

**Instructions:**

1. Fill in Project # & Draft #.
2. Each column has instructions. Hold your mouse over the column title for details.
3. Complete form.
4. Save form on your hard drive. Format for saving this file should include the project number and last name of balloter (ex. P1234\_Smith.xls)
5. Upload to Balloting Center using the Upload File site for this ballot.

**Project # P1076.6/D6 (Rev 1.2, dated 12/9/03)**

Name	Comment Number	Vote	Page	Type of Comment	Comment	Proposed Change	Resolution (for ballot resolution use only)
Bill Anker	BA01	Y	8, 6.1.3	Editorial	>In the "example of <code>async_condition</code>" example, the <code>elsif</code> should be shifted right and vertically aligned with the preceding <code>if</code>.	Easy edit	This change will be made.
	BA02		9, 6.1.3.1 (item b)	Editorial	>I had to read item "b)" about 20 times to understand what the last "unless the ..." clause implied. I'm still not sure why the "unless the ..." clause is needed. It needs more explanation.	Add a "see example X" or otherwise explaining the reason for allowing the <code>async_assignment</code> assignment to itself.	The reason is that a self-assignment retains the previous value, allowing a newly-clocked value to replace definite storage. A note will be added; a new example will be considered for a future revision.
	BA03		9, 6.1.3.1 (item e)	Editorial	Item "e)" would benefit from more explicit phrasing.	Should read "e) The <code>clock_edge</code> is present in the conditions only, and, the <code>clock_edge</code> always expresses the same edge of the same clock signal."	This rewording will be substituted.
	BA04		10, 6.1.3.1	Technical	Although the syntax appears to allow it, an example would make it clear, that sequential statements are allowed inside a process, but, outside the "if reset = '1' .. elsif rising_edge(clk)" part of the process which describes the edge-sensitive storage elements.	Add this example: <pre> AS_DIFF: process (CLOCK, RESET_1, RESET_2, SET, ASYNC_PRELOAD, A, Q) variable RESET : std_logic; begin   RESET := RESET_1 OR RESET_2;   if RESET = '1' then     Q &amp;:= '0';   elsif SET = '1' then     Q &amp;:= '1';   elsif ASYNC_PRELOAD = '1' then     Q &amp;:= A;   elsif rising_edge(CLOCK) then     Q &amp;:= D;   end if;   Qbar &amp;:= not Q; end process;  -- Variable RESET and signal Qbar are set within the process, but, -- outside the edge-sensitive storage description. -- Signal Q models an asynchronous reset/set rising edge triggered -- edge-sensitive storage element. The reset expression is RESET, the set -- expression is SET, and ASYNC_PRELOAD may be either a reset condition or a -- set condition according to the value of A.</pre>	Make recommended change.
	BA05		15, end of page	Editorial	Example 2, "Asynchronous reset modeling", describes 3 states, however, only 2 are labeled. Even though no setting occurs, the third state needs to be labelled for clarity.	Inbetween the last "wait until" and "end loop" lines, put the 2 lines: the "next RESET_LOOP" line -- state three No updates occurring (or put in an assignment)	In this machine, Reset, State 1, and State 2 are the only ones. Until reset or two clocks, State 2 remains. States of a machine without a designated state register are somewhat arbitrary. Indent the loop statements to the right.
	BA06		31, 7.1.2.1.2	Editorial	The last line of the preceding example erroneously became part of the title for 7.1.2.1.2.	Easy edit	This change will be made.
	BA07		109, middle of page	Editorial	I cannot find these 3 listed attributes (TYPE_CONVERSION, ZERO_EXTEND, and, SIGN_EXTEND) defined or referenced anywhere in this document.	If these 3 attributes are meant to be here, the document containing their definitions needs to be referenced.	These attributes were removed from the Std in D5.2. They were left in the Annex by oversight and will be removed from it.
Peter Ashenden	PA01	N		Editorial	In many places, grammar needs to be improved. It may help to have a staff editor advise.		The comment does not provide enough information for correction. A list of passages not identified in other Comments, or a list of grammatical constructs, will be requested from the commentator. Have contacted the author - does not have further time to identify exact issues. No change. Many of the grammar corrections are in subsequent Comments by this Balloter, and IEEE routinely reviews grammar and style of draft standards
	PA02		43, 7.1.9	Editorial	In many places, lists are formatted incorrectly. An example is on page 43, clause 7.1.9, in which the list of alternatives for the attribute value is formatted incorrectly.	Use the unnumbered-list style from the style template instead of using leading double-hyphens.	No change. According to 2001 IEEE SA Electronic Style Guide, 3.11.2, unordered list items each shall be preceded by an em dash or a double hyphen. Note that the IEEE typeset document will differ in appearance from the draft.
	PA03		11, 6.1.3.2(A)	Technical	This clause use the notation ":-" for definitions without indicating what the notation means.	Describe the notation in 1.4.	This notation had no technical meaning. The ":-" symbols on that page will be replaced by the word "is".

PA04	22, 6.3.2	Technical	Incorrect cross reference, shown as "Error! Reference source not found".	Change to "6.1.3.6".	This change will be made.
PA05	25, 6.5.1.2	Editorial	Typo	Change "variabl" to "variable".	This change in the Example 2 VHDL comment will be made.
PA06	26, 6.5.2	Technical	Incorrect declaration for ram_typ.	Change index range to 0 to 2**DEPTH-1.	This change will be made.
PA07	26, 6.5.2	Technical	The concurrent assignment to read the RAM is only sensitive to the address input. Thus, if new data is written with the address unchanged, the new data will not be reflected on the output. Also, the data array needs to be indexed with an integer, not a vector form of address.	Move the statement that reads the ram into the process immediately before the "end process" line. Change the index expression from "a" to "to_integer(unsigned(a))".	The type conversion change will be made. The latched data is not considered an issue in demonstrating a clocked write as distinguished from an unlocked read. Addition of a data-sensitive read process will be considered in a later revision.
PA08	27, 6.5.2	Technical	The index expression in the RAM read operation needs to be an integer, not a vector.	Change the index expression from "a" to "to_integer(unsigned(a))".	This change will be made.
PA09	28, 6.5.2	Technical	Same comment as that on page 26.	Move the statement that reads the ram into the process immediately before the "end process" line. Change the index expression from "a" to "to_integer(unsigned(a))".	The type conversion change will be made. The question of data-sensitive read will be considered in a later revision.
PA10	28, 7	Technical	This clause uses various terminology inconsistently to refer to decoration of parts of a design with attributes. IEEE Std 1076 uses the term "decorate" to refer to the specification of an attribute and associated value for an item.	Use the term "decorate" uniformly to refer to specification of attributes on items.	Rewording will be done. The word "object", which can be confused with a VHDL technical term, will be changed to "item". As in 1076-1993 5.1, decorate will be used where it disambiguates and does not impede generalization.
PA11	29, 7.1.1.1	Technical	The last sentence of the clause refers to logic for an instance, whereas it should refer to logic for the bound design entity.	Insert "the design entity bound to" before "that instance" in the last sentence.	The intention is to describe decoration of the component, not anything bound to it. The suggested rewording won't be done.
PA12	29, 7.1.1.3	Technical	The second sentence of the paragraph above the note refers to "the current module". VHDL does not have modules. Furthermore, no notion of "current" is defined. Finally, the sentence specifies that component instances be dissolved, whereas it should specify that design entities bound to the instances be dissolved.	Change the sentence to read "When this attribute is applied to a component declaration, the design entities bound to all instances of the component shall be dissolved."	"Current module" will be replaced by "immediately enclosing design unit". Binding is not relevant, because of decoration of the component, not the entity (block).
PA13	29, 7.1.1.3	Technical	The last sentence of the second paragraph above the note is unclear.	Change the sentence to read "When this attribute is applied to a component instance, the design entity bound to that instance shall be dissolved."	"Design entity" is ambiguous; the word "entity" is context-dependent in all of VHDL, unfortunately. In this context, it means "component". "Design entity" will be changed to "named entity".
PA14	30, 7.1.2.1	Editorial	In first sentence, referent of "these" is unclear.	Change first sentence to "Definitions for the purpose of the attributes specified in this clause."	The wording will be changed to "Definitions for the purpose of this subclause".
PA15	30, 7.1.2.1.1	Editorial	Formatting and grammar in fourth para.	Delete space at start of para (before "This"). Near end of first line, change "so it can be" to "so that the logic can be". In the third line, insert a space between "device." and "if". In the fourth line, change "terminals" to "terminal". In the fourth line, change "if" to ", provided that".	This attribute might decorate both set and reset of a component, so "pin(s)" will be changed to "pin(s)" and "terminals" to "terminal(s)". The other changes will be made, including "if" to "provided that".
PA16	30, 7.1.2.1.1	Technical	The paragraph preceding the notes does not specify what is to happen if the attribute decorates a construct that contains a mix of both asynchronously and synchronously set or reset storage devices. Should an error or a warning or neither be issued?	Determine the intended behavior and specify it in this paragraph.	An error should be issued, because of an asynchronous input, as described in the next-to-last sentence. No change will be made.
PA17	30, 7.1.2.1.1	Editorial	In note 2, a space is missing after the semicolon.	Insert a space after the semicolon.	This will be done.
PA18	31, 7.1.2.1.1/7.1.2.1.2	Editorial	The last line of 7.1.2.1.1 has been included in the heading of 7.1.2.1.2.	Correct formatting.	This will be done.
PA19	31, 7.1.2.1.2	Editorial	Formatting and grammar in fourth para.	Delete the space at the start of the para. In the first line, change "so it can be" to "so that the logic can be".	This will be done.
PA20	31, 7.1.2.1.2	Editorial	Grammar in the fifth para.	In the second line, change "if" to ", provided that".	This will be done.
PA21	31, 7.1.2.1.2	Editorial	In the seventh para, space missing between "described" and "set/reset".	Insert the space.	This will be done in the fourth parag.
PA22	31, 7.1.2.1.2	Editorial	In the note, a space is missing after the semicolon.	Insert a space after the semicolon.	This will be done.
PA23	32, 7.1.2.1.2	Editorial	Figure placeholders not deleted in two places.	Delete placeholders "Place Interconnection Figure . . . Here" in two places.	These will be deleted after IEEE substitution of the high-resolution versions. The embedded figures are for convenience of Draft review, only.
PA24	33, 7.1.2.2	Technical	As specified, the ONE_HOT and ONE_COLD attributes can only be used to identify a single collection of signals. The mechanism does not provide a way of specifying multiple collections, within each of which only one signal at a time is to be active.	It would be appropriate to use the VHDL group feature to identify a collection of signals. Separate collections could be identified as separate groups, each of which is decorated with an attribute.	This comment will be saved for consideration in a future revision.
PA25	33, 7.1.2.2	Technical	No constraints are specified on the type of signal that can be decorated with this attribute. Presumably it can only be applied to signals that represent bits.	Clarify the intention by specifying the types of signals that can be decorated with the attributes.	Bits clearly are implied by "high". "Collection of signals" will be changed to "collection of one-bit signals".
PA26	33, 7.1.2.2	Technical	The examples suggest that the attributes can decorate both scalar signals and vector signals. The semantics of decorating a vector are not specified. Nor are the semantics of decorating a collection comprising a mix of scalar and vector signals.	Clarify the intention by specifying the semantics.	The change in Comment PA25, plus Example 4, clarifies usage for signal vectors. Any collection of one-bit signals in the attribute specification, is decorable. Possible extension of this attribute to groups of scalars will be held for future revision.
PA27	33, 7.1.2.2	Editorial	Figure placeholder not deleted.	Delete placeholder "Place ... Figure 1 Here".	These will be deleted after IEEE substitution of the high-resolution versions.
PA28	34, 7.1.2.2	Editorial	Figure placeholder not deleted.	Delete placeholder "Place ... Figure 2 Here".	These will be deleted after IEEE substitution of the high-resolution versions.
PA29	35, 7.1.2.2	Editorial	Example 4 lists models that are similar, but does not describe what is being exemplified.	Add descriptive text indicating how the models are similar, how they differ, and how the attributes are used.	The word "similar" will be deleted.

