

The purpose of this proposal is to clarify where concurrent assertions may appear and what kind of variables they may use. A correction was also made to state that the assertion action controls of section 19.11 apply to all assertions, not just concurrent.

Note: referenced text assumes 1995 - assertion in loops

1/28:

- Capitalize the 'a' in 'automatic' in the first bullet point. (Erik and Dmitry)
- Observe region changed to Observed region (Thomas Thatcher)
- Timeslot changed to time slot (Dmitry)

1/29:

- There is use of 'may' and 'must' at lot of places. I think some of these should be converted into 'shall'. (Manisha)

=====

REPLACE IN 16.4 CONCURRENT ASSERTIONS OVERVIEW

Concurrent assertions describe behavior that spans over time. Unlike immediate assertions, the evaluation model is based on a clock so that a concurrent assertion is evaluated only at the occurrence of a clock tick. The values of variables used in the evaluation (except for the special case of loop iterators as described in 16.14.5) are the sampled values. This way, a predictable result can be obtained from the evaluation, regardless of the simulator's internal mechanism of ordering events and evaluating events. This model of execution also corresponds to the synthesis model of hardware interpretation from an register transfer language (RTL) description.

The values of variables used in assertions are sampled in the Preponed region of a time slot, and the assertions are evaluated during the Observe region.

WITH

Concurrent assertions describe behavior that spans over time. Unlike immediate assertions, the evaluation model is based on a clock so that a concurrent assertion is evaluated only at the occurrence of a clock tick. The values of variables used in the evaluation (except for the special case of loop iterators as described in 16.14.5) are the sampled values. This way, a predictable result can be obtained from the evaluation, regardless of the simulator's internal mechanism of ordering events and evaluating events. This model of execution also corresponds to the synthesis model of hardware interpretation from an register transfer language (RTL) description.

All data referenced in a concurrent assertion, with the exception of local variables (see 16.9) and **for** (see 12.7.1) and **foreach** (see 12.7.3) loop indices, shall have a static lifetime (exist for the whole elaboration and simulation time). Similarly, concurrent assertions shall only exist in blocks whose lifetime is also static. So for example,

- Automatic variables and members or elements of dynamic variables shall not be referenced in an assertion. This includes dynamically sized variables and data in automatic tasks, functions, or blocks.
- Class methods (see Clause 8) are only active for the lifetime of the call and therefore shall not contain concurrent assertions or have its elements referenced by a concurrent assertion.
- The lifetime of a **fork...join**, **fork...join_any**, or **fork...join_none** block is limited to the execution of all processes spawned by the block, and the lifetime of a scope enclosing any fork block includes the lifetime of the fork block. **fork...join**, **fork...join_any**, or **fork...join_none** blocks therefore shall not contain concurrent assertions or have its elements referenced by a concurrent assertion.

All variables in an assertion use the value sampled in the Preponed region of a time slot with the exception that local variables and **for** and **foreach** loop variables use the current value. ~~The values of variables used in assertions are sampled in the Preponed region of a time slot, and the~~ The assertions are evaluated during the ~~Observe~~ Observed region.

REPLACE IN 19.11 Assertion action control system tasks

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

WITH

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