Hardware Verification With C : A Practitioner's Handbook: A Practitioner's Approach

This book constitutes the proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2016, which took place in Eindhoven, The Netherlands, in April 2016, held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016. The 44 full papers presented in this volume were carefully reviewed and selected from 175 submissions. They were organized in topical sections named: abstraction and verification; probabilistic and stochastic systems; synthesis; tool papers; concurrency; tool demos; languages and automata; security; optimization; and competition on software verification – SV-COMP.

Based on the highly successful second edition, this extended edition of SystemVerilog for Verification: A Guide to Learning the Testbench Language Features teaches all verification features of the SystemVerilog language, providing hundreds of examples to clearly explain the concepts and basic fundamentals. It contains materials for both the full-time verification engineer and the student learning this valuable skill. In the third edition, authors Chris Spear and Greg Tumbush start with how to verify a design, and then use that context to demonstrate the language features, including the advantages and disadvantages of different styles, allowing readers to choose between alternatives. This textbook contains end-of-chapter exercises designed to enhance students' understanding of the material. Other features of this revision include: New sections on static variables, print specifiers, and DPI from the 2009 IEEE language standard Descriptions of UVM features such as factories, the test registry, and the configuration database Expanded code samples and explanations Numerous samples that have been tested on the major SystemVerilog simulators SystemVerilog for Verification: A Guide to Learning the Testbench Language Features, Third Edition is suitable for use in a one-semester SystemVerilog course on SystemVerilog at the undergraduate or graduate level. Many of the improvements to this new edition were compiled through feedback provided from hundreds of readers.

"As chip size and complexity continues to grow exponentially, the challenges of functional verification are becoming a critical issue in the electronics industry. It is now commonly heard that logical errors missed during functional verification are the most common cause of chip re-spins, and that the costs associated with functional verification are now outweighing the costs of chip design. To cope with these challenges engineers are increasingly relying on new design and verification methodologies and languages. Transaction-based design and verification, constrained random stimulus generation, functional coverage analysis, and assertion-based verification are all techniques that advanced design and verification teams routinely use today. Engineers are also increasingly turning to design and verification models based on C/C++ and SystemC in order to build more abstract, higher performance hardware and software models and to escape the limitations of RTL HDLs. This new book, Advanced Verification Techniques, provides specific guidance for these advanced verification techniques. The book includes realistic examples and shows how SystemC and SCV can be applied to a variety of advanced design and verification tasks." - Stuart Swan

Visit the authors' companion site! http://www.electronicsystemlevel.com/ - Includes interactive forum with the authors! Electronic System Level (ESL) design has mainstreamed – it is now an established approach at most of the world's leading system-on-chip (SoC) design companies and is being used increasingly in system design. From its genesis as an algorithm modeling methodology with 'no links to implementation', ESL is evolving into a set of complementary methodologies that enable embedded system design, verification and debug through to the hardware and software implementation of custom SoC, system-on-FPGA, system-on-board, and entire multi-board systems. This book arises from experience the authors have gained from years of work as industry practitioners in the Electronic System Level design area; they have seen "SLD" or "ESL" go through many stages and false starts, and have observed that the shift in design methodologies to ESL is finally occurring. This is partly because of ESL technologies themselves are stabilizing on a useful set of languages being standardized (SystemC is the most notable), and use models are being identified that are beginning to get real adoption. ESL DESIGN & VERIFICATION offers a true prescriptive guide to ESL that reviews its past and outlines the best practices of today. Table of Contents CHAPTER 1: WHAT IS ESL? CHAPTER 2: TAXONOMY AND DEFINITIONS FOR THE ELECTRONIC SYSTEM LEVEL CHAPTER 3: EVOLUTION OF ESL DEVELOPMENT CHAPTER 4: WHAT ARE THE ENABLERS OF ESL? CHAPTER 5: ESL FLOW CHAPTER 6: SPECIFICATIONS AND MODELING CHAPTER 7: PRE-PARTITIONING ANALYSIS CHAPTER 8: PARTITIONING CHAPTER 9: POST-PARTITIONING ANALYSIS AND DEBUG CHAPTER 10: POST-PARTITIONING VERIFICATION CHAPTER 11: HARDWARE IMPLEMENTATION CHAPTER 12: SOFTWARE IMPLEMENTATION CHAPTER 13: USE OF ESL FOR IMPLEMENTATION VERIFICATION CHAPTER 14: RESEARCH, EMERGING AND FUTURE PROSPECTS APPENDIX: LIST OF ACRONYMS * Provides broad, comprehensive coverage not available in any other such book * Massive global appeal with an internationally recognised author team * Crammed full of state of the art content from notable industry experts

