Four Day Intensive Course on

The Latest Design Verification Methodologies

Target Audience:
- Design Verification engineers
- Design Verification Managers
- Design Engineers looking to cross-train to verification
- Companies trying to move to the latest verification techniques and strategies

For further information contact:
- Sales@tandvsoins.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 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
- Understand and use state-of-the-art formal property checking tools and techniques, their complexities and limitations and how they can be used to complement a traditional dynamic design verification flow
- Carry out functional Design Verification including dynamic and formal techniques at block and system level

Delivery:
- 4 days divided into interactive lectures and hands-on lab exercises
- Lab exercises are arranged both during the morning and during the afternoon 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 computer architecture is assumed.
- Familiarity with the basic functionality of the EDA tools used in the labs is assumed.
- Familiarity with formal logic connectives is of advantage for the lab exercises.
- 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.

DAY 1: DIRECTED TESTING (Unit Level)

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”?
TVS Training Courses: Four Day Intensive Course on The Latest Design Verification Methodologies

- Levels in Design and Verification
- Observability and Controllability
- Black and white box views
- Focus on unit level (DAY 1)

2) Verification Planning
   - Verification planning (based on feature extraction from specification)
     - Methods of plan capture
   - Introduce case study
     - Specification → 'discover' reconvergence models
   - Reconvergence models
   - LAB 1: Feature extraction from FIFO 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

3) 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 FIFO block
     - Turn the test cases from LAB 1 into directed tests to run on the FIFO block
     - Implement drivers and checkers
     - Investigate the FIFO 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

www.testandverification.com
• Overview of speed vs. design size vs. observability and controllability

4) **Assertion-Based Verification (ABV)**

• What is an assertion?
  - Discussion on “Who writes the assertions?”
    - Designers, Verification Engineers, IP providers, standards
  - Implementation (design) and specification (intent) assertions

• Use of assertions
  - Expected behaviour of design and interfaces
    - Combinatorial and sequential
  - Protocol checks
  - Re-use of existing assertions or checks embedded in VIP

• **Identifying Assertions for the FIFO block**
  - Behavioural black box properties
  - Implementation-specific white box properties
  - Discuss options for encoding properties in terms of abstraction levels and observability and implications for ease of bug finding

• Introduction to basics of formal property formalization language
  - How assertions work during simulation
  - SVA, PSL
  - **Write FIFO 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 3: Assertions for the FIFO block**
  - Encoding of assertions for the FIFO block
  - Simulation with assertions – activation of assertions
  - Assertion coverage, collect and analyze

• **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

---

1 Will include a pointer to system level verification on DAY 4 for FPGA and silicon
- Improved localisation of errors for debug
- Re-use of formal properties for formal verification

- Summarise main lessons from DAY 1

DAY 2: CONSTRAINED RANDOM (Unit Level)

5) 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 4: 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

www.testandverification.com
• Need for Verification Management
  o Bug tracking process (applied to identified bug)
  o 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
  o Create a verification report based on a TVS template
  o Verification review \(\rightarrow\) Are we done yet?
  o Should we collect other statistics (instructions run, bug rate, code churn ...)

6) Introduction to formal property checking
• Motivation
  o Ice breaker exercise for class
    ▪ “What is formal verification?”
    ▪ “How does it work?”
    ▪ Develop reconvergence model for formal property checking
    ▪ “Why do we care?”

• DEMO: Re-using assertions from dynamic verification as properties
  o Property initially fails due to lack of environmental constraints
  o Inspect the counter example and discuss what we would need to do?
  o Add one constraint. See property fail because of another missing constraint.
    ▪ Explain that formal can be an iterative process.
    ▪ Point out responsibility of the verification engineer in formalizing properties, especially wrt ensuring validity of constraints.

• Summary after DEMO: How to interpret the results of formal property checking
  o Trivially true properties (vacuity checking)
  o Bugs in properties
    ▪ Bugs in environment constraints
  o Bugs in the design

• Property mutation as a strategy to gain confidence in the correctness of a property

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

• Requirements for effective constrained random generation (establish in discussion)
  o 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 FIFO 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 (eg: pairwise testing)
Coverage-driven verification methodology
Challenge of directing test generation towards unseen coverage

LAB 5: Constrained random test generation for the FIFO block
Implement a constrained random stimulus generator for the FIFO block
Implementation and analysis of the Cross Product functional coverage model for the FIFO 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 6: Checking for the FIFO block
Implement monitors and checkers
Implement/use a scoreboard for the FIFO block

Summarise main lessons from DAY2

DAY 3: WHAT COMES AFTER UNIT LEVEL TESTING?

