Comment	Yes	No	Score	Req. #	Requirements: Syntax Compatability	ID
1 yes	procedural support	14	1	13	R5b	Declarative assertions inside of modules/interfaces	SM6, RH8, AK11
1 yes	same syntax	14	1	13	R7b	Common sequence syntax for procedural and declarative assertions	AK2
1 planned		14	1	13	R8	Define boolean behavior of 4-state, multi-bit expressions	EM1
1 yes	bind feature	12	1	11	R5a	Declarative assertions as well as procedural	RR14, SM6
1 yes		11	0	11	R7c	Define semantic overlap between procedural and declarative assertions	EM4
1 yes		11	2	9	R1c	Assertion and sequential expression syntax compatible in style with the rest of SystemVerilog	TF7
1 yes	bind feature	11	4	7	R5c	Declarative assertions external to modules/interfaces	SD5
1 yes		10	3	7	R7a	Uniformity of expressiveness across multiple layers of behavioral specification	DG2
1 yes	PSL will use same syntax for common subset	10	5	5	R4b	Different syntax from PSL for different semantics	CE1, AK9
1 yes		9	4	5	R1b	Assertion and sequential expression syntax compatible with the rest of SystemVerilog	RR1, SM2
1 yes		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
1 planned	PSL and SVA unification	9	6	3	R4a	Common syntax with PSL for common semantics	CE1, AK9
1 yes		7	6	1	R2	Support all SystemVerilog expressions/types in assertions	SM1
1 yes		7	6	1	R3	Support all SystemVerilog expressions/types in sequential expressions	SM1
1 yes		6	7	-1	R6	Pass/fail statement for declarative assertions	AK1
0 no		4	8	-4	R1a	Must be a superset of SystemVerilog 3.0	DG1

Sequences

1 yes	b[*3]	15	0	15	R14	Ability to specify repeat count in sequences	FV6
1 yes	seq_or_bool within sequence_expr	15	0	15	R18a	Syntax to specify that a boolean cond shall be true within a window	RR5
1 yes	sequence seq1; a ##1 b ##1 c endsequence	15	0	15	R20	Ability to define/declare reusable sequences (and refer to them by name)	SM3, TF3
1 yes	or, and, interset, within, throughout, implication	15	0	15	R23	Ability to combine sequences using logical operators	TF4
1 yes	clocking event defines sampling	14	0	14	R10	Defined sample event when expressions in declarative or procedural assertions are sampled	RR9, SD11, RH2
1 yes	seq_or_bool throughout sequence_expr	14	1	13	R18b	Syntax to specify that a boolean cond shall be true throughout a window	RR6, SD34
1 yes	seq_or_bool throughout sequence_expr	14	1	13	R25	Specify that boolean condition hold true throughout another sequence (istrue)	SD4
1 yes	sample clock is any boolean expression	13	0	13	R11a	Sample clock can be any boolean expression of other signals	SD12
1 yes	seq1 ## seq2, also implication and matched	13	0	13	R13	Ability to refer to and/or synchronize between sequences with different sample clocks	SD26
1 yes	delays windows [n:m] and repeats '1[*n:m]	13	2	11	R16	Syntax shortcut to specify "window" (min/max cycle count) within a sequence	RR11
1 yes	ended	13	2	11	R31a	sequence completion to trigger other operations	RR10
1 yes	\$rose foo; \$fell bar	13	2	11	R9	Ability to specify sequences with relationship between events	DG11, SD16, SD10
1 yes	sequence seq (N=1); a ##1 b[*N] ##1c endsequence;	12	1	11	R21	Ability to define parameterized sequences. i.e: foo(N=1) (a;b[*N];c) foo; foo(2) -> (a;b[*2];c);	TF2
1 yes	sequence seq1; a ##1 b endsequence; sequence seq2; c ##1 d endsequence; sequence seq_com; seq1##1seq2 endsequence;	12	1	11	R22	Ability to compose complex sequences out of other sequences. i.e. seq (...) foo; seq (...) bar; seq (foo;bar) foobar;	TF5
1 yes	assert1: assert not seq1;	12	1	11	R24	Ability to specify that a sequence/expression doesn't occur	DG8
1 yes	seq_or_bool within'1[*length>]	12	1	11	R26	Ability to restrict the length of a sequence	SD3
1 yes	cov1: cover seq1;	12	2	10	R31b	sequence completion for coverage no obligation of proof	RR13
1 yes	sequence seq_new; @(ended seq1) f ##1 g endsequence;	12	3	9	R11b	Sample clock (step control) can be completion of a sequence	DG12
1 yes	sequence multiclockseq; @(posedge clk1) a ##1 b ## @ (posedge clk2) c ##1 d endsequence;	12	3	9	R12	Ability to specify sequences containing boolean conditions evaluated on different sample events	DG10, SD6
1 yes	concurrent coverage statement and SV-EC API	11	3	8	R31c	Coverage supported by declarative assertions	