22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings

First International Haifa Verification Conference, Haifa, Israel, November 13-16, 2005, Revised Selected Papers

Programming Embedded Systems

Blue Book

Hardware Verification

A Practitioner's Handbook

Offers users the first resource guide that combines both the methodology and basics of SystemVerilog Addresses how all these pieces fit together and how they should be used to verify complex chips rapidly and thoroughly. Unique in its broad coverage of SystemVerilog, advanced functional verification, and the combination of the two.

This book is intended as an innovative overview of current formal verification methods, combined with an in-depth analysis of some advanced techniques to improve the scalability of these methods, and close the gap between design and verification in computer-aided design. Formal Verification: Scalable Hardware Verification with Symbolic Simulation explains current formal verification methods and provides an in-depth analysis of some advanced techniques to improve the scalability of these methods and close the gap between design and verification in computer-aided design. It provides the theoretical background required to present such methods and advanced techniques, i.e. Boolean function representations, models of sequential networks and, in particular, some novel algorithms to expose the disjoint support decompositions of Boolean functions, used in one of the scalable approaches.

This book will explain how to verify SoC (Systems on Chip) logic designs using "formal and "semiformal verification techniques. The critical issue to be addressed is whether the functionality of the design is the one that the designers intended. Simulation has been used for checking the correctness of SoC designs (as in "functional verification), but many subtle design errors cannot be caught by simulation. Recently, formal verification, giving mathematical proof of the correctness of designs, has been gaining popularity. For higher design productivity, it is essential to debug designs as early as possible, which this book facilitates. This book covers all aspects of high-level formal and semiformal verification techniques for system level designs. • First book that covers all aspects of formal and semiformal, high-level (higher than RTL) design verification targeting SoC designs. • Formal verification of high-level designs (RTL or higher). • Verification techniques are discussed with associated system-level design methodology.

This volume contains the proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004). TACAS 2004 took place in Barcelona, Spain, from March 29th to April 2nd, as part of the 7th European Joint Conferences on Theory and Practice of Software (ETAPS 2004), whose aims, organization, and history are detailed in a foreword by the ETAPS Steering Committee Chair, Jos´ e Luiz Fiadeiro. TACAS is a forum for researchers, developers, and users interested in ri- rously based tools for the construction and analysis of systems. The conference serves to bridge the gaps between di?erent communities including, but not - mited to, those devoted to formal methods, software and hardware veri?cation, static analysis, programming languages, software engineering, real-time systems, and communication protocols that share common interests in, and techniques for, tool development. In particular, by providing a venue for the discussion of common problems, heuristics, algorithms, data structures, and methodologies, TACAS aims to support researchers in their quest to improve the utility, rel- bility, ?exibility, and e?ciency of tools for building systems. TACASseekstheoreticalpaperswithaclearlinktotoolconstruction,papers describingrelevantalgorithmsandpracticalaspectsoftheirimplementation,- pers giving descriptions of tools and associated methodologies, and case studies with a conceptual message.

6th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2006, Bertinoro, Italy, May 22-27, 2006, Advances Lectures

Hardware Design Verification

Methods and Systems in Comparison

Current Trends in Hardware Verification and Automated Theorem Proving

A Guide to Learning the Testbench Language Features

Simulation and Formal Method-based Approaches

This book constitutes the refereed post-proceedings of the First International Conference on Hardware Verification, Software Testing, and PADTAD held in November 2005. The conference combines the sixth IBM Verification Workshop, the fourth IBM Software Testing Workshop, and the third PADTAD (Parallel and Distributed Systems: Testing and Debugging) Workshop. The 14 revised full papers presented together with three invited contributions were carefully reviewed and selected from 31 submissions. The papers address all current issues in hardware/software verification, software testing, and testing of parallel and concurrent applications.