PA30	36, 7.1.3	Editorial	Grammar in fourth para.	Change first line to "A synthesis tool may determine the implementation of a case statement that assigns to ...".	This will be done.
PA31	36, 7.1.3	Editorial	Missing "Example" heading.	Insert heading before para starting "In the example below...".	Example headers are omitted intentionally, because the code is meant to be read as part of the text.
PA32	36, 7.1.3	Editorial	Keyword "others" not bold in examples.	Change keyword to boldface in two places.	This will be done.
PA33	36, 7.1.4	Technical	In second line of last para on page 36, incorrect terminology.	Change "will be" to "shall be".	"Will be" will be changed to "is to be" to be sure imperative mood is not confused with IEEE grammar.
PA34	37, 7.1.4	Editorial	Missing "Example" heading.	Insert heading before para "Consider the following example:"	Example headers are omitted intentionally, because the code is meant to be read as part of the text.
PA35	37, 7.1.4	Editorial	Figure placeholders not deleted in two places.	Delete placeholders "Place ... Figure ... Here" in two places.	These will be deleted after IEEE substitution of the high-resolution versions.
PA36	37, 7.1.4	Technical	In first line of second last para on page, incorrect terminology.	Change "must" to "shall".	This change will be made.
PA37	37, 7.1.4	Technical	Second sentence of second last para on page is not normative.	Make the second sentence a note.	This will be done.
PA38	37, 7.1.4	Technical	In the last para, grammar and incorrect terminology.	Change the para to "If a function is decorated with the IMPLEMENTATION attribute, the function shall also be decorated with the RETURN_PORT_NAME attribute, the value of which shall be the name of the output port of the design that communicates the return value computed by the function."	This will be done.
PA39	38, 7.1.4	Editorial	Missing "Example" headings.	Insert heading before two paras "In the example below...".	Example headers are omitted intentionally, because the code is meant to be read as part of the text.
PA40	38, 7.1.4	Editorial	In first para, grammar.	Change to "In the example below, the function AND_OR_INVERT is mapped to the component AOI. The RETURN_PORT_NAME attribute specifies that the value computed by the function is mapped...".	This will be changed to "In the example below, the function AND_OR_INVERT is mapped to the component AOI. The RETURN_PORT_NAME attribute specifies ...".
PA41	38, 7.1.4	Technical	The specification of the IMPLEMENTATION attribute decorating predefined subprograms in paras 3 to 6 of the page is unclear and incomplete. Can the attribute only be applied to expressions involving operators from std.standard, or from other packages (e.g., ieee.numeric_bit, etc)? The reference to unary operations presumably means expressions involving exactly one operator application, that being a unary operator; but that is not clear. Similarly, reference to a binary operation presumably means expressions involving exactly one binary operator. Further, no constraints are specified on the form of expression. Must it only involve operators in prefix/infix form? What if the operators are invoked using function-application form? What does that imply for order of mapping of operands to entity ports?	The intention needs to be clarified by a more complete and precise specification. This reviewer cannot provide detailed edits, as the intent is not clear.	The word "simply" will be deleted at the end of p. 36. The second paragraph from the bottom of p. 37 will be, "The port names and directions of the implementation cell shall be matched 1-1 to the formal parameters of the subprogram." The second sentence of this parag. will become a NOTE. The last parag. on p. 37 will be, "A function decorated with the IMPLEMENTATION attribute also shall be decorated with RETURN_PORT_NAME to specify the cell output port." In the first example on p. 38, "end procedure" will be changed to "end function". The first sentence after this example on p. 38 will be deleted, and the second sentence will be changed to, "When the IMPLEMENTATION attribute ...". In the new first sentence, "indicates" will be replaced by "specifies". A new second sentence will be added, "The operand designators shall match by name the ports of the specified cell." In the new first sentence, "indicates" will be replaced by "specifies". A new second sentence will be added, "The operand designators shall match by name the ports of the specified cell."
PA42	38, 7.1.5	Technical	The first part of the first para is vacuous; and the second part of the para is dealt with by clause 5.	Strike the first para.	This will be done.
PA43	39, 7.1.5.1	Editorial	In third line of fourth para, missing space between "a" and "specific".	Insert the space.	This will be done.
PA44	40, 7.1.6	Technical	In the fourth para, incorrect terminology ("infers" instead of "shall infer". Further, the second clause of the first sentence of this para is tautological. If the synthesis tool is required ("shall") to produce combinational logic and doesn't, it is implicitly an error.	Change "infers" to "shall infer". Delete "; in the presence ... error".	Infers changed to implies; also, second "shall" is not necessary because the first one makes the rest mandatory. The second clause must be seen in context of second statement. There is an exception to second clause, which is explained by example. The second clause of first statement and second statement must be re-written. The rewrite is provided with PA45.
PA45	40, 7.1.6	Technical	The clause attempts to specify normative behavior by example.	Change the text from "The attributed object..." to the end of the para to be a normative specification. This reviewer cannot provide specific text as the intention is not clear. The example and associated text can stand as a non-normative illustration.	Change "; in the ...error. The attributed ... ." to - Presence of this attribute on an object which models a sequential logic shall be an error; the only exception is a latch model, where feedback path is modeled using identity statement(s). Such a latch shall be synthesized as combinational logic with a feed-back path.
PA46	40, 7.1.7	Technical	In the fourth para, incorrect terminology and undefined term. Presumably a "load" of a signal is a place where the signal is read.	Change "are to be combined" to "shall be combined". Either use defined terms instead of "loads", or add a definition to Clause 3.	Agreed with comment. Term 'load' or clock network identification must be defined. Replace second sentence with: "drivers or loads of" changed to "ports connected to".
PA47	40, 7.1.7	Editorial	The first three sentences of the last para on the page are a repetition of sentences from the previous para.	Strike the sentences.	The first three sentences of last paragraph shall be stricken off. Last and last but one paragraph shall be combined.
PA48	42, 7.1.8	Technical	The attribute should decorate a type, not a signal or variable.	In the para starting "Attributed object", change "signal, variable" to "type, subtype".	Agreed with Comments.
PA49	42, 7.1.8	Technical	The last para before the note should also allow the attribute to decorate a subtype of an enumeration type.	Append to the sentence, "or a subtype of an enumeration type".	Agreed with Comments.
PA50	42, 7.1.8	Editorial	The last para before the note would be better placed immediately after the para starting "Attributed object".	Move the para.	This is only attribute for which this information is explicitly mentioned. Once we have done PA48, this sentence is not needed.
PA51	42, 7.1.8	Editorial	In the fourth para, grammar. Also, the specification that the attribute value be a string is a repetition.	Change the first two sentences to "The value of this attribute shall specify the encoding of the enumeration type literals. The attribute value shall be made up of ... spaces."	Agree with comments.
PA52	42, 7.1.8	Technical	In fifth para, incorrect terminology.	Change "are to be ignored" to "shall be ignored".	Agree with comment

