Full text available at: http://dx.doi.org/10.1561/1000000013

Applied Assertion-Based Verification: An Industry Perspective

Full text available at: http://dx.doi.org/10.1561/1000000013

Applied Assertion-Based Verification: An Industry Perspective

Harry Foster

Mentor Graphics Corporation
Plano, Texas
USA
Harry_Foster@mentor.com

now
the essence of knowledge
Boston – Delft

Full text available at: http://dx.doi.org/10.1561/1000000013

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is H. Foster, Applied Assertion-Based Verification: An Industry Perspective, Foundations and Trends® in Electronic Design Automation, vol 3, no 1, pp 1–95, 2008

ISBN: 978-1-60198-218-6
© 2009 H. Foster

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1-781-871-0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Full text available at: http://dx.doi.org/10.1561/1000000013

Foundations and Trends® in Electronic Design Automation
Volume 3 Issue 1, 2008

Editorial Board

Editor-in-Chief:
Sharad Malik
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

Editors
Robert K. Brayton (UC Berkeley)
Raul Camposano (Synopsys)
K.T. Tim Cheng (UC Santa Barbara)
Jason Cong (UCLA)
Masahiro Fujita (University of Tokyo)
Georges Gielen (KU Leuven)
Tom Henzinger (EPFL)
Andrew Kahng (UC San Diego)
Andreas Kuehlmann (Cadence Berkeley Labs)
Ralph Otten (TU Eindhoven)
Joel Phillips (Cadence Berkeley Labs)
Jonathan Rose (University of Toronto)
Rob Rutenbar (CMU)
Alberto Sangiovanni-Vincentelli (UC Berkeley)
Leon Stok (IBM Research)

Full text available at: http://dx.doi.org/10.1561/1000000013

Editorial Scope

Foundations and Trends® in Electronic Design Automation will publish survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design

Information for Librarians
Foundations and Trends® in Electronic Design Automation, 2008, Volume 3, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Full text available at: http://dx.doi.org/10.1561/1000000013

Foundations and Trends® in
Electronic Design Automation
Vol. 3, No. 1 (2008) 1–95
© 2009 H. Foster
DOI: 10.1561/1000000013

now
the essence of knowledge

Applied Assertion-Based Verification: An Industry Perspective

Harry Foster

Mentor Graphics Corporation, Plano, Texas, USA
Harry_Foster@mentor.com

Abstract

A wealth of material has been published over the past 30 years specifically related to the theory and technical aspects of property languages and assertion-based techniques. However, as any field of study matures, it becomes necessary to determine if the theories, algorithms, and concepts have grown beyond the bounds of research to become an integral solution to a problem in industry. To understand any solution, it is necessary to understand the problem. For example, debugging, on average, has grown to consume more than 60% of today's ASIC and SoC verification effort. Clearly, this is a topic the industry must address, and some organizations have done just that. Those that have adopted an assertion-based verification (ABV) methodology have seen a significant reduction in simulation debugging time (as much as 50% [1, 47]) due to improved observability. Furthermore, organizations that have embraced an ABV methodology are able to take advantage of more advanced verification techniques, such as formal property checking, thus improving their overall verification quality and results. This paper examines the application of ABV in today's electronic design industry to address specific challenges of poor observability and controllability during the

Full text available at: http://dx.doi.org/10.1561/1000000013

verification process. Statistics illustrating successful application of both low-level and high-level assertions are presented. While the process of writing assertions is fairly well understood by those skilled in the art — the process of creating higher-level assertion-based IP that must communicate with other components in a contemporary transaction-level modeling (TLM) simulation environment, is not. Hence, this paper provides a set of steps (in a tutorial fashion) for creating assertion-based IP.

