

1364.1 Ballot comment resolution document (Revised Sept 10, 2002)								
Commenter	nt Number	for Vote?	Type	Location	Comment	Suggested Resolution	WG resolution / rationale	Change in draft
Stephen A. Bailey	SAB01	N	Editorial	5.3, pg10	Example 16 comment says "If enable is disasserted," Is disasserted a word?	I think "deasserted" is wanted here.	Change "disasserted" to "deasserted".	Change as noted.
	SAB02	N	Technical	5.3, pg 10	Section says: "Blocking assignments may be used for variables that are temporarily assigned and used in an always statement." What is the definition of temporarily? All variables are temporarily assigned. Shouldn't the statement indicate that the variable must be assigned and used only within the same always statement?	Improve the specification and ensure it is something that can be enforced/checked by the synthesis tools.	Change "variables that are temporarily assigned and used in an always statement" to "intermediate variables that are temporarily assigned and used only in the same always statement". Regarding checkability, no change.	Change as noted.
	SAB03	N	Technical	5.4, pg 11	The section states: "For a signal that has multiple drivers,if one of the drivers has an assignment of z , then all drivers shall have at least one assignment to z." This sentence is poorly constructed and is therefore confusing. What is the object z that must have at least one assignment to it? (I know that isn't what is meant, but that is how it is written.) I also don't understand what is the intended requirement.	Rewrite the sentence so that it makes sense.	Change to "If any driver of the signal contains an assignment to the value z, then all the drivers must contain such an assignment."	Change as noted.
Kai Moon Chow	KMC01	N	Editorial	General	The examples are very good to illustrate the standard in a easy understanding manner.		No change required.	No change.
John A. Eldon	JAE01	N	Technical	General	What are the reactions of synthesis tool vendors and users to the discouraging of pragmas and of "translate off ... translate on"? suggested_remedy =		No change required. Don't understand "suggested_remedy =". The WG discussed this issue extensively. The same functionality is provided by `ifdef SYNTHESIS.	No change.

Steve Golson	SG01	N	Editorial	General	<p>Section 1.4 a) states that all Verilog reserved words are boldface. This is not the case throughout the document; there are some places where the boldface is missing.</p> <p>Several of the examples (e.g. Example 4, Example 11, etc.) are split over page breaks. Recommend that you start a new page such that no example code spans a page break.</p> <p>Section 5.6.2, paragraph 2, second sentence, first word, "Uninitialized" is misspelled.</p> <p>Section 6.1 and following, the attribute "synthesis" is not bold, but is colored. This looks grey when printed on a b&w printer.</p> <p>Section 6.1.3, third paragraph (the code example) at the end reads "- applies to reg." Don't you mean "// applies to reg." Did you copy this from a VHDL standard? :-)</p> <p>Section 6.1.4, paragraph just before Example 32, third sentence, replace "eg." with "example"</p> <p>Section 6.2 has the macro SYNTHESIS in color. This looks grey when printed on a b&w printer.</p>	<p>Check that all Verilog keywords are boldfaced.</p> <p>Check that examples are not split over page breaks - pass this requirement to IEEE editors.</p> <p>Change "Unitialized" to "Uninitialized".</p> <p>"synthesis" in red is OK, it prints as bold. However, (* *) in 6.1 Note 1 which are also red do not print as bold. Make them bold.</p> <p>Change "--" to "//".</p> <p>Change "eg." to "example".</p> <p>"SYNTHESIS" in red is OK, it prints like bold. No change.</p>	<p>Change as noted. Remove the convention for all id's to be in uppercase. All id's were made to be in lowercase to be consistent across the document. Make all "synthesis" as in attributes non-bold as they are not reserved words of the language. In 6.1.1, delete "and shall have the following interpretation."</p>	Change as noted.
	SG02	N	Technical	5.4, pg 12	<p>Example 22 is very strange. What are you trying to model? This will *not* model a flop followed by a three-state driver. ENB is some weird combination of a reset and an enable. Compare this with the description in section 5.2.2.1.</p> <p>Consider what happens when ENB is asserted low, then removed, all within a single clock period. What should the value of OUT be? Should it be DIN *now*, or the value of DIN at the previous posedge CLOCK?</p> <p>How many flops do you expect will be synthesized?</p>	Use a better example.	<p>Replace Example 22 with two cases, where the enable is not registered and where it is registered:</p> <p>Example of three-state driver with non-registered enable:</p> <pre>always @(posedge CLOCK) Q <= DIN ;</pre> <p>assign OUT = ENB ? Q : 1'bz ;</p> <p>This generates one FF with a three-state driver on the output.</p> <p>Example of three-state driver with registered enable:</p> <pre>always @(posedge CLOCK) if (!ENB) OUT <= 1'bz ; else OUT <= DIN ;</pre> <p>This generates two FFs, one on DIN, and one on ENB, with a three-state driver on the output of the first FF, controlled by the output of the second FF.</p>	Change as noted.

	SG03	N	Technical	5.4, pg 11	Example 21 will have simulation/synthesis mismatch.	Add "// supported, but simulation mismatch might occur" or add "enb" to the sensitivity list.	Change Example 21 to: always @*	Change as noted.
	SG04	N	Technical	6.1.1, pg 17, 18	"The optional value shall be ignored by synthesis tools." What if the optional value is 0? Shouldn't the attribute be disabled? That's what section 6.1 says.	Delete this paragraph (in two places).	Delete in 6.1.1 (a) and (b) the sentence "The optional value shall be ignored by synthesis tools." In addition to the reason mentioned by the commenter, another reason is that "additional semantics of a non-zero value are not defined by this standard", but synthesis tools are not PROHIBITED to use the value.	Change as noted.
	SG05	N	Technical	6.1.4.a, pg 20	First paragraph reads "This attribute shall apply to an always block..." Current synthesis tools apply this to an entire module.	After the paragraph add a new paragraph that reads "This attribute shall also apply to a module in which case, it shall apply to all always blocks in that module. If no level-sensitive storage devices are inferred for the module, a warning may be issued."	The addition sounds reasonable. Change "may" to "shall" to be consistent with previous para.	Change as noted.
	SG06	N	Technical	6.1.4.h, pg 23	First paragraph reads "This attribute shall apply to an always block..." Current synthesis tools apply this to an entire module.	After the paragraph add a new paragraph that reads "This attribute shall also apply to a module in which case, it shall apply to all always blocks in that module. If no edge-sensitive storage devices are inferred for the module, a warning may be issued."	The addition sounds reasonable. Change "may" to "shall" to be consistent with previous para.	Change as noted.
	SG07	N	Technical	6.1.4.a, pg 20	Fourth paragraph reads "When no signal names are present..." I don't understand this paragraph at all. What does it mean?		Change fourth and fifth paragraphs from "signal names are present" to "signal names are specified in the attribute instance". Same applies to 6.1.4(h). When no signal names are present, the "=" is not required - enclose in square brackets to show that it is optional.	Change as noted.
Jonathan Greenlaw	JG01	N	Technical	4, general	The scope is supposed to enhanced portability of RTL between synthesis tools. This is to be enforced by clause 4 - verification. However, there is no practical conformance requirement. For example, since it would be impossible to apply every possible input vector to any sizeable design, Vendor A may choose a different set of test vectors than Vendor B. Even worse, a vendor may choose a ridiculously small set of vectors to test with.	Specify a detailed PIC for this clause. Include either coverage percentages or better yet a formal method such as logical equivalency.	I do not understand the proposal. What is a PIC? I am not sure it is necessary to specify a "practical conformance requirement". Many standards, such as 1364 itself, do not do so. No change for now.	No change.
T Downing Hopkins	TDH01	N	Editorial	1.5.d, pg 3, line 4	Spelling error: Functtionality	Change to: Functionality	Change "functtionality" to "functionality".	Change as noted.