One of the biggest challenges in chip and system design is determining whether the hardware works correctly. That is the job of functional verification engineers and they are the audience for this comprehensive text from three top industry professionals. As designs increase in complexity, so has the value of verification engineers within the hardware design team. In fact, the need for skilled verification engineers has grown dramatically--functional verification now consumes between 40 and 70% of a project's labor, and about half its cost. Currently there are very few books on verification for engineers, and none that cover the subject as comprehensively as this text. A key strength of this book is that it describes the entire verification cycle and details each stage. The organization of the book follows the cycle, demonstrating how functional verification engages all aspects of the overall design effort and how individual cycle stages relate to the larger design process. Throughout the text, the authors leverage their 35 plus years experience in functional verification, providing examples and case studies, and focusing on the skills, methods, and tools needed to complete each verification task. Additionally, the major vendors (Mentor Graphics, Cadence Design Systems, Verisity, and Synopsys) have implemented key examples from the text and made these available on line, so that the reader can test out the methods described in the text. * Comprehensive overview of the complete verification cycle * Combines industry experience with a strong emphasis on functional verification fundamentals * Includes real-world case studies and downloadable software implementations of key examples from the major vendors (Mentor Graphics, Cadence Design Systems, Verisity, and Synopsys)

One of the biggest challenges in chip and system design is determining whether the hardware works correctly. That is the job of functional verification engineers and they are the audience for this comprehensive text from three top industry professionals. As designs increase in complexity, so has the value of verification engineers within the hardware design team. In fact, the need for skilled verification engineers has grown dramatically--functional verification now consumes between 40 and 70% of a project's labor, and about half its cost. Currently there are very few books on verification for engineers, and none that cover the subject as comprehensively as this text. A key strength of this book is that it describes the entire verification cycle and details each stage. The organization of the book follows the cycle, demonstrating how functional verification engages all aspects of the overall design effort and how individual cycle stages relate to the larger design process. Throughout the text, the authors leverage their 35 plus years experience in functional verification, providing examples and case studies, and focusing on the skills, methods, and tools needed to complete each verification task. Comprehensive overview of the complete verification cycle Combines industry experience with a strong emphasis on functional verification fundamentals Includes real-world case studies

This report describes the partially completed correctness proof of the Viper 'block model'. Viper [7,8,9,11,23] is a microprocessor designed by W. J. Cullyer, C. Pygott and J. Kershaw at the Royal Signals and Radar Establishment in Malvern, England, (henceforth 'RSRE') for use in safety-critical applications such as civil aviation and nuclear power plant control. It is currently finding uses in areas such as the de ployment of weapons from tactical aircraft. To support safety-critical applications, Viper has a particulary simple design about which it is relatively easy to reason using current techniques and models. The designers, who deserve much credit for the promotion of formal methods, intended from the start that Viper be formally verified. Their idea was to model Viper in a sequence of decreasingly abstract levels, each of which concentrated on some aspect ofthe design, such as the flow ofcontrol, the processingofinstructions, and so on. That is, each model would be a specification of the next (less abstract) model, and an implementation of the previous model (if any). The verification effort would then be simplified by being structured according to the sequence of abstraction levels. These models (or levels) of description were characterized by the design team. The first two levels, and part of the third, were written by them in a logical language amenable to reasoning and proof.

Verification Techniques for System-Level Design

Scalable Hardware Verification with Symbolic Simulation

SystemVerilog for Verification

An Object-Oriented Framework

Reasoning about High-Level Constructs in Hardware/Software Formal Verification

Comprehensive Functional Verification

Methods for detecting logical errors in computer hardware designs using symbolic manipulation instead of digital simulation are discussed. A non-procedural register transfer language is proposed that is suitable for describing how a digital circuit should perform. This language can also be used to describe each of the components used in the design. Transformations are presented which should enable the designer to either prove or disprove that the set of interconnected components correctly satisfy the specifications for the overall system. The problem of detecting timing anomalies such as races, hazards, and oscillations is addressed. Also explored are some interesting relationships between the problems of hardware verification and program verification. Finally, the results of using an existing proof checking program on some digital circuits are presented. Although the theorem proving approach is not very efficient for simple circuits, it becomes increasingly attractive as circuits become more complex. This is because the theorem proving approach can use complicated component specifications without reducing them to the gate level. (Author).