1	yes	!\$stable(<bool>) within <seq_expr>	11	4	7	R18d	Syntax to specify that a boolean cond shall change value exactly once within a window	DG4
1	yes	<bool> => <sequence_expr>	11	4	7	R19a	Basic implication operator if (bool) then (seq1) [else (seq2)]; NOTE: Condition may only be boolean, no circular reasoning	SD17
1	yes	<bool> => <sequence_expr>	10	3	7	R19b	Basic implication operator if (seq) then (seq1);	SD17, RR10
1	yes	1 [n]	10	5	5	R15	Syntax shortcut to represent repetition of 'true'	SM4
1	yes	\$stable(<bool>) throughout <sequence_expr>	10	5	5	R18c	Syntax to specify that a boolean cond shall hold value throughout a window	RR7
1	yes	iff supported	9	5	4	R11c	support iff in step-control	SD1
1	yes	accept directive of property	9	5	4	R17	Syntax to specify start/end events for window within a sequence	RR12
1	yes	assertion name is optional in concurrent_assert_statement	8	4	4	R29a	Optional user-specified unique name for assertions	EM5
1	yes	!\$stable (<bool>) within <seq_expr>	9	6	3	R18e	Syntax to specify that a boolean cond shall change value one or more times within a window	RR8
0	no	name is optional for concurrent_assert_statement, but required for property_spec	7	5	2	R29b	Mandatory user-specified unique name for assertions	RR15, AK14
0	no		5	7	-2	R28	Ability to pause a sequence until some condition becomes true possibly infinite repetition of true	SD19
0	yes	seq1 -> seq2	5	8	-3	R19c	Basic implication operator if (seq) then (seq1) [else (seq2)]; NOTE: Condition may only be boolean, no circular reasoning	SD17
0	no		5	10	-5	R27	Explicitly disallow sequential expression implication mutex with 19b	SD18

Set/Reset of Sequences

1	yes	DAS system functions	13	2	11	R32	Capability for dynamic disabling of assertions (i.e. suspend during reset, then restart)	DG3, RH1
1	yes	prop1: assert disable iff(reseta) @(posedge clk) f ##1 g;	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
1	no		10	2	8	R34c	Default reset for assertions in a given region	EM7, SD8
1	no		10	3	7	R33b	Ability to specify explicit synchronous condition to force sequence to terminate as "failed" (i.e. "reset")	SD28
0	no		9	4	5	R34d	Default set for assertions in a given region	EM7, SD8
0	no		8	3	5	R34b	set/reset of assertions from outside the assertion but within the same module	RR16
1	yes	disable iff	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
0	no		6	9	-3	R33d	Ability to specify explicit asynchronous event to force sequence to terminate as "failed" (i.e. "reset")	SD30
0	no		2	6	-4	R35	assertions as atomic objects	EM6
1	yes	DAS system functions	2	7	-5	R34a	set/reset of assertions from outside the assertion via hierarchical name	DG5

Assertions

1	yes	seq @posedge clk seq1 = (a ; b ; c);	14	0	14	R36a	Ability to specify different sample clock for each assertion	SD25
0	yes	default clock	14	0	14	R36b	Ability to specify default clock for all assertions within a scope	SD14, AK13
1	yes	assert	13	2	11	R45a	Ability to specify that assertion must be checked	RH6, EM8, RR17, SD22
0	no		13	2	11	R45b	Ability to specify that assertion is to be a constraint	RH6, EM8, RR17, SD23
1	yes	defined sampling semantics	11	1	10	R43	Automatically prevent false-firing in combinational blocks	AK7
1	yes	assert directive is optional	10	1	9	R44c	Optional specifier for assume/check	
0	no		11	3	8	R36c	Ability to specify default clock for all assertions globally	SD15, AK13
1	yes	ended (same clock domain)	11	3	8	R39a	Operator to match some other sequence	SD20
0	yes	matched (different clock domain)	11	3	8	R39b	Operator to match some other sequence in a different clock domain	SD21
0	yes	clocking domain from testbench SV-EC	10	2	8	R36d	Ability to specify default clock for groups of assertions	SD7
1	yes	assert is open to tool interpretation	10	5	5	R45c	Ability to specify that assertion can be either assumed or proved	RH6, EM8, RR17, SD24
1	yes	semantics for extraction of assertions into declarative assertion	7	6	1	R37a	Specify rules for sample clock to be extracted/inferred from context for procedural assertions (similar to synthesis)	SD13
0	no	no false firing with declarative semantics	6	6	0	R38	Restrict sequential assertions from being used in combinational always blocks (to avoid false firing)	SD38
0	no		6	9	-3	R41c	Automatically extract/infer lhs from usage context	SD50
0	no		6	9	-3	R41d	Automatically extract/infer rhs from usage context	SD51
0	no		6	9	-3	R42	Ability to refer to inferred information from within template	SD52
0	no		5	9	-4	R40	Inference from context for variable values	RH4
0	no		5	10	-5	R41a	Automatically extract/infer reset condition from usage context	SD39
0	no		5	10	-5	R41b	Automatically extract/infer enable condition from usage context	SD40
1	yes	Sample clock extracted from procedural context	3	8	-5	R37b	Specify rules for sample clock to be extracted/inferred from context for declarative assertions	
0	no		2	8	-6	R44b	Mandatory specifier for assume/check	DG6
1	yes	no ability to define assume	2	9	-7	R44a	No ability to specify assume/check	DG7