	TDH02	N	Editorial	4.2.a, pg 5	<p>Ambiguous Clause:</p> <p>When asynchronous data is assigned, the asynchronous data shall not change during period the asynchronous control (the condition under which the data is assigned) is active.</p>	<p>Change to:</p> <p>When asynchronous data is assigned, the asynchronous data shall not change during the period in which the asynchronous control (the condition under which the data is assigned) is active.</p>	<p>Agreed to suggested change.</p>	<p>Change as noted.</p>
	TDH03	N	Editorial	4.2.b, pg 5	<p>4.2.b Third Sentence:</p> <p>"element" should be plural:</p> <p>With level-sensitive storage element, a general rule is that data inputs should be stable before enables go active.</p>	<p>Change to:</p> <p>With level-sensitive storage elements, a general rule is that data inputs should be stable before enables go active.</p>	<p>Change "element" to "elements". There is a typo in the authors statement - author meant "inactive" as described in the draft.</p>	<p>Change as noted.</p>
	TDH04	N	Editorial	5.1, pg 6	<p>5.1, Second Para, Third sentence:</p> <p>Faulty subordination and lack of commas to offset an appositive clause:</p> <p>However, it may be necessary to include all the variables read in the always statement in the event list to avoid mismatches between simulation and synthesized logic.</p>	<p>Change to:</p> <p>However, it may be necessary to include, in the event list, all the variables read in the always statment in order to avoid mismatches between simulation and synthesized logic.</p>	<p>Change to:</p> <p>"However, it may be necessary to include in the event list all the variables read in the always statement to avoid ..." (without additional commas).</p>	<p>Change as noted.</p>
	TDH05	N	Editorial	5.1, pg 6	<p>5.1, Para.4, Sentence 3</p> <p>Awkward sentence structure:</p> <p>For example, variables that are assigned values using blocking assignments inside the always statement before being used by other expressions do not have to appear in the event list.</p>	<p>Change to:</p> <p>For example, a variable whose value is assigned through a blocking assignment within an always statement does not have to appear in the event list if the assignment occurs prior to the variable's use in subsequent expressions.</p>	<p>Change to:</p> <p>"For example, a variable does not have to appear in the event list of an always statement if it is assigned a value with a blocking assignment before being used in subsequent expressions within the same always statement."</p>	<p>Change as noted.</p>
	TDH06	N	Editorial	5.2.2.10, pg 8	<p>5.2.5, Example 10, Second Commented Line</p> <p>//....may optionally have a synchronous reset.</p>	<p>Change to:</p> <p>//....may optionally have a synchronous set.</p>	<p>Change comment to:</p> <p>"// OUT models a positive edge-sensitive storage device with optionally a synchronous set."</p> <p>(Similar change to Example 9. Also, in Examples 7-10, "edge-sensitive" is enough.)</p>	<p>Change as noted.</p>
Andy Ickowicz	AI01	N	Coordination		<p>This draft meets coordination requirements.</p>		<p>No change needed.</p>	<p>No change.</p>

					<p>Coding which is more clear to designers intent:</p> <pre> ----- lat2 : process (gate, en, d) begin if (gate = '1') and (en = '1') then q <= d ; end if ; end process ; lat3 : process (gate, en, d) begin if (gate = '1') then if (en = '1') then q <= d ; else q <= q ; end if ; end if ; end process ; </pre> <p>To address the issue with lat1, I suggest that we add an attribute: latch_enable.</p> <p>subissue conclusions:</p> <pre> ----- </pre> <p>I am not saying that one group is right and the other is wrong in their approach. However, I am saying that with respect to attributes both standards should reflect a coordinated effort.</p> <p>The VHDL attribute leader is Sanjiv Narayan, narayan@cadence.com.</p>			
JL02	Y	Technical	6.1.4, General	<p>Your usage of async_set_reset and sync_set_reset is confusing.</p> <p>Your definition: async_set_reset: applies to resets/sets on latches sync_set_reset: applies to resets/sets on registers</p> <p>subissue 1: -----</p> <p>You usage conflicts with popular tool useage which uses async_set_reset for asynchronous resets/sets on either registers or latches. In fact, when browsing through one vendor's manuals, I saw async_set_reset also being used to identify a latch which only had set-reset conditions (SR Latch).</p>	<p>Work with VHDL SIWG to get one understanding of async_set_reset and sync_set_reset which is more similar to popular tool usage of this attribute.</p>	<p>Subissue 1: It is not clear what the author wants to do when specifying async_set_reset on registers. Maybe this can be addressed in the next revision of the standard.</p>	No change.	

					<p>subissue 2: -----</p> <p>"The presence of the attribute shall cause the set/reset logic to be applied directly to the set/reset terminals of an edge-sensitive storage device if such a device is available in the technology library."</p> <p>If I accept your definition, what happens when sync_set_reset is applied to a register with synchronous reset/set? Your text seems to make it legal to connect this to a library part that uses asynchronous reset/set. Interesting. Good for FPGAs, most likely bad for ASICs.</p>		<p>Subissue 2: I think the author meant to say "...with asynchronous...". We agree a clarification is required. Add a line "It is an error if the attribute is applied to an asynchronous set or reset signal."</p>	<p>Change as noted in subissue 2.</p>
					<p>subissue 3: -----</p> <p>VHDL SIWG understanding: async_set_reset: applies to asynchronous resets/sets on latches and registers sync_set_reset: applies to synchronous resets/sets on registers</p> <p>This is consistent with popular tool usage.</p> <p>VHDL SIWG did extend these some. Usage of these attributes also implies a nonprioritized (one-hot) behavior of the sets and resets.</p> <p>VHDL SIWG took the further interpretation that when async_set_reset is applied to either a latch or a register which has both set and reset, then the set and reset are mutually exclusive.</p>		<p>Subissue 3: duplicate comment as subissue 1.</p>	<p>No change.</p>
	JL03	Y	Technical	6.0, general	<p>Test_Port should not be an attribute: (* synthesis, test_port [= <optional_value>] *)</p> <p>Attributes are for capturing permanent designer intent of a coding style. If a test port is a temporary thing, then this should not be a candidate for an attribute. If a test port is a permanent thing, then it should be brought up through the design as a port (the normal way).</p> <p>It should be a synthesis tool command though. If your argument is that by having it an attribute, we will also get a synthesis tool command that corresponds to this attribute, then I would support this attribute, otherwise, no.</p>	<p>Remove the test_port attribute</p>	<p>There are existing synthesis tools that have a similar attribute and the attribute been found useful.</p>	<p>No change.</p>