This book constitutes the thoroughly refereed post-proceedings of the Second International Haifa Verification Conference, HVC 2006, held in Haifa, Israel, in October 2006. The 15 revised full papers presented together with 2 invited lectures are organized in three topical tracks on hardware verification technologies and methodologies, software testing, and tools for hardware verification and software testing.

This book constitutes the thoroughly refereed post-conference proceedings of the 7th International Haifa Verification Conference, HVC 2011, held in Haifa, Israel in December 2011. The 15 revised full papers presented together with 3 tool papers and 4 posters were carefully reviewed and selected from 43 submissions. The papers are organized in topical sections on synthesis, formal verification, software quality, testing and coverage, experience and tools, and posters- student event.

This book explores C-based design, implementation, and analysis of post-quantum cryptography (PQC) algorithms for signature generation and verification. The authors investigate NIST round 2 PQC algorithms for signature generation and signature verification from a hardware implementation perspective, especially focusing on C-based design, power-performance-area-security (PPAS) trade-offs and design flows targeting FPGAs and ASICs. Describes a comprehensive set of synthesizable c code base as well as the hardware implementations for the different types of PQC algorithms including lattice-based, code-based, and multivariate-based; Demonstrates the hardware (FPGA and ASIC) and hardware-software optimizations and trade-offs of the NIST round 2 signature-based PQC algorithms; Enables designers to build hardware implementations that are resilient to a variety of side-channels.

Third International Haifa Verification Conference, HVC 2007, Haifa, Israel, October 23-25, 2007, Proceedings

10th IFIP WG10.5 Advanced Research Working Conference, CHARME'99, Bad Herrenalb, Germany, September 27-29, 1999, Proceedings

System Design with SystemCTM

Hardware Architectures for Post-Quantum Digital Signature Schemes

Hardware and Software: Verification and Testing

A Prescription for Electronic System Level Methodology

This book constitutes the refereed proceedings of the 12th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods, CHARME 2003, held in L'Aquila, Italy in October 2003. The 24 revised full papers and 8 short papers presented were carefully reviewed and selected from 65 submissions. The papers are organized in topical sections on software verification, automata based methods, processor verification, specification methods, theorem proving, bounded model checking, and model checking and applications.

Describes a small verification library with a concentration on user adaptability such as re-useable components, portable Intellectual Property, and co-verification. Takes a realistic view of reusability and distills lessons learned down to a tool box of techniques and guidelines.

Are you an RTL or system designer that is currently using, moving, or planning to move to an HLS design environment? Finally, a comprehensive guide for designing hardware using C++ is here. Michael Fingeroff's High-Level Synthesis Blue Book presents the most effective C++ synthesis coding style for achieving high quality RTL. Master a totally new design methodology for coding increasingly complex designs! This book provides a step-by-step approach to using C++ as a hardware design language, including an introduction to the basics of HLS using concepts familiar to RTL designers. Each chapter provides easy-to-understand C++ examples, along with hardware and timing diagrams where appropriate. The book progresses from simple concepts such as sequential logic design to more complicated topics such as memory architecture and hierarchical sub-system design. Later chapters bring together many of the earlier HLS design concepts through their application in simplified design examples. These examples illustrate the fundamental principles behind C++ hardware design, which will translate to much larger designs. Although this book focuses primarily on C and C++ to present the basics of C++ synthesis, all of the concepts are equally applicable to SystemC when describing the core algorithmic part of a design. On completion of this book, readers should be well on their way to becoming experts in high-level synthesis.

