

# System Verilog Assertions Language and Methodology

## Comprehensive 1 Day Training Class

**DefineView Consulting**

**<http://www.defineview.com>**

© 2006-2008

# Training :: Abstract

---

- Abstract
  - System Verilog Assertions (SVA) is a powerful subset of the IEEE 1800 System Verilog standard. Its hardware oriented concurrent semantics allow intuitive development of complex multi-clock domain checkers to catch bugs at their source.
  - SVA allows for clean separation of verification and design logic and allows parameterization of properties resulting in a modular reusable methodology.
  - SVA shortens time to tape-out and gain confidence in delivering first pass working silicon.
- Course Highlights
  - Taught from an end user point of view, each operator, feature is explained in detail using comprehensive examples, timing diagrams and simulation logs.
  - The end user perspective distinguishes us from others. We discuss practical hints on why, what and how of Assertion Based Verification (ABV) methodologies derived from real life projects and address issues faced by engineers and managers.
  - Practical applications are discussed to put it all in perspective.
  - A Reference Grade Training Book is provided to the class. It has comprehensive detail to serve as reference material for future.
  - LABS are geared to solidify understanding of key concepts using application oriented designs.

# 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 sequences and properties - legal and illegal usage
  - Multiply clocked properties and 'and', 'or', 'not' operator
  - Multiply clocked properties - Clock resolutions
- Local variables (one of the most powerful features...)
  - Basics and Visibility rules, legal and illegal usage
  - Pipelined behavior (threads)
- Detecting and using endpoint of a sequence
  - .ended, .matched
- The 'expect' statement, 'assume' statement
- Embedding concurrent assertions in procedural code
- Calling subroutines

# 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 3: 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.
- LAB 4: COUNTER Assertions
  - Code different properties to meet various Counter fail conditions.
  - Enforces the use of Local Variables, \$past system task, etc.
- More labs will be conducted if time permits
- Source Code and simulation scripts for the LABs will be provided.

# CUSTOMER TESTIMONIALS ...

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

***John Reykjalin, 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."*

***Shubha Umesh, Senior Logic Engineer, LeCroy Corporation***

*"I like the seminar very much since it's packed full of technical explanations and examples of System Verilog Assertion.*

*You slides are also easy to follow and understand."*

***Gloria Chen, ASIC Verification Engineer, 3PAR***

*"Your seminar clearly showed us that System Verilog Assertions provide new powerful interfaces and how they are implemented as part of the language.*

*Your seminar was one of the best seminar I ever had."*

***Aki Niimura-san, Ponderosa Design***

## About the instructor

Ashok Mehta has worked in the semiconductor industry for over 20 years in hardware design and verification engineering/management positions at companies such as INTEL, Digital, Data General, Philips Semiconductor, AMCC and many startups.

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.

At INTEL, he worked in the Architectural Verification team of the first Pentium and introduced to the company the concepts of verification environments to stress pipelined behavior, directed and constrained random stimulus generation, among other. He also designed a new Bus Functional Language geared to support Pentium's pipelined bus architecture, snooping behavior and deployed it successfully to find numerous bugs in the Pentium Bus Unit and First Level Cache.

At AMCC, he managed a team that employed the latest in SystemVerilog methodologies using class libraries, assertions, functional cover points and scoreboards to verify a complex L2 cache subsystem.

Ashok was also a hands-on manager at startup companies Chameleon Systems, empowerTel Networks and Nazomi communications.

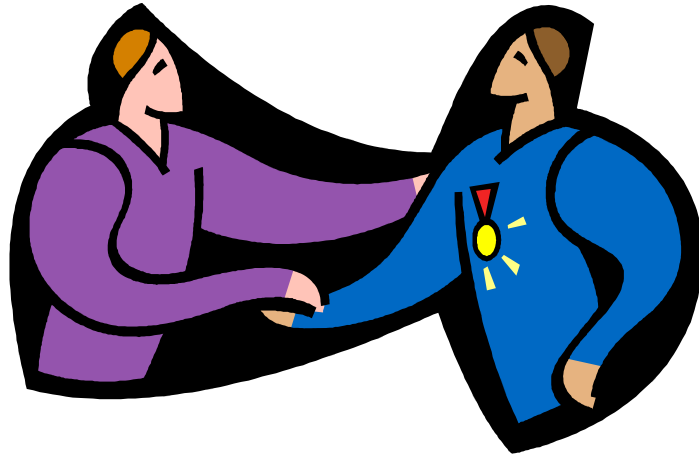
Ashok has been a member of technical sub-committees on IEEE Verilog-SDF, and EIA 576.

Ashok holds a MSEE from University of Missouri-Rolla.

*Ashok brings real life end user perspective to the training class.*

---

# happy asserting...



Please visit our web site for further detail

DefineView Consulting  
([www.defineview.com](http://www.defineview.com))

(email) [info@defineview.com](mailto:info@defineview.com)  
(phone) 408.309.1556

501 Pine Wood Lane  
Los Gatos, CA 95032