Design Verification (DV) training

This course introduces participants to best-practice DV strategies so they can contribute effectively to real projects (see the objectives). It is aimed at new recruits, interns, and managers/engineers wanting to transition to DV, or just understand more about it (target audience). It is delivered fully online with a large number of quizzes and practical exercises to practice the content covered in the lectures.

The training is spread over 3 weeks: the first 2 weeks has 2 hours per day of online lectures and practicals each day (see Table 1, more details can be found by following the links); the 3rd week is 1 hour per day of online going through test bench examples and exercises. Exact times are agreed with the students in advance of the course.

A range of course materials are provided in advance to help participants to prepare (details). Different experience pathways though the examples and exercises are defined based on student prior experience and expertise. There are quizzes on Moodle (with automated grading) to provide feedback on understanding.

It is recommended that during the 3 weeks the students are given time for independent study on Moodle quizzes and exercises outside of the online session.

Table 1: Proposed content and timetable for the DV training with 2 hours of online training per day

Day	Week 1	Week 2
Mon	Lecture: "Introduction", "Simulation-based Verification" Practical: "Introduction to the exercises"	Lecture: Introduction to OSVVM Practical: OSVVM Lecture: Overview of UVVM
Tues	Lecture: "Stimulus generation" Practical: "Introduction to Verilog/SV/VHDL and simulation[1]" "Running directed test benches"	Lecture: "Further UVM" Practical: "Advanced UVM" Lecture: Feature Extraction Practical: Feature Extraction Exercises
Wed	Lecture: "Checking and Assertion-Based-Verification" Practical: "Verification checkers and assertions"	Lecture: "Introduction to debug", "Making debug efficient and effective", "Additional topics" Practical: "Debug and qualification for IP verification"
Thurs	Lecture: "Coverage" Practical: "Coverage"	Lecture: "The Verification Cycle", "IP, subsystem and SoC" Practical: "FIFO subsystem verification"
Fri	Lecture: "The FIFO example" Practical: "FIFO IP verification Lecture: "Introduction to UVM" Practical: "UVM"	Practical: "CPU-based SoC verification" Lecture: ""Verification in practise", Lecture: AI/ML in DV Lecture: Summary

Contents

[1] Note only the languages relevant to the audience will be covered

1 Course Objectives

By the end of the course, participants should be able to:

1. Describe the best-practice DV strategies applied currently to semiconductor digital designs.
2. Understand the main methodologies, tools and languages used in those best-practice DV strategies.
3. Apply those DV methodologies, tools and languages to basic digital designs at IP, subsystem and SoC level.
4. Analyse a “real” semiconductor digital design (IP, subsystem and SoC) and propose a DV strategy.
5. Understand best-practice DV so they can discuss DV topics confidently with colleagues.
6. Have sufficient understanding of DV tools and methodologies to contribute effectively to real projects.

2 Target audience

1. New recruits.
2. University students on placement.

3. Design engineers wanting to learn more about verification and/or how to write re-usable verification.
4. Engineers wanting to transition their career to DV (or just want to learn more about DV).
5. Managers wanting some understanding of DV.

3 Pre-requisites

There are no pre-requisites, although some experience of programming (preferably with an Object-Oriented language) would be useful. The course is suitable for a wide range of people working in semiconductor design and development.

4 Preparation

To allow participants to prepare for the course, the following will be shared in advance:

1. An overview of the practical exercises and explanations on how to run them
2. The database of practical exercises
3. PDF of the initial lecture slides
4. Preparation videos
5. A login in to Moodle to prepare

5 Course overview and structure

1. There are 20 1-hour sessions split as follows:
    - Online one-hour lectures covering the major DV topics (details).
    - Practical sessions (details
    - Sessions on debug (details).
2. A large database of design and verification examples and exercises graded as 3 distinct levels of difficulty and complexity ready for running on a wide variety of simulators (details).
    - These can be downloaded so participants can run them on their machines using their licenses.
3. A number of pre-recorded videos (details) covering running the exercises and using various EDA tools.
4. Use of a FIFO example that builds through the course to show a full end-to-end example of verification.
5. Use of Moodle (details) for testing understanding of the course content and tracking progress on the course.

