Portable Test and Stimulus Standard (PSS) v2.0 Release Notes

PSS 2.0 includes new functionality, errata corrections, and clarifications relative to PSS 1.0a (February 2019). The details of the main new functionality and clarifications are shown here. The section numbers listed are those of PSS 2.0.

- Section 4. Lexical conventions
    - 4.6 Numbers (new section) – Added description of numeric literal formats and specification of bit widths.
    - 4.7 String literals (new section) – Added description of quoted and triple-quoted string literals, escape sequences, and the semantics of valid characters within each.
    - 4.8 Aggregate literals (new section) – Added support for aggregate literals, used to specify the literal value of *array*, *list*, *map*, *set* (see 8.8), and *struct* variables.
- Section 8. Data types
    - 8.4 Enumeration types – Added rules on the semantics of enumeration types and the resolution rules for *enum item* references.
    - 8.6 Chandles – Clarified the default value of a *chandle* variable.
    - 8.8 Collections (new section) – Added support for built-in collection types (*array, list, map, set*), and specified the operators and methods that each supports.
    - 8.9 Reference types (new section) – Added support for declaring reference variables of action, component, flow object, and resource object type.
- Section 9. Operators and expressions (new chapter) – Added descriptions of PSS operators and expression evaluation semantics.
    - 9.4.3 Type inference rules – Added type inference rules that capture the expected type of an expression.
    - 9.5.9 Set membership operator – Moved from 15.1.4 in PSS 1.0a.
    - 9.7 Bit-sizes for numeric expressions – Added description of rules for evaluation of bit-size of expressions of numeric types.
- Section 10. Components
    - 10.7 Pure components (new section) – Added support for lightweight *pure* components, used by the PSS register model (see 24.5).
- Section 11. Actions

  - Added the ability for compound actions to contain more than one activity.

- Section 12. Template types (new chapter) – Added support for templated data types.
- Section 13. Activities
  - 13.3.2 Action handle array traversal (new section) – Added declaration and traversal of arrays of action handles in *sequence*, *schedule*, *select*, and *parallel* activity statements.
  - 13.3.6 Fine-grained scheduling specifiers (new section) – Added support for modifying termination semantics of activities within *parallel* and *schedule* blocks.
  - 13.5.1 Replicate (new section) – Added support for generating expansions of activity statements multiple times with the *replicate* activity statement.
- Section 14. Flow objects
  - 14.4 Using flow objects – Added support for fixed-size arrays of flow object references, and the rules that apply to these arrays.
- Section 15. Resource objects
  - 15.2 Claiming resource objects – Added support for fixed-size arrays of resource object references, and the rules that apply to these arrays.
- Section 17. Randomization specification constructs
  - 17.1.8 forall constraints (new section) – Added the *forall* constraint, which constrains all instances of a type in a context-specific manner.
  - 17.1.10 Default value constraints (new section) – Added support to explicitly determine the value of an attribute with a *default* value constraint, or to *disable* the default value.
- Section 20. Type inheritance, extension, and overrides – Renamed Section 18 from PSS 1.0a.
  - 20.1 Type inheritance (new section) – Describes the support of object-oriented inheritance for actions, components, structs, data flow and resource objects.
  - 20.2 Type extension – Renumbered from 18.1 in PSS 1.0a.
  - 20.3 Combining inheritance and extension (new section) – Clarifies interaction between inheritance and extension, specifying what is and is not allowed when used together.
  - 20.4 Access protection – Moved from 8.9 in PSS 1.0a.
- Section 21. Source organization and processing - Renamed Section 19 from PSS 1.0a. Chapter has been restructured and expanded.
  - Added description of how a PSS model is specified in one or more *source units*, each of which may be a single file or multiple files, and of the relationship between source units.
  - 21.1.1 Package declarations – Added support for declaration of components within packages.

  - 21.1.2 Nested packages (new section) – Added support for declaring packages within packages, and how they can be used.
  - 21.1.3 Referencing package members – Renamed 19.3 from PSS 1.0a. Clarified the ways that package members can be referenced and the semantics of package *import*s.
  - 21.1.4 Package aliases (new section) – Added support for declaring a named *alias* for a package namespace and describing its benefits.
  - 21.2 Declaration and reference ordering (new section) – Added the rules describing whether and when for PSS elements may be referenced before their declaration.
  - 21.3 Name resolution – Was 19.2 in PSS 1.0a. This section now describes in much more detail how name resolution works.

