TVS Training courses

Four Day Intensive Course on

The Latest Design Verification Methodologies

Target Audience:
- Control and instrumentation nuclear safety inspectors

For further information contact:
- Sales@tandvsolns.co.uk
- Mike Bartley on 0044 7796 307958
Learning Outcomes:
Attendees will be able to:

- Understand the process of Design Verification, its complexities and limitations
- Develop a verification plan, set verification goals and select verification levels (block, SoC), methods, techniques and tools to achieve these
- Understand and use state-of-the-art dynamic verification techniques and methods including constrained pseudo-random test generation, coverage collection and analysis, advanced checking, and assertion-based verification
- Be able to develop test benches and tests at SoC level, use appropriate metrics to measure what has been verified and use that data to make tape-out decisions
- Be able to follow a well-defined verification process and produce the various deliverables (plans, reports, etc.) required of that process. The TVS verification process will be used as an example process.

Delivery:
- 4 days divided into interactive lectures and hands-on lab exercises
- Lab exercises are arranged on each day. There is an exercise and a solution sheet for each lab. In addition, the starting point for each lab is pre-coded to include the solution from the previous lab. This ensures that all attendees start a new lab from a consolidated point. To ensure attendees achieve the learning outcomes for each lab the solutions will be discussed at the end of each lab.
- Lectures include small group sessions for hands-on interactive problem solving such as analysis of a specification, identification of features for a verification plan, development of a cross product coverage model, identification of design properties for assertion-based and formal verification, formalization of design properties, analysis and interpretation of formal verification results etc.

Prerequisites:
- No prior knowledge of verification is required.
- Familiarity with the digital design process and basic understanding of SoC architecture is assumed.
- Familiarity with the basic functionality of the EDA tools used in the labs is assumed.
- Basic programming skills are helpful for the lab exercises, so is familiarity with Verilog or SystemVerilog.

Course content:
Interactive sessions are indicated in blue.
Hands-on lab exercises are indicated in green.
TVS verification process in red.

www.testandverification.com
DAY 1:

1) Introduction
   - Overview of 4 Day Course
   - Motivation
     o Ice breaker exercise for class
       ▪ “What is Functional Verification?”
       ▪ “Why do we care?”
     o Bugs
       ▪ Different types of bugs
       ▪ How are they introduced
       ▪ Human dimension
       ▪ Interpretation of specification
       ▪ Complexity vs. understandability
       ▪ How can bugs be found?
     o Cost of bugs
       ▪ Mask cost
       ▪ Late to market cost
       ▪ Lost opportunity cost
       ▪ Recall cost
       ▪ Credibility
       ▪ Reputation
       ▪ Discussion
       ▪ “Which one is the most important?”
   - Shrinking time to market windows
   - Significance of Design Verification for IP providers
   - Chip design process → Where does Verification “fit”?
     o Levels in Design and Verification
     o Observability and Controllability
     o Black and white box views

2) TVS Verification Process
   - Introduction to the TVS verification process
     o Motivation for following a well-defined process
     o Walk-through the TVS process explaining the deliverables at each point and the purpose of each deliverable

3) Introduction to the SoC
   - Introduce case study
     o Overview of the SoC Specification and the various blocks
     o Discussion of the verification approach
     o Documenting the verification approach

www.testandverification.com
4) Feature Extraction and Verification Planning (block vs. system)
- Feature extraction from a block level specification (UART)
- Deciding what to verify at block level and what to verify at system level
- Verification planning (based on the feature extraction)
  - Methods of plan capture
- Introduce UART case study
  - Specification → 'discover' reconvergence models
- LAB 1: Feature extraction from UART specification
  - Definition of test cases for directed testing in enough detail so that these can be turned into directed tests in the next lab, i.e. including what to drive and what to check
- Problems with incomplete and vague specifications
- Corner cases
- Documenting the feature extraction and the verification plan under the TVS verification process

5) Directed Testing
- Verification Tools & Languages
  - Overview dynamic & static (focus in the 3 day course is on dynamic)
  - Languages for Verification including dedicated high-level languages and scripting
- Basic testbench components
- Writing directed tests
  - Black box controllability and observability
- Discussion on how to predict expected results
  - The importance of Driving and Checking to find bugs
    - All 3 are needed: Activation, Propagation and Detection – Why?
- LAB 2: Directed Testing for the UART block
  - Turn the test cases from LAB 1 into directed tests to run on the UART block
  - Implement drivers and checkers
  - Investigate the UART design, record any bugs found
- Discuss limitations of directed testing and simulations using waveforms
  - Effectiveness, efficiency and how big is exhaustive?
  - Need to find bugs early and typical bug finding curve using only directed tests
  - Cost of debug using waveforms and need to increase productivity
- Overview of speed vs. design size vs. observability and controllability
- Documenting the directed testing results under the TVS verification process