Note that participants will get additional support outside the lectures and exercises as needed or requested.

Preparatory and follow-on reading will be suggested but neither are obligatory. There is both formative and summative assessment (details).

6 Course content details

6.1 DV lectures: Details

The online DV lectures will cover the following. These sessions will include online questions and polls to increase the level of interaction.

6.1.1 Introduction	• Course introduction. • What is Design Verification? • Verification complexities. • Preparing for Practical sessions on Hardware design and DV languages. • Verification tools. • The cost of bugs.
6.1.2 Simulation-based Verification	• Driving stimulus and Checking responses. • Observability and Controllability. • Black box vs. white box. • Compare/contrast with formal verification. • Verification hierarchy. • Introduction to the FIFO exercise.
6.1.3 Stimulus generation	• Foundations. • Manual vs. Automation. • Independent vs coordinated generators? • Level of abstraction. • Online vs. offline generation. • Test length. • Randomness. • Stimulus for the FIFO.
6.1.4 Checking and Assertion-Based-Verification	• What and when to check. • Checking mechanisms including behavioural models. • Introduction to assertions. • Benefits and drawbacks of assertions. • Types of assertions. • Mutation testing. • Assertion libraries – to be added – add OVL as an example. • Checkers and assertions for the FIFO.
6.1.5 Coverage	• Introduction. • Code coverage. • Functional coverage. • Coverage analysis and closure. • Coverage for the FIFO.
6.1.6 The FIFO example	• Applying the full verification cycle to the FIFO. • Signing off the FIFO at IP level.
6.1.7 Introduction to UVM	• General structure of a constrained random UVM test bench. • How to create suitable randomisation, sequences, checkers, coverage, assertions etc. • The key UVM structures for building such test benches
6.1.8 Introduction to OSVVM	• General structure of a constrained random OSVVM test bench. • How to create suitable randomisation, sequences, checkers, coverage, assertions etc. • The key OSVVM structures for building such test benches
6.1.9 Further UVM	• Advanced UVM concepts
6.1.10 Additional topics	• Using lint to check design and test bench quality • Test bench qualification using mutation testing
6.1.11 The Verification Cycle	• Verification planning. • Planning, milestones, tracking and completion criteria. • Add examples from Open HW project. • Verification metrics and closure. • Verification methodologies.

6.1.12 Verification in practise	• The verification cycle from start to finish. • Add Design for Verification. • Add Test bench qualification. • IP level verification in practise. • Higher level verification in practise.
6.1.13 IP, subsystem and SoC (System-on-Chip)	• Levels of abstraction. • Differences in designs at IP, subsystem and SoC. • Bus interfaces and connecting IP. • Differences in verification at IP, subsystem and SoC. • Difference between ASIC and FPGA design flows. • Verifying the FIFO integration at subsystem and SoC levels.
6.1.14 Feature Extraction	• Some techniques to apply during feature extraction • Worked examples of feature extraction • Creating a Verification Plan from a feature extraction
6.1.15 AI/ML in DV	• Where/how can AI/ML possibly be applied in DV? • Some example applications

6.2 Practical Sessions: Details

These practical sessions are aimed at giving first-hand experience on developing test benches using a wide range of languages and methodologies. Note that there are also opportunities to develop designs[2]. All practical sessions will include interactive question and answer sessions.