PA53	42, 7.1.8	Editorial	In the text starting "Given the following...", incorrect grammar. The subsequent text "The attribute specification ... literals." should be the completion of the sentence starting "Given the following...", not a sentence in its own right. The complete sentence then should introduce the code block on the line after.	Change "The attribute specification ... literals." to "the attribute specification ... literals, as follows:".	Agree with comment
PA54	43, 7.1.9	Technical	In the para starting "Attributed object", the attribute should presumably decorate a signal or variable, not a type or subtype.	Change "type, subtype" to either "signal" or "signal, variable", depending on intention.	Agree with comment. It can be used on signal, variable, type and subtype.
PA55	43-44, 7.1.9	Technical	The specification indicates that the item decorated by the attribute can be of an enumeration type, but does not specify what other types may be used.	Add a para specifying what types of objects can be decorated with this attribute.	In resolution of PA54, this is taken care of.
PA56	43, 7.1.9	Technical	In the fifth para, incorrect terminology.	Change the para to "The value for this attribute for an FSM with N states shall be one of".	Agree with comment.
PA57	43-44, 7.1.9	Editorial	It would also be clearer to introduce the ENUM_ENCODING-like form of attribute value as part of the list of allowed values, rather than suggesting that it be a special case.	Add to the list: "if the attributed object is of an enumeration type or a subtype of an enumeration type, a string of the form specified for the ENUM_ENCODING attribute". Change the para that crosses pp 43 and 44 to a note describing usage of the two attributes.	With resolution PA54 and add the suggested modification.
PA58	43, 7.1.9	Technical	In the list, typo in two attribute names.	Change "ONEHOT" to "ONE_HOT" and "ONECOLD" to "ONE_COLD".	Agree with comment
PA59	43, 7.1.9	Technical	In the example, it is not legal VHDL to decorate a signal with the same attribute multiple times.	Change the example to have four signals, STATE1, STATE2, STATE3 and STATE4. The first attribute specification should decorate STATE1, the second STATE2, etc.	Agree
PA60	44-45, 7.1.10	Technical	This clause refers to the "others statement" of a case statement and the "else statement" of an if statement. Neither are VHDL statements; they are clauses of enclosing statements.	Change "others statement" to "others clause" and "else statement" to "else clause".	Agree
PA61	44, 7.1.10	Technical	The second and third sentences of the fourth para are very poorly expressed, obscuring the intended meaning.	Change to "The FSM_COMPLETE attribute shall decorate a signal or variable that represents the state vector of a finite-state machine. If the attribute value is TRUE, those states in the synthesized machine for which no transition is specified shall be augmented with a transition to the same state as that specified in the default clause."	The proposed rewrite, technically differs from text in the document. It does not advocate optimization of unreachable states. The text has been reworded and an explanatory note has been added.
PA62	44, 7.1.10	Technical	The text describing the example is inconsistent with the specification of the attribute. The state machine has four states, each of which has transitions specified. According to the specification for the attribute, should the state machine be synthesized with more than four values for the state vector, the additional states should get the transition specified in the others clause (namely, to S0). The fact that S3 is unreachable is not relevant (unless the normative specification for the attribute is incorrect).	Delete the line of example code that specifies the FSM_STATE attribute for STATE. Change the descriptive text to "In the above example, transitions are specified for all states represented by STATE. However, should the synthesis tool generate a state encoding that includes additional values beyond those for S0, S1, S2 and S3, the additional states will have transitions to S0, as specified in the others clause."	The proposed rewrite technically differs from existing text. To clarify, change the encoding values to 3 bit values. See PA61 resolution as well.
PA63	45, 7.1.10	Technical	The para after the example text refers to the "scope" of the FSM_COMPLETE attribute. Technically, the scope of the attribute is part of the declarative region in which the attribute is declared, augmented with other parts of the model, as specified in IEEE Std 1076. This is probably not what is intended.	Change the first sentence to "It shall be an error if a signal or variable is decorated with the FSM_COMPLETE attribute, the attribute value is TRUE and the signal or variable is assigned in more than one case or if statement."	Agree with the comment.
PA64	45, 7.1.10	Technical	The para after the example text specifies that it be an error if the state signal or variable is assigned in more than one case or if statement. However, a state object will often be assigned in multiple if statements, each of which is embedded within an alternative of a case statement and specifies alternate transitions from the a state depending on input conditions.	Revise the paragraph to specify intended error conditions. This reviewer cannot provide specific text, as the intention is not clear.	Agree with comment here. Resolution of PA63 could be further refined to incorporate case or if statements which are directly enclosed within a block. The explanation shall be clarified.
PA65	45, 7.1.11	Technical	Fourth para, incorrect terminology	Change "identifies" to "shall identify".	Agree
PA66	45, 7.1.11	Technical	Fourth para, incorrect term	Change "tokens" to "values"	Agree
PA67	46, 7.2	Technical	First para, incorrect terminology	Change para to "Two metacomments shall provided for conditional synthesis control:".	Agree
PA68	46, 7.2	Technical	The para after the letter-numbered list inadequately specifies use of the metacomments.	A synthesis tool shall ignore any VHDL code after an "RTL_SYNTHESIS OFF" metacomment and before the first subsequent "RTL_SYNTHESIS ON" metacomment.	Agree
PA69	47, 8	Editorial	The last item in an unnumbered list should not be terminated with a period.	Delete the periods.	Agree - this will be done.
PA70	48, 8.1.1.2	Technical	File declarations should be ignored, rather than not supported	In the production for entity_declarative_item, change file_declaration from struck-through to underlined. Delete file_declaration from the list of not-supported categories and include it in a list of ignored categories.	Agree.
PA71	48, 8.1.1.2	Technical	If the ONE_HOT and ONE_COLD attributes are revised to decorate groups, change group_template_declaration and group_declaration from not supported to supported.		This will be held for possible future revision.

PA72	49, 8.1.2.1	Technical	File declarations should be ignored, rather than not supported	In the production for block_declarative_item, change file_declaration from struck-through to underlined. Delete file_declaration from the list of not-supported categories and include it in a list of ignored categories.	Agree
PA73	52, 8.2.1.1	Technical	A subprogram parameter can have different actuals in different calls.	Change "the actual parameter" to "the associated actual parameter in each call to the subprogram".	Agree
PA74	52, 8.2.1.1	Technical	Restricting the actual to an identifier is too restrictive. Would a static name be sufficient?	Change "identifier" to "static name" or other as intended.	Agree
PA75	52, 8.2.1.1	Technical	Reference to "tie" value should only apply to unassociated in-mode parameters.	Change "formal parameter" to "formal parameter of mode in".	Agree
PA76	53, 8.2.2	Technical	The requirement that a use clause "only reference the selected name of a package" is unclear. What is intended to be precluded?	Revise to make the intention clear.	Agree. Rephrase to support all forms of use clause.
PA77	55, 8.2.5	Technical	In sentence referring to constant declarations, incorrect terminology.	Change "A constant declaration must include" to "A constant declaration shall include".	Agree
PA78	55, 8.2.5	Technical	The requirement that a use clause "only reference the selected name of a package" is unclear. What is intended to be precluded?	Revise to make the intention clear.	See resolution of PA-76.
PA79	55, 8.2.6	Technical	The requirement that a use clause "only reference the selected name of a package" is unclear. What is intended to be precluded?	Revise to make the intention clear.	See resolution of PA-76.
PA80	57, 8.3.1.2	Technical	The para starting "It is recommended" mixes recommendations and requirements. It uses incorrect terminology for the recommendations.	Split the paragraph into two, with the second paragraph starting "The synthesis tool shall support integer types...". In the first para of the two, change two occurrences of "shall" to "should".	Agree
PA81	57, 8.3.1.3	Technical	Primary unit and secondary unit declarations should be not supported rather than ignored, since they only occur in physical type declarations, which are not supported.	Change "primary_unit_declaration" and "secondary_unit_declaration" from underlined to struck-through.	Agree
PA82	57-58, 8.3.1.3	Technical	Physical literals of type TIME can occur in initial-value expressions of declarations of objects of type TIME; in those contexts they are ignored.	In singleton list of ignored constructs, after "an after clause" append "or in the initial-value expression of a declaration of an object of type TIME". In the list of not-supported constructs, change "and within an after clause" to "occurring where ignored". In last sentence of clause, change "may occur only ... ignored" to "may occur only within ignored constructs, for example, the after clause".	Agree
PA83	59, 8.3.2.1	Editorial	In para after list of supported constructs, clarification of referents	Change "the discrete range" to "a discrete range". Change "The element subtype" to "An element subtype".	Agree
PA84	59, 8.3.2.1	Technical	In para after list of supported constructs, the scalar type must specifically be an integer or enumeration type. The parens suggest exemplification.	Change "a scalar (integer or enumeration) type" to "an integer or enumeration type".	Agree
PA85	59, 8.3.2.1	Technical	In the third para after the list of supported constructs, the discrete subtype indication need not be in the form of a name.	Change "name" to "denote".	Agree
PA86	61, 8.4.1	Editorial	In the production for type_definition, formatting error	Remove the underlining from the leading space and " " symbol before "protected_type_definition".	Agree
PA87	61, 8.4.1	Editorial	In para after list of ignored constructs, grammar	Change "definition" to "definitions" in two places.	Agree
PA88	63, 8.4.3.1	Editorial	In para after ignored constructs at top of page, referent of "it" is ambiguous.	Change "where it shall have" to "where the declaration shall have".	Agree
PA89	66, 8.4.4	Technical	Not all user-defined attribute declarations are ignored. Those specified in clause 7 should not be.	Change "User-defined attribute declarations" to "User-defined attribute declarations other than declarations of attributes specified in clause 7".	Agree. This shall be changed to, "User-defined attribute declarations other than those of the synthesis-specific attributes in this standard".
PA90	66, 8.4.6	Technical	If the ONE_HOT and ONE_COLD attributes are revised to decorate groups, change the categories in this clause from not supported to supported.		No change. To be considered in future.
PA91	66, 8.4.7	Technical	If the ONE_HOT and ONE_COLD attributes are revised to decorate groups, change the categories in this clause from not supported to supported.		No change. To be considered in future.
PA92	67, 8.5.1	Technical	If the ONE_HOT and ONE_COLD attributes are revised to decorate groups, change entity class "group" in this clause from not supported to supported.		No change. To be considered in future.
PA93	67, 8.5.1	Technical	In the list of not-supported constructs, referring to "use" of user-defined attributes is not clear. presumably it means reading attribute names for user-defined attribute names.	Change "use of user-defined attributes" to "reading of names of user-defined attributes".	Agree

