CH119

Neil

1. Section 12.7.1 wait_order(), 4th paragraph, begins with "The action taken.."

   The else statement is referred to as the fail statement. Didn't we reverse this in the ASWG such that the first statement is the fail statement?

2. Section 12.6, 3rd paragraph, next to last sentence

   In this section of the sentence "...or use the wait()..." get rid of the word 'use'.

3. Section 12.6.3, 3rd paragraph, 2nd sentence

   "The trigger property..." could be re-written to be "The triggered event property..."

4. Section 12.6.3

   There are a couple of words that got hyphenated in the previous write up.
   They now need to be un-hyphenated.

   synchro-nization
   exam-ple

5. Section 12.8.1

   exam-ple

Michael

1. 3.9: "event variables can be assigned another event variable..." should be "event variables can be assigned to other event variables..."

2. 12.6 Paragraph 1 says events can be dynamically allocated and reclaimed. Can you use "new" to dynamically create an event? How is dynamic allocation achieved?

   paragraph 3 adds "use" to "use the wait construct" - I think it should be "using the wait construct".

   paragraph 3 says "Events are always triggered using the -> operator". Not true! We now have non-blocking event triggers and the ->> operator.

3. Francoise has an objection to the statement that waiting for a null event is undefined - furthermore, section 3.9 says, "If the event is assigned null, the event becomes nonblocking, as if it were permanently triggered." We really ought to pick one or the other :)

Francoise

12.6.3
typo: exam-ple (first paragraph after the verilog code example)

12.7.1 wait_order()

"The wait_order construct suspends the calling process until all the specified events are triggered in the given order (left to right), *or any * of the un-triggered events is triggered out of order and causes the operation to fail."

The above sentence needs to be rewritten more clearly as 2 sentences. The wait_order construct suspends the calling process until all the specified events are triggered in the given order (left to right), *or any * of the un-triggered events is triggered out of order. The wiat_order task will fail is any of the untriggered events in the list is triggered out of the specified order.

12.8.1: typo "For example:"

12.8.2:
Did we really vote to make this undefined? I think users will complain about vendor implementation differences if we leave this undefined. They will use the null assignment to cause the event variables to be reclaimed but then they will have runtime errors or different behavior of their code which uses these events unless they enclosed them with a named block and disable that block.

```
event E1 = null;
          @ E1;               // undefined: may block forever or not at
all
          wait( E1.triggered );     // undefined
```

CH121

Neil

a. Section 20.9

I might be wrong, but I thought there was more disagreement on this section in the last meeting than is reflected here.

b. Section 13.2, last sentence

The wording here isn't quite right.

"Unless a hierarchical_expression is used, both the signal and the clocking domain name shall be the same."

It isn't the name of the clocking domain itself that is the same, rather it is the name of the clocking_item that will be the same as the signal name.

    "Unless a hierarchical_expression is used, both the signal and
    the clocking_item names shall be the same."

c. Section 13.14, examples

    Shouldn't all of these be using <= in place of =? (see CH113)

d. Section 5.3

    "The arguments to the new method must be constant literals."

    Aren't expressions also allowed? (see text of 5.3 in LRM).

Francoise

section 13.2
    typo: "The signal_identfier"

CH122

Neil

1. Section 5.4

    expres-sions

Francoise

section 5.4:
    typo "run-time expressions"

section 15.1
    typo: "synchronized, "

section 15.2:
    typo "SystemVerilog "

section 15.2:
    the title of the bnf for program blocks is wrong.
    "Syntax 15-1—Module declaration syntax (excerpt from Annex A)"

    In the program example the comment below should say that p1 and
    p2 are instantiated once iff the module test which contains them
    is instantiated once only. Because if module test is instantiated
    multiple times in the hierarchy so are the programs p1 and p1. //
    p1 and p2 are implicitly instantiated once

section 15.3:
    typo "independent"

section 15.4
    typo : out-puts

section 15.5:

If the same task is called from within a program then the statement S1 will execute when the Reactive region is processed, after variable b may been updated by nonblocking assignments. Statement S2 always executes when the first Active region is processed.

I would modify above paragraph as:

after variable b may *have* been updated by nonblocking assignments made by thedesign calling the task. Is the second sentence correct? I understood that NBA inside a task called from a program would execute in the next NBA region.

Michael

1. Section 5.5

   5.5 is being changed to, "Note that automatic or dynamic variables cannot ... be written with a nonblocking assignment." - I'd like to re-write the last part to "be assigned to with a non-blocking assignment".