Proposal: Exported Tasks

Motivation

SystemVerilog 3.1's DPI supports imported and exported inter-language function calls. An imported function is implemented in C, and it is callable from SystemVerilog. An exported function is implemented in SystemVerilog, and it is callable from C.

Users have requested that the DPI export functionality be extended to include exported tasks. This will enable new and powerful modeling paradigms.

For example, a testbench environment may be written in C, and make use of high-level function calls to effect hardware transactions in the DUT. Such a test environment could be achieved in SystemVerilog as follows. Start up the testbench by calling an imported function from within a program block. The imported function is the gateway to a complex body of C procedural code. When necessary, the C code calls back into the SystemVerilog DUT by making calls to various exported tasks. Each such task effects a certain hardware transaction, which normally will consume simulation time. The C code is impervious to this time consumption since it is transaction-level code, rather than cycle-accurate timed code.

Another application enabled by exported tasks is the seamless interleaving of C-based models with SystemVerilog models. Simulation tools can now offer modeling environments in which C and SystemVerilog models all live on the same kernel threading system. The models call back-and-forth between each other in order to achieve the user's desired functionality. This is especially useful when moving from a transaction-level C-based verification system into actual RTL implementation. The TLM C testbench can be re-used by replacing key function calls into the DUT with calls to SV exported tasks.

Changes

1) Section 26.1.1
   Rename section to "Tasks and Functions"

   Add a new second paragraph as follows:

   "It is also possible to perform task enables across the language boundary. Foreign code may call SystemVerilog tasks, and native Verilog code may call imported tasks. An *imported task* has the same semantics as a native Verilog task: It never returns a value, and it may block and consume simulation time."

   In the existing third paragraph (which starts "Every imported function needs..."), change all uses of the word "function" to read "task or function". However, the last sentence should be rewritten as follows:

   "Imported tasks and functions can have zero or more formal input, inout, or output arguments. Imported tasks always return a void value, and thus can only be used in statement context. Imported functions can return a result or be defined as void functions."

   In the existing fourth paragraph (which starts "DPI is based...", add a concluding sentence as follows:

   "Similar interchangeable semantics exist between native SystemVerilog tasks and imported tasks."

2) Section 26.2.2 erratum
   The final sentence of the section refers to Annex A. This reference should be changed to Annex D.

3) Section 26.3
   Replace all instances of the word "function" with "task or function". Adjust grammar (plurals, correspondences) as needed.

4) Section 26.4, “Imported functions”
Change title to “Imported tasks and functions”.

5) Section 26.4.1, “Required properties...”
Change use of word “functions” to “tasks and functions”.
Change term “function calls” to “task or function calls”

6) Section 26.4.1.1 “Instant completion”
Change title to “Instant completion of imported functions”
Add a new sentence at the end:

“Note that imported tasks may consume time, similar to native SystemVerilog tasks.”

7) Section 26.4.1.3, “Special properties pure and context”
Change all uses of the word “function” to “task or function”.
Similarly, “functions” changes to “tasks or functions”.
However, leave the second paragraph (starting with “A function whose...”) alone, and add the following sentence to the end of the paragraph: “An imported task can never be declared pure.”

8) Section 26.4.3, “Context functions””

Replace the word “function” with the term “task or function”. Similar for “functions”.

9) In Annex A.2.6, and excerpt thereof in Section 26.4.4, “Import Declarations”:
Change BNF as follows:

```
dpi_import_export ::=
        import “DPI” [ dpi_function_import_property ] [ c_identifier= ] dpi_function_proto;
      | import “DPI” [ dpi_task_import_property ] [ c_identifier= ] dpi_task_proto;
dpi_function_import_property ::= context | pure
dpi_task_import_property ::= context
dpi_function_proto ::=
        function named_function_proto (footnotes a, b)
dpi_task_proto ::=
        task named_task_proto (footnote b)
```

Footnote a) dpi_function_proto return types are restricted to small values, as per 26.4.5
Footnote b) formals of dpi_function_proto and dpi_task_proto cannot use pass by reference mode and class types cannot be passed at all; for the complete set of restrictions see 26.4.6.

[Note: These changes are consistent with sv-cc’s LRM-5 adjustment in the 3.1a LRM. (See http://www.eda.org/sv/LRMChanges.html for details)]

10) Section 26.4.4, “Import declarations”
Change word “functions” to “tasks and functions”, except in the last sentence of the 2nd paragraph. Leave that one alone (“Imported functions can return a result or be defined as void functions.”). Add a new sentence after that says “Imported tasks never return a result, and thus are always declared in foreign code as void functions”.

The third paragraph ends with the sentence “An import declaration can also specify an optional function property: context or pure”. Change this sentence as follows: “An import declaration can also specify an optional function property. Imported functions can have properties context or pure; imported tasks can have property context.”

In the examples at the bottom of the section, in “miscellanea”, add a new example like this:

```
import “DPI” task checkResults(input string s, bit [511:0] packet);
```

11) Section 26.4.6, “Types of formal arguments”
Change all uses of the word “function” to “task or function”.

12) Add new section after 26.6 (will be: 26.7), “Exported tasks”

SystemVerilog allows tasks to be called from a foreign language, similar to functions. Such tasks are termed “exported tasks”.

All aspects of exported functions described above in section 26.6 apply to exported tasks. This includes legal declaration scopes as well as usage of the optional c_identifier.

It is never legal to call an exported task from within an imported function. This semantic is identical to native SystemVerilog semantics, in which it is illegal for a function to perform a task enable.

It is legal for an imported task to call an exported task only if the imported task is declared with the `context` property. See section 26.4.3 (Context tasks and functions) for more details.

One difference between exported tasks and exported functions is that SystemVerilog tasks do not have return value types. Thus prototypes of exported tasks in foreign code always have a void return type.

13) Annex A.2.6, Section 26.6
Add new text to Annex A.2.6, and include the excerpt here as is done in section 26.6>
dpi_import_export ::=
| export “DPI” [c_identifier =] task task_identifier;

14) Section D.1, “Overview”
There are two bullet items starting at the 2nd paragraph.
Add two new bullets after the existing two as follows:
— Tasks implemented in SystemVerilog and specified in export declarations can be called from C; such functions are referred to as *exported tasks*.
— Functions implemented in C which can be called from SystemVerilog, and can in turn call exported tasks, are referred to as *imported tasks*.

15) Section D.3, “Portability”
Change “functions” to “tasks or functions”

16) Section D.5, “Semantic Constraints”, and Section D.5.1
Change “imported functions” to “imported tasks and functions”.
Change “exported functions” to “exported tasks and functions”
Change “SystemVerilog function” to “SystemVerilog task or function” (4th paragraph)

17) Section D.5.5, “context and non-context functions”
Change “functions” to “tasks and functions”

18) Section D.7.2, “Calling SystemVerilog functions from C”
Change “functions” to “tasks and functions”

Add the following new paragraph at the end of the section:
“Calling a SystemVerilog task from C is the same as calling a SystemVerilog function from C, with the exception that tasks do not have return types, and thus are always presented as void-returning function calls in C.”

19) Section D.8, and all child D.8.x sections.
Rename to “Context tasks and functions”

Change “functions” to “tasks and functions” throughout the section (exception: VPI/PLI functions should not be changed)