

Preface

We are immersed in the world of computing in our everyday lives. We are aware of computing systems when we use desktops, laptops, or servers. In many scenarios, we interact with systems that have embedded computing in them, such as cell phones, gadgets, and monitoring systems. When we fly on an airplane or even drive a car, there are many computing devices working together to ensure that our journey is pleasant and safe. Can we assume that these embedded computing systems are correct by construction and therefore we can safely rely on them? The simple answer is that no one can prove their absolute infallibility. Yet, since we do not have a choice, we can sit back, relax, and enjoy life!

Consider a simple *adder* to understand the difficulty in verifying today's systems. An adder is one of the simplest computations in a calculator. It adds two input values and produces the result. Typically, the input values are 32-bit integers. Therefore, to verify this adder we have to simulate several trillions ($2^{32} \times 2^{32}$) of test vectors. Clearly, it is too expensive to apply trillions of input vectors to verify an adder. In fact, it is infeasible to simulate an adder using all possible input sequences when the input values are 64-bit integers. If we cannot completely verify a simple adder, what is the hope that we can verify today's embedded and hybrid systems that consist of complex software and hardware including multicore/manycore processors, memories, buses, controllers, interfaces, and so on? Functional validation is widely acknowledged as a major bottleneck in System-on-Chip (SoC) design methodology—up to 70 % of the overall design time and resources are spent on functional validation. In spite of such extensive efforts, many SoC designs fail at the very first time (silicon failures) due to functional errors. The functional verification complexity is expected to increase further due to the combined effects of increasing design complexity and reduced time-to-market constraints.

Existing validation approaches use a combination of simulation-based techniques and formal verification methods. Simulation is the most widely used form of validation using random and constrained-random tests. For instance, in the adder example, instead of trying all possible input combinations, validation engineers can create tests for interesting scenarios based on coverage criteria and corner cases. Since it is impractical to generate and apply all possible tests,

simulation-based validation does not guarantee correctness. On the other hand, formal verification methods (such as model checking) do not use input vectors but explore the state space to ensure correctness. Unfortunately, this exploration can lead to state space explosion for large designs. Today's state-of-the-art verification methodologies use an efficient combination of both approaches. For example, the complete SoC design is simulated using billions of tests whereas very critical components such as controllers or specific protocols are fully verified by formal methods.

There are various promising directions to significantly reduce the overall validation effort by exploiting the synergies between simulation and formal verification. It is beneficial to verify as much as possible at the specification level (which is orders-of-magnitude simpler than implementation but has all the functional details) and automatically reuse high-level validation efforts for verifying lower level implementation. Similarly, it is promising to use directed tests for design validation since directed tests can achieve the same coverage goal using orders-of-magnitude less tests compared to random or constrained-random tests. As a result, simulation using efficient directed tests can drastically reduce the overall validation effort. However, directed test generation is mostly performed by human intervention. Hand-written tests entail laborious and time-consuming effort of verification engineers who have deep knowledge of the design under verification. For a complex design, it is infeasible to manually generate all directed tests to achieve a comprehensive coverage goal. Therefore, it is necessary to develop tools and techniques for automated generation of directed tests.

This book describes innovative ways of combining simulation and formal methods for verifying complex systems. It provides a comprehensive coverage of system-level modeling and validation techniques to both reduce overall validation effort and improve the product quality. It describes challenges associated with verifying complex systems and presents efficient techniques to address these challenges. It studies various system-level modeling approaches using SystemC, UML, and transaction level models. It presents state-of-the-art techniques for automated generation of directed tests focusing on reduction of test generation time and validation effort through clustering, decomposition, and learning techniques. It also describes innovative ways of reusing high-level validation effort across different abstraction levels.

The reader of this book will get a comprehensive understanding of design verification challenges and how system-level validation can significantly improve design quality and reduce overall cost. [Chapter 1](#) highlights the benefits of system-level validation. [Chapter 2](#) describes how to specify complex systems using high-level models. [Chapter 3](#) provides the basic framework for automated generation of directed tests. The next chapter presents how to significantly reduce the number of tests without sacrificing the coverage goal. [Chapter 5–9](#) describe efficient techniques for generation of directed tests. The next two chapters present test generation techniques focusing on validation of multicore/manycore architectures. [Chapter 12](#) describes innovative ways of reusing high-level tests and assertions for validation of implementation. Finally, [Chapter 13](#) concludes the book with a short

discussion of future research directions. We hope you enjoy reading this book and find the information useful for your purpose.

Audience

This book is designed for senior undergraduate students, graduate students, researchers, CAD tool developers, designers, and managers interested in development of efficient tools and techniques for system-level design and verification, directed test generation, and functional validation of heterogeneous SoC designs.

System-Level Validation
High-Level Modeling and Directed Test Generation
Techniques

Chen, M.; Qin, X.; Koo, H.-M.; Mishra, P.

2013, XXII, 250 p., Hardcover

ISBN: 978-1-4614-1358-5