Full text available at: http://dx.doi.org/10.1561/1000000013

Contents

Full text available at: http://dx.doi.org/10.1561/1000000013

Full text available at: http://dx.doi.org/10.1561/1000000013

1

Introduction

Ensuring functional correctness on RTL designs continues to pose one of the greatest challenges for today's ASIC and SoC design teams. Very few project managers would disagree with this statement. In fact, an often cited 2004 industry study by Collett International Research revealed that 35% of the total ASIC development effort was spent in verification [10]. In 2008, a Far West Research study (in conjunction with Mentor Graphics) indicated the verification effort has risen to 46% of the total ASIC development effort [20]. Furthermore, these industry studies reveal that debugging is the fastest growing component of verification, and that it consumes 60% of the total verification effort. Unfortunately, with the increase in verification effort, the industry has not experienced a measurable increase in quality of results. For example, a Collett International Research study that focused on design closure indicated that only 29% of projects developing ASICs were able to achieve first silicon success. To make matters worse, the industry is witnessing increasing pressure to shorten the overall ASIC and SoC development cycle. Clearly, new design and verification techniques, combined with a focus on maturing functional verification process

Full text available at: http://dx.doi.org/10.1561/1000000013

capabilities within an organization (and the industry as a whole) are required.

Assertion-based verification (ABV), although certainly not an end-all to the verification challenge, does directly address today's debugging problem, while providing an integration path for more advanced forms of verification into the design flow (such as formal property checking). This paper provides a survey of today's ABV landscape, ranging from industry case studies to today's assertion language standardization efforts, to emerging challenges and research opportunities.

In addition, this paper directly addresses industry process issues of developing assertion-based IP by introducing a systematic set of planning and development steps. A detailed bus protocol example is provided, which draws together the various concepts introduced throughout the text while demonstrating an effective process for developing assertion and assertion-based verification IP.

1.1 What is an Assertion?

Alan Turing made the following observation over 50 years ago: "How can one check a large routine in the sense of making sure that it's right? In order that the man who checks may not have too difficult a task, the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole program easily flows [46]." In essence, this view is at the heart of ABV.

Informally, an assertion is a statement of design intent that can be used to specify design behavior. Assertions may specify internal implementation design behaviors (such as a specific FIFO structure) or external specification design behavior (such as a bus protocol or even higher-level, end-to-end behavior that spans multiple design blocks). One key characteristic of assertions is that they allow the user to specify *what* the design is supposed to do at a high level of abstraction, without having to describe the details of *how* the design intent is to be implemented. Thus, this abstract view of the design intent is ideal for the verification process — whether we are specifying high-level

Full text available at: http://dx.doi.org/10.1561/1000000013

requirements or lower-level implementation behaviors — by means of assertions.

Properties versus assertions: The recent flurry of interest in assertion-based techniques has prompted considerable published research and industry articles on the subject. Often, the authors of these publications interchange the terms *property* and *assertion*, which leads to confusion. For our discussion, a property is informally defined as follows:

Property — A statement of design intent.

For instance, the statement, *grant0 and grant1 are mutually exclusive* is an example of a property, which is actually a partial specification for an arbiter. Notice that we have not stated how we intend to use this property during the verification process. For example, we might choose to use this property as a constraint specification, which is a requirement on the environment, and *assume* the property during verification. In this case, we want to eliminate traces from the verification tool that violate our constraint. Or, we might choose to use this property as a coverage specification, and *cover* the property such that the verification tool notifies us at the particular point in which the coverage specification holds on a trace. Or, we might choose to use the property as a specification of design intent and *assert* that the property holds on all traces produced by a verification tool.

For our discussion, an assertion is informally defined as follows:

Assertion — An implementation of a property that is evaluated or executed by a tool to validate design intent.

Assertions are used as targets during the verification process (for example, a checker for simulation or a proof obligation for formal) to help us identify and isolate unexpected behavior.

1.2 Controllability and Observability

Fundamental to the discussion of ABV is understanding the concepts of *controllability* and *observability* [13, 19]. Informally, controllability refers to the ability to influence or activate an embedded finite state machine, structure, or specific line of code within the design by stimulating various input ports. Note that, while in theory a simulation

Full text available at: http://dx.doi.org/10.1561/1000000013

testbench has high controllability of the design model's input ports during verification, it can have very low controllability of an internal structure within the model. Observability, in contrast, refers to the ability to observe the effects of a specific internal finite state machine, structure, or stimulated line of code. Thus, a testbench generally has limited observability if it only observes the external ports of the design model (because the internal signals and structures are often indirectly hidden from the testbench).

To identify a design error using a simulation testbench approach, the following conditions must hold:

1. The testbench must generate proper input stimulus to activate a design error.
2. The testbench must generate proper input stimulus to propagate all effects resulting from the design error to an output port.

