

Accellera PSL - Status and Plans



Harry Foster, Chair
Erich Marschner, Co-Chair

Accellera Board Meeting - 10 Sep 3003

Committee Charter and Goals

- Charter
 - The FVTC is responsible for the maintenance and evolution of the Accellera Property Specification Language.
- Goals
 - Creation of the Accellera PSL Standard (done)
 - Alignment of PSL and SVA (in progress)
 - Extension of PSL for greater utility (starting)
 - Adoption of PSL as an IEEE standard (starting)



Subcommittees

- **Alignment Subcommittee**
 - Chair: Erich Marschner
 - Participants: JohnH, SurrendraD, CindyE, DanaF, RoyA, BassamT, JosephL, HarryF, AmbarS, SitvanitR
- **LRM Subcommittee**
 - Chair: Harry Foster
 - Participants: SteveW, SitvanitR, JohanA, GalV, BenC, AvigalO
- **Extensions Subcommittee**
 - Chair: Ben Cohen
 - Participants: BenC, AlokJ



LRM Subcommittee

- **Goal: Create the 1.1 LRM document**
 - Address issues raised in the 1.01 LRM
 - Correct errors and typos in the 1.01 LRM
 - Fold in the final work of the Alignment and Extension Subcommittees to create the 1.1 LRM
- **Process:**
 - Held kickoff meeting in May
 - Now conducting weekly working meetings to address issues and make corrections
 - Issues list can be view at:
 - > Alignment Issues (www.eda.org/vfv/docs/alignment.issues.pdf)
 - > Extension Issues (www.eda.org/vfv/docs/extension.issues.pdf)
 - > LRM Issues (www.eda.org/vfv/docs/LRM.issues.pdf)



Alignment Subcommittee

- Goal: Alignment of PSL and SVA
 - Priorities:
 - > 0. maintain a sound formal semantic foundation
 - > 1. avoid different semantics for same syntax
 - > 2. allow same syntax for same semantics
- Process:
 - Weekly meetings since May 21
 - Formal Semantics Analysis/Mapping/Alignment
 - Syntactic Mapping / Extensions Proposals



Alignment - Steps

- Define boundaries of alignment task (**Done**)
- Define formal semantics of SVA, PSL independently using same concepts (**Done**)
- Identify issues that need to be addressed (semantic, syntactic) (**Done**)
- Refine both SVA and PSL semantic definitions to fix semantic problems and align (**Done**)
 - e.g., align semantic concepts (e.g., strong/weak/neutral semantics)
- Prove equivalence between SVA and PSL semantics for base operators (**Done**)
- Prove equivalence between SVA and PSL derived operators (**TBD**)
- Define syntactic mappings based on proven semantic equivalences (**Started**)
 - 1-to-1 wherever possible (with exceptions/caveats)
 - map from SVA -> PSL first; possibly from PSL subset -> SVA also
- Define syntactic extensions that can simplify mappings (**Starting**)
- Consider merging the formal semantic definitions to create a single common definition (**TBD**)



Alignment - Boundaries

- Includes:
 - refinement of PSL semantic definition as planned for PSL v1.1 in Dec 2002
 - cleanup or adjustment of both semantic definitions to facilitate mapping
 - temporal operators (for which there is a formal semantics)
 - assert directive
- Assumes:
 - both SVA and PSL in a SystemVerilog context (i.e., SystemVerilog 'flavor' of PSL)
 - standalone statements, with any influence from the environment folded in
 - > PSL default clock / SVA clock extraction has been made explicit
 - > disabling of processes in System Verilog not considered
 - mapping from SVA LRM syntax to SVA abstract syntax is defined
- Excludes:
 - procedural assertions (none in PSL)
 - local variables in sequences (none in PSL) (but could perhaps be mapped to forall)
 - strong Booleans in PSL (not in SVA)
 - declarations, verification units, etc. (not formally defined)
 - directives other than assert directive