This book constitutes the thoroughly refereed post proceedings of the 5th International Haifa Verification Conference, HVC 2009, held in Haifa, Israel in October 2009. The 11 revised full papers presented together with four abstracts of invited lectures were carefully reviewed and selected from 23 submissions. The papers address all current issues, challenges and future directions of verification for hardware, software, and hybrid systems and present academic research in the verification of systems, generally divided into two paradigms - formal verification and dynamic verification (testing).
Advanced Verification Techniques
7th International Haifa Verification Conference, HVC 2011, Haifa, Israel, December 6-8, 2011, Revised Selected Papers
Using Data Mining to Increase Controllability and Observability in Functional Verification
10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings
12th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2003, L'Aquila, Italy, October 21-24, 2003, Proceedings
Tools and Algorithms for the Construction and Analysis of Systems

This book presents 8 papers accompanying the lectures of leading researchers given at the 6th edition of the International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2006). SFM 2006 was devoted to formal techniques for hardware verification and covers several aspects of the hardware design process, including hardware design languages and simulation, property specification formalisms, automatic test pattern generation, symbolic trajectory evaluation, and more.
I am glad to see this new book on the e language and on verification. I am especially glad to see a description of the e Reuse Methodology (eRM). The main goal of verification is, after all, finding more bugs quicker using given resources, and verification reuse (module-to-system, old-system-to-new-system etc.) is a key enabling component. This book offers a fresh approach in teaching the e hardware verification language within the context of coverage driven verification methodology. I hope it will help the reader und- stand the many important and interesting topics surrounding hardware verification. Yoav Hollander Founder and CTO, Verisity Inc. Preface This book provides a detailed coverage of the e hardware verification language (HVL), state of the art verification methodologies, and the use of e HVL as a facilitating verification tool in implementing a state of the art verification environment. It includes comprehensive descriptions of the new concepts introduced by the e language, e language syntax, and its as- ciated semantics. This book also describes the architectural views and requirements of verifi- tion environments (randomly generated environments, coverage driven verification environments, etc.), verification blocks in the architectural views (i. e. generators, initiators, c- lectors, checkers, monitors, coverage definitions, etc.) and their implementations using the e HVL. Moreover, the e Reuse Methodology (eRM), the motivation for defining such a gui- line, and step-by-step instructions for building an eRM compliant e Verification Component (eVC) are also discussed.
CHARME'99 is the tenth in a series of working conferences devoted to the dev- opment and use of leading-edge formal techniques and tools for the design and veri?cation of hardware and systems. Previous conferences have been held in Darmstadt (1984), Edinburgh (1985), Grenoble (1986), Glasgow (1988), Leuven (1989), Torino (1991), Arles (1993), Frankfurt (1995) and Montreal (1997). This workshop and conference series has been organized in cooperation with IFIP WG 10. 5. It is now the biannual counterpart of FMCAD, which takes place every even-numbered year in the USA. The 1999 event took place in Bad Her- nalb, a resort village located in the Black Forest close to the city of Karlsruhe. The validation of functional and timing behavior is a major bottleneck in current VLSI design systems. A predominantly academic area of study until a few years ago, formal design and veri?cation techniques are now migrating into industrial use. The aim of CHARME'99 is to bring together researchers and users from academia and industry working in this active area of research. Two invited talks illustrate major current trends: the presentation by G ?erard Berry (Ecole des Mines de Paris, Sophia-Antipolis, France) is concerned with the use of synchronous languages in circuit design, and the talk given by Peter Jansen (BMW, Munich, Germany) demonstrates an application of formal methods in an industrial environment. The program also includes 20 regular presentations and 12 short presentations/poster exhibitions that have been selected from the 48 submitted papers.
*Hardware/software co-verification is how to make sure that embedded system software works correctly with the hardware, and that the hardware has been properly designed to run the software successfully -before large sums are spent on prototypes or manufacturing. This is the first book to apply this verification technique to the rapidly growing field of embedded systems-on-a-chip(SoC). As traditional embedded system design evolves into single-chip design, embedded engineers must be armed with the necessary information to make educated decisions about which tools and methodology to deploy. SoC verification requires a mix of expertise from the disciplines of microprocessor and computer architecture, logic design and simulation, and C and Assembly language embedded software. Until now, the relevant information on how it all fits together has not been available. Andrews, a recognized expert, provides in-depth information about how co-verification really works, how to be successful using it, and pitfalls to avoid. He illustrates these concepts using concrete examples with the ARM core - a technology that has the dominant market share in embedded system product design. The companion CD-ROM contains all source code used in the design examples, a searchable e-book version, and useful design tools. * The only book on verification for systems-on-a-chip (SoC) on the market * Will save engineers and their companies time and money by showing them how to speed up the testing process, while still avoiding costly mistakes * Design examples use the ARM core, the dominant technology in SoC, and all the source code is included on the accompanying CD-Rom, so engineers can easily use it in their own designs*
With C and GNU Development Tools
Introduction to Formal Hardware Verification
5th International Haifa Verification Conference, HCV 2009, Haifa, Israel, October 19-22, 2009, Revised Selected Papers