It is possible, however, to set up a condition where the input stimulus activates a design error that does not propagate to an observable output port. In these cases, the first condition cited above applies; however, the second condition is absent, as illustrated in Figure 1.1.

Embedding assertions in the design model increases observability. In this way, the verification environment no longer depends on generating input stimulus to propagate a design error to an observable port. Thus,

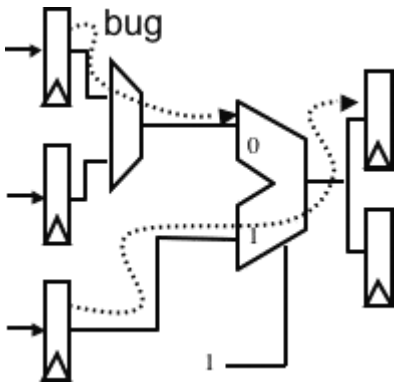

Fig. 1.1 Poor observability and controllability misses bugs.

Full text available at: http://dx.doi.org/10.1561/1000000013

any improper or unexpected behavior can be caught closer to the source of the design error, in terms of both time and location. Thus resulting in an overall reduction of debugging time.

While embedded assertions help solve the observability challenge in simulation, they do not help with the controllability challenge. However, the existence of assertions within the flow does open up the possibility for utilizing formal property checking to target critical or high-value assertions, thus addressing the controllability challenge.

1.3 Assertion Stakeholders

Assertions added at any level of hierarchy (or abstractions) clearly benefit verification by reducing debugging time while clarifying design intent. Certainly multiple stakeholders within the design and verification process can contribute to the assertion development process — thus reducing ambiguities while improving observability.

Figure 1.2 illustrates a typical design refinement process through various levels of abstraction and the stakeholders associated with each level. Adoption of assertions in the industry, at the time of this writing, has predominately occurred within the block and module level. They can be called implementation assertions. This adoption trend is partially due to the lack of effective guidelines for assertion use at higher levels of design hierarchy (or abstraction) and confusion about which stakeholders should contribute to the assertion development process.

Although an architect can contribute to the assertion development effort by defining global properties (derived from the architecture and micro-architectural specification) that must hold across multiple possible implementations, the design engineer contributes by writing internal white-box assertions derived from the implementation. In addition, the verification engineer contributes by developing assertions that specify correct interface behavior between units and between blocks. The verification engineer also contributes by developing black-box, end-to-end assertions across design components.

Although it is easy to identify the various stakeholders and roles they play in the assertion development process, these stakeholders

