VHDL-200x Telecon Meeting
15 Apr 2004

Attendees:
Tim Schneider
John Ries
Chuck Swart
Charlie Guy
Deepak Pant
Ajay Varikat
Stephen Bailey
Peter Ashenden
Erich Marschner
Francoise Martinolle
John Shields

Discussion:

1) Jim started meeting in Steve's absence and read IEEE-SA Patent policy and DASC modifications read to meeting attendees.

2) 1164 Integration
   Reviewed std_logic_1164 change list.

   Jim to integrate 1164 modifications into Fast Track changes.

   Peter to send Jim "golden" version of package std_logic_1164 for updates.

   std_logic_1164 package details will be included into 1076 document as a new chapter or as part of the chapter on the standard packages.

   John Ries and Chuck Swart are looking into leveraging the advantages of the integration of std_logic_1164. John Ries is writing a proposal for case (either the current one or a new one) that understands the std_logic value '-'. They will also look into specifying that the type conversion functions in 1164 as locally static.

3) Interfaces
   Discussed different possibilities for interfaces. Protected Types with shared variables, named events, signals.

   Do not think we can accomplish this in the next 4-6 weeks, so it will need to be deferred.

   Steve Bailey to write up a requirements specification so we will be ready to move on this for the language revision that follows vhdl-200x-ft.

4) Generics
   Discussed adding generics to packages. We want capabilities that will address Bhasker's TBV team's need for generic container objects (FIFO, associative array, etc.). Since we do not have

time to define all the enhanced generic capabilities we ideally want due to the constraints of time for the FT revision, we also want to be careful that the proposal will be forward compatible with planned future directions.

Peter has agreed to provide a proposal that meets these goals.

5) Library IEEE
We determined that all the LRM should specify is IEEE standard packages and which library they are located. If the 1076 standard identifies a package, then it is required of implementations to support them. (As all packages are written in legal VHDL, required support should be a non-issue. Optimized support of any package is up to the implementation/vendor.)

Note: This means that the issue of non-IEEE standard packages being present in library STD or IEEE will not be directly addressed in the LRM. It was the consensus of those at the meeting that the LRM should not directly reference packages that are not IEEE standards.

6) PSL
We tried to identify as many issues as possible that need to be addressed to properly incorporate PSL (by reference) into VHDL. These issues included:

a. vunits as a VHDL design unit or some form of configuration enhancement to allow binding of verification units (entity/architectures that contain assertion/coverage checkers) to the design hierarchy.
b. When are values of objects referenced in PSL sampled? We should define this to ensure portability. As VHDL does not suffer from the race conditions, this should be relatively straight-forward.

   Clocked sequences are activated when the clock expression triggers (just like a VHDL wait).

   PSL directives and endpoints are evaluated in that cycle prior to evaluation of user processes. This ensures no race conditions with shared variables that are referenced in the PSL being changed by a user process after the clock trigger but prior to PSL evaluation.

   Unclocked sequences are evaluated whenever a signal referenced in the sequence has an event. (Again, PSL is evaluated prior to user processes.)

c. Erich Marschner reported that the labeling and report clauses being added in the PSL 1.1 LRM are consistent with existing VHDL syntax.

   However, PSL does not define a severity clause. Therefore, there is some difference in the PSL assert syntax and VHDL syntax. The optionality of the severity clause in VHDL makes this more manageable. I could see where nonnormative guidance could be provided to not use the severity clause if strict portability with PSL is desired.

I (Steve Bailey) noted after the meeting that VHDL allows concurrent assertions to be postponed. Preserving the postponed capability would provide a Verilog-like "strobe" evaluation of the PSL assertion. This could be useful in some contexts. Again, such usage would not conform to strict PSL portability and we could note this as well.

d. SystemVerilog Assertions (SVA) allows specification of "action blocks" when an assertion passes or fails. Erich reported that the Accellera FVTC considered this but decided that assertions should remain passive. It seems reasonable that we not attempt to build "action blocks" infrastructure within VHDL at this time. We can respond in a future release if users request it.
e. SVA also defines procedural assertions (assertions that are embedded within an always block – process). SV defines semantics for the equivalent concurrent assertion. Again, we decided to not provide this capability now. We can revisit it for a future revision, if users request it. (NOTE: This has nothing to do with the existing VHDL sequential assertion statement.)
f. Francoise Martinolle has agreed to work on defining an assertion/coverage API. She will target finishing a draft by DAC or shortly after.
g. PSL supports const parameters (which must be integers), Boolean parameters (which need not be constant and include bit/std_logic boolean equivalence) and sequence parameters. These parameters apply to parameterized sequences and properties. Issue is how to resolve the different parameter passing mechanism/limitations, etc.
h. Need to define where PSL constructs can appear in VHDL. We could treat them as specifications (and not declarations) and allow specifications in statement parts – users have requested such an enhancement to allow attribute specifications near the thing being attributed.

   If they are treated as declarations, then either users must place them in declarative parts or we need to allow all declarations outside of declarative parts.

   Maybe we need to define a new "class" of language construct to allow PSL to be specified in declarative and statement parts without worrying about side-effect changes on existing declaration and specification rules.
i. PSL's assume and restrict directives are used for stimulus generation. (Formal tools use them to reduce the input or state space possibilities which is a form of stimulus.) In simulation, we can define that they imply an equivalent assert to ensure assumptions/restrictions are held. Or, we can define that they are ignored. There seems to be consensus for treatment as an assert.

7) Pathname issue that came up in VHPI LRM editing
   Peter Ashenden and John Shields will visit the language's use of path names and see if we can define a canonical form within the standard that meets all needs.

The result could be different from the current output of 'Path_name and 'Instance_name. If that should occur, these attributes would remain unchanged for backward compatibility. New attributes could be defined that comply with the new canonical forms, if needed.

Erich mentioned that PSL has issues with ':' as a path separator since colon is used in Verilog for ranges and PSL needs to work with both languages. He also mentioned the general desirability to have hierarchical naming conventions that are standard across both VHDL and Verilog as this would improve the ability to write language-neutral PSL and would be helpful for the many users who do mixed HDL simulations every day.

8) Chuck Swart mentioned that he and Jose Torres have been discussing merging the math packages (1076.2) into 1076. There was concern about whether doing so would require the packages to be supported to be 1076 compliant. In general, it is possible to identify optionally supported capabilities. However, as long as the packages are fully specified in VHDL, the only issue should be how well an implementation supports it. That is, what optimizations and/or other tool-specific capabilities may exist in regards to the package.