PA94	73, 8.7.3.1	Technical	Physical literals of type TIME can occur in initial-value expressions of declarations of objects of type TIME; in those contexts they are ignored.	In the list of not-supported constructs, change "and within an after clause" to "occurring where ignored". In last sentence of clause, change "may only appear ... ignored" to "may occur only within ignored constructs, for example, the after clause".	Analysis: Comment also potentially impacts section 8.3.1.3, p58.  Recommendation: In section 8.3.1.3, under ignored, change: physical_literal of type TIME within an after clause. to: physical_literal of type TIME within an ignored construct, for example, an after clause. In section 8.3.1.3, change: References to objects and literals of type TIME may occur only within constructs such as after clauses and shall be ignored. To: References to objects and literals of type TIME may occur only within ignored constructs (such as an after clause). In section 8.7.3.1 not-supported list, change: physical literal, except of type TIME and within an after clause to: physical literal, except of type TIME occurring where ignored. In section 8.7.3.1, last sentence, change: Physical literals of type TIME and floating point literals may only appear in after clauses, where they shall be ignored. to: Physical literals of type TIME and floating point literals may occur only within ignored constructs, for example, an after clause. See also PA82.
PA95	76, 8.8.1	Technical	This clause is inconsistent about whether sensitivity clauses and sensitivity lists are supported, ignored or not-supported.	Clarify intention and revise to be consistent.	Analysis: Agree Resolution: Mark sensitivity lists and sensitivity clauses as supported.
PA96	76, 8.8.1	Technical	In second to last para of clause, incorrect terminology	Change "are supported" to "shall be supported".	Analysis: Agree Resolution: make recommended change
PA97	76, 8.8.1	Technical	The last paragraph should probably not be normative.	Make the para a note.	Resolution: Agree. Make it so
PA98	78, 8.8.6	Editorial	Typo	Change "8.4.3.2b)" to "8.4.3.2(b)".	Resolution: Agree. Make it so
PA99	79, 8.8.7	Technical	The last para of the clause should probably not be normative.	Change to a note	Resolution: Agree. Make it so
PA100	79, 8.8.8	Technical	The first para after the list of supported constructs should probably not be normative.	Change to a note	Resolution: Agree. In addition expand case note to match if note: If a signal or variable is assigned under some values of the conditional expressions in the case statement, but not for all values, storage elements may result; see 6.1 and 6.2.
PA101	79, 8.8.8	Technical	In the second para after the list of supported constructs: incorrect terminology	Change "may never occur" to "shall never occur".	Analysis: Offending statement is: "If a metalogical value occurs as a choice, or as an element of a choice, in a case statement that is interpreted by a synthesis tool, the synthesis tool shall interpret the choice as one that may never occur."  Resolution: Already one shall, I am not sure what is correct in this case. "May" is part of the issue. Perhaps the following is better: If a metalogical value occurs as a choice, or as an element of a choice, in a case statement that is interpreted by a synthesis tool, the synthesis tool shall interpret the choice as one that does not occur.
PA102	79, 8.8.8	Editorial	In note 2, space missing.	Insert space between "choice" and "including".	Resolution: Agree. Make it so
PA103	82, 8.9.2	Technical	If the ONE_HOT and ONE_COLD attributes are revised to decorate groups, change group_template_declaration and group_declaration from not supported to supported.		No change. This will be considered for a future rev.
PA104	82, 8.9.2	Technical	Not all user-defined attribute declarations are ignored. Those specified in clause 7 should not be.	Change "user-defined attribute declarations and their specifications" to "user-defined attribute declarations and their specifications, other than declarations and specifications of attributes specified in clause 7".	Analysis: Attributes are marked as supported. 8.9.2 statement under ignored states: "user-defined attribute declarations and their specifications." Furthermore, in 8.9.2 p92 the 2nd to last paragraph it states: Attribute declarations and specifications as described in 7.1 shall be the only ones supported. This aligns with Peter's comments.  Peter's statement belongs somewhere, but where is the appropriate place, probably not here, otherwise, it also belongs everywhere else attributes can be specified  Sections with comments on attributes support/ignored: 8.5.1 Under not supported: use of user-defined attributes. 8.9.2 Under Ignored Recommendation: Syntax wise, attributes need to be marked as supported. For the ignored statement there are two choices: 1) it goes in all sections referring to attribute specification 2) it goes in section 8.5.1, Attribute Specification The ignored statement = "User-defined attribute declarations and their specifications, other than declarations and specifications of attributes specified in clause 7, shall be ignored". Choice (2) is preferred. Add an ignored category to 8.5.1 as follows: ===== Ignored: -- User-defined attribute declarations and their specifications, except those of the synthesis-specific attributes of Clause 7 =====

	PA105		83, 8.9.2	Technical	The requirement that a use clause "may only contain the selected name of a package" is unclear. What is intended to be precluded?	Revise to make the intention clear. Also, use "shall" instead of "may".	Analysis: There is a significant consistency issue here. Sections that contain "use_clause" 8.1.1.2, 8.1.2.1, 8.1.3, 8.1.3.1, 8.2.2, 8.2.5, 8.2.6, 8.9.2, 8.10.4, 8.11.3. No restrictions on use_clause: 8.1.1.2, 8.1.2.1, 8.1.3, 8.1.3.1, 8.10.4, 8.11.3 has "shall" 8.2.2, 8.2.5, 8.2.6, has "may" 8.9.2. Resolution: These must be consistent. Delete may statement in 8.9.2. Delete all statements about shall.
	PA106		83, 8.9.2	Technical	The last sentence specifies that aliases shall not be supported, but 8.4.3.3 specifies that they shall be supported.	Change "file objects, access objects (variables of access type) and aliases" to "file objects and access objects (variables of access type)".	Resolution: Agree. Make it so Revised text should read: Use of file objects and access objects (variables of access type) in a process shall not be supported.
	PA107		83-85, 8.9.5	Editorial	Since the structure of clause 8 mirrors that of IEEE Std 1076, the production for options should be moved to page 83 and not duplicated in 8.9.5.1 and 8.9.5.2.		Resolution: Agree Move options to section 8.9.5 following the production for concurrent_signal_assignment_statement
	PA108		83-85, 8.9.5	Technical	The production for options has "options" shown as ignored, but leaves the optional guarded keyword as supported. This is inconsistent.	Remove the underlining from the category name "options". In the list of supported constructs on page 83, change "options" to "guarded in options".	Resolution: Agree. Make it so.
	PA109		88, 8.13	Technical	Real literals are ignored in initial value expressions of time-typed object declarations (which are ignored). The first para also uses incorrect terminology.	Change first para to "Real literals shall occur only in contexts where they are ignored, for example, in after clauses."	Analysis: Offending statement is already covered correctly in section 8.3.1.4. Floating point types. Resolution: Delete the following sentence from 8.13 Real literals are allowed only in after clauses.
	PA110		93, A	Editorial	Ensure this annex is updated to reflect any changes made in clause 8.		Resolution: Agree. Change Annex occurrences of: sensitivity lists & clauses, file declaration, delay_mechanism, after_time_expression
John Aynsley	JA01	N		TECHNICAL	In my opinion, much of what is new in 1076.6/D6 will be perceived in the user community as being bad coding style, violating the rules of best practice in the industry and devaluing the investment made in in-house coding standards. In my view, it will be perceived by EDA tool vendors as an irrelevance in a mature market where de facto standards dominate. Neither the user community nor the tool vendors will want to re-invest to support a standard that does not add any value. On the other hand, the extensions to 1076.6-1999 to support VHDL-93 are positive.	None.	Reject. Agree with the observation that this standard allows bad coding styles. However, this standard is not about creating good coding styles. To create a standard that enumerates only good coding styles would be time prohibitive and would be out of date shortly after it was written. In addition each vendor would have to do lots of work to determine how to support the coding styles. Rather this standard enumerates what "should be" synthesizable through a small set of simple rules. You will note that the coding styles you wished for in the review of the previous version of this standard are now supported.
	JA02		10 Incorrect Example 6	TECHNICAL	The meaning of "lacks an assignment under <sync_condition>" is unclear, and it is unclear why this example "violates Rule a" (the signal or variable has a <sync_assignment>). Q <= D; appears to be a <sync_assignment> - it is "controlled by" the clock edge rising_edge(clk)	Either clarify the definitions in 6.1.3 and 6.1.3.1, or change the example.	Analysis: To understand this, I had to review the definition of <sync_assignment>. An assignment to a signal or variable which is controlled by <clock_edge> in all execution paths. Resolution: Change comment on example 6 from: Lacks an assignment under <sync_condition>; violates Rule a To: Violates rule a. Is not a <sync_assignment> since it is not controlled by a <clock_edge> in all execution paths.
	JA03		11 6.1.3.2 para C)	EDITORIAL	Grammar used gives ambiguous/incorrect meaning.	1"...is not specified explicitly (viz., A) or IS SPECIFIED implicitly (viz., B)."	Recommendation: Change: This includes the following wait statements in which the clock edge is not specified explicitly (viz., A) or implicitly (viz., B). To: This includes forms of wait statements from either A or B where a clock edge is not specified either explicitly or implicitly.
	JA04		11 6.1.3.2 para C)	TECHNICAL /EDITORIAL	Missing semicolons at end of wait statements.		Analysis: Sections A and B have wait statement templates that end with ";". Section C should be consistent Resolution: Add ";" to the end of templates in C.1, C.2, C.3
	JA05		12 Edge-sensitive storage shall be modeled for a signal or variable...	TECHNICAL	Consider the following example:  process variable v1, v2: std_logic; begin wait until rising_edge(clock); v1 := a; v2 := v1 and b; q <= v2 or c; end process;  Variable v1 and v2 meet the prescribed conditions, but it would be wrong to say that they modeled edge-sensitive storage.	The treatment of synchronous assignments to variables need to be more sophisticated to accurately reflect the functional behavior of VHDL.	Analysis: In section 6.1.3, there is the following statement: "A variable read on a clock edge previous to one on which it is written shall represent a storage element." Note that "previous" typically implies "the one before" and for a variable a register should be created if the variable was written any time in the past and not just the previous clock edge. In section 6.1.3.1, there is the following statement: Edge-sensitive storage shall be modeled for a signal or variable assigned inside a process with sensitivity list when all the following apply: ... There is nothing in 6.1.3.1 that says that the above condition of 6.1.3 is also required to be met. Resolution: Part 1) Add the following condition below to 6.1.3.1: I) For a variable, the value that was written during a given clock edge is read during a future clock edge. Part 2) In section 6.1.3, remove the statement about variables: A variable read on a clock edge previous to one on which it is written shall represent a storage element.

