System Verilog Assertions
&
Functional Coverage

Methodology and Language

2 Day Training Class
Agenda

Ashok B. Mehta

DefineView Consulting

http://www.defineview.com

2 Day 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. It allows for clean separation of DV logic from the design logic and allows for parameterization of properties resulting in a modular reusable methodology.
  - Functional Coverage (FC) is another subset of System Verilog that allows you to measure how much of design intent have you covered with your tests/regressions.
  - Combined SVA and FC allow for a modular, reusable and objective methodology that shortens time to develop&debug and gain much higher confidence in delivering a first pass working silicon.
- **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 language 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.
  - Class also explains practical ways to deploy SVA into your existing Verilog/System Verilog methodology and delineates real life methodology components that you can apply right away.

2 Day Training :: Agenda

- **DAY 1**
- **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 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'

2 Day Training :: Agenda

- **DAY 1 (contd.)**
  - Property operators
    - 'not' operator
    - If … else
    - 'disable iff'
  - Recursive property
    - Mutually exclusive
    - 0 delay infinite loop
    - Restrictions
  - System functions
    - $onehot, $onehot0, $isunknown, $countones
- **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.

2 Day Training :: Agenda

- **DAY 2**
  - 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 (be careful with them !!).
  - Multiple implications in a property; blocking action_blocks
- **LABs**
  - 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.

2 Day Training :: Agenda

- **DAY 2 (contd.)**
  - **System Verilog Functional Coverage**
    - Code coverage vs. Functional coverage
    - Coverage driven methodology
    - Features
      - ‘covergroup’
      - ‘coverpoint’
      - ‘bins’
      - ‘cross’ coverage
      - Transition coverage
      - Wildcard bins
      - ‘ignore_bins’ and ‘illegal_bins’
      - ‘binsof’ ‘intersect’
    - Coverage options
      - Instance specific
      - ‘covergroup’ type
    - System tasks for coverage
    - Coverage methods for use in procedural code

CUSTOMER TESTIMONIALS …

"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

"I like the training 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

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

Ravi Reddy, President, Aurora VLSI

"The training 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

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

John Reykjalin, President, Grizzly Peak Engineering, Inc.

About the instructor

Ashok Mehta has worked in the semiconductor industry for 24+ years in hardware design and verification engineering / management positions at companies such as Digital, Data General, Intel, Philips Semiconductor, AMCC and many startups.

At Digital, he worked on the VAX System design/verification team that introduced to the company the industry's first truly concurrent HDL called HiLo, trained groups of engineers who were mostly familiar with internally developed procedural languages and used it for design and verification. He worked on introducing the concepts of Transaction level modeling, constrained random stimulus, clock accurate reference models/checkers (now known as scoreboards) and embedded checker libraries (now known as assertions).

At Data General, he worked with the team that was the first to introduce and fully deploy Verilog/PLI for design and verification for DG's High End Systems Division. He also taught Verilog (which was very new to most designers) to various internal groups at DG.

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 the processor verification team that employed the latest in SystemVerilog methodologies using class libraries, assertions, functional cover points and scoreboards to verify a brand new processor and L2 cache subsystem.

Ashok also managed SoC Verification teams at startups such as Lara/EmpowerTel Networks, Chameleon systems and Nazomi Communications.

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

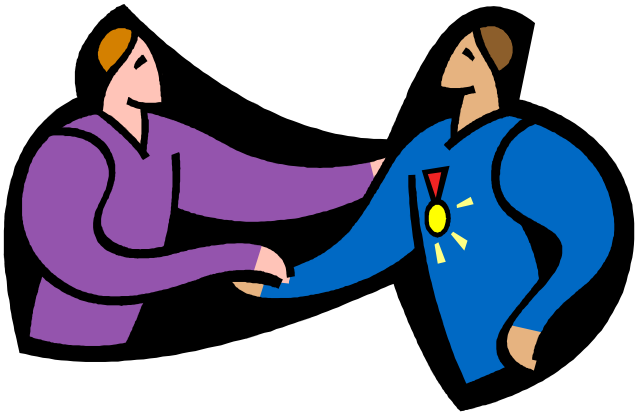
Ashok brings real life experience as a user of HDL and HVL languages and methodologies to the training class.

Web: www.defineview.com

Email: ashok@defineview.com

Phone: (408) 309-1556

happy asserting…



Please visit our web site for detail on scheduling and pricing.

We can also customize the 2-day class to a 1-day class.
Please call us to discuss your requirements.

DefineView Consulting
(www.defineview.com)

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