Full text available at: http://dx.doi.org/10.1561/1000000013

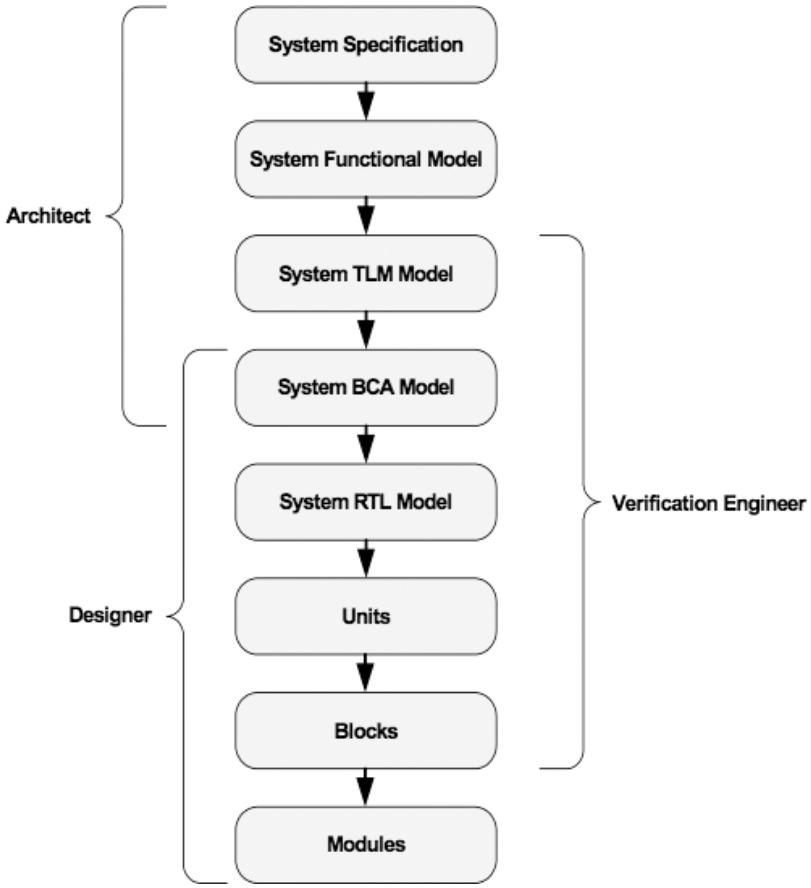

Fig. 1.2 Stakeholders and levels of abstraction.

often lack a process for systematically developing their assertions. In fact, a classic mistake many engineers make when first adopting assertion-based techniques is to jump into coding assertions too soon — without fully understanding the behavior they are trying to specify. This *ad hoc* approach leads to incomplete (or incorrect) property sets and non-reusable assertion-based IP. This problem can be addressed by introducing a systematic process in which a natural language list of properties is created prior to coding the assertion-based IP, as discussed later in this paper.

Full text available at: http://dx.doi.org/10.1561/1000000013

1.4 Assertions Within a Flow

When applying assertion-based techniques, it is important to first understand where they can be effectively applied. For example, in addition to the verification flow stakeholders, Figure 1.2 also illustrates a typical design refinement process.

The flow begins with a system specification, which is typically a natural language properties document. There has been recent interest in executable forms of system specification, such as UML [39]. The first refinement of the system specification is often a system function model to explore the proposed algorithm, which is often written in C or C++. At this point, hardware–software partitioning and architectural mapping decisions have not been made.

The system transaction-level model (TLM) is generally an untimed (or partially timed) model that is created after architectural mapping. This model is often used for firmware development, system and architectural performance analysis, and software development. Furthermore, the TLM is often further refined into a bus cycle-accurate (BCA) model as architectural decisions begin to gel.

RTL refinement occurs next. During this phase, the system is partitioned into multiple units. Each unit is partitioned into blocks. Each block is partitioned into modules consisting of RTL code.

Certainly, a natural language list of properties can (and should) be developed at all levels of abstraction. However, today's assertion language standards lack the proper formalism necessary to express properties at all of the levels of abstraction illustrated in Figure 1.2. For example, in an untimed TLM, one concurrent transaction might overlap with a different transaction. That is, it can begin and end in the middle execution of a different untimed transaction.

Existing assertion language standards lack the semantics (and syntax) necessary to express assertions related to this type of transaction behavior. Researchers have proposed solutions to the TLM assertion specification problem, which might result in semantic and syntactical extensions to existing standards [14, 15, 16]. Today, successful assertion specification in an industry setting occurs at and below the BCA abstraction level, illustrated in Figure 1.2. Later in this paper,

Full text available at: http://dx.doi.org/10.1561/1000000013

techniques are presented for developing assertion-based IP that can be reused in a transaction-level testbench using existing standards by introducing what is essentially an abstraction converter between a timed RTL model and untimed transaction environment.

1.5 ABV Discussion Preview

The remainder of this paper is organized as follows: In Section 2, industry motivations for adopting ABV are presented. Included are a number of successful case studies. Section 3 is focused on the recent emergence of assertion language and library standards. As a foundation for the assertion language discussion, a basic introduction to temporal logic and extended regular expressions is presented, followed by a brief introduction to IEEE Property Specification Language (PSL), SystemVerilog Assertions (SVA), and the Open Verification Library (OVL). Section 4 provides a tutorial for developing assertion-based IP with a focus on contemporary simulation environments found in industry. Section 5 integrates the concepts presented in all the previous sections into a demonstration on how to create assertion-based IP for a simple nonpipelined bus protocol example. Finally, Section 6 provides a summary discussion.