Semantics

1	yes	semantics definition is completed	15	0	15	R46	Formally defined semantics for assertions	AK8, CE2, SD37
1	yes	pass/fail statements can only be verification code	14	1	13	R50a	Assertion evaluation shall have no side effects	CE5
1	yes		13	0	13	R48	Semantics independent of implementation	CE4
1	yes	subset is everything except immediate assertions	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
1	yes	local variable construct, variable per evaluation	12	2	10	R54	Automatic variables, localized to a particular execution of a given assertion	SD55
1	yes	sampling semantics, just before next timestep at sampling clock	10	2	8	R56	Expressions in assertions evaluated at end of timestep	AK6
1	yes	local variable construct, variable per evaluation	11	4	7	R53	Recognition of multiple concurrent (overlapping) data-dependent instances of the same behavior described by a single property	FV7
1	yes	pass/fail statements can only be verification code	9	4	5	R50b	Assertion statement may not cause side effects in the design NOTE: Incompatible with SV3.0	RH9
0	no		8	3	5	R51	Fifo semantics for handshake protocol of two expressions	AK10

1	yes	DAS system functions	7	3	4	R49	Ability to disable non-simulation properties	AK12
1	yes	sample expressions evaluated prior to beginning of timestep	7	3	4	R55	Sample expressions in declarative or procedural assertions at predefined point in a timestep	AK6
1	yes	both per assertion and per attempt variables are supported	9	6	3	R52	Allow assignment/reference to variables within an assertion	EM3, SD53
Past/Future Values								
1	yes	past operator supports variable # reference	15	0	15	R58b	refer to past value of signal constant number of cycles in the past	AK5
1	yes	\$rose	15	0	15	R59a	Function/keyword to evaluate if signal was false on previous cycle and is now true (i.e. "posedge")	AK3, RR4
1	yes	\$fell	15	0	15	R59b	Function/keyword to evaluate if signal was true on previous cycle and is now false (i.e. "negedge")	AK3, RR4
1	yes	past operator supports variable # reference	14	1	13	R58a	refer to past value of signal one cycle in the past	AK5, RR3
1	yes	\$stable	11	2	9	R59c	Function/keyword to evaluate if signal has the same value it had on the previous cycle (i.e. "stable")	AK3
0	no		10	4	6	R57a	refer to future values of signal one cycle in the future	RR2, SD54
0	no		7	5	2	R58c	refer to past value of signal constant number of cycles in the past with explicit enable	AK5
0	no		6	6	0	R57b	refer to future values of signal constant number of cycles in the future	
1	yes	past operator supports variable # reference	4	10	-6	R58d	refer to past value of signal variable number of cycles in the past	
0	no		2	11	-9	R57c	refer to future values of signal variable number of cycles in the future	

System Functions

1	yes	SystemVerilog boolean expressions	11	3	8	R61a	Ability to operate on compile-time constant sets	SD35
0	no		7	4	3	R60	Extend 3.0 system functions, \$onehot, \$onecold, \$isunknown to accept multiple arguments (i.e. make the "[]" optional)	AK4
0	no		5	8	-3	R61b	Ability to operate on dynamic-sized sets	SD36

Formal

1	yes	first_match	14	1	13	R62	Recognition of first occurrence of an event or completion of sequence	FV1, SD9
1	yes	matched	13	2	11	R63	Recognition of all occurrences of an event	FV2, SM5
0	no	no formula operators	13	2	11	R64	Support Safety and Liveness properties	FV3
0	no	no explicit weak and strong, automatically inferred based on context/use	11	3	8	R67	Support for Weak and Strong clocks	FV9
1	yes	any glitch free boolean expression	11	3	8	R68	Level-sensitive clocks	FV10
0	no	support for only sequence implication and top level always, never. No formula composition.	11	3	8	R70	Allow composition of formulae with Temporal operators (i.e. before, until, etc)	DG9
1	yes	negation of top level formulas only	10	4	6	R65	Closed under negation of a formula, not sequences	FV5
0	no		9	3	6	R66	Vacuity Check	FV8
1	yes		6	6	0	R69	Require Finite Time Reasoning (i.e. Explicitly disallow Infinite Time Reasoning)	FV4
1	yes	non overlapping implication ->	6	6	0	R71	Easy way to specify non-overlapping suffix implications	EM2

Usage

1	yes	template feature	12	3	9	R77	Assertion encapsulation as element/template with arguments	RH7, SD31
1	yes	instantiation in both procedural and declarative	11	4	7	R78	Ability to instantiate template	SD32
1	yes		10	3	7	R72	Easy to use	RR18, SM8
1	yes		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
1	yes	system task	9	4	5	R73	Ability to set severity of assertions	RH5
1	yes	Scheduling semantics defines separation and link from design to verification.	8	5	3	R74	Define recommended method to separate design code from verification code Tool/methodology requirement rather than a language requirements	PN1
1	yes		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
1	yes		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
1	yes		8	6	2	R79	Ability to instantiate template without adding hierarchy	SD33, TF6
1	yes		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

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