- Section 22. Test realization – This chapter has been substantially restructured and enhanced.
  - 22.1.3 exec block kinds – Components now support a two-phase top-down + bottom-up initialization process using *init_down* and *init_up* exec blocks.
  - 22.2.3 Default parameter values (new section) – Function parameters can now have default values.
  - 22.2.4 Generic and vararg parameters (new section) – PSS-defined library functions may be variadic and be templatized on parameter types.
  - 22.2.5 Pure functions (new section) – Functions whose return value depends only on the parameter values and that do not introduce side-effects can be declared as *pure*.
  - 22.3 Native PSS functions (new section) – PSS native functions can now be defined, using the new procedural constructs in 22.7.
  - 22.7 Procedural constructs (new section) – Added procedural constructs for environment-agnostic definition of native functions and exec blocks.

- Section 23. Conditional code processing
  - Restricted expressions legal in conditional compilation statements. Clarified name resolution rules for compilation process.

- Section 24. PSS core library (new chapter) – Defined a library of standard functions and types that PSS implementations must support.
  - 24.1 Executors – Covers management of execution contexts within target environments.
  - 24.2 Address spaces – Covers modeling of memory and other types of target storage.
  - 24.3 Allocation within address spaces – Covers allocation of target storage.
  - 24.4 Data layout and access operations – Covers management and access routines to target storage.
  - 24.5 Registers – Covers modeling of system hardware registers.

- Annex B. Formal syntax – In addition to support for new language constructs, the BNF was substantially refactored for clarity without changes to language semantics, in particular:
    - The BNF form of expressions was simplified since operator precedence is now captured in Chapter 9.
    - Optional semicolons were removed from declarations, such as actions, and added as an independent stmt_terminator element that can appear in multiple contexts.
- Annex C. C++ header files – Extensive updates to reflect enhanced functionality.
- Annex D. Core library package (new chapter) – Code listing for core library defined in Section 24.
    - D.1 Package executor_pkg
    - D.2 Package addr_reg_pkg
- Annex E. Foreign Language bindings – Enhanced rules for foreign language binding to include compound and user-defined data types.
    - E.1 Function prototype mapping – Added formal definitions for parameter function prototype mappings from a PSS function to a foreign language function along with parameter direction, if supported.
    - E.2 Data type mapping – Added PSS data type bindings to C/C++ and SystemVerilog for specific parameter types (primitive, user-defined, fixed-sized arrays of these types).
    - E.3 C language bindings – Added support for mapping PSS function to an identical function name in C.
        - E.3.2 C return types – Added a table that outlines mappings for PSS primitive types to C types for input, output/inout, and return.
        - E.3.3 C Arrays – Added support for mapping function arguments of fixed-sized arrays from PSS to C. There is no support for mapping fixed-sized arrays for function return types.
        - E.3.4 C structs – Define the binding of PSS struct type to a C type for input, output/inout and return. The binding for strut field primitive types to C types are defined in a table.
        - E.3.5 Enumeration types – Define the binding for PSS enumeration to C as a plain numeric type (char, short, int, or long long).
    - E.4 C++ language bindings
        - E.4.1 Function name mapping and namespaces – Clarification of PSS to C++ type mapping functions and namespaces (packages and components).
        - E.4.2 Primitive types – Added text spec mappings for primitive types outlined in a table.

        - E.4.3 Arrays – Added mappings for PSS arrays to C++ arrays.
        - E.4.4 Structs – Added mappings for PSS struct to C++ struct types for input, output/inout, and return as well as field mappings.
        - E.4.5 Enumeration types – Added mappings for PSS enumeration to C++ unscoped enumeration type binding.
    - E.5 SystemVerilog language bindings
        - E.5.1 Function names – Added function name binding from PSS to SystemVerilog function or task names.
        - E.5.2 Primitive types – Added a table defining the mapping from PSS primitive to SystemVerilog types.
        - E.5.3 Numeric value mapping – Added specification for mapping numeric types or values passed from PSS to SystemVerilog.
        - E.5.4 Arrays – Defined the mapping from PSS fixed-size arrays to SystemVerilog dynamic arrays. Arrays are passed by value between PSS and SystemVerilog.
        - E.5.5 Structs – Added the specification for mapping PSS structs to classes in SystemVerilog with fields whose types and names match. Values of all fields are deep-copied between mapped elements.
        - E.5.6 Enumeration types – Added the specification for the mapping of PSS enumeration types to SystemVerilog enums.

The following functionality has been removed or deprecated in PSS 2.0:

- Removed "**`while (`** *`expression`* **`)`** *`activity_stmt`*" (Section 11.5.2.1 in PSS 1.0a)
- Deprecated array properties `.size` and `.sum`.
    - The methods `.size()` and `.sum()` should be used instead.
- Deprecated declarations of the form **`int/bit[`**`N:M`**`]`**
    - The form **`int/bit[`**`N`**`]`** should be used instead.
    - If **`int/bit[`**`N:M`**`]`** is used, M is required to be 0.
- Deprecated **`exec init`**
    - **`exec init_up`** is equivalent and should be used instead.