Purpose of 1987: Change “verification statement” to “assertion statement” and make it a special term, meaning, italicize it where it defined and add it to the glossary. It needs to be clarified that all assertions are affected, immediate and concurrent, asserts, assumes, and covers. The expect statement is not included.

In addition, Mantis 2205, was combined since it references this new term to solve the issue that the LRM is ambiguous relative to whether $asserton/off/kill and assertion action controls affect all assertions or just the “assert” directives. The text that relates to the system tasks and controls now specifically references assertion statements.

In addition, because the term “assertions” is used broadly throughout the LRM, and in places like VPI that cannot be changed to “assertion statement”, the definition of “assertions” in the glossary was changed to encompass all assertion statements. I did a search for the word “assertions “to ensure that all references in the LRM make sense with this new definition and modified the few that did not.

Aligned with p1800-2008-draft 4

Replace

16.2 Overview

An assertion specifies a behavior of the system. Assertions are primarily used to validate the behavior of a design. In addition, assertions can be used to provide functional coverage and generate input stimulus for validation..

There are two kinds of assertions: concurrent and immediate.

With

16.2 Overview

An assertion specifies a behavior of the system. Assertions are primarily used to validate the behavior of a design. In addition, assertions can be used to provide functional coverage and to ensure that ~~generate~~ input stimulus that is used for validation conforms to assumed requirements.

Assertions appear as an *assertion statement* that states the verification function to be performed. The statement can be one of the following:

- — **`assert`** to specify the property as a checker to ensure that the property holds for the design
- — **`assume`** to specify the property as an assumption for the environment. Simulators check that the behavior holds while formal tools use the information to generate input stimulus.
- — **`cover`** to monitor the property evaluation for coverage

There are two kinds of assertions: concurrent and immediate.

Replace

16.14 Concurrent assertions

A property on its own is never evaluated for checking an expression. It must be used within a verification

statement for this to occur. A verification statement states the verification function to be performed on the property. The statement can be one of the following:
— **assert** to specify the property as a checker to ensure that the property holds for the design
— **assume** to specify the property as an assumption for the environment
— **cover** to monitor the property evaluation for coverage

With

16.14 Concurrent assertions

A property on its own is never evaluated for checking an expression. It must be used within an assertion statement(note to editor: add a link to the definition in 16.2) for this to occur. ~~A verification statement states the verification function to be performed on the property. The statement can be one of the following:~~
~~— **assert** to specify the property as a checker to ensure that the property holds for the design~~
~~— **assume** to specify the property as an assumption for the environment~~
~~— **cover** to monitor the property evaluation for coverage~~

Add to the Glossary (place before the definition of Verilog)

Assertion statement: An assertion statement states the verification function to be performed on the property. It includes immediate and concurrent assertions. The statement can be one of the following:
— **assert** to specify the property as a checker to ensure that the property holds for the design
— **assume** to specify the property as an assumption for the environment. Simulators check that the behavior holds while formal tools use the information to generate input stimulus.
— **cover** to monitor the property evaluation for coverage
NOTE—See 16.2 for a discussion of assertion statement.

Replace in the Glossary

assertion: A statement that a certain property must be true, e.g., that a `read_request` must always be followed by a `read_grant` within two clock cycles. Assertions allow for automated checking that the specified property is true and can generate automatic error messages if the property is not true.

With

assertion: An assertion statement that describes a behavior . ~~a certain property must be true~~, e.g., that a `read_request` is ~~must always be~~ followed by a `read_grant` within two clock cycles. The assertion directive describes the verification function to be performed. For example, an **`assert`** directive ~~Assertions allow~~ allows for automated checking that the specified property is true and ~~can generate~~ generates automatic error messages if the property is not true.

REPLACE in 16.3

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*.

WITH

When ~~Because the~~ an assertion is a statement that something must be true (e.g. an **`assert`** or an **`assume`**), the failure of such ~~an~~ assertion shall have a severity associated with it. By default, the severity of an assertion failure is *error*.

REPLACE in 16.14.2

The biasing feature is only useful when properties are considered as assumptions to drive random simulation. When a property with biasing is used in an assertion or coverage, the **`dist`** operator is equivalent to inside operator, and the weight specification is ignored.

WITH

The biasing feature is only useful when properties are considered as assumptions to validate input stimulus. ~~drive random simulation~~. When a property with biasing is used ~~in an~~ with an **`assert`** or **`cover`** assertion directive, ~~assertion or coverage~~, the **`dist`** operator is equivalent to inside operator, and the weight specification is ignored.

REPLACE in 19.10

SystemVerilog provides three system tasks to control assertions.

WITH

SystemVerilog provides three system tasks to control the evaluation of assertion statements ~~assertions~~.

REPLACE in 19.11

SystemVerilog provides six system tasks to control execution of assertion action blocks for concurrent assertions:

WITH

SystemVerilog provides six system tasks to control the execution of assertion action blocks ~~for concurrent assertions~~ that are associated with assertion statements and the `expect` statement: