

Declaration of Rachel J. Watters on Authentication of Publication

I, Rachel J. Watters, am a librarian, and the Head of Resource Sharing for the General Library System, Memorial Library, located at 728 State Street, Madison, Wisconsin, 53706. Part of my job responsibilities include oversight of Wisconsin TechSearch (“WTS”), an interlibrary loan department at the University of Wisconsin-Madison. I have worked as a librarian at the University of Wisconsin library system since 1998, starting as a graduate student employee in the Kurt F. Wendt Engineering Library and WTS, then as a librarian in Interlibrary Loan at Memorial Library. I began professional employment at WTS in 2002 and became WTS Director in 2011. In 2019, I became of Head of Resource Sharing for UW-Madison’s General Library System. I have a master’s degree in Library and Information Studies from the University of Wisconsin-Madison. Through the course of my studies and employment, I have become well informed about the operations of the University of Wisconsin library system, which follows standard library practices.

This Declaration relates to the dates of receipt and availability of the following:

Abarbanel, Y., Beer, I., Gluhovsky, L., Keidar, S., and Wolfsthal, Y. (2000). FoCs – automatic generation of simulation checkers from formal specifications. In E.A. Emerson and A.P. Sistla (Eds.), *Computer Aided Verification: 12th International Conference, CAV 2000 Chicago, IL, USA, July 15-19, 2000 Proceedings* (Lecture Notes in Computer Science, Vol. 1855), p. 538-542. Springer-Verlag.

Standard operating procedures for materials at the University of Wisconsin-Madison Libraries. When a volume was received by the Library, it would be checked

Declaration of Rachel J. Watters on Authentication of Publication

in, added to library holdings records, and made available to readers as soon after its arrival as possible. The procedure normally took a few days or at most 2 to 3 weeks.

Exhibit A to this Declaration is a true and accurate copy of the front matter of the *Computer Aided Verification: 12th International Conference, CAV 2000 Chicago, IL, USA, July 15-19, 2000 Proceedings* (2000) publication, which includes a stamp on the verso page showing that this book is the property of the Kurt Wendt Library at the University of Wisconsin-Madison. Exhibit A also includes an excerpt of pages 538 to 542 of that volume, showing the article entitled *FoCs – automatic generation of simulation checkers from formal specifications* (2000).

Attached as Exhibit B is the cataloging system record of the University of Wisconsin-Madison Libraries for its copy of the *Computer Aided Verification: 12th International Conference, CAV 2000 Chicago, IL, USA, July 15-19, 2000 Proceedings* (2000) publication. As shown in the “Receiving date” field of this Exhibit, the University of Wisconsin-Madison Libraries owned this volume and had it cataloged in the system as of October 4, 2000.

Members of the interested public could locate the *Computer Aided Verification: 12th International Conference, CAV 2000 Chicago, IL, USA, July 15-19, 2000 Proceedings* (2000) publication after it was cataloged by searching the public library catalog or requesting a search through WTS. The search could be done by title, author,

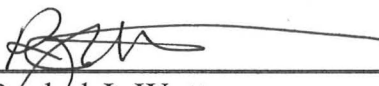
Declaration of Rachel J. Watters on Authentication of Publication

and/or subject key words. Members of the interested public could access the publication by locating it on the library's shelves or requesting it from WTS.

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: December 11, 2023

Memorial Library
728 State Street
Madison, Wisconsin 53706



Rachel J. Watters
Head of Resource Sharing

EXHIBIT A

Lecture Notes in Computer Science

1855

E. Allen Emerson A. Prasad Sistla (Eds.)

Computer Aided Verification

12th International Conference, CAV 2000
Chicago, IL, USA, July 2000
Proceedings



Springer

Cadence Design Systems, Inc. - Ex. 1037, Page 00005

Lecture Notes in Computer Science

This series reports new developments in computer science, quickly, informally, and at a high level. The type of material considered for publication is

- drafts of original papers or monographs,
- technical reports of high quality and broad interest,
- advanced-level lectures,
- reports of meetings, provided they are of exceptional interest and focused on a single topic.

Publication of Lecture Notes is intended as a service to the computer science community in that the publisher Springer-Verlag offers global distribution of documents which would otherwise have a restricted readership. Once published and copyrighted they can be cited in the scientific literature.

Manuscripts

Lecture Notes are printed by photo-offset from the master copy delivered in camera-ready form. Manuscripts should consist of no fewer than 100 and preferably no more than 500 pages of text. Authors of monographs and editors of proceedings volumes receive 50 free copies of their book. Manuscripts should be printed with a laser or other high-resolution printer onto white paper of reasonable quality. To ensure that the final photo-reduced pages are easily readable, please use one of the following formats:

Font size (points)	Printing area		Final size %
	(cm)	(inches)	
10	12.2 x 19.3	4.8 x 7.6	100
12	15.3 x 24.2	6.0 x 9.5	80

On request the publisher will supply a leaflet with more detailed technical instructions or a T_EX macro package for the preparation of manuscripts.

Manuscripts should be sent to one of the series editors or directly to:

Springer-Verlag, Computer Science Editorial III, Tiergartenstr.17,
D-69121 Heidelberg, Germany

ISBN 3-540-67770-4



ISSN 0302-9743

<http://www.springer.de>

89064908908



B89064908908A

LNCS 1855

Emerson · Sistla (Eds.)

CAV 2000

Computer Aided Verification

S

QA
76.76
E93
C38
2000

Lecture Notes in Computer Science

1855

E. Allen Emerson A. Prasad Sistla (Eds.)

Computer Aided Verification

12th International Conference, CAV 2000
Chicago, IL, USA, July 2000
Proceedings



Springer

Cadence Design Systems, Inc. - Ex. 1037, Page 00006

E. Allen Emerson A. Prasad Sistla (Eds.)

Computer Aided Verification

12th International Conference, CAV 2000
Chicago, IL, USA, July 15-19, 2000
Proceedings

Madison, WI 53706-1350
University Avenue
100 University Hall
University of Wisconsin - Madison
Physics Library



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

E. Allen Emerson
University of Texas, Computer Sciences Department
Taylor Hall 2.124, Austin, TX 78712, USA
E-mail: emerson@cs.utexas.edu

A. Prasad Sistla
University of Illinois at Chicago
Electrical Engineering and Computer Science Department
851 South Morgan Street, Chicago, IL 60607, USA
E-mail: sistla@surya.eecs.uic.edu

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Computer aided verification : 12th international conference ;
proceedings / CAV 2000, Chicago, IL, USA, July 15 - 19, 2000. E. Allen
Emerson ; A. Prasad Sistla (ed.). - Berlin ; Heidelberg ; New York ;
Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo :
Springer, 2000
(Lecture notes in computer science)
ISBN 3-540-67770-4

Kurt K. Wendt Library
University of Wisconsin-Madison
215 N. Randall Avenue
Madison, WI 53706-1629

Classification (1998): F.3, D.2.4, D.2.2, F.4.1, B.7.2, C.3, I.2.3

ISSN 0302-9743

ISBN 3-540-67770-4 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a company in the BertelsmannSpringer publishing group.
© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN: 10722167 06/3142 5 4 3 2 1 0

QA
76.76
E93
C38
2000

Preface

This volume contains the proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000) held in Chicago, Illinois, USA during 15-19 July 2000.

The CAV conferences are devoted to the advancement of the theory and practice of formal methods for hardware and software verification. The conference covers the spectrum from theoretical foundations to concrete applications, with an emphasis on verification algorithms, methods, and tools together with techniques for their implementation. The conference has traditionally drawn contributions from both researchers and practitioners in academia and industry. This year 91 regular research papers were submitted out of which 35 were accepted, while 14 brief tool papers were submitted, out of which 9 were accepted for presentation. CAV included two invited talks and a panel discussion. CAV also included a tutorial day with two invited tutorials.

Many industrial companies have shown a serious interest in CAV, ranging from using the presented technologies in their business to developing and marketing their own formal verification tools. We are very proud of the support we receive from industry. CAV 2000 was sponsored by a number of generous and forward-looking companies and organizations including: Cadence Design Systems, IBM Research, Intel, Lucent Technologies, Mentor Graphics, the Minerva Center for Verification of Reactive Systems, Siemens, and Synopsys.

The CAV conference was founded by its Steering Committee: Edmund Clarke (CMU), Bob Kurshan (Bell Labs), Amir Pnueli (Weizmann), and Joseph Sifakis (Verimag).

The conference program for this year's CAV 2000 was selected by the program committee: Parosh Abdulla (Uppsala), Rajeev Alur (U. Penn and Bell Labs), Henrik Reif Andersen (ITU Copenhagen), Ed Brinksma (Twente), Randy Bryant (CMU), Werner Damm (Oldenburg), David Dill (Stanford), E. Allen Emerson, co-chair (U. Texas-Austin), Steven German (IBM), Rob Gerth (Intel), Patrice Godefroid (Bell Labs), Ganesh Gopalakrishnan (U. Utah), Mike Gordon (Cambridge), Nicolas Halbwachs (Verimag), Warren Hunt (IBM), Bengt Jonsson (Uppsala), Kim Larsen (Aalborg), Ken McMillan (Cadence), John Mitchell (Stanford), Doron Peled (Bell Labs), Carl Pixley (Motorola), Amir Pnueli (Weizmann), Bill Roscoe (Oxford), Joseph Sifakis (Verimag), A. Prasad Sistla, co-chair (U. Illinois-Chicago), Fabio Somenzi (U. Colorado), and Pierre Wolper (Liege).

We are grateful to the following additional reviewers who aided the reviewing process: Will Adams, Nina Amla, Flemming Andersen, Tamarah Arons, Eugene Asarin, Mohammad Awedh, Adnan Aziz, Clark Barrett, Gerd Behrmann, Wendy Belluomini, Michael Benedikt, Saddek Bensalem, Ritwik Bhattacharya, Tom Bienmueller, Per Bjesse, Roderick Bloem, Juergen Bohn, Bernard Boigelot, Ahmed Bouajjani, Olivier Bournez, Marius Bozga, P. Broadfoot, Udo Brock-

meyer, Glenn Bruns, Annette Bunker, Paul Caspi, Prosenjit Chatterjee, Hubert Common, Jordi Cortadella, Sadie Creese, David Cyrluk, Pedro D'Argenio, Satyaki Das, Luca de Alfaro, Willem-P. de Roever, Juergen Dingel, Dan DuVarney, Joost Engelfriet, Kousha Etessami, David Fink, Dana Fisman, Martin Fraenzle, Laurent Fribourg, Malay Ganai, Vijay Garg, Jens Chr. Godskesen, Jeff Golden, M. H. Goldsmith, Guarishankar Govindaraju, Susanne Graf, Radu Grosu, Aarti Gupta, Dilian Gurov, John Havlicek, Nevin Heinze, Holger Hermanns, Thomas Hildebrandt, Pei-Hsin Ho, Holger Hermanns, Ravi Hosabettu, Jae-Young Jang, Henrik Hulgaard, Thomas Hune, Hardi Hungar, Anna Ingolfsdottir, Norris Ip, Purushothaman Iyer, Hans Jacobson, Damir Jamsek, Jae-Young Jang, Henrik Ejersbo Jensen, Somesh Jha, Michael Jones, Bernhard Josko, Vineet Kahlon, Joost-Pieter Katoen, Yonit Kesten, Nils Klarlund, Josva Kleist, Kare Jelling Kristoffersen, Andreas Kuehlmann, Robert P. Kurshan, Yassine Lakhnech, Rom Langerak, Salvatore La Torre, Ranko Lazic, Jakob Lichtenberg, Orna Lichtenstein, Jorn Lind-Nielsen, Hans Henrik Løvengreen, Enrico Macii, Angelika Mader, Oded Maler, Pete Manolios, Monica Marcus, Abdelillah Mokkedem, Faron Moller, Jesper Moller, Oliver Möller, In-Ho Moon, Laurent Mounier, Chris Myers, Luay Nakhleh, Kedar Namjoshi, Tom Newcomb, Flemming Nielson, Kasper Overgård Nielsen, Marcus Nilsson, Thomas Noll, David Nowak, Aletta Nylen, Manish Pandey, George Pappas, Atanas Parashkevov, Abelardo Pardo, Catherine Parent-Vigouroux, David Park, Justin Pearson, Paul Pettersson, Nir Piterman, Carlos Puchol, Shaz Qadeer, Stefano Quer, Theis Rauhe, Antoine Rauzy, Kavita Ravi, Judi Romijn, Sitvanit Ruah, Theo Ruys, Jun Sawada, Alper Sen, Peter Sestoft, Ali Sezgin, Elad Shahr, Ofer Shtrichman, Arne Skou, Uli Stern, Kanna Shimizu, Scott D. Stoller, Ian Sutherland, Richard Trefer, Jan Tretmans, Stavros Tripakis, Annti Valmari, Helmut Vieth, Sergei Vorobyov, Bow-Yaw Wang, Farn Wang, Poul F. Williams, Chris Wilson, Hanno Wupper, Jason Yang, Wang Yi, Tsay Yih-Kuen, Sergio Yovine, and Jun Yuan.

Finally, we would like to give our special thanks to John Havlicek for his enormous assistance overall including maintaining the CAV web site, the cav2k account, and in preparing the proceedings. We appreciate the assistance of the UTCS computer support staff, especially John Chambers. We are also most grateful, to Richard Gerber for kindly lending us his "START" conference management software as well as his prompt assistance when a file server error masqueraded as a web server error.

June 2000

E. Allen Emerson and A. Prasad Sistla

Table of Contents

Invited Talks and Tutorials

Keynote Address: Abstraction, Composition, Symmetry, and a Little Deduction: The Remedies to State Explosion	1
<i>A. Pnueli</i>	
Invited Address: Applying Formal Methods to Cryptographic Protocol Analysis	2
<i>C. Meadows</i>	
Invited Tutorial: Boolean Satisfiability Algorithms and Applications in Electronic Design Automation	3
<i>J. Marques-Silva, K. Sakallah</i>	
Invited Tutorial: Verification of Infinite-State and Parameterized Systems .	4
<i>P.A. Abdulla, B. Jonsson</i>	

Regular Papers

An Abstraction Algorithm for the Verification of Generalized C-Slow Designs	5
<i>J. Baumgartner, A. Tripp, A. Aziz, V. Singhal, F. Andersen</i>	
Achieving Scalability in Parallel Reachability Analysis of Very Large Circuits	20
<i>T. Heyman, D. Geist, O. Grumberg, A. Schuster</i>	
An Automata-Theoretic Approach to Reasoning about Infinite-State Systems	36
<i>O. Kupferman, M.Y. Vardi</i>	
Automatic Verification of Parameterized Cache Coherence Protocols.....	53
<i>G. Delzanno</i>	
Binary Reachability Analysis of Discrete Pushdown Timed Automata	69
<i>Z. Dang, O.H. Ibarra, T. Bultan, R.A. Kemmerer, J. Su</i>	
Boolean Satisfiability with Transitivity Constraints	85
<i>R.E. Bryant, M.N. Velez</i>	
Bounded Model Construction for Monadic Second-Order Logics	99
<i>A. Ayari, D. Basin</i>	
Building Circuits from Relations	113
<i>J.H. Kukula, T.R. Shiple</i>	

Combining Decision Diagrams and SAT Procedures for Efficient Symbolic Model Checking	124
<i>P.F. Williams, A. Biere, E.M. Clarke, A. Gupta</i>	
On the Completeness of Compositional Reasoning	139
<i>K.S. Namjoshi, R.J. Trefler</i>	
Counterexample-Guided Abstraction Refinement	154
<i>E. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith</i>	
Decision Procedures for Inductive Boolean Functions Based on Alternating Automata	170
<i>A. Ayari, D. Basin, F. Klaedtke</i>	
Detecting Errors Before Reaching Them	186
<i>L. de Alfaro, T.A. Henzinger, F.Y.C. Mang</i>	
A Discrete Strategy Improvement Algorithm for Solving Parity Games	202
<i>J. Vöge, M. Jurdziński</i>	
Distributing Timed Model Checking – How the Search Order Matters	216
<i>G. Behrmann, T. Hune, F. Vaandrager</i>	
Efficient Algorithms for Model Checking Pushdown Systems	232
<i>J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon</i>	
Efficient Büchi Automata from LTL Formulae	248
<i>F. Somenzi, R. Bloem</i>	
Efficient Detection of Global Properties in Distributed Systems Using Partial-Order Methods	264
<i>S.D. Stoller, L. Unnikrishnan, Y.A. Liu</i>	
Efficient Reachability Analysis of Hierarchical Reactive Machines	280
<i>R. Alur, R. Grosu, M. McDougall</i>	
Formal Verification of VLIW Microprocessors with Speculative Execution .	296
<i>M.N. Velev</i>	
Induction in Compositional Model Checking	312
<i>K.L. McMillan, S. Qadeer, J.B. Saxe</i>	
Liveness and Acceleration in Parameterized Verification	328
<i>A. Pnueli, E. Shahar</i>	
Mechanical Verification of an Ideal Incremental ABR Conformance Algorithm	344
<i>M. Rusinowitch, S. Stratulat, F. Klay</i>	

Model Checking Continuous-Time Markov Chains by Transient Analysis ..	358
<i>C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen</i>	
Model-Checking for Hybrid Systems by Quotienting and Constraints Solving	373
<i>F. Cassez, F. Laroussinie</i>	
Prioritized Traversal: Efficient Reachability Analysis for Verification and Falsification	389
<i>R. Fraer, G. Kamhi, B. Ziv, M.Y. Vardi, L. Fix</i>	
Regular Model Checking	403
<i>A. Bouajjani, B. Jonsson, M. Nilsson, T. Touili</i>	
Symbolic Techniques for Parametric Reasoning about Counter and Clock Systems	419
<i>A. Annichini, E. Asarin, A. Bouajjani</i>	
Syntactic Program Transformations for Automatic Abstraction	435
<i>K.S. Namjoshi, R.P. Kurshan</i>	
Temporal-Logic Queries	450
<i>W. Chan</i>	
Are Timed Automata Updatable?	464
<i>P. Bouyer, C. Dufourd, E. Fleury, A. Petit</i>	
Tuning SAT Checkers for Bounded Model Checking	480
<i>O. Shtrichman</i>	
Unfoldings of Unbounded Petri Nets	495
<i>P.A. Abdulla, S.P. Iyer, A. Nylén</i>	
Verification Diagrams Revisited: Disjunctive Invariants for Easy Verification	508
<i>J. Rushby</i>	
Verifying Advanced Microarchitectures that Support Speculation and Exceptions	521
<i>R. Hosabettu, G. Gopalakrishnan, M. Srivas</i>	
Tool Papers	
FoCs: Automatic Generation of Simulation Checkers from Formal Specifications	538
<i>Y. Abarbanel, I. Beer, L. Gluhovsky, S. Keidar, Y. Wolfsthal</i>	
IF: A Validation Environment for Timed Asynchronous Systems	543
<i>M. Bozga, J.-C. Fernandez, L. Ghirvu, S. Graf, J.-P. Krimm, L. Mounier</i>	

Integrating WS1S with PVS 548
 S. Owre, H. Rueß

PET: An Interactive Software Testing Tool 552
 E. Gunter, R. Kurshan, D. Peled

A Proof-Carrying Code Architecture for Java 557
 C. Colby, P. Lee, G.C. Necula

The STATEMATE Verification Environment – Making It Real 561
 T. Bienmüller, W. Damm, H. Wittke

TAPS: A First-Order Verifier for Cryptographic Protocols 568
 E. Cohen

VINAS-P: A Tool for Trace Theoretic Verification of Timed Asynchronous
Circuits 572
 T. Yoneda

XMC: A Logic-Programming-Based Verification Toolset 576
 *C.R. Ramakrishnan, I.V. Ramakrishnan, S.A. Smolka, Y. Dong, X. Du,
 A. Roychoudhury, V.N. Venkatakrishnan*

Author Index 581

This series reports new developments in computer science, teaching, quickly, informally, and at a high level. The script is more important than its form, which may vary. The type of material considered for publication is:

- drafts of original papers or monographs,
- technical reports of high quality and broad interest,
- advanced-level lectures,
- reports of meetings, provided they are of exceptional interest and focused on a single topic.

Publication of Lecture Notes is intended as a service to the computer science community in that the publisher Springer-Verlag offers global distribution of documents which would otherwise have a restricted readership. Once published and copyrighted they can be cited in the scientific literature.

Manuscripts

Lecture Notes are printed by photo-offset from the master copy delivered in camera-ready form. Manuscripts should consist of no fewer than 100 and preferably no more than 500 pages of text. Authors of monographs and editors of proceedings volumes receive 50 free copies of their book. Manuscripts should be printed with a laser or other high-resolution printer onto white paper of reasonable quality. To ensure that the final photo-reduced pages are easily readable, please use one of the following formats:

Font size (points)	Printing area		Final size %
	(cm)	(inches)	
10	12.2 x 19.3	4.8 x 7.6	100
12	15.3 x 24.2	6.0 x 9.5	80

On request the publisher will supply a leaflet with more detailed technical instructions or a $\text{\TeX}$ macro package for the preparation of manuscripts.

