

Issues sub-committee Update

6 Aug 2007

The Topics Currently Discussed

- New architecture for vunits
- Formal Semantics of Local Variables
- Semantics of verification directives

New architecture for vunits

Name space, scoping rules

- if vunit A inherits vunit B, or vunit A is bound to HDL module or instance B, then
 - the names visible in B become visible in A
 - A may override assignments made to objects in B

New architecture for vunits

- Definitions of vmode and vprop according to the above
- Tools are free to go beyond what the LRM requires, for example
 - to assume a given assertion in order to prove another assertion

New architecture – reusable verification IP

- parameterized vunits
- A parameterized vunit need not be in the same flavor as the context in which it is used
- Instantiating a parameterized vunit in one flavor with parameters of another flavor -
same as cross-language instantiation of an HDL module
- A new kind of verification unit – vpkg (temporary name)
 - a vunit that contains only declarations
 - typically parameterized sequence and/or property decls

New Architecture - Status

- It remains to:
 - Define the syntax for parameterized vunits
 - write the new text for the LRM
 - More can be found in
<http://www.eda.org/ieee-1850/Issues/Group-C.1.html>

Formal semantics of Local variables

- There are two proposals for formal semantics
- The difference is in overlapping operators: $\vdash\rightarrow$ and $:$
- Proposal A resembles SVA's semantics
 - $\{\text{new}(i<-0) [*]; a; b[*]; c, i++\} \vdash\rightarrow \{d; e; f\}$ i will be incremented even if d doesn't hold
- In proposal B
 - $\{\text{new}(i<-0) [*]; a; b[*]; c, i++\} \vdash\rightarrow \{d; e; f\}$ d is evaluated first. i is not incremented if d does not hold.

Formal semantics of Local variables

- In the SC we decided to continue with proposal A
 - closer to SVA
 - nicer semantics
- Difference of Proposal A from SVA's semantics:
 - Simpler semantics
 - The scope of a variable is defined syntactically
 - distributivity is not broken
 - hide() - preserves the notion of intersection of && for a corner case
 - the hiding cannot be done automatically

Local variables - Status

- Looking for a keyword to replace hide()
- Refinement of the formal semantics
- More can be found in
<http://www.eda.org/ieee-1850/Issues/Group-P.1.html>

Semantics of verification directives

Clarified the semantics of restrict

- New: restrict! r - The path models tightly the SERE
- restrict r - assume {r;EOP} or assume{r;false}.
- Some prefix of the path matches the SERE r tightly - assume r!

Verification directives

Decided to deprecate assume-guarantee and restrict-guarantee

- not useful
- The use model determines how each assume/restrict is used
- Remaining
 - Semantics of assume