JL04	Y	Technical	6.0, general	<p>6.3.2 "translate_on/translate_off" pragmas deprecated Unfortunately in VHDL we need this functionality. 1076.6-1999 standard defines these pragmas as: -- RTL_SYNTHESIS OFF -- RTL_SYNTHESIS ON</p> <p>Since you are deprecating features, which is marking them for no future use, it would be nice if these features were more similar to VHDL's usage of these features:</p> <pre>//RTL_SYNTHESIS OFF //RTL_SYNTHESIS ON</pre>	<pre>Use the metacommments: //RTL_SYNTHESIS OFF //RTL_SYNTHESIS ON</pre>	No change. The idea is to get rid of them entirely.	No change.
JL05	Y	Technical	5.1, general	<p>subclause 5.1 Modeling combinational logic Is there a way to tighten up your sensitivity rules to require a synthesis to produce an error and stop if combinational logic is produced that ignores required sensitivity list elements. In VHDL this is easy, but we don't use global variables (blocking assignments).</p>	See comment	The WG felt that such a restriction was not necessary.	No change.
JL06	Y	Technical	5.2.2.1, pg 10	<p>P10, Subclause 5.2.2.1, last note before subclause 5.3 NOTE--No specific style is required to infer edge-sensitive storage device with synchronous set/reset. A synthesis tool may optionally choose to or not to infer such a storage device.</p> <p>This is confusing. If I were the John Cooley of VHDL, I would conclude from this statement that if you code a synchronous reset register in Verilog, the synthesis tool may or may not give you synchronous reset functionality.</p>	<p>I would recommend that you put a coding style for synchronous reset in your document. Then if you must, put a note that indicates the quality of the implementation is both synthesis tool and silicon vendor library dependent. Either a single register library part or combinational logic + a library part are valid solutions as long as the RTL simulation and Gate simulation match.</p>	Extend note at end of 5.2.2.1 to say "See the sync_set_reset attribute on how it can be used to infer a device with synchronous set/reset." In addition, move entire note to 5.2.2.	Change as noted.
JL07	Y	Editorial	5.7, pg 15	<p>P15 subclause 5.7 first paragraph. A RAM shall be modeled using a Verilog memory (a two-dimensional reg array) that has the attribute ram_block associated with it. A RAM element may either be modeled as an edge sensitive storage element or as a level sensitive storage element. A RAM data value may be read synchronously or asynchronously.</p> <p>Is "reg array" a Verilog syntax thing? It is confusing to see "reg array" followed by a sentence later "as an edge sensitive storage element or as a level sensitive storage element."</p>	See comment	<p>A reg array is a Verilog thingie. The paragraph looks reasonable. FYI: 1364-2001 was very careful about removing confusion between "reg" and "register". 1364-2001 states, "NOTE In previous versions of the Verilog standard, the term register was used to encompass both the reg ,integer , time ,real and realtime types;but that the term is no longer used as a Verilog data type." Now the term "variable" is used to mean any of these data types, whereas "reg" is reserved for reg data type specifically.</p>	No change.
JL08	Y	Technical	5.7, pg 15	<p>P15 subclause 5.7 Modeling random access memories (RAM) In general RAM arrays can also have registers on the outputs. Do you intend to provide a coding style for this?</p>	Add coding style for supporting RAMs with registered outputs	The registered outputs need to be coded separately from the memory block. 5.2.2 describes how to accomplish this.	No change.

	JL09	Y	Technical	6.1, pg 17	subclause 6.1: NOTE--It is recommended that if a synthesis tool supports attributes other than those listed as part of this standard, then the syntax for specifying such an attribute be identical with the format described in this section.	For future expandability of your standard, I would recommend that the attribute synthesis be reserved exclusively to prefix 1364.1 standardized attributes. Otherwise, what happens when two different vendors create a new attribute of the same name but slightly different implementation? If synthesis belongs to 1364.1 exclusively, then only your attributes can be added.	Good point. Add a second note "It is recommended that a synthesis tool not use the attribute "synthesis" in any other form or meaning other than its intended use as described in this standard."	Make change as noted.
	JL10	N	Editorial	6.1.2, general	Section 6.1.2 RAM/ROM inference attributes The structure of this section is confusing to have the a, b, c, in the middle of the paragraph rather than a classic header approach: a: ROM Inference b: RAM Inference c: Logic Inference (disallow ROM or RAM inference)	Fix paragraph heading structure	Agreed. Fix para heading structure to contain only english text. Move syntax and format as part of the paragraphs. This is to be done in all of Sec 6.	Change as noted.
	JL11	N	Editorial	pg 22	P22, in note 2, just before example 39: gaurantee	Fix the typo	Agreed.	Change as noted.
	JL12	Y	Technical	pg 22	Is this example 39 really legal? 1) There is nothing in the case expression. 2) for ADD, SUB, Z = array; for GT, Z = scalar logic value Example 39: (* synthesis, op_sharing = 1 *) module ALU (. . .) always @(*) case () ADD: z = a + b; SUB: z = a - b; GT : z = a > b; default : z = 'bz; endcase endmodule	Fix the example. Even if it is legal, it is a bad coding practice to put a bit size result in the same object that holds an array size result.	Good point. Add z, a, b, op_code to port list and provide sizes. In addition, fill in case condition as op_code. Also fix the bit assignment problem by assigning to bit 0. Make the z assignment to the full range of variable z. Well, change variable z to alu_out as it causes confusion with the value z.	Change as noted.
	JL13	N	Editorial	6.3	6.3 Deprecated features How can you have deprecated features if this is the first version of your standard?	Perhaps these should not be part of the standard? If they are already in use, then they are a defacto standard.	The para explains that there are some current common practices. These are the ones that are planned to be deprecated.	No change.
James Markevitch	JM01	N	Editorial	1.5, pg 3	In paragraph d) the word "functionality" is misspelled as "functionality".	Correct the typo.	Agreed.	Change as noted.
	JM02	N	Editorial	3, pg 4	In paragraph 3. j) the "x" and "z" should be in a bold font.	Change to bold.	Agreed.	Change as noted.

