

System Verilog Assertions

Language and Methodology

Training Agenda

Ashok B. Mehta
DefineView Consulting

<http://www.defineview.com>

© 2006-2016

Training :: Abstract

- **Abstract**

- System Verilog Assertions (SVA) is a powerful subset of the IEEE 1800 System Verilog standard.
- Its hardware oriented concurrent semantics allow for intuitive development of complex multi-clock domain checkers to catch those elusive bugs at the source (white box observability).
- SVA also allows for clean separation of DV logic from the design logic and allows for parameterization of properties resulting in a modular reusable methodology.

- **Course Highlights**

- Each operator/feature is explained in detail using comprehensive examples, timing diagrams and simulation logs.
- Real life applications are discussed to put it all in perspective.
- A reference grade handout book is provided to the class. It has comprehensive detail on each page that can serve as excellent reference material for future.
- Labs are geared to solidify understanding of key concepts using application oriented designs.
- The course includes IEEE 1800 - Features from LRM 2005, 2009 and 2012.

Training :: Agenda

- **Introduction to Assertions**
 - What's an assertion? Why can't I just use Verilog?
 - Advantages of Assertion Based Verification (ABV) .
 - Assertion Based Verification (ABV) Methodology components
- **System Verilog Assertions :: Syntax and Semantics (with applications)**
 - Immediate assertions
 - Concurrent assertions - Basics
 - clocking basics; formal arguments; severity levels; threads
 - Sequence introduction
 - Property introduction (with/without an implication)
 - Vacuous pass?
 - Binding properties.
 - Threading (*what are the performance implications?*)
 - Sampled value functions (in property/sequence and procedural)
 - Functions that return Boolean pass/fail: \$rose, \$fell, \$stable
 - Function that return sampled value; \$past (with/without gating expr.)
 - Sequence Operators
 - ##m and ##[m:n] clock delay (*SVA allows only fixed delays. So what if you want variable delays??*)
 - [*] and [*m:n] - Consecutive repetition operator
 - [=] and [=m:n] - Non-consecutive repetition operator
 - [->] and [-> m:n] - Goto (non-consecutive) repetition operator
 - Pros/Cons of infinite (\$) range
 - 'throughout', 'within', 'intersect', 'first_match'
 - 'and' and 'or' of sequences with/without delay range
 - 'intersect' vs. 'and'

Training :: Agenda (contd.)

- Property operators
 - 'not' operator; If ... else ; 'disable iff'
- Recursive property
 - Mutually exclusive, 0 delay infinite loop, Restrictions
- System functions
 - \$onehot, \$onehot0, \$isunknown, \$countones
- Multiple Clocks / Multiply clocked properties and sequences
- Local variables (one of the most powerful features...)
 - Pipelined behavior (multiple threads)
- Detecting and using endpoint of a sequence
 - .ended, .matched, .triggered
- The 'expect' statement, 'assume' statement
- 'let' declaration
- 'checker'
- 'strong' and 'weak' properties, abort system tasks, deferred immediate assertions
- \$sampled, \$changed, \$inferred_clock, \$inferred_disable
- past and future global clock sampling functions such as \$rose_gclk, \$fell_gclk, \$rising_gclk, \$falling_gclk, etc. .
- 'followed by' property operators: #-# and #=#
- 'always', 'eventually', 'until', 'until_with', 's_until', 's_until_with', 'nexttime', 'case', \$inferred_clock and \$inferred_disable, etc.

Training :: Agenda :: LABs

- **LABs**
 - LAB 1: Learn how to 'bind' property module with design module.
 - Understand vacuous pass and properties with/without implication
 - LAB 2: Enforces how pipelined threads of a property work.
 - LAB 2: FIFO
 - A simple FIFO design is presented. You will code different properties to meet various FIFO fail conditions.
 - FIFO assertions are some of the most useful assertions to code for any design. This lab teaches how to do that so that you can apply them directly to your design.
 - *If time permits, more Labs will be conducted.*

CUSTOMER TESTIMONIALS ...

"Ashok is a very good instructor - very impressive."

John Reykjalín, President, Grizzly Peak Engineering, Inc.

*"The class was excellent, well distributed between the fundamentals and the practical examples. With a little tweak, we can use those example assertions presented right now in our design development!
In addition I would like to thank you for seminar associated text material. The handout (book) is a few levels above anything I have previously seen. The rules and example descriptions cover the associated topic completely."*

Thomas Slee, Sr. Electrical Engineer, ASIC Verification Lead, Space System Loral

*"The seminar on SVA was very educative and informative.
The material was in-depth, was from a hardware design/verification person's perspective and it was vendor neutral. The information was very good and I hope to have a chance to use it in the future."*

Shubha Umesh, Senior Logic Engineer, LeCroy Corporation

"Ashok, I take this opportunity to personally thank you for your dedication. You have very good knowledge on the subject. I intend to now use assertions heavily on my next verification project and will use your examples extensively. This class helped me strengthen my knowledge on SVA."

Mohammad Ashraf, Sr. Electrical Engineer, Space System LORAL

About the instructor

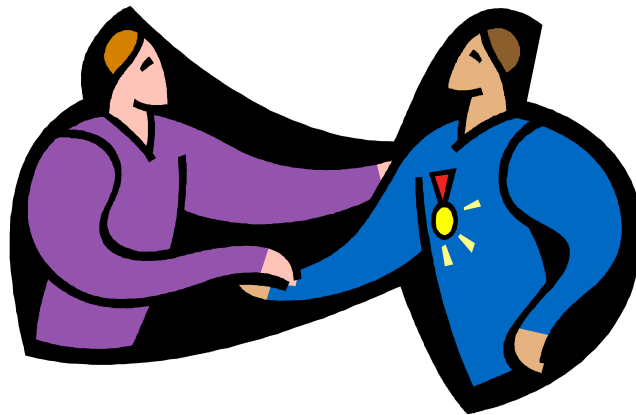
- [Ashok Mehta](#) has been working in the ASIC/SoC design and verification field for over 30 years. He started his career at Digital Equipment Corporation (DEC) working first as a CPU design engineer, moving on to hardware verification of the VAX11-785 CPU design. He then worked at Data General, Intel (first Pentium CPU design team) and after a route of a few startups, worked at Applied Micro and TSMC.
- He was a very early adopter of Verilog and participated in Verilog, VHDL, iHDL (Intel HDL) and SDF (standard delay format) technical subcommittees. He has also been a proponent of ESL (Electronic System Level) designs and at TSMC he released two industry standard Reference Flows that take designs from ESL to RTL while preserving the verification environment for reuse from ESL to RTL.
- Lately, he has been involved with 3DIC design verification challenges at TSMC which is where SystemVerilog Assertions played an instrumental role in stacked die SoC design verification.
- [Ashok is author of the popular book "SystemVerilog Assertions and Functional Coverage: A guide to language, methodology and applications"](#).
- Ashok earned an MSEE from University of Missouri. [He holds 13 U.S. Patents in the field of SoC and 3DIC design verification.](#)
- He brings to the class real life experience as an end user of HDL/HVL languages and methodologies that he personally deployed working on many successful silicon tape-outs. He provides practical in-sight to each feature/operator of the language to show exactly how it will help you solve your problem. He has an enthusiastic style of teaching welcoming any/all questions from the class and strives to provide utmost clarity in the answers.

Web: www.defineview.com

Email: ashok@defineview.com

Phone: (408) 309-1556

happy asserting...



Please visit our web site for further detail

DefineView Consulting
(www.defineview.com)

(email) ashok@defineview.com
(phone) 408.309.1556