JA06	12 Example "Using the transformation in form C above..."	EDITORIAL	The process is in form A2 (not form C) whilst the transformations themselves are unnumbered.	Number the transformations.	Agree. Number the three transformations to T1, T2, and T3. Change sentence to "Using the transformations described above, the goal . . .".
JA07	13 6.1.3.3 "...shall be taken as the functional clock."	TECHNICAL	The text seems to assume that each process has a unique functional clock.	This assumption should be stated explicitly, and the handling of simulation and synthesis semantics for multiple clocks mentioned, bearing in mind the cases of scan flip-flops, dual edge flip-flops and multi-phase clocking schemes. I don't suggest that there is necessarily anything wrong with the assumption, just that it needs to be made more explicit to avoid mis-interpretation and confusion.	Analysis: In 6.1.3.3., there is a sentence that states, "The signal in the first clock edge expression (textually) shall be taken as the functional clock." In my mind this is very explicit. Since this is an implication, perhaps a note would be in order. Resolution: Add note: The determination of the functional clock is made on a process by process basis. Make sure to code the functional clock first.
JA08	15 Example 1	TECHNICAL	Using type std_logic_vector for variables A, B and M violates section 4 of this standard "No array type, other than those listed in (e) and (f), shall be used to represent signed or unsigned numbers."  Assignment M := B*"0000"; will cause a run-time error. The array widths are wrong.  Variable COUNT and the assignment count := 0; are redundant.	You might as well tweak it such that it actually does a multiply, as follows:  process variable A, B: unsigned(7 downto 0); variable M : unsigned(8 downto 0); begin wait until clk = '1'; if start = '1' then A := "0000" & A_PORT; -- init state B := B_PORT & "0000"; done <= '0'; M := (others => '0');  for count in 0 to 3 loop wait until clk = '1'; if A(0) = '1' then -- compute state 1 M := M + B; end if; wait until clk = '1'; A := shift_right(A, 1); -- compute state 2 M := shift_right(M, 1); end loop;  wait until clk = '1'; M_OUT <= M(7 downto 0); -- final state M_OUT = A_PORT * B_PORT done <= '1'; end if; end process;	Analysis: All observations made by reviewer are correct. Both solutions need to be simulated. One thing I observe is that the done flag is on from end of one cycle and remains on until one clock after the start of a new cycle (this may be erroneous). In addition, a shift left algorithm may be more efficient. All the additions are done with an 8 bit + 8 bits rather than 8 bits + 4 bits resource. I ran the provided code and it did not seem to work right. Below is a shift left algorithm that does the 8 bit + 4 bit suggested above. Resolution: Replace existing example with the following tested code: library ieee ; use ieee.std_logic_1164.all ; use ieee.numeric_std.all ; Entity Mult is port (clk : in std_logic ; start : in std_logic ; done : out std_logic ; A : in unsigned (3 downto 0) ; B : in unsigned (3 downto 0) ; Y : out unsigned (7 downto 0) ) ; end Mult ; Architecture ImplicitFSM of Mult is signal intY : unsigned(7 downto 0) begin MultProc : process begin wait until clk = '1'; if start = '1' then done <= '0'; intY <= (others => '0'); for j in A'range loop wait until clk = '1'; if A(j) = '1' then intY <= (intY(6 downto 0) & '0') + B ; else intY <= (intY(6 downto 0) & '0') ; end if; end loop; done <= '1'; end if; end process; Y <= intY ; -- final state Y = A * B end ;
JA09	18 6.2.1.1	TECHNICAL	The rules for level-sensitive storage are incomplete	Add another rule as follows:  ;-) There exists at least one explicit assignment to the signal (or variable) within the process.	Analysis: This working is already in condition a: a) There is no execution path of the process with <clock_edge> expression as condition, under which the signal (or variable) has an explicit assignment. Resolution: No Change
JA10	21 6.2.1.3 Example	TECHNICAL	First line missing from architecture in example	Insert the following:  architecture CONCUR_SUB of level_sensitive is	Resolution: Add the architecture statement as suggested
JA11	22 6.3.2 Example	TECHNICAL /EDITORIAL	Syntax error in example. Missing keyword "block" after "end".		Analysis: Agree (hopefully we will fix 1076 in 200X) Resolution: Replace end ; with end block ;
JA12	31 Example	TECHNICAL /EDITORIAL	Syntax error in second line of example	Replace colon with semicolon following the word "boolean"	Resolution: Agree. Fix it.
JA13	31 7.1.2.1.2	EDITORIAL	The para number (7.1.2.1.2) has gotten mixed up with the VHDL text of the preceding example	Replace with:  end process; end architecture EdgeSensitive;  7.1.2.1.2 For Level-Sensitive Storage Elements	Resolution: Agree, Fix it.
JA14	32 Example	TECHNICAL /EDITORIAL	Syntax error in second line of example	Replace colon with semicolon following the word "boolean"	Resolution: Agree. Fix it.