Alignment - Issues

- **Scope issues:**
 - PSL is larger - VHDL flavor, keyword temporal operators, OBE; more directives
 - vunits & inherit in PSL don't map easily to SVA
 - coverage is not well defined in either SVA or PSL
- **Semantic issues:**
 - need for strong/weak/neutral finite semantics in both PSL and SVA
 - abort vs. disable semantics
 - suffix implication semantics
 - within - different meanings in PSL, SVA
 - @(clk) - different meanings in PSL, SVA
- **Syntactic issues:**
 - `:/ ;` vs. `##0 / ##1`
 - operator precedence issues (fusion vs. concatenation)
 - leading delay in SVA vs. PSL
 - `rose/fell` vs. `$rose/$fell`
 - `inf` vs. `$`
 - `[=]`, `[->]` vs. `[*=]`, `[*->]`



Strong/Weak/Neutral Semantics

- For simulation (finite traces), we need strong/weak/neutral semantics
 - **weak** for constrained random simulation, because test will stop at an arbitrary place, and pending obligations should not be treated as failures
 - **strong**, for directed tests, because test will presumably run a full scenario to completion, and pending obligations should be treated as failures
 - **neutral**, the traditional LTL finite trace semantics, which is effectively a compromise between the weak and strong cases
- PSL originally addressed strong semantics via strong clocks
- Dec 2002 paper proposed addition of strong/weak/neutral semantics in PSL v1.1
- Mar 2003 SVA formal semantic definition adopted strong/weak semantics, but left out neutral semantics (an oversight)
- We've refined both PSL and SVA formal semantics definitions to include strong/weak/neutral semantics



Strong/Weak/Neutral Result “Modes”

- Holds strongly:
 - no bad states have been seen
 - all future obligations have been met
 - the property will hold on any extension of the path
- Holds:
 - no bad states have been seen
 - all future obligations have been met
 - the property may or may not hold on any extension of the path
- Pending:
 - no bad states have been seen
 - future obligations have not been met
 - (the property may or may not hold on any extension of the path)
- Fails:
 - a bad state has been seen
 - (future obligations may or may not have been met)
 - (the property may or may not hold on any extension of the path)



Refinement of PSL Formal Definition

- Done:

- restatement of abort semantics [planned Dec 02]
- refinement of finite trace semantics (strong/weak/neutral) [planned Dec 02]
- change to strengthless clocks
- addition of strong booleans
- definition of next suffix implication: $\{r1\} \mid=> \{r2\} == \{r1; true\} \mid-> \{r2\}$
- addition of non-degeneracy requirement for both LHS and RHS seres in suffix impl.
- alignment of formal syntax with BNF
- refinement of definition of $r[*0]$
- refinement of clock tick definition
- refinement of strong suffix implication definition (derivation from $\{r\} (f)$)

- Pending:

- refinement of weak suffix implication definition (derivation from $\{r\} (f)$)
- possible definition of FL_Property ::= $\{r\}$, where $\{r\} == (! (\{r\} (FALSE@TRUE)))$

- Impact:

- changes are virtually transparent to users - relate to extreme corner cases at worst
- strong/weak/neutral semantics change simplifies handling of finite traces



Formal Mapping - Unclocked Sequences

- DEFINITION 1: Let b be a boolean expression, and let R, R_1, R_2 be unclocked SVA sequences.
 - $M(b) = b$
 - $M((R)) = \{M(R)\}$
 - $M((R_1 \##1 R_2)) = \{M(R_1) ; M(R_2)\}$
 - $M((R_1 \##0 R_2)) = \{\{M(R_1)\} : \{M(R_2)\}\}$
 - $M((R_1 \text{ or } R_2)) = \{\{M(R_1)\} | \{M(R_2)\}\}$
 - $M((R_1 \text{ intersect } R_2)) = \{\{M(R_1)\} \&\& \{M(R_2)\}\}$
 - $M(R[*0]) = M(R)[*0]$
 - $M(R[*1:\$]) = M(R)[+]$
- Proof for each case
- Derived operators remain to be mapped/proven



Syntactic Mapping - SVA ## to PSL ; :

- For sequence operands SVA Os, Os1, Os2, and equivalent PSL Op, Op1, Op2,

- Prefix:

