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 import 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
   Remove first sentence of second paragraph.

2) Section 26.4.1
   Add "or time consuming (See section 26.4.4 for details)" after "or context".

3) Section 26.4.1.3
   Change section heading to "Special properties pure, context, and timeconsuming".

   Add new paragraph at end of section as follows

   An imported function that is intended to consume time must be attributed as `timeconsuming`. An example of consuming time would be that an imported function makes a call to an exported task which executes a delay control or event control statement.

4) Modify section 26.4.3, sentence in fourth paragraph starting with "For imported functions not...":

   The part of the sentence that says "SystemVerilog functions" should be changed to "SystemVerilog functions or tasks".

5) Add new section after 26.4.3 (will be: 26.4.4), "Time consuming functions":

   An imported function that consumes simulation time must be attributed as `timeconsuming`. An imported function can consume time by making a call to an exported task that contains a delay control or an event control statement. Note that any use of such statements is considered time-consuming, even if the statements result in simulation time only being suspended for one or more "delta" cycles. (e.g. #0;)

The `timeconsuming` property is orthogonal to the context property. That is, an imported function which calls a time consuming task must be attributed as both `context` and `timeconsuming`. An imported function which calls a task that does not consume time need only be attributed as `context`.

Note that time consuming imported functions can return values, just like normal imported functions. Such functions are not meant to be direct proxies for SystemVerilog task calls, which don't have return value types.

6) In Annex A.2.6, and excerpt thereof in Section 26.4.4, "Import Declarations":
Change BNF to have additional optional dpi_import_property field "timeconsuming".

7) 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 field.

It is not legal to call an exported task from within a `pure` or plain (un-attributed) imported function. For an imported function to call an exported task, the imported function must be attributed as `context`.

A major difference between exported functions and tasks is that tasks may consume simulation time. If an exported task executes any delay or event control statements, the calling imported function must be attributed as `timeconsuming`. Failure to do so is an error and may result in unpredictable simulator behavior.

One other 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.

<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;

8) In Section D.1, "Overview", add a third bullet item near the top:
— Tasks implemented in SystemVerilog and specified in export declarations can be called from C; such functions are referred to as *exported tasks*.

9) Add new section after Section D.5.5 (will be: D.5.6) called "Time consuming functions":

Also refer to Section 26.4.4.

Some DPI imported functions, or perhaps other C functions supported by given simulation tools, may result in simulation time being consumed. This may happen by way of making a call to an exported SystemVerilog task, or perhaps by some API specific to a given simulation tool. Such functions are called time consuming functions, and they are attributed with the `timeconsuming` property in SystemVerilog source code.

Note that time consuming imported functions are often also `context` imported functions, since in order to consume time by calling an exported SystemVerilog task, the imported function must be attributed with the `context` property.

10) Change section D.7.2 to be called "Calling SystemVerilog functions and tasks from C"
At the end, add this:
"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."

11) In Section D.8:
Add a third introductory paragraph as follows:
"For the purposes of the discussion in this section, consider that context issues associated with exported tasks are identical to context issues associated with exported functions. All capabilities and limitations of context functions described herein apply to timeconsuming functions as well. In the interest of brevity, no further mention will be made of exported tasks in this section."