	JA15		33 7.1.2.2	TECHNICAL	The definition of one-hot as given here ("in which only one signal in the collection is active at a given time") is useful one, but is not the normal definition of one-hot. (In OVL, the Open Verification Library, this check is known as zero_one_hot.) This will cause confusion.	Change the definition of ONE_HOT to "...in which one and only one signal in the collection is active at a given time", and add a second attribute ZERO_ONE_HOT with the definition "...in which at most one signal in the collection is active at a given time". Similarly for ONE_COLD and ONES_ONE_COLD.  N.B. Example 2 requires the use of ZERO_ONE_HOT, examples 3,4 require the use of ONE_HOT	Analysis: Currently the VHDL definition of the ONE_HOT attribute matches the OVL definition of ZERO_ONE_HOT. However we believe that this matches the ONE_HOT attribute of Synopsys. The definitions for OVL make sense for OVL. Disagree with observation about examples. All of the examples are ovl ZERO_ONE_HOT (or SIWG ONE_HOT). If we had an attribute for ovl ONE_HOT, we are not sure where it would be needed to be used. Recommendation: No change. We should lean toward the future on use OVL's name for the attribute ZERO_ONE_HOT we also agree that in the future the right way to do this will be through the OVL as that way the RTL simulation and gate simulation will match. Unfortunately OVL was not able to be considered for this effort due to timing and duration of the work.
	JA16		33 7.1.2.2	TECHNICAL	There is a technical issue with using VHDL attributes to identify a collection of signals (by decorating each with the ONE_HOT attribute): VHDL attributes don't identify collections of signals! VHDL groups exists for this purpose. Hence, if you had two or more sets of one-hot signals within the same architecture, after parsing there would be no way to identify which signals belonged to which set. For example:  attribute ZERO_ONE_HOT of set1, reset1: signal is true; attribute ZERO_ONE_HOT of set2, reset2: signal is true;  Downstream tools might see:  reset1 [ZERO_ONE_HOT=true] reset2 [ZERO_ONE_HOT=true] set1 [ZERO_ONE_HOT=true] set2 [ZERO_ONE_HOT=true]  Of course, a tool might still extract useful information from these attributes, but it would have to rely on analyzing the topology of the circuit to establish which signals fell into which one-hot sets. In special cases, such as flip-flops, this is okay. In general, this may not be obvious. For example, multiple sets of one-hot signals might feed into the same block of combinational logic.	Think again! The simplest "solution" would be to put heavy caveats on the use of the one-hot attributes, but this would make for a very weak standard. I recommend dropping these attributes, acknowledging that the standards OVL and PSL have overtaken this effort.	No change. This is resolved by the 'one-bit' PA25 Resolution.  In Example 2, for example, the 'collection' of signals is decorated as such. If there were other collections, each would be decorated separately. That's how this attribute is defined. If existing parsers can't handle this, it is something to implement in a new way.  For each such specification, the synthesizer should create one ONE_HOT register.
	JA17		34-35 Examples 2,3,4	TECHNICAL	These three examples are contradictory in that example 2 assumes a different definition of one-hot than examples 3-4. Example 2 assumes the definition ZERO_ONE_HOT as given above, and examples 3-4 assume the definition ONE_HOT as given above.	See above.	Analysis: See Above. All of the examples cited are ovl ZERO_ONE_HOT (or SIWG ONE_HOT). I am not sure what use a ONE_HOT would be. No change.
	JA18		44 7.1.10	TECHNICAL	One specific solution is proposed to a well-rehearsed issue in writing state machines in VHDL. The proposed solution violates the spirit of VHDL in introducing possible (and unnecessary) mismatches between simulation and synthesis. Several other technical solutions are possible and are in widespread use in the VHDL community. It is inappropriate to impose this as a standard.	Remove attribute FSM_COMPLETE	No change. Analysis: Issue with FSM complete has to do with the behavior when an unreachable state is removed. Creation of a safe statemachines is critical feature. Dealing with unreachable states is not critical. Resolution: Two choices: 1) Prohibit tools from removing unreachable states when FSM complete is true 2) Make it an error if there exists unreachable states when FSM complete is true. To me, 1 is the right answer. If unreachable states exist, it is either an error or it was done deliberately to help drive results in a particular direction. I am likely to do it deliberately. Consider the following statemachine, process(ps, in1) begin out1 <= '0'; case ps is when S0 => if (in1 = '1') then ns <= S1; else ns <= S0; end if; when S1   S3 => out1 <= '1'; if (in1 = '1') then ns <= S2; else ns <= S0; end if; when S2 => if (in1 = '0') then ns <= S0; else ns <= S2; end if; end case; end process; If I force coding of ps to be S0=00, S1=01, S2=10, S3=11, and the synthesis tool is forced to keep the unreachable state, S3, as it is, then out1 = ps(ps/left);
Stephen Bailey	SB01	Y	1.3	Typo/formatting	The formatting of the section providing the Supported, Ignored and Not Supported categories should be improved to ensure the alignment of the text descriptions.		Agreed. Will fix.

SB02	6.1.3	Functional correctness of the example	The modeling style for synthesis places asynchronous control conditions prior to the synchronous condition. In this example, the synchronous condition comes before the async reset condition. Thus, if rising_edge(clk) occurs at the same time as reset, reset will not occur. The example should show the correct/recommended coding style unless the purpose of the example is to show an erroneous situation.		No change. The example does show the correct modeling style. If both clk edge and reset occur at the same time, the Q=D will not occur if reset is 1 as shown in the condition. OOPS. Forget what I wrote here previously. Still not my issue, but ...  Analysis: Reset is active high ('1'). If reset and clock occur simultaneously, then the clock condition is false (it tests for 0) and the elsif condition is true. Resolution: No Change.
SB03	6.1.3	Typo/formatting	The indentation level of "elsif (en = '1' and rising_edge(clk) then" is not consistent and can lead to confusion.		Agreed. Make recommended change.
SB04	6.1.3.1	Typo/formatting	1st paragraph, the 2nd line is indented and should not be.		
SB05	6.1.3.1	Typo/formatting	Item c) in the list: "It is possible statically to enumerate ..." The adverb modifier "statically" is not placed next to the verb it modifies. This violates preferential grammar style. Recommend rewriting as "It is possible to statically enumerate ..."		Agreed. Make recommended change.
SB06	6.1.3.6	Question	Why include guarded blocks in the standard? How many synthesis tools support this? Although it is not my intent to limit the standard to the lowest common denominator, I do not see any value in requiring compliance to a coding style that almost no one uses and is probably not widely supported by synthesis tools today (and unlikely to be supported in the future). NOTE: Same question applies to section 6.2.1.4 and any other section employing guarded blocks/signals.		No change. The purpose of this rev was to encompass everything that VHDL can support for synthesis. In addition, the request to support guarded blocks explicitly cam from users.
SB07	6.1.3.7 Note 2	Normative vs. informative	I finally found the normative specification that allows subprogram recursion when statically bound. It is in section 8.2.2. The note should contain a reference to that section. (Note that this note is repeated as Note 3 in the next section as well.)		Agreed. Make recommended change.
SB08	6.2.1.2 Examples.	Question	I just noticed that many (all?) of the synchronous concurrent signal assignment examples are of the form "when RESET". On the other hand, the process examples are of the form "if RESET = '1' then". The point being that the concurrent signal assignment examples are either assuming that the async control signals are Boolean or they are all wrong. Personally, I'd prefer to see equivalent examples where the concurrent signal assignment examples assume that the async control signals are std_ulogic and use the form "when RESET = '1'".		Agreed. Make recommended change. Good observation.
SB09	6.5.1.1 Example	Normative/Informative	The example contains the following comment: "OK not to use ieee.rtl_attributes package, but the attribute must be defined identically as in the package." This comment is OK, but when I quickly looked in section 7.1, I could not find any normative statement that the attributes could be used from the rtl_attributes package or locally defined as long as the local definitions are identical to the definitions in the package. Therefore, we are lacking a normative specification of this flexibility.		Agreed. Make recommended change.
SB10	6.5.2	Efficiency	It is unacceptable that RAMs must be modeled using signals. The memory contents can often be more easily (and usually more efficiently) be modeled using a variable. I see no reason for this restriction. The standard should allow variables in this case. This is my most important comment and is serious enough to consider a NO vote.		Agreed. Modify clause "... using a signal or a variable that may have ..." in first sentence. In addition change Ex 2 to make the RAM as a variable instead of a signal. In addition, make Z and X as bus vector constants as assigning a single bit is illegal.
SB11	7.1.5	Ambiguity	"An exact match in this context means the same as when the VHDL includes a component instantiation and the synthesis tool must search for the component named." I find the wording of this sentence to be poor which leads to potential ambiguity in the specification. I assume that what the sentence is saying is that the same default binding rules specified in 1076 for binding component and entity names is used in this context. Unfortunately, the sentence does not clearly state this and I believe it can be better stated. I also believe that it would be best to state the matching independently of the 1076 default binding rules as the context is different.		Agreed. Change sentence to "An exact match in this context means that the same default binding rules specified in 1076 for binding component and instance names is used."

SB12		7.1.10	Undesired or dangerous specification	In the example 1 for this section, state S3 is unreachable. The code says that if S3 is ever reached (due to some error condition?) the next state is S1. However, with the attribute specified, the synthesis tool is to implement a transition out of S3 to S0. Thus simulation and synthesis behavior would not match. The synthesis result would not match the code. At a minimum, the section must note that the potential difference between simulation and synthesis results. I don't like this example as it indicates to users how to create unreadable and difficult to maintain code. The second example is much better and demonstrates a valid use of the attribute.		There is no problem with a mismatch as in simulation, state S3 will never be reached. In a synthesized netlist as well, S3 will never be reached. However recommend that Example 1 and 2 be switched so that the better example be shown first.
SB13		7.2	Ambiguity	Is the space between "--" and "RTL_SYNTHESIS" optional? I think it should be, but either way, it should be specified. Also, can we have a longer keyword than RTL_SYNTHESIS? (That's sarcasm.) Actually, how about an abbreviated alternative? We definers of VHDL standards constantly look to create the most typing possible for users even when not justified.		The space is optional. Add a line saying so. Also recommend that the name "RTL_SYN" can alternately be used for the directive such as: a) -- RTL_SYNTHESIS OFF (or, abbreviated, -- RTL_SYN OFF) b) -- RTL_SYNTHESIS ON (or, abbreviated, -- RTL_SYN ON)
SB14		8.1.1.2	Formatting/Typo	The strike-through on group_template_declaration is lighter than the others. (The others appear to be a bold strike-through.) There are other instances of this as well. It is probably something with how Acrobat is displaying the PDF, but please check and verify.		Looks ok in text. No need for change; it seems to be an aliasing error in the PDF viewer.
SB15		8.4.1	Too restrictive limitation	Why aren't protected types supported? While the case can be made for not supporting shared variables, protected types can be the type for a process-local or subprogram-local variable.		Good suggestion. This will be considered for the next rev of the standard.
SB16		8.8.1	Inconsistent specification	[sensitivity_clause] is struck-through in the wait_statement production, but it is not listed as not supported. The strike-through seems to be wrong as:  wait on clk until clk = '1';  Would (or should be) a valid wait statement.		Remove the strike-through.
SB17		8.8.4	Too restrictive limitation	Why isn't unaffected supported? The draft standard supports:  If rising_edge(clk) then Q <= D; Else Q <= Q; End if;  What's the difference between Q <= Q and Q <= unaffected?		Good point. It should be supported. Recommend removing the strike-through and moving from not supported to supported area. Also change in Annex A: waveform; & in 8.9.5.
SB18		8.8.5.1	Ambiguous specification	What is the significance of calling out array variable assignments separately from variable assignments? Does this imply that record variable assignments are not supported?		Recommend remove section 8.8.5.1.
SB19		8.8.8	Formatting/Typo	Note 2 appears to have a space missing between "choice" and "including".		Agrees. Fix typo.
SB20		8.9.1	Too restrictive limitation	Why aren't block headers supported? (We go to all this trouble to support guarded blocks and bus/register signal kinds, but we don't support protected types for process/subprogram local variables, unaffected nor block headers?)		It was felt that no one would ever use this feature. However, it is worth reconsidering it in a future rev.