High-level Synthesis
Vos

Verification is increasingly complex, and SystemVerilog is one of the languages that the verification community is turning to. However, no language by itself can guarantee success without proper techniques. Object-oriented programming (OOP), with its focus on managing complexity, is ideally suited to this task. With this handbook—the first to focus on applying OOP to SystemVerilog—we'll show how to manage complexity by using layers of abstraction and base classes. By adapting these techniques, you will write more "reasonable" code, and build efficient and reusable verification components. Both a learning tool and a reference, this handbook contains hundreds of real-world code snippets and three professional verification-system examples. You can copy and paste from these examples, which are all based on an open-source, vendor-neutral framework (with code freely available at www.trusster.com). Learn about OOP techniques such as these: Creating classes—code interfaces, factory functions, reuse Connecting classes—pointers, inheritance, channels Using "correct by construction"—strong typing, base classes Packaging it up—singletons, static methods, packages
As part of the Modern Semiconductor Design series, this book details a broad range of e-based topics including modelling, constraint-driven test generation, functional coverage and assertion checking.
This advanced textbook presents an almost complete overview of techniques for hardware verification. It covers all approaches used in existing tools, such as binary and word-level decision diagrams, symbolic methods for equivalence and temporal logic model checking, and introduces the use of higher-order logic theorem proving for verifying circuit correctness. Each chapter contains an introduction and a summary as well as a section for the advanced reader, aiding an understanding of the advantages and limitations of each technique. Backed by many examples and illustrations, this text will appeal to a broad audience, from beginners in system design to experts. XXXXXXX Neuer Text This is a complete overview of existing techniques for hardware verification. It covers all approaches used in existing verification tools, such as symbolic methods for equivalence checking, temporal logic model checking, and higher-order logic theorem proving for verifying circuit correctness. The book helps readers to understand the advantages and limitations of each technique. Each chapter contains a summary as well as a section for the advanced reader.
The ever shrinking feature size of modern electronic chips leads to more designs being done as well as more complex chips being designed. These in turn lead to greater use of high-level specifications and to more sophisticated optimizations applied at the word -level. These steps make it more difficult to verify that the final design is faithful to the initial specification. We tackle two steps in this process and their formal equivalence checking to help verify the correctness of the steps. First, we present LEC, a combinational equivalence checking tool that is learning driven. It focuses on data-path equivalence checking with the goal of transforming the two logics under comparison to be more similar in order to reduce the complexity of a final Boolean (bit-level) solving. LEC does equivalence checking of combinational logic between two RTL (word-level) designs, one the original and one an optimized RTL version. LEC features an open architecture such that users and developers can learn with the system as new designs and optimizations are met, and then it can be modularly extended with new proof procedures as they are discovered. To address the use of higher level specifications, we build a simple trusted C to Verilog translation procedure based on the LLVM compiler infrastructure. The translator was designed to implement an almost vertatim translation of the C language operators and control structures into the Verilog \emph{always_ff} and \emph{always_comb} blocks through traversing LLVM Bytecode programs. The procedure reliably bridges the language barrier between software and hardware and allows hardware synthesis and verification techniques to be applied readily. In combination, these two procedures allow for equivalence checking between a software-like specification and an optimized word-level RTL implementation.
Verification Methodology Manual for SystemVerilog
Second International Haifa Verification Conference, HVC 2006, Haifa, Israel, October 23-26, 2006, Revised Selected Papers
The Complete Industry Cycle
Hardware Verification with C++
Formal Methods for Hardware Verification
Validation, Verification, and Testing of Computer Software

