

Spec/Category/SpecVariable handling proposals for P1450.4

In all discussions below, the term “spec variables” refer to variables defined within a spec block, and which appear in one or more category subsections of a spec block or blocks. Further, spec variables will have one or more selector subvalues (i.e., min, typ, max, or meas), as specified by IEEE 1450-1999, chapter 19 “Spec and Selector Blocks”, and selected by the Category and Selector statements of the PatternExec block (IEEE 1450-1999, chapter 16) or by parameters of P1450.4 Test or Flow blocks.

As defined in IEEE 1450-1999, spec variables (used for specifying timing or levels values) are global in scope (accessible to all levels). The combination of the category name + variable name must be named unique.

Finally, even though the categories and variables are syntactically contained within a spec block, a spec block has no semantic meaning. It is not used to provide additional scoping.

1. Status quo for dot0, dot1, dot2
 - a. No changes, no additions, to what’s specified in STIL dot0, dot1, dot2
 - b. All spec variables are global, and combination of category+variable name **MUST** be unique.
 - c. The set of variables within a particular category can be contained in one spec block, or spread out over multiple spec blocks.
2. Constrain all categories for a set of variables to a single spec block.
 - a. No changes to scoping of spec variables, but add constraint that all categories for a particular variable must be contained in a single spec block, and add spec block (category, variable) iteration syntax.
 - b. As in #1 above, all spec variables are global, and combination of category+variable name **MUST** be unique.
3. Add spec block name to scoping hierarchy, but keep “category+variable” name uniqueness.
 - a. All spec variables and categories are scoped to a particular spec block, which **MUST** be named in a particular context. If variables or categories are **NOT** named in the currently active spec block(s), they cannot be resolved.
 - b. However, the constraint that the combination of “category+variable name” must be unique is not relaxed.
 - c. This adds semantic meaning to the spec block, and introduces a change to existing STIL practice.
4. Add spec block name to scoping hierarchy, and allow duplication of “category+variable name” in multiple (uniquely named) spec blocks.
 - a. All spec variables and categories are scoped to a particular spec block, which **MUST** be named in a particular context. If variables or categories are **NOT** named in the currently active spec block(s), they cannot be resolved. Can reuse category and variable names in different spec blocks.

- b. Note that this is a major change from existing STIL practice, and will present some issues when translating to LTX Envision (which uses a scheme very similar to #2 above). We believe that these issues can be dealt with by creating Envision spec variable names which are a concatenation of the spec block name and spec variable name. It's a bit clumsy, but it will work. When translating Envision code to STIL (if that is ever done), no problems arise, since an Envision spec variable is global in scope, and would not be reused in different spec blocks.

Variable scoping rules.

1. Within a test or flow, local scope consists of local variables and parameters. Each name must be unique in that context.
2. Variables, constants, and parameters (including spec variables) declared in the local scope will hide any variables, constants, or parameters of the same name which are declared in the global scope. Global scope includes specs declared in Spec blocks, global variables declared in named or unnamed variables blocks at the global scope, TestProgram variables, and all Test or Flow objects (created by instantiating TestTypes or FlowTypes).
3. To access the variables in the upper level scope, the complete variable or spec access syntax can be used.