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 but students will have the opportunity to share designs in advance that can be used during the course (those parts are highlighted in green).

2-day Bootcamp

<table>
<tr><th colspan="2">Day 1</th></tr>
<tr><td>9.00</td><td><h2>1 Introduction</h2><h3>1.1 Writing Basic System Verilog Assertions</h3>
Introduction to the language
The main combinatorial language constructs (syntax and semantics)
Constraints and properties
Examples of constraints
Reading and understanding
Writing from fresh
Examples of properties
Reading and understanding
Writing from fresh
<h3>1.2 Writing properties for formal vs. simulation</h3></td></tr>
<tr><td>10.00</td><td><h3>1.3 Proving a basic property</h3>
Using a tool to prove a basic property on a real design (e.g. FIFO or a student example design)</td></tr>
<tr><td>11.00</td><td>Break</td></tr>
<tr><td>11.30</td><td><h2>2 Advanced SVA</h2><h3>2.1 Writing Complex System Verilog Assertions</h3>
The main sequential language constructs (syntax and semantics)
Examples of sequential constraints
Reading and understanding
Writing from fresh
Examples of sequential properties
Reading and understanding
Writing from fresh
<h3>2.2 Proving a complex property</h3>
Using a tool to prove a complex property on a real design (or a student example design)
Interpreting the three possible outcomes: Proved, Failed, Unproven</td></tr>
<tr><td>13.00</td><td>Lunch</td></tr>
<tr><td>14.00</td><td><h2>3 Debugging a failing property</h2>
Using a tool to debug a failing property on a real design (or a student example design)</td></tr>
</table>

15.30	Break
16.00	**4 Formal metrics** • Measuring coverage and completeness • Measuring coverage with a particular tool on a real design (or a student example design)
17.00	End
Day 2	

9.00	**5 Full formal verification of a block**
	• Developing constraints for a real design (or a student example design)
	    ○ Over constraint
	    ○ Under constraint
	    ○ Determining a “full” set of properties
	• Run the constraints and properties using the tool on a real design (or a student example design)

13.00	Lunch

14.00	**6 Formal in the design flow**
	
	6.1 Formal verification applications or “apps”
	• What are the main formal verification apps (e.g. X propagation) and why are they useful
	• Running apps on a real design (or a student example design)
	
	6.2 The AHAA model
	• Understanding the various applications of formal verification
	    ○ Bug avoidance
	    ○ Bug hunting
	    ○ Bug absence
	    ○ Bug analysis
	• Understanding when and where to apply formal during a project
	
	6.3 Formal for designers
	• Designer bring-up
	    ○ Design visualisation, advanced lint, using apps, reachability
	• Possible reuse by verification
	
	6.4 Formal reuse
	• Reuse of assertions between dynamic and static verifications
	• Running simulations with our formal assertions added on a real design (or a student example design)
	• Reuse in assume/guarantee relationships
	• Assertions as VIP
	
	6.5 Formal in context
	• Formal as part of a verification plan
	• Combining formal results with other activities for a signoff decision
	
	6.6 What type of design and size of design is suitable?

17.00	Close

1-day Advanced Formal

Day 1	
9.00	**1 Introduction** • What makes some properties hard to prove? ○ Examples of hard-to-prove properties

	○ Why are they hard to prove? • Interpreting Unproven results • Example of an Unproven property on a real design (or a student example design)
10.00	**2 Overview of techniques for resolving Unproven's** • 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** • 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 (or a student example 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** • Start with the following advanced techniques: abstractions, cut points, free variables, undriven signals, • Example of applying these techniques on a real design (or a student example design)
15.30	Break
16.00	**5 Applying advanced techniques for resolving Unproven's** • 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 (or a student example design) **6 Summary** • 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