9) Are we done 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**
  - Update the verification report
  - Verification review → Are we done yet?
    - Focus on “sign-off” at unit level
    - Identify areas for formal verification

10) Developing properties
• Introduction to basics of formal property formalization language
  - SVA, PSL
• The environment and assumptions
• Under-constrained and over-constrained properties
• Invariants
• Trivially true properties (vacuity checking)
• Pre-conditions and use of cover properties
  - Assume/Guarantee approach
• Sequential behaviour, deadlock and livelock
• **How do you know whether you have “enough” properties?**
• **LAB 7: Property Formalization and Checking for FIFO design**
  - Convert all of the assertions from the FIFO into properties
• Add constraints
• Try to prove the properties in the tool

- Property specification strategies
  o Properties on internal signals and state machines
  o Properties capturing required input/output behaviour
  o Properties on block interface signals

11) Writing system tests
- Top Level Verification planning. What do you verify at top level?
  o What has been verified at block level?
    ▪ How can these results be re-used at top level?
    ▪ What remains to be verified?
  o top level functionality
  o connectivity
  o configurability
  o register map
  o changing clock ratios
  o power on/reset

• The architecture of the top level test bench
• Writing top level tests (run via CPU or as transactions using BFM’s)
  o Architecture compliance testing
• LAB 8: Top Level Test Plan
  o Definition of test cases for various top level scenarios 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
• Improving controllability
  o verification components driven by the tests
• Forcing state
  o eg: by hot loading memories/caches or check pointing simulations
• Introducing elements of advanced constrained random
  o Compile and run time configurations
  o Instruction Stream Generators for CPU verification (RIS)
  o Chicken bits and irritators

12) Predicting expected results
- Reference models
  o Levels of abstraction
    ▪ Functionally accurate
    ▪ Cycle accurate
  o Impure reference models
  o Use of abstract reference models (C/C++/SystemC) and DPI
- System level properties and assertions

www.testandverification.com
• System level debug
• Coverage at top level
  o Evaluation of coverage metrics
    ▪ code,
    ▪ structural,
    ▪ functional,
    ▪ toggle coverage and
    ▪ full top-level requirements coverage
• LAB 9: Reference Model and System Level Properties
  o Turn the test cases from LAB 8 “Top Level Test Plan” into directed
tests to run on the top level test bench with mechanisms for checking
expected results including promoting unit level assertions
  o Run tests, debugging any fails and ensuring all tests are passing
  o Add suitable metrics such as toggle coverage
• Power aware verification
  o Power management, power modes and state retention
  o Specifying and verifying power intent - UPF

• Summarise main lessons from DAY3

DAY 4: BRINGING IT ALL TOGETHER

13) Using formal verification
• Strengths of formal verification:
  o Speed of set-up
  o Flexibility of verification environment
  o Full / exhaustive proof
  o Intensive stressing of design
  o Corner cases
• Potential issues with formal:
  o False failures
  o False proofs
  o Non-exhaustive checks
  o Non-uniform / unpredictable run times
• LAB 10: Bug hunting on the FIFO design (corner case bug)
  o For the read/write when empty/full corner case, develop properties to
    identify the root cause of the bug
• Summary of expected effort expended and effort saved elsewhere

www.testandverification.com
• **Describe and discuss various applications of formal and how they exploit the strengths and avoid the weaknesses of formal**
  o Dedicated bug hunting
  o Bug absence – single properties and whole blocks
  o Bug hunting and bug absence combined
  o Bug avoidance – correct by construction
  o Bug analysis
  o Checking bug fixes
  o Coverage closure

14) **Combining static and dynamic verification**
• Writing a verification plan including both static and dynamic verification
  o **How much can be re-used?**
  o Mixed verification strategy: which technique is best suited for what?
• Trading off dynamic and static during execution of the plan
• Integrating environment assumptions from formal verification into the dynamic verification environment for monitoring during simulation
• **LAB 11: Combining static and dynamic on the FIFO block example**
• **Third verification review**
  o Update the verification report, include results from formal verification
  o Verification review → Are we done yet?
    ▪ **Focus on “sign-off” at unit level**

15) **System verification, FPGA and silicon**
• **Is block level verification and top level verification sufficient?**
  o Validate complete systems (both HW and SW)
  o Validating correct operation with related IP
• **How to go faster**
  o Size of test space
  o Required scale of testing
  o **Speed of different simulators vs. observability and controllability**
    ▪ **Discussion of tradeoffs**
• **Synthesisable test benches**
  o Including synthesisable transactors for VIPs
  o Synthesisable assertions
  o Connecting 'real' external hardware
  o ICE debug
  o Backdoor memory access
• **Performance verification and QoS**

16) **Complete Verification Flow (Verification Cycle) – Bringing it all together**
• **Functional Specification**
TVS Training Courses: Four Day Intensive Course on The Latest Design Verification Methodologies

- 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
  - Verification review → Are we done yet?
    - Focus on system level “sign off”
    - Requirements coverage
- **Summarise main lessons from DAY4**

**Recap of 4 Day intensive course**
- What has been covered and main lessons learned
- Other techniques not covered in detail