6) When is verification done?
- Sign off criteria
  - Discuss what we expect from signoff criteria:
    - objective,
    - measures readiness to ship,
    - easy to measure,
    - progresses through the project (so can track progress)
- Coverage
  - Idea of coverage models
    - Mention all that we have covered so far in the course.
    - Re-emphasize importance of assertion coverage
  - Code coverage
    - From statement/block to expression, MC/DC and toggle coverage
      - Discuss strengths of the metrics
      - What does 100% code coverage mean?
        - Discuss limitations
          - Multi cycle scenarios
          - Concurrent events
          - Cross-correlations
          - Corner cases (still just a lower bound metric)
      - Merging of results from multiple test runs and multiple testbenches
  - Coverage closure challenge
    - What leads to coverage holes?
    - Typical coverage closure curve: coverage over time
  - LAB 3: Code Coverage Collection for the directed testing exercise
    - Collect and analyze the code coverage obtained in LAB 2
    - Identify coverage holes
    - How would you close these?
      (Forward pointer to generating properties from coverage holes)
- Corner cases
  - incorrect error handling
  - undefined behaviours
- Need for Verification Management
  - Bug tracking process (applied to identified bug)

---

1 Will include a pointer to system level verification on DAY 4 for FPGA and silicon

www.testandverification.com
TVS Training Courses: Four Day Intensive Course on The Latest Design Verification Methodologies

- **Configuration control and regression testing**
  - Did you just test the right version of the design?
  - Is the design quality stable and improving?
- **First verification review**
  - Create a verification report based on a TVS template
  - Verification review → Are we done yet?
  - Should we collect other statistics (instructions run, bug rate, code churn ...)
- Setting and documenting signoff criteria for the UART block

- **Summarise main lessons from DAY 1**

**DAY 2:**

7) **Introduction to Constrained random generation**

- Motivation for constrained random generation
  - “Fair weather sailing” (the common cases) vs. making rare events happen more often

- **Requirements for effective constrained random generation (establish in discussion)**
  - Repeatability
  - Random stability
  - Self-checking / monitors
  - Need to separate checking from stimulus generation – Why?

- **Coverage**
  - Code and structural (as from DAY 1)
  - Functional coverage models
    - Emphasize need to find appropriate level of abstraction
    - Detailed introduction to Cross Product functional coverage models
      - Importance of knowing restrictions
      - Legal and illegal coverage tasks
  
  - **Small group exercise to develop a Cross Product functional coverage model for the UART block followed by discussion of the options proposed by the groups plus one by the instructor (if needed)**
  - Coverage model completeness
    - Known knowns, known unknowns, etc

- **Transactors (abstracting away from signals)**
- **Constrained random drivers (grey box and white box controllability)**
- **Managing configurability and features (e.g: pairwise testing)**
- **Coverage-driven verification methodology**
TVS Training Courses: Four Day Intensive Course on The Latest Design Verification Methodologies

- Challenge of directing test generation towards unseen coverage

- **LAB 4: Constrained random test generation for the UART block**
  - Implement a constrained random stimulus generator for the UART block
  - Implementation and analysis of the Cross Product functional coverage model for the UART block

- Documenting the results from the constrained random verification for the UART block

8) **Checking: Predicting the expected results**

- Isolation of checking from stimulus generation
- Monitors (grey box and white box observability) and checkers
- Scoreboards for checking data I/O (matching data, timing, order, ...)
- Coping with coherency and non-determinism

- **LAB 5: Checking for the UART block**
  - Implement monitors and checkers
  - Implement/use a scoreboard for the UART block

9) **Are we done at block level yet?**

- **Combining coverage: Code, structural and functional coverage**
  - Discussion of pros/cons of each and combined strengths
  - Re-emphasize importance of assertion coverage
  - Coverage analysis
    - Coverage hole identification
  - Coverage closure strategy:
    - **How to prioritize?**
    - **How to close coverage?**
      - Coverage-driven verification
      - Introduce use of formal methods for coverage closure. (Generating properties from coverage holes)

- How to interpret results from formal verification, how to they contribute to the overall coverage picture?