Manuscripts should be sent to one of the series editors or directly to:

Springer-Verlag, Computer Science Editorial III, Tiergartenstr.17,
D-69121 Heidelberg, Germany

ISBN 3-540-67770-4



9 783540 677703

ISSN 0302-9743

FoCs - Automatic Generation of Simulation Checkers from Formal Specifications

Yael Abarbanel, Ilan Beer, Leonid Gluhovsky,
Sharon Keidar, and Yaron Wolfsthal

IBM Haifa Research Laboratory, Israel
{yaell,beer,leonid,sharon,wolfstal}@il.ibm.com

1 Introduction and Motivation

For the foreseeable future, industrial hardware design will continue to use both simulation and model checking in the design verification process. To date, these techniques are applied in isolation using different tools and methodologies, and different formulations of the problem. This results in cumulative high cost and little (if any) cross-leverage of the individual advantages of simulation and formal verification.

With the goal of effectively and advantageously exploiting the co-existence of simulation and model checking, we have developed a tool called **FoCs** ("**Formal Checkers**"). FoCs, implemented as an independent component of the RuleBase toolset, takes RCTL¹ properties as input and translates them into VHDL programs ("checkers") which are integrated into the simulation environment and monitor simulation on a cycle-by-cycle basis for violations of the property.

Checkers, also called Functional Checkers, are not a new concept: manually-written checkers are a traditional part of simulation environments (cf. [GB+99]). Checkers facilitate massive random testing, because they automate test results analysis. Moreover, checkers facilitate the analysis of intermediate results, and therefore save debugging effort by identifying problems directly - "as they happen", and by pointing more accurately to the source of the problems.

However, the manual writing and maintenance of checkers is a notoriously high-cost and labor-intensive effort, especially if the properties to be verified are complex temporal ones. For instance, in the case of a checker for a design with overlapping transactions (explained in Section 3), writing a checker manually is an excruciating error-prone effort.

Observing the inefficient process of manual checker writing in ongoing IBM projects has inspired the development of FoCs, as a means for automatically generating checkers from formal specifications. For each property of the specification, represented as an RCTL formula, FoCs generates a checker for simulation. This checker essentially implements a state machine which will enter an error

¹ RCTL includes a rich and useful set of CTL safety formulas and regular expressions, see [BBL98]

state in a simulation run if the formula fails to hold in this run. The next section will describe the checker generation process in more detail.

Experience with FoCs in multiple projects has been very favorable in terms of verification cost and quality. Verification effort is reduced by leveraging the same formal rules for model checking of small design blocks as well as for simulation analysis across all higher simulation levels. An equally important benefit of FoCs is the conciseness and expressiveness of RCTL formulas. Formulas consisting of just a few lines can efficiently represent complex and subtle cases, which would require many lines of code if described in a language such as VHDL. This makes maintenance, debugging, porting and reuse of specifications and checkers highly cost-effective.

2 Tool Architecture and Implementation

Figure 1 shows the overall environment in which FoCs operates. The user provides a design to be verified, as well as formal specifications and a set of test programs generated either manually or automatically. FoCs translates the formal specification into checkers, which are then linked to the design and simulated with it. During simulation, the checker produces indications of property violations. It is up to the user to decide what action to take: fix the design, the property, or the simulation environment.

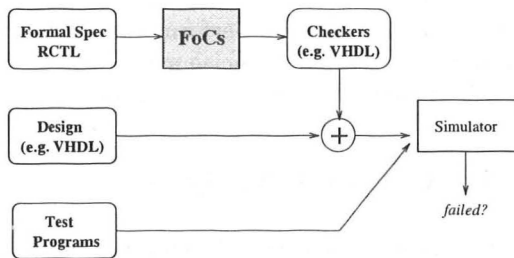


Fig. 1. FoCs Environment

FoCs translates RCTL into VHDL as follows: First, each property is translated into a non-deterministic finite automaton (NFA) and a simple $AG(p)$ formula, where p is a Boolean expression. The NFA has a set of distinguished error states, and the formula specifies that the NFA will never enter an error state (entering an error state means that the design does not adhere to the specification under the test conditions). The translation details are described in [BBL98].

Since contemporary simulators do not support non-determinism, the NFA has to be converted into a deterministic automaton (DFA). The DFA, in turn, is translated into a VHDL process - the FoCs checker. The $AG(p)$ formula is translated into a VHDL *Assert(p)* statement that prints a message when the VHDL process reaches a state where the underlying property is violated, and possibly stops simulation.