George Economakos	GE01		10 Incorrect Example 6	Technical	The signal assignment Q<=D is implicitly controlled by a <sync_condition> (rising_edge(clk) in the enclosing if statement). This may bring misunderstandings in inferring edge sensitive storage components.	<p>Either:</p> <p>a) Remove the example.</p> <p>b) Change clause 6.1.3.1.a (page 9) or the definition of the &lt;sync_assignment&gt; (page 8) to:</p> <p>"a) The signal or variable has an explicit &lt;sync_assignment&gt;"</p> <p>or</p> <p>"&lt;sync_assignment&gt;. An assignment to a signal or variable which is explicitly controlled by &lt;clock_edge&gt; in all execution paths."</p>	The author is recommended to add the word "explicitly" in the defn of sync assignment 6.1.3 as in clause "which is explicitly controlled". The clause has been updated.
Peter Flake	PF01	Y	22, 6.3.2	Editorial	I do not understand "Error! Reference source not found".		This shall be fixed.
Ian Andrew Guyler	IAG01	Y	40, 7.1.6	Editorial	<p>The text is unclear about the behaviour of the COMBINATIONAL</p> <p>attribute. The first statement should apply when the COMBINATIONAL attribute is set to true, not just on the presence of the attribute.</p> <p>It says that it is an error if sequential logic is synthesised in the presence of the attribute, however it does not say whether</p> <p>this would mean that the tool was not-compliant, or whether the tool should issue an error under these circumstances.</p> <p>Also, I would expect that attaching COMBINATIONAL=TRUE to</p> <p>a clocked process or COMBINATIONAL=FALSE to a purely combinational process would also be erroneous.</p>	See above.	<p>Analysis: This is a technical issue. The definition of combinational differs slightly from my memory. My belief is that when combinational is true then if sequential logic is present, it is an error. Application is to a statemachine. The example shown is confusing as it is more of a side-effect than a representative example. We need to resolve what setting combinational to false means. I think it means the same as not having the attribute set. The balloter asserts that it could potentially mean that no combinational logic is intended. I do not agree with this interpretation.</p> <p>Resolution:</p> <p>Replace: "The COMBINATIONAL attribute shall indicate to the synthesis tool that the attributed object infers only combinational logic; in the presence of this attribute, if sequential logic is synthesized, it shall be an error. The attributed object must contain identity assignment(s) as shown in the following example, or attachment of the COMBINATIONAL attribute shall be an error." With: "The COMBINATIONAL attribute shall indicate to the synthesis tool that the attributed object infers only combinational logic. If this attribute is set to true on an object, and the code implies sequential logic, then an error shall be indicated. Setting the combinational attribute to false has no effect (or the same effect as the attribute not being declared)."</p> <p>Note: the intended use of this attribute is to catch unintended latches in combinational logic (such as statemachine present state - next state logic).</p> <p>An additional use of this attribute is to determine whether the following code should generate a latch or a multiplexer with feedback:</p> <pre>attribute COMBINATIONAL of P1: process is TRUE; P1 : process ( ENABLE, D) begin if ENABLE = '1' then Q &lt;= D; else Q &lt;= Q ; -- Identity assignment for Q end if; end process;</pre> <p>When the combinational attribute is set to true, the code shall generate a multiplexer with feedback. Otherwise the code shall create a latch.</p> <p>See also JL04 and JL05 for resulting updates 6.2.1.1</p>

						<p>Above resolution continued: =====</p> <p>To help clarify the TRUE and FALSE meanings, the first paragraph of 7.1 shall be changed to:</p> <p>For the boolean-valued attributes described in this subclause, the effects refer to a value of TRUE; a boolean FALSE value of the attribute shall result in behavior identical to the behavior when the attribute specification has been omitted entirely.</p> <p>In addition, the 4th paragraph of 7.1.6 shall be changed to:</p> <p>The COMBINATIONAL attribute shall indicate to the synthesis tool that the decorated item implies only combinational logic. Decoration by COMBINATIONAL of an object modelling sequential logic shall be an error except in the one case of a latch model in which the feedback path is defined by identity statements; such a latch shall be synthesized as combinational logic with a feedback path.</p> <p>=====</p>	
William A. Hanna	WAH01	Y	genera	Technical	<p>I did approve the standard because I am very familiar with all previous version and Logic Synthesis in general.</p> <p>I would like to see us someday able to synthesiz records and files by adopting generic: Read, Write, init, status, Format.</p> <p>A File is nothing but a collection of records that can be accessed randomly or sequentially. Hope you consider that for next revision.</p> <p>I will be glad to discuss it. Thanks.</p> <p>William A. Hanna; Ph.D.EE william.a.hanna@boeing.com</p>	Consider a new task for the Logic Synthesis of Records and Files in the next version of 1076.6	Agree. Consider support for Records and Files in the next version. Particularly since Verilogo supports files in this version.
Jim Lewis	JL01	Y		Editorial	Figures in section 7 need to have figure numbers rather than editorial comments.	Add proper figure numbers.	Agree. Will be fixed.
Michael McNamara	MM01	Y	47 7.1.4	Technical	In Clause 7.1.4, (on Page 47 in the PDF) : two example blocks have editorial directives 'Place Subprogram Implementation Figure 1 Here' in addition to the actual figure.	Eliminate place holder text	Figure placeholders shall be removed.
D.C. Mohla	DCM01	Y	General-definitions	Technical	<p>In principal we need to have the entire definition there or in Std or in IEEE- 100 by reference</p> <p>Definitions in Section 3.3, 3.5, 3.9, and 3.15 are tied to either IEEE-1164-1993, IEEE-1076.31997, or IEEE-1076.6- 1999</p> <p>Definition in 3.3 " don't care value" has two definitions. One from 1076.3 and second from 1076.6 . Both definitions indicated, "As defined by IEEE 1164- 1993. Please consider using one definition for consistency and simplicity</p>	Provide complete definitions in this document for inclusion in IEEE-100 such that they are not dependent on another document. I am available for consultation and additional details if needed	The definitions themselves are not copied from the other standards, the values such as '1', 'Z', are. However the first para in 3 clarifies where the terms are defined from. For clarity, we recommend to delete "defined by . . ." clause in 3.5, 3.5, 3.9. Also in 3.15, change to ". . . compliant to this standard.

E. Molenkamp	EM01	Y	16	Technical	<p>Example:  COND_SIG_ASSGN: Q &lt;= '0' when RESET else  '1' when SET else  A when ASYNC_LOAD else  D when CLOCKEVENT and CLOCK = '1';</p> <p>This is the only example that uses RESET in stead of resets='1'. Consider changing this example in:  Example:  COND_SIG_ASSGN: Q &lt;= '0' when reset='1' else  '1' when set='1' else  A when async_load='1' else  D when CLOCKEVENT and CLOCK = '1';</p>	Resolution: fix it as suggested.
	EM02		21	editorial	Add an empty line after "6.2.1.4. Level-sensitive storage from guarded block" before the text of this paragraph.	Resolution: fix it as suggested.
	EM03		22	editorial	Missing reference in document: " logic in Error! Reference source not found. or 6.2.1.4 are not fulfilled."	Resolution: fix it as suggested.
	EM04		38	editorial	"It is an error if the specified entity or library cell is not in the target synthesis library." There are two 'points' at the end of this line.	Resolution: fix it as suggested.
	EM05	39 just above 7.1.5.2		editorial	"The value of the attribute shall indicate the name of aspecific" change "aspecific" in "a specific".	Resolution: fix it as suggested.
	EM06	79 Note 2		editorial	2--A case choiceincluding a metalogical value such as *1X1* indicates a branch that never can be taken by the synthesized circuit (IEEE Std 1076.3-1997).  Change ""choiceincluding" in "choice including"	Resolution: fix it as suggested.
	EM07		91	Technical	the following functions with arguments of type "universal integer" or INTEGER: -- "?", "mod", and "rem" when neither operand is static or the second argument is not a static power of two -- "****" when the first argument is not a static value of two	EM07 and EM08 are part of the same issue. Issue: Some places in the standard use the term first for the left argument. Some use left for the left argument. Observation: This inconsistency existed in the previous version of the standard. Resolution: No change. Neither term causes ambiguity. It would be nice to have consistent language. It could be considered for the next time we work on the standard, but on a risk reward tradeoff, I don't think it is worth it.
	EM08	73 (8.7.2.7)		Technical	The ** (exponentiation) operator shall be supported only when both operands are static or when the left operand has the static value 2.  No consequent usage of 'left' and 'first'.	Resolution: See above.
	EM09	109 (Annex B)		Technical	attribute TYPE_CONVERSION : boolean; attribute ZERO_EXTEND : boolean; attribute SIGN_EXTEND : boolean;  I could not find an explanation of these attribute in the standard. Notice also that in the index (page 111,112) there is no reference to other pages than the Annex B. Remove these three attributes.	Resolution: I don't remember why we need these. I agree that they can be removed. Fix it as suggested.
Serafin A. Perez Lopez	SAP01	Y	3. Definitions (p.3)	Technical	3.9 metalogical value: I think that the literal 'Z' should not be considered as a metalogical value, because it does have a hardware meaning: a high-impedance output. In fact, it was not considered as a metalogical value in last version of this standard.	Agree with recommended change. In addition, wherever metalogical was intended to include the value 'Z', it shall be replaced with "metalogical or high-impedance"
	SAP02		6. Modeling hardware elements 6.1.3.7. Example: (p.18)	Technical	I think it may be useful to include some wait sentence in this example to show or take into account the importance of NOTE 1 (p.17)	Resolution: While an example with a wait statement would be nice, it is not required. This is not a user's guide of examples.
	SAP03		7.1.2.1.2. (p.31)	Technical	There is an error in the title: It must be deleted "end architecture EdgeSensitive;". That sentence belongs to the example above and thus the words in boldface must be interchanged: "end architecture EdgeSensitive;"	Resolution: agree. Fix it.
	SAP04		7.1.8.	Technical	Enumeration encoding attribute (p.42) : Attribute object must be type, subtype instead of signal, variable	Resolution: Agree. Fix it.
	SAP05		7.1.9.	Technical	Finite State Machine attribute (p.42) : Attribute object must be signal, variable instead of type, subtype	Resolution: Agree. Fix it.

