

22.8 Assertion control system tasks, pp 384

The LRM does indicate how the control tasks `$asserton`, `$assertoff` and `$assertkill` affect the behavior of verification statements that are placed in initial blocks. If the text is applied as it is in the LRM now, then when if `$assertoff` is issued to disable assertions during the design reset period, then assertions in the initial block would never run.

The proposal tries to remedy this situation. In summary, if there is a call to `$assertoff` before any clocking event of the assertion occurs, then the assertion will not start until a `$asserton` is issued. At that point it will execute its unique evaluation attempt.

Replace

SystemVerilog provides three system tasks to control assertions.

— `$assertoff` shall stop the checking of all specified assertions until a subsequent `$asserton`. An assertion that is already executing, including execution of the pass or fail statement, is not affected.

— `$assertkill` shall abort execution of any currently executing specified assertions and then stop the checking of all specified assertions until a subsequent `$asserton`.

— `$asserton` shall re-enable the execution of all specified assertions.

When invoked with no arguments, the system task shall apply to all assertions. When the task is specified with arguments, the first argument indicates levels of the hierarchy, consistent with the corresponding argument

to the Verilog `$dumpvars` system task. Subsequent arguments specify which scopes of the model to control. These arguments can specify entire modules or individual assertions.

With

SystemVerilog provides three system tasks to control assertions.

— `$assertoff` shall stop the checking of all specified assertions until a subsequent `$asserton`. An assertion that is already executing, including execution of the pass or fail statement, is not affected.

— `$assertkill` shall abort execution of any currently executing specified assertions and then stop the checking of all specified assertions until a subsequent `$asserton`.

— `$asserton` shall re-enable the execution of all specified assertions.

The evaluation attempt that could start in the same time step as the call to one of the tasks is affected only if the task is called before entering the reactive scheduling region. When an assertion is placed in an initial block, there is only one attempt evaluated starting at the first occurrence of its clocking event. If `$assertoff` is called before entering the reactive scheduling region that follows the first clocking event, the behavior of the assertion in an initial block is as follows:

Assuming that no clocking event occurs before `$assertoff` or `$assertkill` is executed, the assertion is not started until an `$asserton` is executed. At that point the unique attempt is evaluated starting in the time step of the nearest clocking event of the assertion. If `$assertkill` is executed on the assertion before the assertion evaluation attempt (if any) could complete, the evaluation attempt is aborted and cannot be restarted.

If the tasks are called in the reactive scheduling region, their effect takes place only in the next time step.

When invoked with no arguments, the system task shall apply to all assertions. When the task is specified with arguments, the first argument indicates levels of the hierarchy, consistent with the corresponding argument to the Verilog `$dumpvars` system task. Subsequent arguments specify which scopes of the model to control. These arguments can specify entire modules or individual assertions.