Farn Wang (Ed.)

LNCS 3299

Automated Technology for Verification and Analysis

Second International Conference, ATVA 2004
Taipei, Taiwan, ROC, October/November 2004
Proceedings

Springer

Farn Wang (Ed.)

Automated Technology for Verification and Analysis

Second International Conference, ATVA 2004
Taipei, Taiwan, ROC, October 31–November 3, 2004
Proceedings

Springer

Volume Editor

Farn Wang
National Taiwan University
Department of Electrical Engineering
1, Sec. 4, Roosevelt Rd., Taipei, Taiwan 106, ROC
E-mail: farn@cc.ee.ntu.edu.tw

Library of Congress Control Number: 2004113833

CR Subject Classification (1998): B.1.2, B.2.2, B.5.2, B.6, B.7.2, C.2, C.3, D.2, D.3, F.3

ISSN 0302-9743
ISBN 3-540-23610-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 11339656 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3299

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

It was our great pleasure to hold the 2nd International Symposium on Automated Technology on Verification and Analysis (ATVA) in Taipei, Taiwan, ROC, October 31–November 3, 2004. The series of ATVA meetings is intended for the promotion of related research in eastern Asia. In the last decade, automated technology on verification has become the new strength in industry and brought forward various hot research activities in both Europe and USA. In comparison, eastern Asia has been quiet in the forum. With more and more IC design houses moving from Silicon Valley to eastern Asia, we believe this is a good time to start cultivating related research activities in the region.

The emphasis of the ATVA workshop series is on various mechanical and informative techniques, which can give engineers valuable feedback to fast converge their designs according to the specifications. The scope of interest contains the following research areas: model-checking theory, theorem-proving theory, state-space reduction techniques, languages in automated verification, parametric analysis, optimization, formal performance analysis, real-time systems, embedded systems, infinite-state systems, Petri nets, UML, synthesis, tools, and practice in industry.

As a young symposium, ATVA 2004 succeeded in attracting 69 submissions from all over the world. All submissions were rigorously reviewed by three reviewers and discussed by the PC members through the network. The final program included a general symposium and three special tracks: (1) Design of secure/high-reliability networks, (2) HW/SW coverification and cosynthesis, and (3) hardware verification. The general symposium consisted of 24 regular papers and 8 short papers. The three special tracks together accepted 7 papers. The final program also included three keynote speeches by Bob Kurshan, Rajeev Alur, and Pei-Hsin Ho; and three invited speeches by Jean-Pierre Jouannaud, Tevfik Bultan, and Shaoying Liu. The symposium was also preceded by three tutorials by Bob Kurshan, Rajeev Alur, and Pei-Hsin Ho.

We want to thank the National Science Council, Ministry of Education, and Academia Sinica of Taiwan, ROC. Without their support, ATVA 2004 would not have come to reality. We thank the Department of Electrical Engineering, Center for Information and Electronics Technologies (CIET), SOC Center, and Graduate Institute of Electronic Engineering (GIEE) of National Taiwan University for their sturdy support, and we thank Synopsys, Inc. for sponsoring ATVA 2004. We thank all the tutorial–keynote speakers, invited speakers, committee members, and reviewers of ATVA 2004. Finally, we thank Mr. Rong-Shiung Wu, for his help in maintaining the webpages and compiling the proceedings, and Mr. Lin-Zan Cai, for his help in all the paperwork.

August 2004 Farn Wang

Organization

Steering Committee

E.A. Emerson (USA) Oscar H. Ibarra (USA)
Insup Lee (USA) Doron A. Peled (UK)
Farn Wang (Taiwan) Hsu-Chun Yen (Taiwan)

Organizing Chair

Hsu-Chun Yen

Program Chair

Farn Wang

Program Committee

Tommaso Bolognesi (Italy) Tevfik Bultan (USA)
Sungdeok Cha (Korea) Yung-Ping Cheng (Taiwan)
Jin-Young Choi (Korea) Jing-Song Dong (Singapore)
Jifeng He (China) Teruo Higashino (Japan)
Pao-Ann Hsiung (Taiwan) Chung-Yang Huang (Taiwan)
Oscar H. Ibarra (USA) Insup Lee (USA)
Huimin Lin (China) Doron A. Peled (UK)
Scott D. Stoller (USA) Yih-Kuen Tsay (Taiwan)
Bow-Yaw Wang (Taiwan) Farn Wang (Taiwan)
Hsu-Chun Yen (Taiwan) Tomohiro Yoneda (Japan)

Special Tracks

1. Design of Secure/High-Reliability Networks, Chair: Teruo Higashino
   Additonal PC members:
   Ana R. Cavalli (France) Tai-Yi Huang (Taiwan)
   Masakatsu Nishigaki (Japan) Shoji Yuen (Japan)
2. HW/SW Coverification and Cosynthesis, Chair: Pao-Ann Hsiung
   Additonal PC members:
   Rong-Guey Chang (Taiwan) Tai-Yi Huang (Taiwan)
   Jung-Yi Kuo (Taiwan) Alan Liu (Taiwan)
   Win-Bin See (Taiwan)
3. Hardware Verification, Co-chairs: Chung-Yang Huang, Bow-Yaw Wang
   Additonal PC members:
   Tai-Yi Huang (Taiwan) Masakatsu Nishigaki (Japan)