JM03	Y	Technical	4, pg 4	<p>The final sentence in the second paragraph reads "There is no matching requirement placed on any internal nodes." It seems that one of the primary values of the "keep" attribute is to ensure that certain nodes ARE maintained.</p> <p>Is the intent that one of the effects of "keep" is that internal nodes will continue to match pre- and post-synthesis? Given that the Draft explicitly indicates that "keep" cancels "fsm_state" it appears that this is the intent.</p> <p>If so, then this sentence should be modified.</p> <p>If this is not one of the intents of the "keep" attribute, then that should be made explicit in 6.1.4 e).</p>	<p>Assuming the intent is to keep the signal, a suggested replacement sentence is:</p> <p>There is no matching requirement placed on any internal nodes unless the keep attribute is specified (see 6.1.4) for such a node. If the keep attribute is specified for a node, then matching shall be ensured.</p> <p>In the above sentences, the word "keep" should be in bold.</p>	Agreed.	Change as noted.
JM04	Y	Technical	4, pg 4	<p>The second paragraph in clause 4 discusses how the pre- and post-synthesis netlists shall agree in simulations. However, some reasonable designs might result in the intentional (i.e. non-bug) X output from the pre-synthesis netlist. This paragraph should be modified to indicate that this is acceptable. Clause B.10 indicates that this may be the result of errors, but the idea of X's in pre-synthesis and non-X's in post-synthesis should be made acceptable in the paragraph in Clause 4.</p> <p>An example of a reasonable design which intentionally drives X outputs in pre-synthesis might be an instruction decoder driving the data to a register file:</p> <pre> write_ena = 1'b0; write_data = 16'bx; case (instruction) `ALU_OP: begin write_ena = 1'b1; write_data = alu_out; end `LOAD_OP: begin write_ena = 1'b1; write_data = mem_bus; end // Lots of other instructions ... </pre>	<p>The second sentence of the paragraph, beginning with "Equivalent in this context" might be the best place to implement appropriate wording. It is not clear what an "unambiguous definition" is, but maybe that already means that it is legal to match X with any of X, 0, or 1.</p> <p>Without discussing this in more detail with the WG members, it is difficult to fully understand the intent of this paragraph and, therefore, to suggest specific text to change.</p>	The note at the end of sec 4 (before 4.1) states that it is ok for outputs to possibly have metalogical values.	No change.
JM05	Y	Technical	4.2, pg 5	In 4.2 a) the fourth sentence begins "Immediately after the clock edge". This fails to properly account for hold time constraints which might be present in the design. Paragraph 4 c) near the top of the same page indicates that setup/hold times must be taken into account before allowing data transitions. This sentence should be modified to reflect that.	Replace the entire sentence beginning with "Immediately after the clock edge" with "The input values may be changed after the clock edge and after sufficient time has elapsed to ensure that no hold time violations will occur."	Agreed.	Change as noted.
JM06	N	Editorial	5.1, pg 6	In the third paragraph of Clause 5.1, beginning with "A variable assigned in an always statement cannot", the word "cannot" should be replaced with "shall not".	Change "cannot" to "shall not" in the sentence.	Agreed.	Change as noted.

	JM07	Y	Technical	5.4, pg 11	<p>The second paragraph in Clause 5.4, reads "z assignments shall not propagate across variable assignments". Does this apply to implicit assignments, such as those which occur with module instantiation in hierarchical designs? Those are a common and important part of many designs (e.g. an I/O ring which is a sub-module of a complete ASIC).</p> <p>This sentence should be modified to explicitly allow propagation of z values across module ports (both up and down the hierarchy).</p> <p>A related comment is that simple assignments, such as assign bit1 = en ? data : 1'bz; assign bit2 = bit1; may occur in some design methodologies. Such trivial assignments, where no operators are involved, should also be allowed.</p>	Modify the sentence to allow one or both of the above two suggestions.	Z values cannot propagate across implicit assignments as well. Append sentence "...across variable assignments (including implicit assignments, such as those which occur with module instantiations).	Change as noted.
	JM08	N	Editorial	5.6, pg 12	<p>In 5.6 note 1, the word "absence" is misspelled as "absense".</p> <p>The same misspelling occurs on page 15 in note 2 of Clause 5.7</p>	Correct the spelling.	Agreed.	Change as noted.
	JM09	N	Editorial	5.6.1, pg 13	In the fourth line of the example, an extra closing parenthesis occurs after the variable "a".	Remove the typo.	Agreed.	Change as noted.
	JM10	N	Editorial	5.6.2, pg 13	In the second sentence of the second paragraph of Clause 5.6.2, the word "Uninitialized" is misspelled as "Unitialized".	Correct the misspelling.	Agreed.	Change as noted.
	JM11	N	Editorial	6.1, pg 16	In the sixth paragraph of Clause 6.1, beginning "If the attribute has an" the word "can" should be replaced with either "shall" or "may", depending on the intent here. I believe that the correct word is "shall".	Replace the word "can" with "shall" (or "may" if that is the intent).	Replace with "may".	Change as noted.
	JM12	Y	Technical	6.1.1, pg 17	<p>In Clause 6.1.1 the third sentence indicates that "The optional value shall be ignored". For consistency with other attributes, this should accept 0 and non-zero to disable/enable the feature.</p> <p>This same comment applies to both the full_case attribute as well as the parallel_case attribute near the top of page 18.</p>	For consistency with other attribute, just remove the sentence from each of the full_case and parallel_case descriptions.	Agreed with author's recommendation. Make same change for ROM/RAM attributes as well.	Change as noted.
	JM13	N	Editorial	6.1.1, pg 18	In note 5 beginning with "Adding a default statement to a case" the word "nullifies" should be changed to "nullify".	Change "nullifies" to "nullify".	Agreed.	Change as noted.
	JM14	N	Editorial	6.1.3, pg 19	The beginning of the fsm_state description, which is the third "paragraph" beginning "(synthesis, fam_state [=encoding_scheme" is not in a font consistent with the rest of the document.	Change the font as appropriate.	Agreed.	Change as noted.
Gregory A. Maston	GAM01	N	Editorial	Annex A, pg 63	I was surprised that there were still keywords that had been struck-out in Annex A. I appreciate in the "full syntax" description the need to identify unsupported Verilog constructs, but this annex is presented (I thought) to identify just the constructs supported for RTL synthesis. So I am confused why this section does not remove all the struck-out constructs and clearly identify just what is supported (or ignored; underlined constructs are acceptable here).	either consider removing all the struck-out keywords in Annex A or put a note in identifying why they are still present here.	Agree to add a note at end of first para. "Note: The BNF presented here is the complete Verilog BNF that identifies the supported, ignored and unsupported constructs for synthesis."	Change as noted.