Full text available at: http://dx.doi.org/10.1561/1000000013

References

[1] Y. Abarbanel, I. Beer, L. Gluhovsky, S. Keidar, and Y. Wolfsthal, "FoCs — Automatic generation of simulation checkers from formal specifications," in *Proceedings of 12th International Conference Computer Aided Verification*, pp. 414–427, 2000.

[2] Accellera Standard OVL Library Reference Manual, www.accellera.org, 2008.

[3] ARM, AMBA specification version 2.0, 1999.

[4] R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M. Y. Vardi, and Y. Zbar, "The Forspec temporal logic: A new temporal property-specification language," in *TACAS'2002, LNCS,* vol. 2280, pp. 296–311, 2002.

[5] I. Beer, S. Ben-David, and A. Landver, "On-the-fly model checking of RCTL formulas," in *Computer Aided Verification, Proceedings of 10th International Conference, Lecture Notes in Computer Science,* vol. 1427, pp. 184–194, Springer-Verlag, 1998.

[6] J. Bergeron, E. Cerny, A. Hunter, and A. Nightingale, *Verification Methodology Manual for SystemVerilog.* Springer, 2006.

[7] D. Bustan and J. Havlicek, "Some complexity results for SystemVerilog assertions," in *CAV 2006 Conference Proceedings*, pp. 205–218, CAV, July 2006.

[8] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Proceedings of Workshop on Logic of Programs, Lecture Notes in Computer Science,* vol. 131, pp. 52–71, Springer-Verlag, 1981.

[9] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking.* The MIT Press, 2000.

Full text available at: http://dx.doi.org/10.1561/1000000013

[10] R. Collett, "2004 IC/ASIC functional verification study," Industry Report from *Collett International Research*, p. 34, 2004.
[11] P. Dasgupta, *A Roadmap for Formal Property Verification*. Springer, 2006.
[12] S. Dellacherie, H. Foster, E. Marschner, S. Ruah, and S. Smith, "Tutorial: Assertion-Based Verification," *40th Design Automation Conference*, 2003.
[13] S. Devadas, A. Ghosh, and K. Keutzer, "An observability-based code coverage metric for functional simulation," in *Proceedings of the 33rd Design Automation Conference*, pp. 418–425, 1996.
[14] W. Ecker, V. Esen, and M. Hull, "Execution semantics and formalisms for multi-abstraction TLM assertions," *Proceedings of MEMOCODE*, 2006.
[15] W. Ecker, V. Esen, M. Hull, T. Steininger, and M. Velten, "Requirements and concepts for transaction level assertions," *ICCD*, 2006.
[16] W. Ecker, V. Esen, T. Steininger, M. Velten, and M. Hull, "Interactive presentation: Implementation of a transaction level assertion framework in SystemC," *DATE 2007*, pp. 894–899, 2007.
[17] C. Eisner and D. Fisman, *A Practical Introduction to PSL*. Springer, 2006.
[18] E. A. Emerson and J. Y. Halpern, "'Sometimes' and 'not never' revisited: on branching versus linear time temporal logic," *Journal of the ACM*, vol. 33, no. 1, pp. 151–178, 1986.
[19] F. Fallah, S. Devadas, and K. Keutzer, "OCCOM: Efficient computation of observability-based code coverage metrics for functional simulation," in *Proceedings of the 35th Design Automation Conference*, pp. 152–157, 1998.
[20] FarWest Research 2008 industry study in conjunction with Mentor Graphics, 2008.
[21] H. Foster and C. Coelho, "Assertions targeting a diverse set of verification tools," in *Proceedings of International HDL Conference*, 2001.
[22] H. Foster, A. Krolnik, and D. Lacey, *Assertion-Based Design*. Kluwer Academic Publishers, Second ed., 2004.
[23] H. Foster, K. Larsen, and M. Turpin, "Introducing the new accellera open verification library standard," in *Proceedings of DVCon*, 2006.
[24] H. Foster, L. Loh, B. Rabii, and V. Singhal, "Guidelines for creating a formal verification testplan," in *Proceedings of DVCon*, 2006.
[25] M. Glasser, A. Rose, T. Fitzpatrick, D. Rich, and H. Foster, "The verification cookbook," http://www.mentor.com/go/cookbook, 2007.
[26] M. J. C. Gordon, "Validating the PSL/Sugar semantics using automated reasoning," in *Formal Aspects of Computing*, vol. 15, pp. 406–421, Springer-Verlag, London, December 2003.
[27] F. Haque, J. Michelson, and K. Khan, *The Art of Verification with SystemVerilog Assertions, Verification Central*. First ed., 2006.
[28] IEEE Standard 1800-2005 SystemVerilog: Unified Hardware Design, Specification and Verification Language, IEEE, Inc., New York, NY, USA, 2005.
[29] IEEE Standard 1850-2005 Property Specification Language (PSL), IEEE, Inc., New York, NY, USA, 2005.
[30] International Technology Roadmap for Semiconductors, 2003 Report, www.itrs.net/reports.html.