The number of states of the DFA may be exponential in the number of states of the NFA, but simulation is sensitive to the size of the representation (the number of VHDL lines) rather than to the number of states. The number of VHDL lines in the resulting VHDL checker is at most quadratic in the size of the property. Practically, it is almost always linear because of the types of properties that people tend to write.

The above translation process is implemented within the RuleBase model checker [BBEL96].

3 Example

The following example will demonstrate the conciseness and ease of use of RCTL for checker writing and the advantage of automatic checker generation. Assume that the following property is part of the specification: **If a transaction that starts with tag t has to send k bytes, then at the end of the transaction k bytes have been sent.** The user can formulate this property in RCTL as follows:

*forall k:*²

$\{*, start \ \& \ start_tag=t \ \& \ to_send=k, !end*, end \ \& \ end_tag=t\} \rightarrow \{sent=k\}$

Manual writing of a checker for this property may become complicated if transactions may overlap, which means that a new transaction may start while previous transactions are still active. The checker writer has to take into consideration all possible combinations of intervals and perform non-trivial bookkeeping. The RCTL formula is evidently much more concise and readable than the resulting VHDL file or a manually written VHDL or C program.

4 Using FoCs for Coverage Analysis

The quality of simulation-based verification depends not only on thorough checking of results, but also on the quality of the tests used (a.k.a. input patterns or test vectors) [KN96]. FoCs checkers can serve to enhance the quality of tests by providing a means for measuring *test coverage*. Test coverage measurement is the process of recording if certain user-defined events occurred during simulation of a set of tests. When used for coverage purposes, the FoCs checkers will evaluate the quality of the test suite by discovering events, or scenarios, that never happen during simulation. This feedback will guide the user which further tests are needed in order to cover scenarios that have not been exercised.

The implementation of coverage checkers is similar to that of functional checkers. The only difference is that instead of reporting an error, a coverage checker provides a positive indication when covering the relevant scenario. An

² The semantics of *forall* are intuitive. The implementation, however, is not trivial. It involves spawning a new automaton whenever a new value of k is encountered. Neither the formal semantics nor the implementation are included here due to lack of space.

example of a scenario to cover is: **a snoop event happens twice between read and write**, which can be formulated in RCTL as follows:

$$\{*, read, \{!snoop*, snoop, !snoop*, snoop\} \ \& \ \{!write*\} \} \rightarrow \{cover\}$$

A special case of coverage analysis is detection of *simulation vacuity*. While vacuity in model checking is defined as a failure of a subformula to influence the model checking results [BBER97], simulation vacuity refers to the failure of a set of tests to trigger a functional checker, which means that the checker did not influence simulation results. To detect simulation vacuity, FoCs attaches a coverage checker to each functional checker it generates; the coverage checker indicates whether the functional checker was triggered in at least one simulation run.

5 Experience

The FoCs toolset has been deployed in several projects in IBM, notably in the GigaHertz processor development effort in Austin and in the IBM Haifa ASIC development laboratory. FoCs has also been successfully used by the formal verification team of Galileo Technology, Inc. The experience with FoCs in these projects has been very favorable in terms of verification cost and quality. Using FoCs, verification effort was reduced - reportedly by up to 50% - by using the same formal rules for model checking at the unit level and for simulation analysis in the subsystem and system levels. This reduction was achieved despite the fact that the addition of checkers increases simulation time considerably (up to a factor of two). Thousands of FoCs checkers were written so far by virtue of the great ease of writing RCTL specifications and translating them to FoCs checkers.

6 Related Work

In a previous work, Kaufmann et al [KMP98] described an approach to verification in which all specifications and assumptions have the form $AG(p)$, where p is a Boolean formula. Both specifications and assumptions of this form can be used in simulation, with specification being translated into simple checkers, and assumptions being translated into testbench code which avoids leading the simulation into undesired states.

Canfield et al [CES97] describe a platform called Sherlock, aiming to serve both model checking (through translation to CTL) and simulation. Sherlock includes a high level language for specifying reactive behavior, while we use RCTL - a regular expression based language. While Sherlock postprocesses simulation traces, our method works during simulation. No experience has been described in [CES97]. Our own experience has demonstrated that the simple, concise syntax and semantics of RCTL, coupled with online simulation checking, is highly useful to the verification teams with whom we work.