Michael D. McKinney	MDM01	N	Technical	7	This comment may be "editorial" instead of "technical". In Section 7 "Syntax" of the PDF file, I noted that many strikeouts on many pages were incomplete. That is, a knockout would occur for the left-hand token, leaving the "::- <something>" in place. Or, a knockout would occur in the middle of a list of 'or-ed' items, leaving "token ::= <something>" in place. If just the strikeouts are removed from the list of syntax items at final printing the resulting words would not be very understandable.	This section will have to be cleaned up to make syntax elements that are understandable if the section is to be included in the final standard.	The strikeouts are part of the standard and there is no intention to take these out.	No change.
Paul J. Menchini	PJM01	N	Editorial	1.4, Item f) (page 2)	A double dash (--) is used when an em-dash (—) is indicated	Use an em-dash	Agreed. Double-dashes are used everywhere for a note. But the author is correct in saying that an em-dash ought to be used. All double dashes following Note will be changed to an em-dash. Also NOTE is uppercased.	Change as noted.
	PJM02	N	Editorial	1.5, Item d) (page 3)	"functionality" is misspelled	Correct spelling	Agreed.	Change as noted.
	PJM03	N	Substantive	1.5, Items h and l (page 3)	The introduction to this section clearly identifies the annexes as informative, yet item i again identifies Annex B as informative; moreover, item h does not identify Annex A as informative.	Strike the word "informative" from item l	Agreed.	Change as noted.
	PJM04	N	Editorial	1.5 Item h	The description of Annex A is a sentence fragment while the remaining descriptions in this subclause are complete sentences.	Add "This clause" at the beginning of the description.	Agreed. Change to "This annex" to be consistent with the following annex.	Change as noted.
	PJM05	N	Editorial	4 (page 4)	The first and second paragraphs each contain the phrase " - " where an em-dash is indicated	Use an em-dash in both places	Agreed.	Change as noted.
	PJM06	N	Editorial	4 (page 5), 5.2.2.1 (page 5)	The note contains a double dash	Use an em-dash	Agreed.	Change as noted.
	PJM07	N	Substantive	4.1 (page 5)	The second sentence starts with "Typically, this shall" The use of "shall" implies a requirement, but the use of "typically" implies it's not a requirement	Reword to eliminate the use of "shall"	Change to "Typically, this is done in ...".	Change as noted.
	PJM08	N	Editorial	4.2, item a (page 5)	This item contains two paragraphs. However, the start of the second paragraph is not indicated clearly, either by indentation of the first line or a blank line between the paragraphs	Adopt IEEE style for this item	Merge the two paragraphs into one.	Change as noted.
	PJM09	N	Editorial	5 (page 6)	The clause starts with "This section"	Change to "This clause"	Agreed.	Change as noted.
	PJM10	N	Editorial	5.1 (page 6)	Antecedent for "this" missing in the parenthetical phrase in the first sentence.	Add the appropriate antecedent	Change from "Combinational logic shall be modeled using a continuous assignment (this shall also include a net declaration assignment)." to "Combinational logic shall be modeled using a continuous assignment or a net declaration assignment." Rationale: Clarification	Change as noted.
	PJM11	N	Editorial	5.2.2.1 (page 9)	The line "Example 14" is widowed.	This line should start page 10	Move "Example 14:" to start of page 10. Rationale: Readability	Change as noted.
	PJM12	N	Editorial	5.3, paragraph 3	Antecedent for "this" is missing in the second sentence	Add the appropriate antecedent	No change to draft. See SAB02. Rationale: The sentence is clear the way it is.	No change.
	PJM13	N	Editorial	5.3, example 13	"i.e." is missing a terminal comma	Add the comma	Page 11, change "// i.e. Q is" to "// i.e., Q is". It is example 17. Rationale: English	Change as noted.

PJM14	N	Editorial	5.6, all notes, and following	Double dash used instead of an em-dash	Use an em-dash	change All "-" to "—". Rationale: Conformance to LRM style	Change as noted.
PJM15	N	Editorial	5.6.1, 5.6.2	The titles of these subclauses are awkward	Use an article ("a") after "Using" and before "case"	change Delete the word "Using" in heading Rationale: Conform with titles in list shown in 5.6	Change as noted.
PJM16	N	Substantive	5.6.1	The phrase "shall only be allowed to be read" is awkward	Use "shall be read only"	change to draft. Applies to 5.6.2, not 5.6.1 from "shall only be allowed to be read" TO "shall be read only" Rationale: Clarity	Change as noted.
PJM17	N	Editorial	5.6.1	The line "Example 24" is widowed	This line should start page 14	Move "Example 24:" to start of next page Rationale: Readability	Change as noted.
PJM18	N	Editorial	5.6.2	The phrase "2D reg array" appears here, but elsewhere (e.g. 5.7) as "two-dimensional reg array"	Use the longer form	Change "'2D reg array" to "two-dimensional reg array" Rationale: readability	Change as noted.
PJM19	N	Editorial	5.7	The phrases "edge sensitive" and "level sensitive" appear here, but elsewhere as "edge-sensitive" and "level-sensitive"	Use "edge-sensitive" and "level-sensitive" throughout	Replace throughout "edge sensitive" and "level sensitive" to "edge-sensitive" and "level-sensitive" Rationale: Compliance to LRM	Change as noted.
PJM20	N	Substantive	6.1	Notes both precede and follow normative text; moreover, there are multiple unnumbered notes and notes with the same number in this subclause	Consolidate notes to the end of the subclause and number appropriately	No change to draft. Rationale: LRM uses same style	No change.
PJM21	N	Substantive	6.1	The undefined phrase "Verilog standard" appears	Use "LRM"	Change "Verilog standard" TO "LRM". Rationale: Consistency and clarity. Verilog staNDARD IS NOT DEFINED IN DOCUMENT	Change as noted.
PJM22	N	Editorial	6.1.1, item a, note 2	The phrase "Contrary to popular belief" is condescending and unnecessary	Strike the phrase	Delete "'Contrary to popular belief" Rationale: Add no information, and is informal	Change as noted.
PJM23	N	Editorial	6.1.1, item b, note 5	There is a case mismatch in the phrase "does not nullifies"	Change to "does not nullify"	Change "does not nullifies" to 'does not nullify" Rationale: English usage.	Change as noted.
PJM24	N	Editorial	6.1.3	The initial sentence is unnecessary	Strike it	Change from "Here is one attribute that applies to ?nite state machine (FSM) extraction." TO "These attributes apply to ?nite state machine (FSM) extraction." Rationale: Explanation of purpose. More formal notation. Deleting is more severe than we intended.	Change as noted.
PJM25	N	Substantive	7	The introductory note should be normative	Make it so	No change to draft. Rationale: Note seems clear	No change.
PJM26	N	Editorial	7.7.5.1	The phrase "a *x*" should be "an *x*"	Make it so	Change from "Case expression in a casex statement shall not have a x or a z (or ?) value." TO "Case expression in a casex statement shall not have an x or a z (or ?) value." Rationale: English correction.	Change as noted.
PJM27	N	Editorial	7.8.2.1	The sentence "The keyword <code>_automatic_</code> is not optional." is not italicized	Italicize it	Agreed.	Change as noted.