Full text available at: http://dx.doi.org/10.1561/1000000013

[31] H. Iwashita and T. Nakata, "Forward model checking techniques oriented to buggy designs," *International Conference on Computer Aided Design, ICCAD*, 1997.
[32] M. Kantrowitz and L. Noack, "I'm done simulating; Now what? Verification coverage analysis and correctness checking of the DECchip 21164 alpha microprocessor," *Proceedings of Design Automation Conference*, pp. 325–330, 1996.
[33] A. Krolnik, *Cyrix M3 Phase 1 Report*. Cyrix Inc. internal report, 1998.
[34] A. Krolnik, *Cyrix M3 Phase 2 Report*. Cyrix Inc. internal report, 1999.
[35] T. Kropf, *Introduction to Formal Hardware Verification*. Springer, 1998.
[36] J. Long and A. Seawright, "Synthesizing SVA local variables for formal verification," in *Proceedings of the 44th Design Automation Conference, DAC 2007*, pp. 75–80, 2007.
[37] J. Long, A. Seawright, and H. Foster, "SVA local variable coding guidelines for efficient use," in *Proceedings of DVCon*, 2007.
[38] E. Marschner and H. Foster, "Assertion-based verification," in *EDA for IC System Design, Verification, and Testing (Electronic Design Automation for Integrated Circuits Handbook)*, (L. Scheffer, L. Lavagno, and G. Martin, eds.), CRC Press, 2006.
[39] G. Martin, "UML for embedded systems specification and design: motivation and overview," *Proceedings of Design, Automation and Test in Europe*, 2002.
[40] Open Verification Methodology (OVM), http://www.ovmworld.org/.
[41] OSCI TLM-1.0 Transaction Level Modeling Standard, SystemC kit with whitepaper available on http://www.systemc.org/.
[42] A. Piziali, *Functional Verification Coverage Measurement and Analysis*. Kluwer Academic Publishers, 2004.
[43] A. Pnueli, "The temporal logic of programs," in *Proceedings of 18th IEEE Symposium on Foundation of Computer Science*, pp. 46–57, 1977.
[44] V. Singhal, Coverage: The link between SIMULATION and FORMAL, 2006 discussion by Oski Technology, http://oskitech.com/papers/coverage-0506.pdf.
[45] S. Taylor, M. Quinn, D. Brown, N. Dohm, S. Hildebrandt, J. Huggins, and C. Ramey, "Functional verification of a multiple-issue out-of-order, superscalar alpha processor — the DEC Alpha 21264 microprocessort," *Proceedings of Design Automation Conference*, pp. 638–643, 1998.
[46] A. Turing, in *Report of a Conference on High Speed Automatic Calculating Machines*, pp. 67–69, University of Mathematical Laboratory, Cambridge, 1949.
[47] B. Turumella and M. Sharma, "Assertion-based verification of a 32 thread SPARC™ CMT microprocessor," in *Proceedings of the 45th Design Automation Conference, DAC 2008*, pp. 256–261, 2008.
[48] M. Y. Vardi, "Branching vs linear time: Final showdown," in *Proceedings of 7th International Conference on Tolls and Algorithms for the Construction and Analysis of Systems (TACAS 2001), LNCS*, vol. 23, Springer, 2001.
[49] M. Y. Vardi and P. Wolper, "Reasoning about infinite computations," *Information and Computation*, vol. 115, no. 1, pp. 1–37, 1994.
[50] P. Wolper, "Temporal logic can be more expressive," *Information and Control*, vol. 56, no. 1/2, pp. 72–99, 1983.