DefineView Consulting

Los Gatos, CA

Real Life, in-depth, application oriented

System Verilog Assertions
&
Functional Coverage

Language & Methodology Training

Design Verification Consulting Services
by
Seasoned Design and Verification professionals

Contact Us

Web: http://www.defineview.com

Email: info@defineview.com

Phone: (408) 309-1556

Why choose DefineView for your training needs?

- **Taught by seasoned Design/Verification *end users* of HDL, HVL languages with years of experience towards first pass silicon success.**
- **The *end user perspective* distinguishes us from others. We provide practical hints on what, how and why of Assertion Based Verification (ABV) and Functional Coverage methodologies and applications derived from *real life projects*.**
- **The training is in-depth and clearly explains language semantics with detailed timing diagrams/simulation logs coupled with plenty of *Practical Applications*.**
- **LABs are geared to confirm understanding of key concepts using application oriented designs.**
- **Class includes a 300+ page Reference Grade Training Book and 60+ page LAB book.**

Customer Testimonials

"Ashok brings invaluable practical insights on how SVA will help you survive in the DV trenches."

Matt Cossoul, Director Applications, Abound Logic

"The training class was excellent, well distributed between the fundamentals and the practical examples. With a little tweak, we can use those examples right now in our design development!

The training book is a few levels above anything I have previously seen. The rules, example descriptions cover the topics completely."

Thomas Slee, ASIC Verification Lead, Space System Loral

"Ashok is a very good presenter. Meticulous and thorough with the right amount of technical detail without losing the audience."

Ravi Reddy, President, Aurora VLSI

Comprehensive 2 Day Training Agenda

- **Introduction to Assertions**
  - What's an assertion? Why can't I just use Verilog? SVA advantages over Verilog.
  - Advantages of Assertion Based Verification (ABV) .
  - Assertion Based Verification (ABV) Methodology components
    - Who writes them? What types of assertions do you write? etc.
- **System Verilog Assertions :: Syntax/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; $past (with/without gating expr.)
  - Property/Sequence Operators
    - ##m and ##[m:n] clock delay
    - SVA allows only fixed delays. So what if you want variable delays?
    - [*] and [*m:n] – Consecutive repetition operator
    - [=] and [=m:n] – Non-consecutive repetition operator
    - [->] and [-> m:n] – Goto (non-consecutive) repetition operator
    - Pros/Cons of infinite ($) range
    - 'throughout', 'within', 'intersect', 'first_match'
    - 'and' and 'or' of sequences with/without delay range
    - 'intersect' vs. 'and'
  - Property operators
    - 'not' operator
    - If … else
    - 'disable iff'
  - Recursive property
    - Mutually exclusive, Restrictions
    - 0 delay infinite loop
  - 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)
    - Special consideration for 'and' and 'or'
  - Detecting and using endpoint of a sequence
    - .ended, .matched
  - The 'expect' statement, 'assume' statement
  - Embedding concurrent assertions in procedural code
  - Calling subroutines
  - Asynchronous Assertions (*careful with them)*
  - Multiple implications in a property;
  - blocking action_blocks
- **System Verilog Functional Coverage (*with applications*)**
  - Code coverage vs. Functional coverage
  - Coverage driven methodology (when/how/what should you cover?)
  - Features
    - 'covergroup' , 'coverpoint' , 'bins'
    - 'cross' coverage and Transition coverage
    - Wildcard bins , 'ignore_bins' and 'illegal_bins'
    - 'binsof' 'intersect'
  - Coverage options
    - Instance specific , 'covergroup' type
  - System tasks and Coverage methods for use in procedural code

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 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, $past system task, etc.
- **LAB 5: DATA TRANSFER BUS PROTOCOL Assertions**
  - Code different properties to meet bus protocol fail conditions.
  - Exemplifies temporal domain assertions coding ($stable, $rose, throughout, etc.)
  - Shows two different ways to code the same property.
- **LAB 6: PCI PROTOCOL Assertions**
  - Create a test plan for a basic PCI Read Bus Transaction.
  - Write properties to catch key temporal domain protocol violations.

About the Instructor

Ashok Mehta is the principal instructor at DefineView who has worked in the semiconductor industry for over 20 years in hardware design and verification engineering/management positions at companies such as INTEL, Digital, 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 striving to provide utmost clarity to 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.

Feel free to drop an email or call Ashok with any questions on System Verilog in general and Assertions in particular.

Email: ashok@defineview.com

Phone: 408-309-1556