- Some advanced ideas
  - State and transition coverage and structural coverage
  - design mutation and mutation coverage

- **Regressions**
  - Purpose of regression testing
  - Properties of a regression suite
  - How to collect tests for a regression suite

- **Extended soak testing using constraint random test generation**
  - How to determine an effective soak testing strategy?
  - How to use simulation farms to achieve effective soak testing?
• Second verification review
  o Update the verification report
  o Verification review → Are we done yet?
    ▪ Focus on “sign-off” at unit level
    ▪ Identify areas for formal verification

10) System level integration verification planning
• Defining the UART integration verification plan
  o Small group exercise to develop a UART integration verification plan
• Introducing use of BFM to create bus traffic
  o Advantages and disadvantages of using a bus traffic generator
• LAB 6: Writing the integration tests using BFM traffic generator
  o Implement system level integration tests
• Documenting the results from the integration verification for the UART block

• Summarise main lessons from DAY2

DAY 3:

11)Assertion-Based Verification (ABV)
• What is an assertion?
  o Discussion on “Who writes the assertions?”
    ▪ Designers, Verification Engineers, IP providers, standards
  o Implementation (design) and specification (intent) assertions
• Use of assertions
  o Expected behaviour of design and interfaces
    ▪ Combinatorial and sequential
  o Protocol checks
  o Block level vs. System Level assertions.
  o Re-use of existing assertions or checks embedded in VIP
• Identifying Assertions for the UART block
  o Behavioural black box properties
  o Implementation-specific white box properties
  o Discuss options for encoding properties in terms of abstraction levels and observability and implications for ease of bug finding
  o Block level vs. System Level assertions. Which to use?
  o Reuse of assertions
• Introduction to basics of formal property formalization language
  o How assertions work during simulation
  o SVA, PSL
  o Write UART properties using SVA
Focus on use of correct timing connectives

- Importance of Assertion Coverage
  - What does it mean that an assertion was never violated?
  - How do you know assertions are correct?
  - How do you know assertions are triggered?

- LAB 7: Assertions for the UART block
  - Encoding of assertions for the UART block
  - Simulation with assertions – activation of assertions
  - Assertion coverage, collect and analyse
  - Add assertions to both the block and system level verification of the UART

- Documenting the assertions for the UART block

- Discuss costs v benefits of using assertions
  - Costs include:
    - Simulation speed
    - Writing the assertions
    - Maintaining the assertions
  - Benefits include:
    - Assertion capture process can help avoid “silly” bugs being introduced during design capture
    - Explicit expression of designer intent and specification requirements
    - Improved localisation of errors for debug
    - Re-use of formal properties for formal verification

12) CPU-based SoC verification

- What is CPU-based verification
- Discuss BFM-based vs. CPU-based verification
  - What are the benefits and drawback of each approach?
  - When is it best to use each approach?

- LAB 8: Writing the integration tests using BFM traffic generator
  - Implement system level integration tests

DAY 4:

13) The complete verification of the SPI

- LAB 9: Repeat the UART verification for the SPI
  - From block through to system level
14) System verification, FPGA and silicon

- **Is block level verification and top level verification sufficient?**
  - Validate complete systems (both HW and SW)
  - Validating correct operation with related IP
  - Are there any more tests we need

- **How to go faster**
  - Size of test space
  - Required scale of testing
  - Speed of different simulators vs. observability and controllability
    - Discussion of tradeoffs

- **Synthesisable test benches**
  - Including synthesisable transactors for VIPs
  - Synthesisable assertions
  - Connecting 'real' external hardware
  - ICE debug
  - Backdoor memory access

- **Performance verification and QoS**

- **Do we need more system level tests for the UART?**
  - What are the benefits and drawback of each approach?

- **LAB 10: Write additional system level tests**
  - For those system level features not covered by only considering the blocks

15) Complete Verification Flow (Verification Cycle) – Bringing it all together

- **Functional Specification**
  - Have all the top level requirements been tested?
  - Requirements-based testing methodology linking requirements to tests

- **The Verification Plan as a specification of the verification process**
  - The role of Formal Verification

- **Development of the verification environment**

- **Debugging**

- **Regressions**

- **HW debug**

- **Escape analysis**

- **Errata management**

- **Re-use strategy between projects**
  - Developing with re-use in mind
  - Verification IP (VIP)

- **Final verification review**
  - Bug review
  - Verification report
o Verification review → Are we done yet?
  ▪ Focus on system level “sign off”
  ▪ Requirements coverage

16) TVS Verification Process
  • Review of all process and all of the documents generated and the reviews
    o Is this sufficient?