	SAP06		8.8.1. (p.76)	Technical	The wait statement syntax indicates that sensitivity_clause shall not be supported (sensitivity_clause). Nevertheless a few lines below it is included as supported syntax. And, in fact, it is used in section 6.1.3.2. examples. So I find some contradiction regarding it all. I guess that sensitivity_clause must no be struck-through.		Resolution: sensitivity_clause shall be supported
	SAP07		8.8.8. (p.79) NOTE 2	Technical	A case choice including .... It must be: A case choice including .... Simply, a blank character is missing.		Resolution: Agree fix it.
	SAP08		8.14.1.1.	Technical	I don't understand why the attributes VALUE, POS, VAL, SUCC, PRED, LEFTOF and RIGHTOF are no longer considered as predefined. It is clear that the Working Group has decided to change it regarding the last version. But maybe there should be some kind of explanation included. The last paragraph is one of the reasons I think it must be included another Annex explaining the changes adopted in this version, just like it is usually done in any standard revision.		Resolution: Instead of "no longer considered as predefined", we presume the balloter meant "not supported". In addition, the balloter has correctly identified that these attributes were supported in the 1999 std. So our conclusion is that we have a typo in our draft and we ought to mark these (VALUE, POS, VAL, SUCC, PRED, LEFTOF, RIGHTOF) as supported. In addition, add the following NOTE immediately after "8 Syntax": NOTE--Subclause titles in this clause match those of IEEE Std 1076.
John Shields	JS01	Y	22 6.3.2	editorial	Text "Error! Reference source not found." should be fixed		Agree. Fix it.
	JS02		23,25 6.5.1.1,3	editorial	recommend that all examples use ieee_rtl_attributes.all. Explain local declaration alternative in Chapter 7.		Disagree. Either or is ok.
	JS03		26 6.5.2	technical	type ram_type is incorrect	change DEPTH to 2**DEPTH	Agree. Fix it.
	JS04		26 6.5.2	editorial	commented out version of attr ram_block should be removed or made more relevant		Agree. Fix it.
	JS05		26 6.5.2	technical	in q <= ram(a), a is an illegal index	change ram(a) to ram(to_integer(unsigned(a)))	Agree. Fix it.
	JS06		27 6.5.2	technical	in q <= ram(a), a is an illegal index	change ram(a) to ram(to_integer(unsigned(a)))	Agree. Fix it.
	JS07		28 6.5.2	technical	in q <= ram(a), a is an illegal index	change ram(a) to ram(to_integer(unsigned(a)))	Agree. Fix it.
	JS08		40 7.1.7	editorial	2cd paragraph, 2cd and 3rd sentence are redundant	remove	Agree. Merge paras 1 and 2 and remove redundant sentences as identified by balloter.
Mark Tillinghast	MT01	N	general	Technical	This represents a somewhat long train of abuse insofar as the IEEE standards organization goes. I am sorry that RTL has to bear the brunt of this comment but it goes far deeper and historically farther back in time than P1076.6 D6.  Never  Supported, Ignored or Not supported is a categorization more befitting a Guide than a standard. Clause 1.3 gives "a nod and a wink" to the "Pick and Choose" implementations of the standard.	Say what is supported as a requirement. Delete any information about what is not supported. Do not refer to things that are ignored or not supported, if it is not a part of the specification.  Alternatively it should be recast as a guide in the following Sense as a Guide.  IEEE-SA Standards Board Operations Manual (2003):  — Guides: documents in which alternative approaches to good practice are suggested but no clear-cut recommendations are made.	Reject. These definitions are critical to usefulness of the standard.
	MT02		33 line 30 7.1.2.2	Editorial	All figures must be labeled as follows(from IEEE Standards Style Manual May 2000)  Figure 3—Typographical specifications for figure callouts  A figure shall be labeled by the word Figure followed by a number, a dash, and a title as exemplified in Figure 3 above.	Replace Suprogram Implementation Figure x Here with the requirements set forth in Clause 16 of IEEE Standards Style Manual May 2000.	Agree. Fix Figure Labels and style
	MT03		line 20 6.1.3.2	Editorial	Example 4 refers to condition b  Incorrect Example 6 violates Rule a  should be consistent: either rule or condition.	Change it to rule or condition and use it uniformly.	Agree. Standardize on either the term rule or the term condition and update document.

	MT04		13 line 1 6.1.3.2	Technical	The outer if statement is superfluous:  process begin wait on SET, reset, clock ;  if SET = '1' or reset = '1' or rising_edge(clock) then  if reset = '1' then Q <= '0';  elsif SET = '1' then  Q <= '1';  elsif rising_edge(clock) then  Q <= D;  end if ;  end if;  end process ;	replace with  process begin wait on SET, reset, clock ;  if reset = '1' then Q <= '0';  elsif SET = '1' then Q <= '1';  elsif rising_edge(clock) then Q <= D;  end if ;  end process ;	Reject. Agree with simplification, however, this standard is not about best coding styles. It is about what should be synthesizable and what a user should be able to input.
	MT05		44 line 13 7.1.10	Technical	Finite State Machine completion attribute covers up errors. It is a sloppy alternative to a fatal error.	Unreachable states should not automatically transition to the specific default statement.	Agree. Fix it. The intent of FSM_COMPLETE attribute is for the synthesis tool to preserve the intent of the transitions coded in the statemachine.
John Williams	JW01	Y	22 6.3.2, parag. 2	Editorial	"Error! Reference source not found." appears where a section number should.	Replace the MS Word error message with "6.1.3.6".	Agree. Fix it.
	JW02		31 7.1.2.1.2	Editorial	The 7.1.2.1.2 heading is confused with some text from the preceding paragraph.	Remove "end architecture EdgeSensitive" from the heading and into preceding paragraph, in computer code style.	Agree. Fix it.
Mark Zwolinski	MZ01	Y	6.1.2	Technical	Rising_edge and falling_edge are defined for std_ulogic in Std_logic_1164. The functions are defined for Bit in numeric_bit.		Agree. Reword appropriately.
	MZ02		6.3.2	Technical	There's an "Error! Reference source not found."		Agree. Fix it.
	MZ03		6.5	Technical	All the commented out attributes are wrong. They should be logic_block not rom_block or ram_block and the Boolean value should not be quoted.		Agree. See User's resolutions below. In addition in example 2, where logic block is used, the word variable is mis-spelled. See next two comments.
	MZ04		6.5.1.1	Technical	Should be -- attribute logic_block of ROMINIT : constant is true;		Agree. Solutions to above.
	MZ05		6.5.1.2	Technical	Example 1 Should be -- attribute logic_block of Z : signal is true; Example 2 Should be -- attribute logic_block of rom : variable is true;		Agree. Solutions to above.
	MZ06		6.5.2	Technical	Example 1 Should be -- attribute logic_block of z : signal is true; Example 2 should be -- attribute logic_block of ram : signal is true; Example 3 Should be -- attribute logic_block of ram : signal is true; Note also that the Purpose, 2 lines below is incorrect – it is an Asynchronous Ram definition		Agree. Fix it. Note on synchronous RAM, it is unusual to have a tristate on on-chip memory. Furthermore, it is typical for the RE to be connected directly to the tristate. The example has a register on RE. Recommendation: Assign Q_int <= Ram(A) ; Then combinationally create the tristate: with RE select Q<= Q_int when, 'Z' when '0', 'X' when others ;
	MZ07		7.1.2.1.2	Technical	The title has been confused with the last line of the previous example.		Agree. Fix it.
	MZ08		7.1.9	Technical	The description of the FSM_STATE attribute is very confused. The attributed objects are stated to be type or subtype, but the examples all use signals. The special case, where the state encoding is explicitly given, is not listed in the set of possible values and also overlaps with the ENUM_ENCODING attribute. I note there is a distinction, but it seems unnecessary.		The typos are in the 3rd lines of paras 7.1.8, 7.1.9. In 7.1.8, should be "Attributed object: type, subtype" In 7.1.9, should be "Attributed object: signal, variable" These shall be fixed. See SAP04, SAP05.

















Comment Recirculated? (Y/N/NA)  
resolution use only)

(for ballot













































