Section **19.9**

Severity System Tasks

Description

Currently System Verilog does not provide any standard way to specify severity information while printing user-defined general error messages. The Assertion severity system tasks could be used for that purpose, but they are currently only permitted in SVA action blocks. It would be valuable for users if these tasks were allowed anywhere $display is allowed. This will provide an easy and standard way to print info, warning, and error messages from general Verilog code.

Suggested Resolution

In Clause 19.9, change:

19.9 ~~Assertion severity~~ Severity tasks

In Table Syntax-19-9, change:

```
~~assert_~~severity_message_task ::=
fatal_message_task
| nonfatal_message_task

fatal_message_task ::= $fatal [ ( finish_number [ , list_of_arguments
~~message_argument { , message_argument }~~ ] ) ] ;
nonfatal_message_task ::= severity_task [ ( list_of_arguments ~~{~~
~~message_argument { , message_argument] } ]~~ ) ] ;

~~message_argument ::= string | expression~~
```

Table Syntax-19-9, change the table description:

Syntax 19-9—~~Assertion s~~Severity system task syntax (not in Annex A)

In the body of Clause 19.9, change:

~~SystemVerilog assertions have has a severity level associated with any assertion failures detected. By default, the severity of an assertion failure is "error". The severity levels can be specified by including one of the following severity system tasks in the assertion fail statement:~~

~~— $fatal shall generate a run-time fatal~~ assertion ~~error, which terminates the simulation with an error code. The first argument passed to $fatal shall be consistent with the corresponding argument to the Verilog $finish system task, which sets the level of diagnostic information reported by the tool. Calling $fatal results in an implicit call to $finish.~~

~~— $error shall be a run-time error.~~

~~— $warning shall be a run-time warning, which can be suppressed in a tool-specific manner.~~

— $info shall indicate that the failure carries no specific severity.

All of these severity system tasks shall print a tool specific message, indicating the severity of the failure, and specific information about the failure, which shall include the following information:

— The file name and line number of the assertion statement

— The hierarchical name of the assertion if it is labeled or the scope of the assertion if it is not labeled.

For simulation tools, these tasks shall also report the simulation run time at which the severity system task is called.

Each of the severity tasks can include optional user defined information to be reported. The user defined message shall use the same syntax as the Verilog $display system task and can include any number of arguments.

SystemVerilog provides special text messaging system tasks that can be used to flag various exception conditions. The tasks are defined as follows:

— `$fatal` shall generate a run-time fatal error, which terminates the simulation with an error code. The first argument passed to `$fatal` shall be consistent with the corresponding argument to the `$finish` system task (see 19.2), which sets the level of diagnostic information reported by the tool. Calling `$fatal` results in an implicit call to `$finish`.

— `$error` shall indicate a run-time error.

— `$warning` shall indicate a run-time warning.

— `$info` shall indicate that the message carries no specific severity.

Each of the severity system tasks can include optional user-defined information to be reported. The user-defined message shall use the same syntax as the `$display` system task (see 20.2.1) and thus can include any number of arguments.

All of the severity system tasks shall print a tool-specific message, indicating the severity of the exception condition and specific information about the condition, which shall include the following information:

— The file name and line number of the severity system task call. The file name and line number shall be same as __LINE__ and __FILE__ compiler directives respectively.

— The hierarchical name of the scope in which the severity system task call is made.

— For simulation tools, the simulation run time at which the severity system task is called.

The tool-specific message shall include the user-defined message if specified.

In Clause 19.1, change:

Assertion severity tasks (19.9)

To

Assertion sSeverity tasks (19.9)

In Clause 16.3, change:

The immediate **`assert`** statement is a *statement_item* and can be specified anywhere a procedural statement is specified. The execution of immediate assertions can be controlled by using assertion control system tasks (See 19.10).

~~NOTE—The assertion control system tasks are described in 19.10.~~

~~Because the assertion is a statement that something must be true, the failure of an assertion shall have a severity associated with it. By default, the severity of an assertion failure is error. Other severity levels can be specified by including one of the following severity system tasks in the action block: fail statement:~~
The information about assertion failure can be printed using one of the following severity system tasks in the action block.

- `$fatal` is a run-time fatal.
- `$error` is a run-time error.
- `$warning` is a run-time warning.
- `$info` indicates that the assertion failure carries no specific severity.

The syntax for these severity system tasks is shown in 19.9.

If an assertion fails and no **`else`** clause is specified, the tool shall, by default, call `$error`, unless `$assertfailoff` is used to suppress the failure. ~~a toolspecific option, such as a command-line option, is enabled to suppress the failure~~.

In Clause 16.3, delete:

~~All of these severity system tasks shall print a tool-specific message indicating the severity of the failure and specific information about the specific failure, which shall include the following information:~~

~~— The file name and line number of the assertion statement.~~

~~— The hierarchical name of the assertion, if it is labeled, or the scope of the assertion if it is not labeled.~~

~~For simulation tools, these tasks shall also include the simulation run-time at which the severity system task is called.~~

~~Each system task can also include additional user-specified information using the same format as $display.~~

If more than one of these system tasks is included in the ~~**`else`** clause~~ action block, then each shall be executed as specified.

In clause 16.3, change:

~~The display of messages of warning and information types can be controlled by a tool-specific option, such as a command-line option.~~

The severity system tasks can be used in assertion pass or fail statements. These tasks shall print the same tool-specific message when used either in a pass or a fail statement. For example:

*assert_foo : **assert**(foo) $info("passed"); **else** $error("failed");*

In Clause 16.14.1, change:

The *action_block* shall not include any concurrent **assert**, **assume**, or **cover** statement. The *action_block*, however, can contain immediate assertion statements.

The conventions regarding default severity (error) and the use of severity system tasks in concurrent assertion action blocks shall be the same as those specified for immediate assertions in 16.3.

The pass and fail statements of an assert statement are executed in the Reactive region. The regions of execution are explained in the scheduling semantics in Clause 9.