

Formal Verification Training Outline

This document gives an overview of the formal verification training course. It is split into 2 courses:

- A 2-day “Bootcamp”: an introduction to writing and proving basic properties and how they can be used to verify a complete block or used alongside simulation-based verification.
- A 1-day “Advanced”: effective use of writing and for formal verification and proving them, and how to

Although the course will not focus on any particular tool, a tool will be used to demonstrate and practise the concepts introduced during the day (those parts of the course are highlighted in yellow). Example designs will be provided.

2-day Bootcamp

Day 1	
9.00	1 Introduction 1.1 Writing Basic System Verilog Assertions <ul style="list-style-type: none">• Introduction to the language• The main combinatorial language constructs (syntax and semantics)• Constraints and properties• Examples of constraints<ul style="list-style-type: none">○ Reading and understanding○ Writing from fresh• Examples of properties<ul style="list-style-type: none">○ Reading and understanding○ Writing from fresh 1.2 Writing properties for formal vs. simulation
10.00	1.3 Proving a basic property <ul style="list-style-type: none">• Using a tool to prove a basic property on a real design (e.g. FIFO)
11.00	Break
11.30	2 Advanced SVA 2.1 Writing Complex System Verilog Assertions <ul style="list-style-type: none">• The main sequential language constructs (syntax and semantics)• Examples of sequential constraints<ul style="list-style-type: none">○ Reading and understanding○ Writing from fresh• Examples of sequential properties<ul style="list-style-type: none">○ Reading and understanding○ Writing from fresh 2.2 Proving a complex property <ul style="list-style-type: none">• Using a tool to prove a complex property on a real design• Interpreting the three possible outcomes: Proved, Failed, Unproven
13.00	Lunch
14.00	3 Debugging a failing property <ul style="list-style-type: none">• Using a tool to debug a failing property on a real design
15.30	Break

16.00	4 Formal metrics <ul style="list-style-type: none"> Measuring coverage and completeness Measuring coverage with a particular tool on a real design
17.00	End
Day 2	
9.00	5 Full formal verification of a block <ul style="list-style-type: none"> Developing constraints for a real design <ul style="list-style-type: none"> Over constraint Under constraint Determining a “full” set of properties Run the constraints and properties using the tool on a real design
13.00	Lunch
14.00	6 Formal in the design flow <p>6.1 Formal verification applications or “apps”</p> <ul style="list-style-type: none"> What are the main formal verification apps (e.g. X propagation) and why are they useful Running apps on a real design <p>6.2 The AHAA model</p> <ul style="list-style-type: none"> Understanding the various applications of formal verification <ul style="list-style-type: none"> Bug avoidance Bug hunting Bug absence Bug analysis Understanding when and where to apply formal during a project <p>6.3 Formal for designers</p> <ul style="list-style-type: none"> Designer bring-up <ul style="list-style-type: none"> Design visualisation, advanced lint, using apps, reachability Possible reuse by verification <p>6.4 Formal reuse</p> <ul style="list-style-type: none"> Reuse of assertions between dynamic and static verifications Running simulations with our formal assertions added on a real design Reuse in assume/guarantee relationships Assertions as VIP <p>6.5 Formal in context</p> <ul style="list-style-type: none"> Formal as part of a verification plan Combining formal results with other activities for a signoff decision <p>6.6 What type of design and size of design is suitable?</p>
17.00	Close

1-day Advanced Formal

Day 1	
9.00	1 Introduction <ul style="list-style-type: none"> What makes some properties hard to prove? <ul style="list-style-type: none"> Examples of hard-to-prove properties Why are they hard to prove? Interpreting Unproven results

	<ul style="list-style-type: none"> • Example of an Unproven property on a real design
10.00	2 Overview of techniques for resolving Unproven's <ul style="list-style-type: none"> • Review some techniques for trying to get Unproven properties to Proven or Failed – e.g. reduce input space, reduce data widths, use of additional constraints, abstractions, cut points, free variables, undriven signals, black boxes, models, etc • How to select the best technique for the given Unproven
11.00	Break
11.30	3 Applying simple techniques for resolving Unproven's <ul style="list-style-type: none"> • Start with simple techniques such as reduce input space, reduce data widths, use of additional constraints, • Example of applying these techniques on a real design • abstractions, cut points, free variables, undriven signals and variables, black boxes, models, etc
13.00	Lunch
14.00	4 Applying advanced techniques for resolving Unproven's <ul style="list-style-type: none"> • Start with the following advanced techniques: abstractions, cut points, free variables, undriven signals, • Example of applying these techniques on a real design
15.30	Break
16.00	5 Applying advanced techniques for resolving Unproven's <ul style="list-style-type: none"> • Start with the following advanced techniques: black boxes, models • Consider some tool specific tips and tech • Example of applying these techniques on a real design 6 Summary <ul style="list-style-type: none"> • What makes some properties hard to prove? • The main techniques for overcoming Unproven's • Selecting the right technique for the right property
17.00	End