Hardware verification currently takes more than 50% of the whole verification time. There is a sustained effort to improve the efficiency of the verification process, which in the past helped deliver a large variety of supporting tools. The past years though did not see any major technology change that would bring the improvements that the process really needs (H. Foster 2013) (Wilson Research Group 2012). The existing approach to verification does not provide that type of qualitative jump anymore. This work is introducing a new tactic, providing a modern alternative to the existing approach to the verification problem. The novel approach I use in this research has the potential of significantly improve the process, way beyond incremental changes. It starts with acknowledging the huge amounts of data that follows the hardware development process from inception to the final product and in considering the data not as a quantitative by-product but as a qualitative supply of information on which we can develop a smarter verification. The approach is based on data already generated throughout the process currently used by verification engineers to zoom into the details of different verification aspects. By using existing machine learning approaches we can zoom out and use the same data to extract information, to gain knowledge that we can use to guide the verification process. This approach allows an apparent lack of accuracy introduced by data discovery, to achieve the overall goal. The latest advancements in machine learning and data mining offer a base of a new understanding and usage of the data that is being passed through the process. This work takes several practical problems for which the classical verification process reached a roadblock, and shows how the new approach can provide a jump in productivity and efficiency of the verification process. It focuses on four different aspects of verification to prove the power of this new approach: reducing effort redundancy, guiding verification to areas that need it first, decreasing time to diagnose, and designing tests for coverage efficiency.
This state-of-the-art monograph presents a coherent survey of a variety of methods and systems for formal hardware verification. It emphasizes the presentation of approaches that have matured into tools and systems usable for the actual verification of nontrivial circuits. All in all, the book is a representative and well-structured survey on the success and future potential of formal methods in proving the correctness of circuits. The various chapters describe the respective approaches supplying theoretical foundations as well as taking into account the application viewpoint. By applying all methods and systems presented to the same set of IFIP WG10.5 hardware verification examples, a valuable and fair analysis of the strenghts and weaknesses of the various approaches is given.
Authored by two of the leading authorities in the field, this guide offers readers the knowledge and skills needed to achieve proficiency with embedded software.
This volume contains the proceedings of the 3rd Haifa Veri?cation Conference (HVC 2007),whichtookplacein Haifa during October 2007.HVC isa forumfor researchers from both industry and academia to share and advance knowledge in the veri?cation of hardware and software systems. Academic research in veri?cation is generally divided into two paradigms – formal veri?cation and dynamic veri?cation (testing). Within each paradigm, di?erent algorithms and techniques are used for hardware and softwaresystems. Yet,attheircore,allofthesetechniquesaimtoachievethesamegoalofensuring the correct functionality of a complicated system. HVC is the only conference that brings together researchers from all four ?elds, thereby encouraging the migration of methods and ideas between domains. With this goal in mind we established the HVC Award. This award rec- nizes a promising contribution to veri?cation published in the last few years. It is aimed at developments that signi?cantly advance the state of the art in veri?cation technology and show potential for future impact on di?erent ver- cation paradigms. The winners of the HVC Award are chosen by an indep- dent committee with experts from all ?elds of veri?cation – both formal and dynamic, software and hardware. The winners of the 2007 HVC Award were Corina Pas? ? areanu and Willem Visser, for their work on combining static and dynamic analysis. This year we received 32 submissions, out of which 15 were accepted after a thorough review conducted by the Program Committee (PC) and additional reviewers.Eachpaper wasreviewedby atleastthree reviewers,sometimes more.
A SystemC Based Approach for Successful Tapeout
Hardware and Software, Verification and Testing
Formal Verification of Floating-Point Hardware Design
Hardware Verification with System Verilog
Correct Hardware Design and Verification Methods
ESL Design and Verification