Finally, in [SB+97], Schlipf et al describe a methodology and tool that inspired our work. They unify formal verification and simulation efforts by automatically translating state machines which represent the environment specification either to simulation behavioral models or to input for a model checker. Boolean assertions attached to the state machines serve as $AG(p)$ formulas in model checking or $Assert(p)$ in simulation. In contrast, our solution represents the specification as temporal logic formulas rather than state machines, for the sake of conciseness and readability.

7 Future Plans

Although focused on checker generation for functional testing and coverage analysis, we view FoCs as a step towards a full methodology of "Formal Specification, Design and Verification". We intend to provide a set of integrated, complementary tools that will facilitate the use of formal specification for multiple purposes. Once written, the formal specification will:

- Serve as an executable specification; architects can experiment with it while defining the specification
- Be used as a golden model to resolve ambiguities and misunderstandings during the implementation stage
- Be translated into temporal formulas for model checking
- Be translated into simulation checkers
- Be used for derivation of coverage criteria
- Provide hints for automatic test generation

We believe that such a methodology, once supported by the appropriate tools, will significantly contribute to the quality and efficiency of the design process.

References

- [BBEL96] I. Beer, S. Ben-David, C. Eisner, A. Landver, "RuleBase: an Industry-Oriented Formal Verification Tool", Proc. DAC'96, pp. 655-660.
- [BBER97] I. Beer, S. Ben-David, C. Eisner, Y. Rodeh, "Efficient Detection of Vacuity in ACTL Formulas", CAV'97, LNCS 1254, pp. 279-290.
- [BBL98] I. Beer, S. Ben-David, A. Landver, "On-The-Fly Model Checking of RCTL Formulas", CAV'98, LNCS 1427, pp. 184-194.
- [CES97] W. Canfield, E.A. Emerson, A. Saha, "Checking Formal Specifications under Simulations" Proc. ICCD'97.
- [GB+99] D. Geist, G. Biran, T. Arons, Y. Nustov, M. Slavkin, M. Farkash, K. Holtz, A. Long, D. King, S. Barret, "A Methodology for Verification of a System on Chip", Proc. DAC'99.
- [KN96] M. Kantrowitz, L. M. Noack, "I'm Done Simulating; Now What? Verification Coverage Analysis and Correctness", Proc. DAC'96.
- [KMP98] M. Kaufmann, A. Martin, C. Pixley, "Design Constraints in Symbolic Model Checking", CAV '98, LNCS 1427, pp. 477-487.
- [SB+97] T. Schlipf, T. Buechner, R. Fritz, M. Helms and J. Koehl, "Formal verification Made Easy", IBM Journal of R&D, Vol. 41, No. 4/5, 1997.

EXHIBIT B



Resource description

[Computer aided verification : 12th international conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000 ; proceedings / E. Allen Emerson, A. Prasad Sistla \(eds.\) CAV \(Conference\) Chicago, Ill\) 2000 : \(12th : Springer, Berlin ; London : 2000 \[3540677704\]](#)

Holdings

Verona Shelving Facility: Shelving Verona : QA76.76 E93 C38 2000

Holdings ID

22783666490002122

Barcode

89064908908

Process type

[Loan](#)

Status

Item not in place

Item ID

23633281420002122

MMS ID

9950629923602122

[View all holdings](#)

[View all items](#)

[Browse shelf listing](#)

General

ENUM/CHRON

Notes

History

General Information

Barcode

89064908908

Copy ID

1

Material type

Book

Item policy

general (book)

Provenance

-

Is magnetic

No

PO Line

Issue date

-

Receiving date

10/04/2000

Expected receiving date

-

Enumeration A

-

Enumeration B

-

Chronology I

-

Chronology J

-

Description

-

Pages

-

Pieces

1

Replacement cost

-

Receiving operator

import

Physical condition

-

Process type

Inventory Information

Inventory number

-

Inventory date

-

Inventory price

-

Location Information

Permanent location

Verona Shelving Facility: Shelving Verona (vsf)

Item call number type

-

Item call number

-

Source (Subfield 2)

-

Storage location ID

-

Temporary Location Information

Item is in temporary location

No

Temporary location

-

Temporary call number type

-

Temporary call number

-

Temporary source (Subfield 2)

-

Temporary item policy

-

Due back date

-

Retention Information

Committed to Retain

-

Retention Reason

-

Retention Note

-

