

SystemVerilog: a Training Vendor's Perspective

Tim Corcoran
VP of Engineering



Who we are

- **WHDL teaches a broad range of languages (broad)**
 - SystemVerilog
 - Verilog / VHDL
 - SystemC
 - C++
 - Other languages
- **Teaching since 1993 (deep)**
- **Uniquely positioned to present SV in context**

Sources I've drawn upon

- **Observations from our own customer training**
 - SV End-User training
 - SV EDA training
 - Consulting
- **Input from**
 - End users
 - EDA companies

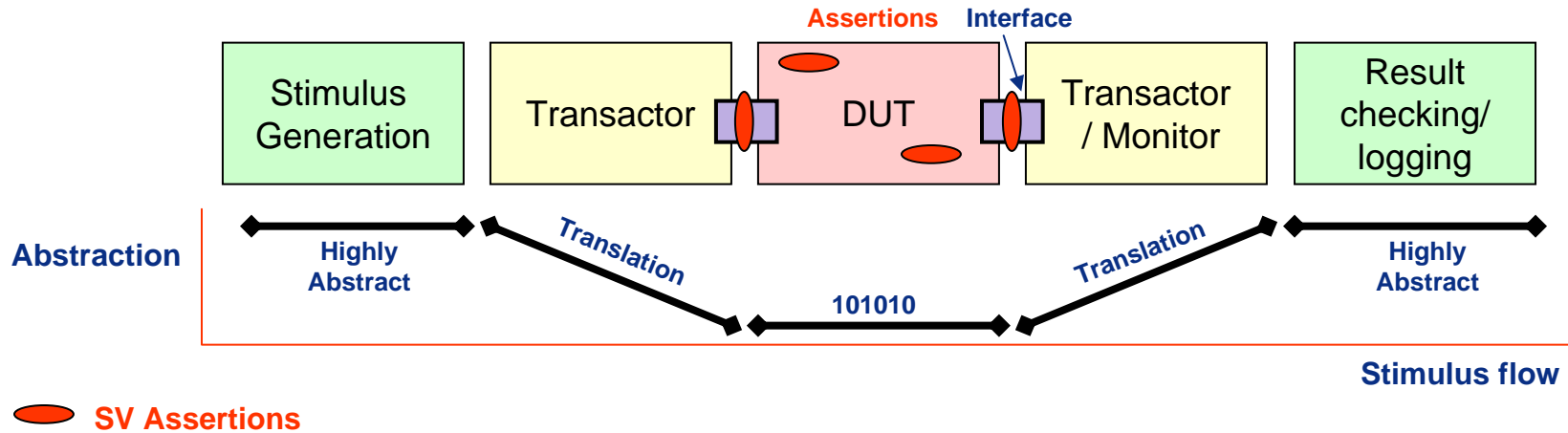
What attracts users to SystemVerilog today?

- **#1 reason: Verification**
- **#2 reason: Assertions**
- **#3 reason: Object Orientation**
- **#4 reason: Randomization / Constraints**
- **#5 reason: Functional Coverage**

Verification

- **SV supports:**
 - Modern syntax
 - Single language for design/verification
 - TB/DUT communication issues
 - ◆ Program Blocks & Clocking Domains
 - Thread spawning and control
 - Interfaces
 - Transaction-level modeling and verification
 - Packages

Verification - Transaction Based Verification



■ TBV means

- Earlier system level verification
- Adapters (transactors) buffer abstraction levels
- Testing is done at human-friendly level (e.g. read, write, dma tasks)

Assertions

■ SV supports:

- Assertion based Verification
- Dynamic feedback (constraint updating?)
- SVA coding style (Do's and don'ts)

■ Missing:

- A common coding style for assertions to maximize tool use (Sim, Synth, Formal)

Very Important!

Assertions – Observations

- **Assertion based methodology addresses the increasing difficulty of functional verification**
- **Rapid advances in standard languages and new verification tools are making assertions more practical and effective**
- **Assertions can be reused across a broad range of verification tools**
 - Simulation, Formal, Coverage and testbench
 - Emulation, Fast prototype, etc....