I am honored and delighted to write the foreword to this very first book about SystemC. It is now an excellent time to summarize what SystemC really is and what it can be used for. The main message in the area of design in the 2001 International Te- nologyRoadmapfor Semiconductors (ITRS) isthat"cost ofdesign is the greatest threat to the continuation ofthe semiconductor roadmap. " This recent revision of the ITRS describes the major productivity improvements of the last few years as "small block reuse," "large block reuse ," and "IC implementation tools. " In order to continue to reduce design cost, the - quired future solutions will be "intelligent test benches" and "embedded system-level methodology. " As the new system-level specification and design language, SystemC - rectly contributes to these two solutions. These will have the biggest - pact on future design technology and will reduce system implementation cost. Ittook SystemC less than two years to emerge as the leader among the many new and well-discussed system-level designlanguages. Inmy op- ion, this is due to the fact that SystemC adopted object-oriented syst- level design—the most promising method already applied by the majority of firms during the last couple of years. Even before the introduction of SystemC, many system designers have attempted to develop executable specifications in C++. These executable functional specifications are then refined to the well-known transaction level, to model the communication of system-level processes.
Hardware Verification with C++A Practitioner's HandbookSpringer Science & Business Media
The Practical, Start-to-Finish Guide to Modern Digital Design Verification As digital logic designs grow larger and more complex, functional verification has become the number one bottleneck in the design process. Reducing verification time is crucial to project success, yet many practicing engineers have had little formal training in verification, and little exposure to the newest solutions.Hardware Design Verificationsystematically presents today's most valuable simulation-based and formal verification techniques, helping test and design engineers choose the best approach for each project, quickly gain confidence in their designs, and move into fabrication far more rapidly. College students will find that coverage of verification principles and common industry practices will help them prepare for jobs as future verification engineers. Author William K. Lam, one of the world's leading experts in design verification, is a recent winner of the Chairman's Award for Innovation, Sun Microsystems' most prestigious technical achievement award. Drawing on his wide-ranging experience, he introduces the foundational principles of verification, presents traditional techniques that have survived the test of time, and introduces emerging techniques for today's most challenging designs. Throughout, Lam emphasizes practical examples rather than mathematical proofs; wherever advanced math is essential, he explains it clearly and accessibly. Coverage includes Simulation-based versus formal verification: advantages, disadvantages, and tradeoffs Coding for verification: functional and timing correctness, syntactical and structure checks, simulation performance, and more Simulator architectures and operations, including event-driven, cycle-based, hybrid, and hardware-based simulators Testbench organization, design, and tools: creating a fast, efficient test environment Test scenarios and assertion: planning, test cases, test generators, commercial and Verilog assertions, and more Ensuring complete coverage, including code, parameters, functions, items, and cross-coverage The verification cycle: failure capture, scope reduction, bug tracking, simulation data dumping, isolation of underlying causes, revision control, regression, release mechanisms, and tape-out criteria An accessible introduction to the mathematics and algorithms of formal verification, from Boolean functions to state-machine equivalence and graph algorithms Decision diagrams, equivalence checking, and symbolic simulation Model checking and symbolic computation Simply put,Hardware Design Verificationwill help you improve and accelerate your entire verification process--from planning through tape-out--so you can get to market faster with higher quality designs.
This is the first book to focus on the problem of ensuring the correctness of floating-point hardware designs through mathematical methods. Formal Verification of Floating-Point Hardware Design advances a verification methodology based on a unified theory of register-transfer logic and floating-point arithmetic that has been developed and applied to the formal verification of commercial floating-point units over the course of more than two decades, during which the author was employed by several major microprocessor design companies. The book consists of five parts, the first two of which present a rigorous exposition of the general theory based on the first principles of arithmetic. Part I covers bit vectors and the bit manipulation primitives, integer and fixed-point encodings, and bit-wise logical operations. Part II addresses the properties of floating-point numbers, the formats in which they are encoded as bit vectors, and the various modes of floating-point rounding. In Part III, the theory is extended to the analysis of several algorithms and optimization techniques that are commonly used in commercial implementations of elementary arithmetic operations. As a basis for the formal verification of such implementations, Part IV contains high-level specifications of correctness of the basic arithmetic instructions of several major industry-standard floating-point architectures, including all details pertaining to the handling of exceptional conditions. Part V illustrates the methodology, applying the preceding theory to the comprehensive verification of a state-of-the-art commercial floating-point unit. All of these results have been formalized in the logic of the ACL2 theorem prover and mechanically checked to ensure their correctness. They are presented here, however, in simple conventional mathematical notation. The book presupposes no familiarity with ACL2, logic design, or any mathematics beyond basic high school algebra. It will be of interest to verification engineers as well as arithmetic circuit designers who appreciate the value of a rigorous approach to their art, and is suitable as a graduate text in computer arithmetic.
Design Verification with E
A Formal Hardware Verification System User's Guide
A Mathematical Approach
The e Hardware Verification Language
Co-verification of Hardware and Software for ARM SoC Design

Formal Hardware Verification