Myths and Realities of SVA and PSL

• **Is it a war?**
  – **Not from Accellera viewpoint**
    • Each language services different needs
    • PSL supports multi-flavors for multi-HDLs
    • SVA supports SystemVerilog
      – Actually does / can support multi-HDLs, 1 ABV flavor
  – **EDA Vendors have different view**
    • Retain customer base
      – Early adoption of ABV linked to retention
    • Add new customers
    • Provide more language support
Myths and Realities of SVA and PSL

• How do customers feel?
  – Ambivalent
    • Mixture of opposite feelings
    • Uncertain
  – Verilog customers
    • SV and SVA natural progression
    • Users need multi-HDL, 1 flavor ABV
  – VHDL customers
    • Tempted toward PSL but leery
    • Awaits VHDL200x with caution
    • Must understand all issues and realities
Issues with PSL and SVA?

• **Maturity and support**
  - Users want stability in the language and tools
  - Users do not want to be debuggers for vendors

• **Multi-HDL support**
  - Users now are multi-lingual
    • Verilog, VHDL, C, SystemVerilog, SystemC

• **Multi-flavor support for ABV**
  - Favors users who are uni-lingual
  - Impacts IPs, tools, users’ background

• **Description of language's expressiveness**
  - Constructs of the assertion language
  - Advantages and limitations of each
Issues with PSL and SVA? (2)

- **Power of base HDL**
  - Role of base HDL into ABV language

- **Verification environment**
  - Coverage
  - Debugability
  - Reactive testbench
Issues: Maturity and support

- **Standardization**
  - PSL Accellera: LRM 1.01 April 2003, LRM 1.1 DAC 2004
  - SVA Accellera: SV 3.1a DAC 2004
  - IEEE: Both Targeted for June 2005

- **Many Vendors support**
  - Simulation
  - Formal verification

- **The BIG questions:**
  How much of the LRM **IS** supported?
  How much of what is supported is sufficient?
  - Very rapid and competitive progress for both languages
  - Constantly building upon known models / algorithms
  - Constantly improving support of language subsets
Issues: Multi-HDL / Multi-flavor

- Real need because of variety of applications
- **PSL**: Multi-flavored, syntax targeted toward each flavor
  - Reduces need to learn new syntax
    - Familiarity with targeted HDL
    - Need to learn PSL syntax and rules
  - Complicates design of IPs and mix of languages
- **SVA**: Originally targeted for SystemVerilog
  - Can support in a uni-flavored manner multi-HDLs
    - External module with `bind` to HDL module
    - External module with hierarchical access to variables
    - External on top of SystemC
    - Inline code with pragmas
  - Need to learn SVA syntax and rules
**Issue: Description of assertions and choice of base HDL**

- **Base HDL impacts style of assertions**
  - **SV:** interfaces, associative array, semaphores, queues, multi-dimensional arrays, structures
  - **VHDL’95:** multi-dimensional arrays, structures
  - **VHDL’200x:** toward standardization, features similar to SV

- **Base HDL impacts coverage**
  - **SV + SVA, SV + PSL:** Control and data coverage (e.g., bins)
  - **PSL (VG, VHDL, GDL):** Control coverage only

- **Choice of ABV impacts quality of assertions**
  - Constructs impact style and capabilities
    - **SVA variables:** In sequences and properties
      - Counters for dynamic range calculations, object count, data hold for future comparisons, data calculations
    - **PSL forall:** In properties only
      - Limited to data hold for future comparisons
Issue: Verification environment

- **ABV impacts design of verification environment**
  - **Debugability**
    - **PSL**: Rely on tool interface
    - **SVA**: Display of user-defined messages
      - Control of severity level
      - Call to user defined task
      - On/OFF switch for assertions
  - **Reactive testbench**
    - **PSL**: Acts as a monitor only
      - Cannot impact testbench
    - **SVA**: Call to user-defined tasks
      - * upon success or failure of assertion
      - Modify testbench variables
      - * change flow of verification
Differences between PSL and SVA

- **Common:** Many features supporting ABV and multi-clocking
- **Flavors**
  - PSL: Verilog, SystemVerilog, VHDL, GDL
    - GDL a place holder, not well defined
  - SVA: Integral part of SystemVerilog,
    - Can support Verilog, VHDL, SystemC(?) with
      - Module bind and pragmas (inline a la PSL)
- **Sensitivity**
  - PSL: Level and clock sensitive
    - Assertion scheduling dependent on language flavor
  - SVA: Clock sensitive
    - Assertion scheduling and processing in SystemVerilog
- **Property primitive**
  - PSL: SERE, non-SERE
    - SERE requires the use of curly braces {{..}}
  - SVA: Everything is SERE
    - NO curly braces in properties
Differences between PSL and SVA (2)

- Local variables in properties
  - PSL: None directly, limited emulation with the `forall` construct
  - SVA: Provides flexibility (value compare, iteration counters, etc)

- Local variables in sequences
  - PSL: None
  - SVA: Provides flexibility (value compare, iteration counters, etc)

- Abort
  - PSL: Asynchronous, Nested aborts allowed
    - IEEE PSL to provide synchronous
  - SVA: Asynchronous, only at top level
    - can be synchronous with `triggered` method

- Immediate / Concurrent / in block / Expect
  - PSL: Concurrent only
  - SVA: Immediate, concurrent, in Procedural Block, expect
    - Enables embedded assertions in `always` blocks
Differences between PSL and SVA (3)

- Action blocks in assertions upon success or failure
  - PSL: None
  - SVA: Full support with user defined severity levels
    - Display messages
    - Modify variables for dynamic testbench control

- Assertion control (ON/OFF/Kill)
  - PSL: None, must use the *abort* in every property
  - SVA: Yes, eliminates false reporting of errors during initialization

- Binding
  - PSL: With vunit, need design interfaces
    - Need HDL shell if design interfaces are not known
  - SVA: Bind of verification module to DUT
    - Design interfaces not needed to construct verification module

- Recursion in properties
  - PSL: None
  - SVA: Yes, can be used to create complex properties
Differences between PSL and SVA (4)

- **Statistical distribution in properties**
  - PSL: None
  - SVA: Yes, in sequences and properties

- **Operator strength**
  - PSL: Strong and weak operators
    - implication, until, before, within, next
  - SVA: Four levels of satisfaction:
    - Holds strongly, holds, pending, fails

- **Verification directives**
  - PSL: assume, assume_guarantee, restrict, restrict_guarantee, cover, fairnes, strong fairness
  - SVA: assume, assert, cover, expect