	PJM28	N	Substantive	7.11.3	The note is unclear; for example, separate from what, and why the subjunctive?	Clarify the note	Delete note as it is confusing. Configurations are supported. Of course a synthesizable configuration would be one where instances point to library elements that are themselves synthesizable.	Change as noted.
	PJM29	N	Editorial	B.3 and following	"Mismatch" is misspelled as "mis-match"	Correct spelling	change "mis-match" to "Mismatch" Rationale: spelling	Change as noted.
	PJM30	N	Editorial	B.4	"b input" is misspelled as "b-input"	Correct spelling	Change :b-input" to "b input". Rationale: spelling	Change as noted.
	PJM31	N	Editorial	B.4	"2-input" should be spelled "two-input"	Correct spelling	change."2-input" to "two-input" Rationale: spelling	Change as noted.
	PJM32	N	Editorial	B.7	The last sentence, "Using Verilog functions should be done with caution" is awkward.	Change to "Verilog functions should be used with caution...."	Change "Using Verilog functions should be done with caution" TO "Verilog functions should be used with caution...." Rationale: English	Change as noted.
	PJM33	N	Editorial	B.8, B.9	A comma is needed after "the outputs would most likely go unknown"	Add the comma	Change "In a synthesized gate-level model, the outputs would most likely go unknown indicating a design problem." TO "In a synthesized gate-level model, the outputs would most likely become unknown, thus indicating a design problem." Rationale: Language structure	Change as noted.
	PJM34	N	Editorial	B.8, B.9	The phrase "For this reason, in general, the casex/casez statement should be avoided/is safe to use" is awkward	Rephrase to "For this reason, the casex/casez statement should, in general, be avoided/is safe to use"	Change "For this reason, in general, the casex/casez statement should be avoided/is safe to use" TO "For this reason, the casex/casez statement	Change as noted.
Fumiaki Nagao	FN01		Editorial		The terms "meta-comment" and "statically computable expression" which are defined at the clause 3, are not used as the words in this document. We can see "metacomments" and "statically computable" in this document. I think that these word have to be matched or these word definitions are needless.		Change in 6.3 "metacomments" TO "metacomments" Rationale: match definition. Also "Statically computed" is used in document.	Change as noted.
Karen L. Pieper	KLP01	Y	Technical	A.6.2, pg 74	Initial_construct is underlined indicating that it can be ignored, yet the Rom inferencing requires that it be analyzed.	Find a way to mark specifically those initial blocks which must be analyzed (an attribute?)	change underlined 'initial_construct ::= initial statement" to non-underlined "initial_construct ::= initial statement" // SUPPORTED ONLY FOR ROM DEFINITION Rationale: Initial statement supported for ROMs or CONSTANT definitions	Change as noted.
	KLP02	Y	Technical	5.6, pg 12	Because there are no examples of the logic_block attribute, I am unclear as to where it would apply (can I put it on the initial block?)		Change from: "If the logic_block attribute is used, then it shall imply that no ROM is to be inferred." TO: "If the logic_block attribute is used, then it shall imply that no ROM is to be inferred, and combinational logic be used instead." Rationale: Clarification	Change as noted.

KLP03	Y	Technical	5.6.2, pg 13	Can any variables be used in calculating memory values for a ROM? Can those variables be used in other synthesizable constructs after that? Is there any assumption that they obtain an initial value because of the calculation?		No CHange Rationale: Stated that restriction that the assignments to the ROM, which include data and address, shall be statically computable, as defined also in 5.6.1	No change.
KLP04	Y	Technical	5.6.2, pg 13	Can any variables be used in calculating memory values for a ROM? Can those variables be used in other synthesizable constructs after that? Is there any assumption that they obtain an initial value because of the calculation?	Eliminate initial block based Rom initiation	No CHange Rationale: Stated that restriction that the assignments to the ROM, which include data and address, shall be statically computable, as defined also in 5.6.1	No change.
KLP05	Y	Editorial	5.6.2, pg 13	On the last sentence on the page change the wording from: either of the attribute logic_block or .. to: either of the attributes logic_block or ...		Agreed.	Change as noted.
KLP06	Y	Editorial	6.1.2.c, pg 19	I do not see a logic_block example	Add one.	changein 5.6.2. example 24: FROM "(* synthesis, rom_block = "ROM_CELL XYZ01" *) reg [3:0] rom[0:7];" TO "(* synthesis, rom_block = "ROM_CELL XYZ01" *) reg [3:0] rom[0:7]; // (* synthesis, logic_block*) reg [3:0] rom[0:7]; rationale: Clarity	Change as noted.
KLP07	Y	Technical	6.1.4, pg 20	To ensure proper interpretation of the standard, pictures of the resulting logic after synthesis are needed for the async_set_reset comment		We believe the text provides sufficient information.	No change.
KLP08	Y	Technical	6.1.4.d, pg 21	Does the standard require that the indicated implementation be used? The language does not require that.	Clarify what actions are to be taken when the attribute is seen	Agree with author. The implementation is only a recommendation, not to be enforced. Add a note to this effect.	Add a note as stated.
KLP09	Y	Technical	6.1.4, pg 21	keep attribute: If a net is kept, what does that imply about the objects connected to the net? Do they need to be kept? Could the net be kept in the design and yet not be connected to anything?	Clarify the semantic and required actions for all cases.	The objects connected to the "keep" net do not need to be kept unless they have a "keep" attribute on them. It could happen that a "keep" net ends up to be just hanging without any connection to an object. In such a case, it makes sense for a synthesis tool to issue a warning about the same. Add a note to this effect.	Add a note as stated.
KLP10	Y	Technical	6.1.4.f, pg 22	What is the label used for? What does it apply to? Anything in the resulting netlist? If so, what in the resulting netlist? A net, the top level gate? How is it used? Given the current semantics it looks like the attribute can be ignored. Is that true?	Clarify the semantics.	The use of the label attribute is not defined by the standard. The attribute provides a standard way to label a sentence or an item. If a synthesis tool does not support use of labels (which is does not to support this standard), then it may safely ignore this attribute. Add a note to this effect.	Add a note as stated.

