

Preface

Having been a design and verification engineer of CPUs and SoCs for over 20 years, I've come to realize that the design verification field is very exhaustive in its breadth and depth. Knowing only SystemVerilog and UVM may not suffice. Sure, you need to know UVM (Universal Verification Methodology) but also SVA (SystemVerilog Assertions), SFC (SystemVerilog Functional Coverage), CRV (constrained random verification), CDC (clock domain crossing) verification, interconnect NoC (Network on Chip) verification, AMS (analog/mixed signal) verification, low-power verification (UPF), hardware acceleration and emulation, hardware/software co-verification, and static formal (aka static functional aka formal property check) verification technologies and methodologies.

I noticed that there isn't a book that gives a good overview (high level but with sufficient detail) of the technologies and methodologies at hand. Engineers rely on white papers, blogs, and EDA vendor literature to get some understanding of many of these topics. That was the impetus for this book. I have covered all the aforementioned topics in this book.

The book is written such that the reader gets a good comprehensive overview of a given topic but also enough detail to get a good grasp on the topic. The book has a comprehensive bibliography that points to further reading material that the reader can pursue.

The book is meant for managers, decision makers, as well as engineers: managers who want a quick introduction to the technologies and methodologies and engineers who want sufficient detail with applications, which they can then further pursue.

Chapter 1. This chapter introduces the current state of design verification in the industry. Where do the bugs come from? Are they mostly functional bugs?

Chapter 2. This chapter discusses the overall design verification (DV) challenges and solutions. Why is DV still such a long pole in the design cycle? We will discuss a comprehensive verification plan and assess the type of expertise required and how Develop => Simulate => Debug => Cover loop can be improved.

Chapter 3. This chapter discusses what SystemVerilog is. Is it a monolithic language? Or is it an umbrella under which many languages with different syntax and semantics work through a single simulation kernel? How did SystemVerilog come about?

Chapter 4. This chapter describes in detail the architecture of UVM and UVM hierarchy and discusses each hierarchical component (testbench, test, environment, agent, scoreboard, driver, monitor, sequencer, etc.) in detail. Two complete examples of UVM are provided to solidify the concepts.

Chapter 5. This chapter describes constrained random verification (CRV). CRV allows you to constrain your stimulus to better target a design function, thereby allowing you to reach your coverage goal faster with accuracy. From that sense, functional coverage and CRV go hand in hand. You check your coverage and see where the coverage holes are. You then constrain your stimulus to target those holes and improve coverage.

Chapter 6. This chapter discusses SVA (SystemVerilog Assertions) methodology, SVA and functional coverage-driven methodology, and plenty of applications to solidify the concepts. SystemVerilog Assertions (SVA) are one of the most important components of SystemVerilog when it comes design verification. SVA is instrumental in finding corner cases, ease of debug, and coverage of design's sequential logic.

Chapter 7. This chapter discusses differences between code and functional coverage and SFC (SystemVerilog Functional Coverage) fundamentals such as “cover-group,” “coverpoint,” “cross,” “transition,” etc. along with complete examples. SystemVerilog Functional Coverage (SFC) is an important component that falls within SystemVerilog.

Chapter 8. This chapter starts with the understanding of metastability and then dives into different synchronizing techniques. It also discusses the role of SystemVerilog Assertions in verification of CDC (clock domain crossing). We will then discuss a complete methodology. CDC has become an ever-increasing problem in multi-clock domain designs. One must solve issues not only at RTL level but also consider the physical timing.

Chapter 9. This chapter discusses challenges and solutions of low-power verification. It goes into sufficient depth of UPF (Unified Power Format) and how it applies to design from Spec to GDSII level. Finally, it discusses low-power estimation and verification at ESL level.

Chapter 10. This chapter discusses static verification, aka formal-based verification. Static verification is an umbrella term, and there are many different technologies that fall under it: for example, logic equivalency check (LEC), clock domain crossing check (CDC), X-state verification, low-power structural checks, ESL $\leftrightarrow$ RTL equivalency, etc. This chapter discusses all these topics and a lot more including state-space explosion problem and the role of SystemVerilog Assertions.

Chapter 11. This chapter discusses ESL (electronic system-level) technology, methodology and OSCI TLM2.0 standard definition, virtual platform examples, and how to use a virtual platform for design verification, among other topics.

Chapter 12. This chapter discusses the methodologies to develop software such that it is ready when hardware is ready to ship. What kind of platform do you need? How does ESL virtual platform play a key role? How do emulators and accelerators fit in the methodology equation?

Chapter 13. This chapter discusses major challenges and solutions of AMS (analog/mixed signal), the current state of affair, analog model abstraction levels, real number modeling, SystemVerilog Assertions-based methodology, etc.

Chapter 14. This chapter discusses challenges, solutions, and methodologies of SoC interconnect verification and a couple of EDA vendor solutions, among other things. Today's SoC contains hundreds of pre-verified IPs, memory controller, DMA engines, etc. All these components need to communicate with each other using many different interconnect technologies (cross-bus based, NoC (Network on Chip) based, etc.).

Chapter 15. This chapter describes a complete product life cycle. Simply designing the ASIC and shipping it is not enough. One needs to also consider the board design into which this ASIC will be used. This chapter describes board-level design issues and correlation between the board-level system design and the ASIC design and verification.

Chapter 16. This chapter discusses, in detail, verification of a complex SoC, namely, a voice over IP network SoC. We will go through a comprehensive verification plan and describe each verification step with VoIP SoC-based real-life design.

Chapter 17. This chapter discusses, in detail, verification of a cache subsystem of a large SoC. We will go through a comprehensive verification plan and describe each verification step with a real-life Cache subsystem SoC verification strategy. This chapter discusses the verification methodology using UVM agents.

Chapter 18. This chapter discusses, in detail, verification of a cache subsystem of a larger SoC. We will go through a comprehensive verification plan and describe each verification step with a real-life Cache subsystem SoC. This chapter discusses the verification methodology using an instruction set simulator (ISS) (as opposed to an UVM agent which is described in Chap. 17).

<http://www.springer.com/978-3-319-59417-0>

ASIC/SoC Functional Design Verification
A Comprehensive Guide to Technologies and
Methodologies

Mehta, A.B.

2018, XXXI, 328 p. 175 illus., 160 illus. in color.,

Hardcover

ISBN: 978-3-319-59417-0