- SVA (##0 Os) => PSL (Op)
- SVA (##1 Os) => PSL ([*1] ; Op)
- SVA (##N Os) => PSL ([*N] ; Op)

- Infix:

- SVA (Os1 ##0 Os2) => PSL ({Op1} : {Op2}) [1][2]
- SVA (Os1 ##1 Os2) => PSL (Op1 ; Op2)
- or PSL ({Op1} : {[*1]} ; Op2)
- SVA (Os1 ##N Os2) => PSL (Op1 ; [*N-1] ; Op2) [3]
- or PSL ({Op1} : {[*N]} ; Op2)

[1] Note the braces required on the PSL operands - placed to preserve original precedence.

[2] Both SVA, PSL require both operands to be non-empty.

[3] N must be >0, and for the alternative forms, Op1 must be non-empty



Possible Syntactic Extensions to PSL

- Suppose we define the following “syntactic sugar” operators in PSL:
 - (prefix case) $##K r \implies [*K] ; r$ if $K > 0$, else r
 - (infix case) $r1 ##K r2 \implies \{r1\} : \{[*K]\} ; r2$ if $K > 0$, else $\{r1\} : \{r2\}$
- Then the mapping becomes simpler, augmented by the new definitions:
for sequence operands SVA Os , $Os1$, $Os2$, and equivalent PSL Op , $Op1$, $Op2$,
 - Prefix:
 - SVA (##0 Os) \implies PSL (##0 Op) \implies PSL (Op)
 - SVA (##1 Os) \implies PSL (##1 Op) \implies PSL ([*1] ; Op)
 - SVA (##N Os) \implies PSL (##N Op) \implies PSL ([*N] ; Op)
 - Infix:
 - SVA ($Os1$ ##0 $Os2$) \implies PSL ($Op1$ ##0 $Op2$) \implies PSL ({ $Op1$ } : { $Op2$ })
 - SVA ($Os1$ ##1 $Os2$) \implies PSL ($Op1$ ##1 $Op2$) \implies PSL ({ $Op1$ } : {[*1]} ; $Op2$)
 - SVA ($Os1$ ##N $Os2$) \implies PSL ($Op1$ ##N $Op2$) \implies PSL ({ $Op1$ } : {[*N]} ; $Op2$)



Status of the Mapping Document

- Basic operators mapped, and mappings proven
- Not yet addressed:
 - derived operators
 - matched, ended, first match, edge operators, \$past
- Needs extension to create syntactic mapping:
 - select most intuitive equivalences (of the many that exist)
 - extend to derived operators
 - > in particular, fusion and concatenation
 - figure out how to express 'M' and 'T' functions
 - make the mapping bidirectional (i.e., add PSL to SVA mapping)
- For mapping from PSL to SVA, we would need to identify the PSL subset that can be mapped
 - would stick to SEREs and SERE implication, with negation and top-level aborts
 - won't attempt more subtle mappings from LTL and CTL operators



Refinement of SVA Formal Definition

- Done:
 - addition of finite neutral semantics (27 minor changes)
 - substantive corrections (3 changes)
 - wording change to simplify SVA/PSL alignment proofs (1 change)
 - clarification (1 change)
 - font correction (1 change)
- Pending:
 - correction for missing 'bar' in first-match semantics
 - addition of non-degeneracy requirement
 - addition of strong/weak/neutral mode explanation
- Impact:
 - changes have essentially no impact on users
 - strong/weak/neutral semantics clarify how implementors should deal with finite traces



IEEE Adoption of PSL

- **VHDL 1076-200x**

- Assertions subcommittee has decided to incorporate by reference the Simple Subset of PSL (VHDL Flavor) into VHDL 200x
- This will extend the existing Concurrent Assertion Statement capability

- **Verilog 1364-2005**

- A proposal has been made to incorporate by reference PSL (Verilog Flavor) into Verilog 2005.
- This would probably end up being the Simple Subset as for VHDL.
- This would lay the groundwork for adoption of System Verilog Assertions at the appropriate time

- **Standalone IEEE standard**

- PSL v1.1 should be ready to propose as an IEEE standard, if the Board decides to do so