	KLP11	Y	Technical	6.1.4.g, pg 23	op_sharing is only applied to a module. Do you need to indicate what kinds of operators need to be shared? Is sharing allowed across always blocks. Is The absence of the attribute a requirement that the sharing not be performed?	Clarify the semantics	We do not want to restrict as to which operators need to be shared. Sharing may be done across always blocks if a synthesis tool can figure it out. In the absence of the attribute, a synthesis tool may still perform operator sharing. Add a note that includes the above lines.	Add a note as stated.
	KLP12	Y	Technical	6.1.4.l	test_port seems to be a misleading name for this functionality. probe_point seems to better get to the intent of the attribute, and does not lead to user expectation that this affects post production test.	Rename the attribute to probe_point	Recommend authors change.	Change as noted.
	KLP13	Y	Technical	7.3.1.5	Multiplication can be supported beyond those where one operand is a power of 2 and so on. Limiting the behavior to this for a tool to be compliant is not in the best interests of the user community.		Agree with author's recommendation. For multiplication, division and modulus, there is no restriction on what the operands may be.	Change as noted.
David R. Smith	DRS01		Comment		This caught me on the way out of the door to a foreign hiking trip. I am taking the draft with me but am abstaining now because I cannot guarantee internet access and there was no response to my request to mail in my vote		We are sorry the ballot caught you exactly in time for your vacation.	No change.
Stuart Sutherland	SS01	N	Editorial	Several	All comments are based on IEEE P1364.1/Draft 2.2, April 26,2002. They are all minor, and do not affect the technical correctness of the LRM. 1) Section 1.5, item d): the word "functionality" is misspelled as "functionality". 2) Section 5.2.2.1: example of code form does not consistently put angle brackets around user-defined names. 3) Section 5.2.2.1: examples 18, 19, 21 do not follow conventions set forth in 1.4 4) Section 5.6.1: example 23 does not follow conventions set forth in 1.4 5) Section 5.6.2: example 24 does not follow conventions set forth in 1.4 6) Section 5.6.3: example 25 does not follow conventions set forth in 1.4 7) Section 5.7: examples 26 and 27 do not follow conventions set forth in 1.4 8) Section 6.1: some text is in red. Usage of red is not defined in the conventions set forth in section 1.4. If color is to be used, it must be explained, and used consistently.		1) Agreed. Will fix. 2) ok. The problem seems to be only on the first line - will fix. 3) ok. Will fix. 4) ok - will fix. 5) ok - will fix. 6) ok - will fix. 7) ok - will fix. 8) Text will be changed to all black.	Make changes as identified.

				<p>9) Section 6.1, 3rd paragraph (not counting notes): The sentence "Additional semantics of a non-zero value is not defined by this standard." mixes plural noun with singular verb.</p> <p>10) Section 6.1: examples 28 and 29 do not follow conventions set forth in 1.4</p> <p>11) Section 6.1.1: examples are not numbered, the way previous examples are</p> <p>12) Section 6.1.1: examples do not follow conventions set forth in 1.4</p> <p>13) Section 6.1.3: attribute syntax for fsm_state is not numbered as in previous sections.</p> <p>14) Section 6.1.3: attribute syntax for fsm_state is not bolded as in previous sections.</p> <p>15) Section 6.1.3: attribute syntax for fsm_state uses a VHDL style comment.</p> <p>16) Section 6.1.3: example 30 does not follow conventions set forth in 1.4</p> <p>17) Section 6.1.4: examples 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, do not follow conventions set forth in 1.4</p> <p>18) Section 6.1.4, NOTE 2, just before example 39: the word "guarantee" is misspelled as "gaurantee".</p>		<p>9) Change "of" to "for". 10) ok -will fix. 11) Not numbered since part of a note. 12) ok - will fix. 13) Since there was only one attribute, there was no need to number this. 14) Agreed - will fix. 15) ok - will fix. 16) ok - will fix. 17) ok - will fix. 18) Typo - will fix.</p>	
--	--	--	--	---	--	--	--

					<p>19) Section 6.1.4: examples 31, 37, 40, should use nonblocking assignments, per guidelines set forth in 5.2.2 and 5.3</p> <p>20) Section 6.1.4, NOTE 1, at very end of section: the word "simulation" is misspelled as "simulatation".</p> <p>21) Section 6.2: some words are in red.</p> <p>22) Section 6.2: example 43 does not follow conventions set forth in 1.4</p> <p>23) Section 6.3: some words are in red.</p> <p>24) Section 7: some text is in red or blue colors, and some text is underlined.</p> <p>25) Annex A: some text is in red or blue colors, and some text is underlined.</p> <p>26) Annex B: All examples do not follow conventions set forth in 1.4</p>		<p>19) Agreed - will fix. Also noticed Example 17 needs to be corrected in a similar fashion. 20) Will fix typo. 21) will remove color - was unintentional. 22) ok - will fix. 23) will remove color. 24) This is intentional. Will add a section (f) to 1.4 explaining the coloring scheme. 25) See (24). 26) ok - will fix.</p>	
Atsushi Takahara	AT01	N	Technical	general, pg 98	<p>The constraints for synthesis (e.g. clock cycle, I/O to the outside or other modules timing, area issues) should also be described in Verilog for Synthesis.</p> <p>In the practical sense, the constraints are very important part of design as well as functional designs. The synthesis tools should know such constraints value.</p> <p>I strongly recommend that the synthesis language has the part of defining constraints for synthesis.</p>		<p>The synthesis constraints are being standardized in a different working group - DCWG. For status, check http://www.eda.org/dcwg. It would most probably be a separate standard.</p>	No change.
Scott E. Wecht	SEW01	N	Editorial	4.2.a, pg 5, line 8	grammar error "during period the"	change to "during the period"	Agreed to the suggested grammatical change.	Make change as recommended.
	SEW02	N	Editorial	6.1.1.b.5, pg 18, line 25	verb use error: "does not nullifies"	change to "does not nullify"	Agreed.	Make clarification as recommended.

