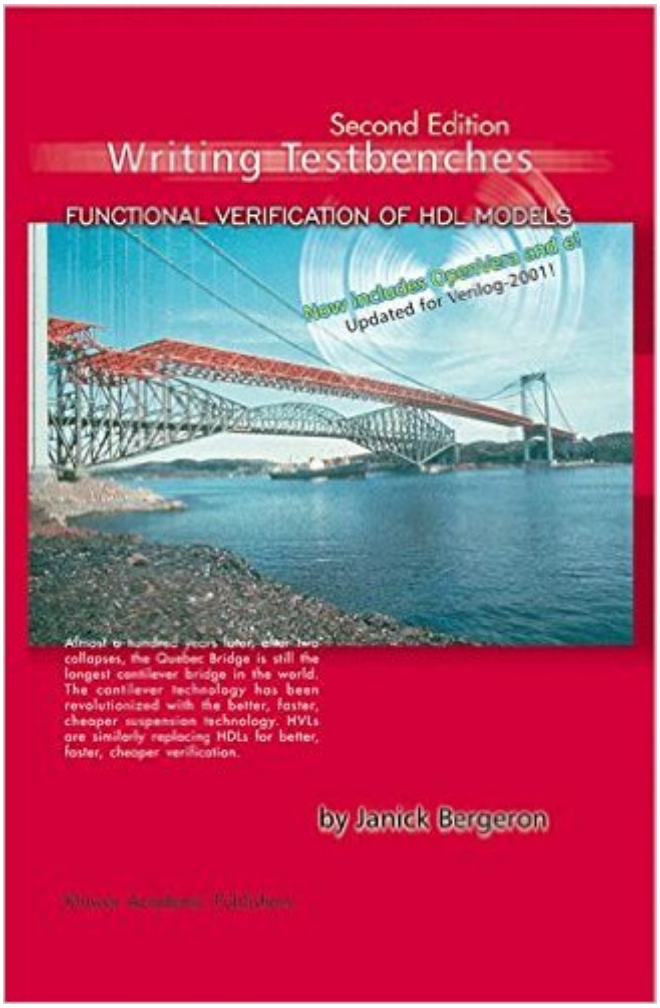
Writing Testbenches: Functional Verification Of HDL Models



Synopsis

mental improvements during the same period. What is clearly needed in verification techniques and technology is the equivalent of a synthesis productivity breakthrough. In the second edition of Writing Testbenches, Bergeron raises the verification level of abstraction by introducing coverage-driven constrained-random transaction-level self-checking testbenchesÂ- all made possible through the introduction of hardware verification languages (HVLs), such as e from Verisity and OpenVera from Synopsys. The state-of-art methodologies described in Writing TestÂ- benches will contribute greatly to the much-needed equivalent of a synthesis breakthrough in verification productivity. I not only highly recommend this book, but also I think it should be required reading by anyone involved in design and verification of today's ASIC, SoCs and systems. Harry Foster Chief Architect Verplex Systems, Inc. xviii Writing Testbenches: Functional Verification of HDL Models PREFACE If you survey hardware design groups, you will learn that between 60% and 80% of their effort is now dedicated to verification.

Book Information

Hardcover: 478 pages

Publisher: Springer; 2nd edition (February 1, 2003)

Language: English

ISBN-10: 1402074018

ISBN-13: 978-1402074011

Product Dimensions: 6.1 x 1.1 x 9.2 inches

Shipping Weight: 1.8 pounds (View shipping rates and policies)

Average Customer Review: 3.8 out of 5 starsÂ Â See all reviewsÂ (9 customer reviews)

Best Sellers Rank: #1,972,026 in Books (See Top 100 in Books) #131 inÂ Books > Computers & Technology > Programming > Functional #374 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compilers #562 inÂ Books > Textbooks > Medicine & Health Sciences > Medicine > Clinical > Anesthesiology

Customer Reviews

This book covers many facets of the task of creating testbenches. However, it doesn't seem to follow a very well thought out plan, and there are holes in the coverage.Most of the book is a 'tips and tricks' coverage of how to get each language to do what it wasn't designed to do. He walks through various situations and says that something is easy to do in Specman (shows a short code fragment), but then goes into long detail in how to get around VHDL's limitations and get the same

result. I realize these are probably pretty cool tricks, but not at all the approach for me (a beginner to writing sizable testbenches). If he kept up the coverage of all 4 languages throughout, it might be useful, but the focus shifts from language to language at whim. You won't learn how to write a testbench as much as you will learn some pitfalls to avoid.One more gripe before I get to the parts I liked. Each chapter ends with a summary. The summary lists the author's favorite tricks, not a summary of the whole chapter. I found these to be not at all helpful in either deciding whether to read the chapter, or as a review of what was covered.I did like the explanations of:-- The importance of verification (now I know why I was hired)-- Overview of all the lingo (I can sound like I know what I'm talking about now, even if I don't)-- Merits of the various types of coverage (code/functional/transition ...)-- Aspect Oriented Programming (e) and why it is useful (cool stuff!)-- Using coverage to drive a random benchThat is only about 10% of the book, however. That 10% was really pretty good.I see one of the other reviewers complained about lack of downloadable sourcecode. It is available at [...

quality requirements for rotors in a constant (rigid) state -- Part 1: Specification and verification of balance tolerances Verification and Validation in Scientific Computing