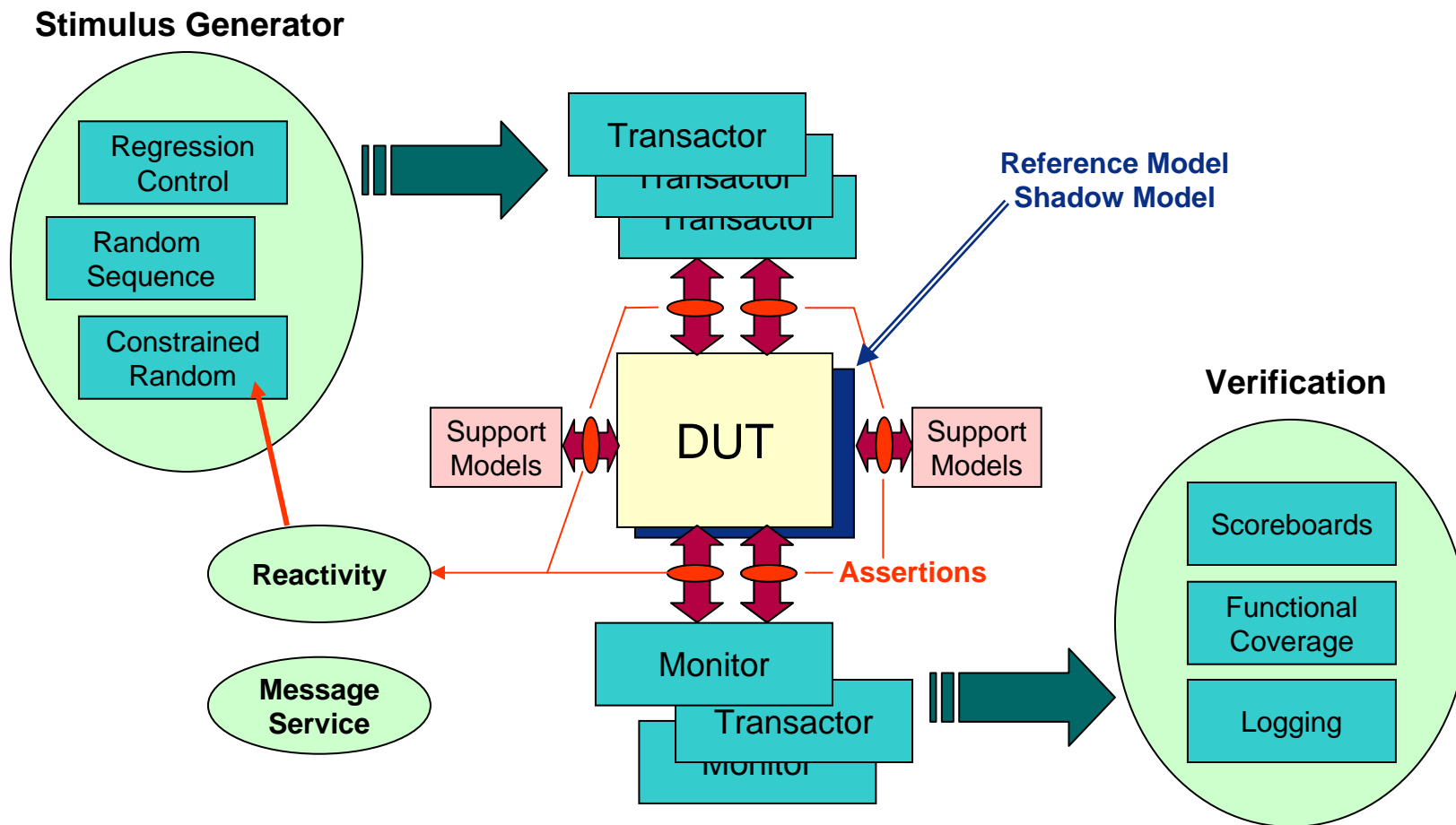
Object Orientation

■ SV supports:

- Packetized structures
- Reusable transactors (base and derived classes)
- Verification monitors (base and derived classes)
- Inter-transactor/monitor communication (mailboxes etc)
- Inheritance
- Polymorphism
- Object destruction (memory reclaim)
- Verification Framework

Object Orientation - Verification Framework

Modern Verification combines many techniques/ideas to form a reusable framework



Randomization / Constraints

■ SV supports

- New Test Methodologies: Exhaustive, directed, random, constrained
- Randomization of data and control paths
- Dynamic constraint modification
- Seeding and solver state save/reload

Functional Coverage

- **SV Supports:**
 - Code Coverage & Functional Coverage
 - Coverpoints / groups / bins

Summary

- **Verification is the big attraction to SV**
 - Single unified language
 - Transaction-level Modeling & Verification
 - Reusability of OO structures
 - SV users are hungry for hi-level capabilities
e.g. SC-Channels (SV Interfaces with blocking methods)
- **Designers are “interested” in new SV RTL feature set**
- **Assertions need a standardized coding style**
 - Assertions must be usable by all tools in flow
- **Trend is towards higher level of modeling to manage complexity**