	SEW03	N	Editorial	6.1.3, pg 19, line 27	formatting inconsistency; the synthesis attribute is not formatted the same as in other examples. It is in a larger font and is not bold. Also, invalid comment on same line.	change from: "(* synthesis, fsm_state [=<encoding_scheme>] *) -- applies to a reg" change to (where "_" underline implies bold font): "(* synthesis, fsm_state [=<encoding_scheme>] *)" " " "	Agreed. All synthesis attributes to be made cor	Make clarification as recommended.
	SEW04	N	Editorial	6.1.4.a.31, pg 20	Example 31 does not reinforce the consequence of its code due to the lack of a closing comment. The reader is left to guess at the synthesis outcome. A one line comment could be added following the last line of code (as done in many other examples).	Add a closing comment to Example 31 such as: "// reset and enable logic connect through the data input"	Good point. Agreed to add comment.	Make clarification as recommended.
	SEW05	N	Editorial	6.1.4.I.42, pg 25	two grammar errors in NOTES: 1-- "in verifier" and "It may also needed"	change to "in a verifier" and "It may also be needed"	Agreed to the suggested grammatical change.	Make clarification as recommended.
John Michael Williams	JMW01	Y	Editorial		1. Beginning in section 7, there are change markers (blue text) all over. These never should appear in a document to be balloted, because approval becomes ambiguous: Is the approval of the text as already changed, or of the text before the changes, it being OK just to promise the changes in a future version? 2. At several points in the text, characters are in red color, presumably for emphasis. This has ambiguous meaning and discriminates against users with color-vision deficiency.	I would change my vote to Affirmative with Comments if all red and blue colored text were changed to black, and if my other negative comment was fulfilled.	These are not change markers but color-coded so that PDF readers can easily identify the not supported constructs with the ignored constructs and with the keywords. A hard copy of the spec still shows the difference by highlighting, underlining and crossed-out. By coloring the spec, PDF users can take advantage of the PDF capabilities. No new information is provided by coloring or the lack thereof.	No change.
	JMW02	Y	Editorial		As another reason for a negative vote, Annex A includes BNF that should appear in section 7, if anywhere. For example, this Annex includes unsupported UDP syntax.	Delete the present Annex A.	The purpose of Annex A was to show the synthesis syntax subset as compared to the complete Verilog language in one place. The WG feels that Annex A is an important informative part of the standard.	No change.
	JMW03	Y	Editorial		Examples should not be numbered sequentially; they should be numbered within each section or perhaps named after the section. For example, the first example appearing in section 5.4 would be numbered, "Example 5.4.1". As it is now, adding or deleting one example near the middle of the document would require all subsequent examples to be renumbered, and all references to them to be edited and verified.	Make numbering scheme more localized.	This issue is more one of appearance. At this point, changing the example numbering may be hazardous. We recommend that it be changed in a future revision of this standard.	No change.

	JMW04	Y	Editorial		The usage, "supported" versus "unsupported" is not appropriate for a Std document.	I suggest the words, "supported" and "unsupported" be replaced with others. These terms are not proper in a standard document, which merely presents standards and does not "support" (implement) them. I recommend: "supported" --> "allowed" "unsupported" --> "forbidden". "Unsupported" features in software are those desired and allowed, but which the developer has not or could not implement in the current release. "Unsupported" constructs in a Std are those not allowed if the Std is correctly applied.	The specified terminology is prevalent in industry and using any other words will only lead to confusion. We do not see any problems with the words "supported" and "not supported", as these are being used in a very specific context (not the English context).	No change.
	JMW05	Y	Editorial	Pg 2, 1.3 terminology	The "Not supported" construct includes the wording, "RTL synthesis does not expect . . .". This anthropomorphism adds nothing to the definition and should be deleted.	I suggest, "The RTL synthesis tool shall fail upon encountering the construct, and the failure mode shall be undefined."	The remedy seems reasonable and we agree to reword as suggested.	Make clarification as recommended.
	JMW06	Y	Technical	Pg 3, 3 Definitions	"b) combinational logic" defined as "logic that does not have any storage device" doesn't seem valid, because most logic will store a value between clocks or for some period after power is removed; also, how can logic gates inside a device have their own "device" to store (whatever)?	I suggest this definition: Logic which always combines to settle to the same output values, given the same set of input values and independent of all previous history of input values. The definitions for d) edge-sensitive and g) level-sensitive also should be revised: They almost seem to be a joke the way they are stated now.	We feel the definition of combinational logic is appropriate, as the context of a storage device is clear in this document. However we agree that the definitions of d) and g) can be improved. Since the author has not suggested a remedy for this, we recommend changing the sentence in the paras to "...edge-sensitive to a clock, for example, ..." and "...level-sensitive to a clock, for example, ...".	Make clarification as recommended.
	JMW07	Y	Editorial	5, pg 6	Second line has a typo: "three-states" should be "three-state".		Agreed.	Make typo change as suggested.
	JMW08	Y	Technical	5.1, pg 6	The "shall" here seems misused: The first two paragraphs both have "shall", but they are mutually exclusive!	Both paragraphs probably should use "may", not "shall".	The English is not correct here as pointed out. Clearly the intent was that both always statement and continuous statement can be used to model combinational logic.	Change first two paras to.. modeled using a continuous

	JMW09	Y	Editorial	6.1.1, pg 17-18	The two Notes 1 on pp. 17 and 18 are phrased as "is providing"; proper usage would be "provides".		Agreed.	Make typo change as suggested.
	JMW10	Y	Editorial	6.1.3, pg 19	Second line of 1st paragraph should read, "... where the hardware ...".		Agreed . Change "then" to "the".	Make typo change as suggested.
	JMW11	Y	Editorial	7.2.6, pg 34	Says "Shall not be supported".	I suggest either change this to "Not supported" or change all the others to "Shall not be supported".	Change to "Not supported" to be consistent with others.	Make editorial change as suggested.
	JMW12	Y	Technical	Annex B.7, pg 96	In the first paragraph, second line says, "behaves like a function"	Should be "behaves like a latch". Also, I'm not so sure of the point here: If a synthesis tool inferred combinational logic not equivalent to a latch, it would be a tool bug, not something someone should avoid when writing Verilog according to the Std.	Agreed to the typo. Re: the second point, the standard explicitly states that functions are synthesized into combinational logic. This section is simply warning the user that functions ought to be used with caution.	Make typo change as suggested.
Mark Zwolinski	MZ01	N	Editorial	6.1.1, pg 18	"does not nullifies"	should read "does not nullify"	Agreed.	Make change as suggested.
	MZ02	N	Editorial	General	An observation. This standard has been strongly influenced by 1076.6. I think that's excellent. Given that the style of Verilog proposed here is similar to the style of VHDL proposed in 1076.1, I think a comment about 1076.6 would be useful in the introduction. At the very least this would give some confidence to anyone moving from one language to the other or to anyone working in a mixed-language environment.	Include something like this in the introduction (3rd paragraph): "This standard has a similar structure to that of IEEE 1076.6 (VHDL RTL Synthesis)."	While the commenter is correct in his observation, we feel that it is inappropriate to add such a statement as both standards may digress in the future.	No change.
Shalom Bresticker (non-voting member)	SB01	N	Editorial	General	(1) Ex 4, should add else part, for eg. "else out = 1'b1;" (2) Ex5, should be "in2". (3) 5.2.1.2, delete "clock" in "negative clock edge expression".	As suggested.	Agreed.	Make change as suggested.