6.2.1 Introduction to the exercises	• Overview of the exercises. • How to run the exercises. • Some simple simulation exercises. • Overview of the videos. • Selecting which exercises to run and how to get feedback.
6.2.2 Introduction to Verilog/SV/VHDL and simulation	• Overview of Hardware Description Languages (HDLs). • Verilog, SV and VHDL (only the languages relevant to the audience will be covered)
6.2.3 Running directed test benches	• Running directed test benches in the exercises. • Adding some randomisation to those test benches.
6.2.4 Verification checkers and assertions	• Automated verification checks. • Scoreboards. • SV Assertions.
6.2.5 Coverage	• Functional coverage. • Code coverage. • Structural coverage.
6.2.6 FIFO IP verification	• FIFO IP and test bench. • Can your test bench find the FIFO bugs?
6.2.7 UVM	• Developing a basic UVM test bench from scratch.
6.2.8 Advanced UVM	• Adding advanced UVM features into the test bench
6.2.9 OSVVM	• Developing a basic OSVVM test bench from scratch
6.2.10 Debug and qualification for IP verification	• FIFO test bench qualification. • FIFO debug exercises.
6.2.11 FIFO subsystem verification	• Review of subsystem test bench solution.
6.2.12 CPU-based SoC verification	• Introduction to CPU-based test bench verification.
6.2.13 Feature Extraction Exercises	• Exercises on simple feature extraction

[2] Development of designs have been added to give participants the opportunity to better understand the designs they are trying to verify.

6.3 Debug content

The online DV lectures will cover debug. Specifically, the following content will be covered.

6.3.1 Introduction to debug	What is debug and where do bugs lie? Debugging designs and verification environments. Working with (not against) the design team.
6.3.2 Making debug efficient and effective	Techniques to reduce debug time. Ensuring a bug is fixed. Making use of "bug clustering". Ensure my verification environment is designed for ease of debug?

The practical sessions will also cover techniques and examples of how to perform debug in simulation. The exercise database contains examples and exercises where designs have bugs that need to be found using the EDA tools.

There will be pre-recorded videos showing how to debug with the various EDA tools.

7 Moodle

Moodle (https://moodle.com/) is an online learning platform. Students have a personalised login for the course that allows them to

- Access course materials in advance
- Attempt online quizzes with automated marking and model answers to give them instant feedback on their understanding
- Update on their progress on the exercises

The lecturer is also able to use this to monitor participant understanding and progress.

8 Pre-recorded videos

There will be pre-recorded videos explaining how to run the exercises on all the supported EDA tools. They will also cover how to debug using the various debug examples in the database.

9 Database of examples and exercises

Participants will be given access to numerous small DV examples and exercises to support what is being taught in both the DV and the debug lectures. The exercises will be available in both SystemVerilog (some using UVM), VHDL (some using OSVVM and UVVM) and Python (using cocotb) that can be run on a wide range of simulators[3]. Exercises are also graded into "simple, medium, hard" to allow students to find exercises suited to their level of expertise:

- Small design exercises are provided for those participants who want to learn about design[4].
- Small test bench exercises are provided to allow participants to implement the concepts learned in the lectures.
- There is documentation and videos explaining how to run the examples on a wide range of tools.
- The exercises will be kept small and simple due to time limitations:
    - There are directed test exercises and some randomisation is added.
    - There are separate exercises for assertions, coverage and debug.
    - There are small complete test bench UVM/OSVVM/UVVM and cocotb exercises .
    - Participants will mostly be expected to complete a small part of a design or change to a design or test bench.
    - There will be full solutions available for every exercise.

[3] Including commercial simulators Cadence Xcelium, Siemens Questa, Synopsys VCS, and Aldec ActiveHDL. And free tools such as GHDL + GTK wave for VHDL and Verilator for SV UVM

[4] As mentioned, design exercises have been added to give participants the opportunity to better understand the designs they are trying to verify. This course is not aimed at teaching hardware design.

9.1 Access to exercises and DV tools

Participants will be able to download the exercises for running on their own machines with their own tools and licenses.

9.2 Mapping a path through the examples and exercises

A number of pathways though the examples and exercises for the participants, based on their level of experience.

10 Assessment

The following assessment activities are available to participants:

- Participants who attend the full course will receive a TechWorks Academy certificate of attendance.
- There will be a multiple-choice end of course test[5].

[5] This is seen as higher value than an attendance certificate so attendees can still acquire a high value certificate even if they do not attend all sessions