Reviewers

Madhukar Anand	Constantinos Bartzis	Jing Chen
Ting-Shuo Chou	Edward T.H. Chu	Zhe Dang
Arvind Easwaran	Xiang Fu	Dimitra Giannakopoulou
Kiyoharu Hamaguchi	Ping Hao	Hiromi Hiraishi
Geng-Dian Huang	Kuang-Li Huang	Ranjit Jhala
Li Jiao	Jesung Kim	Moonzoo Kim
Masaaki Kondo	Rom Langerak	Dongdai Lin
Xinxin Liu	Zhiming Liu	Stephane Maag
Franco Mazzanti	Chris Myers	Kozo Okano
Hong Pan	Usa Sammapun	Oleg Sokolsky
Jin Sun	Kenji Taguchi	Yu-Che Tsai
Tatsuhiro Tsuchiya	Razvan Voicu	Liqiang Wang
Rui Xue	Ping Yang	Tuba Yavuz-Kahveci
Karen Yorav	Fang Yu	Jian Zhang

Sponsoring Institutions

National Science Council, Taiwan, ROC
Ministry of Education, Taiwan, ROC
Institute of Information Science, Academia Sinica, Taiwan, ROC
National Taiwan University (NTU), Taiwan, ROC
Center for Information and Electronics Technologies (CIET), NTU, Taiwan, ROC
SOC Center, NTU, Taiwan, ROC
Graduate Institute of Electronic Engineering, NTU, Taiwan, ROC
Synopsys, Inc.

Lecture Notes in Computer Science

For information about Vols. 1–3193

please contact your bookseller or Springer

Vol. 3305: P.M.A. Sloot, B. Chopard, A.G. Hoekstra (Eds.), Cellular Automata. XV, 883 pages. 2004.

Vol. 3302: W.-N. Chin (Ed.), Programming Languages and Systems. XIII, 453 pages. 2004.

Vol. 3299: F. Wang (Ed.), Automated Technology for Verification and Analysis. XII, 506 pages. 2004.

Vol. 3293: C.-H. Chi, M. van Steen, C. Wills (Eds.), Web Content Caching and Distribution. IX, 283 pages. 2004.

Vol. 3292: R. Meersman, Z. Tari, A. Corsaro (Eds.), On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. XXIII, 885 pages. 2004.

Vol. 3291: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. XXV, 824 pages. 2004.

Vol. 3290: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. XXV, 823 pages. 2004.

Vol. 3287: A. Sanfeliu, J.F.M. Trinidad, J.A. Carrasco Ochoa (Eds.), Progress in Pattern Recognition, Image Analysis and Applications. XVII, 703 pages. 2004.

Vol. 3286: G. Karsai, E. Visser (Eds.), Generative Programming and Component Engineering. XIII, 491 pages. 2004.

Vol. 3284: A. Karmouch, L. Korba, E.R.M. Madeira (Eds.), Mobility Aware Technologies and Applications. XII, 382 pages. 2004.

Vol. 3281: T. Dingsøyr (Ed.), Software Process Improvement. X, 207 pages. 2004.

Vol. 3280: C. Aykanat, T. Dayar, İ. Körpeoğlu (Eds.), Computer and Information Sciences - ISCIS 2004. XVIII, 1009 pages. 2004.

Vol. 3274: R. Guerraoui (Ed.), Distributed Computing. XIII, 465 pages. 2004.

Vol. 3273: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), <<UML>> 2004 - The Unified Modelling Language. XIII, 454 pages. 2004.

Vol. 3271: J. Vicente, D. Hutchison (Eds.), Management of Multimedia Networks and Services. XIII, 335 pages. 2004.

Vol. 3270: M. Jeckle, R. Kowalczyk, P. Braun (Eds.), Grid Services Engineering and Management. X, 165 pages. 2004.

Vol. 3269: J. Lopez, S. Qing, E. Okamoto (Eds.), Information and Communications Security. XI, 564 pages. 2004.

Vol. 3266: J. Solé-Pareta, M. Smirnov, P.V. Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, R.J. Gibbens (Eds.), Quality of Service in the Emerging Networking Panorama. XVI, 390 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004. (Subseries LNAI).

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004. (Subseries LNAI).

Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), Object-Oriented and Internet-Based Technologies. XII, 239 pages. 2004.

Vol. 3262: M.M. Freire, P. Chemouil, P. Lorenz, A. Gravey (Eds.), Universal Multiservice Networks. XIII, 556 pages. 2004.

Vol. 3261: T. Yakhno (Ed.), Advances in Information Systems. XIV, 617 pages. 2004.

Vol. 3260: I.G.M.M. Niemegeers, S.H. de Groot (Eds.), Personal Wireless Communications. XIV, 478 pages. 2004.

Vol. 3258: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004. XVII, 822 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004. (Subseries LNAI).

Vol. 3256: H. Ehrig, G. Engels, F. Parisi-Presicce, G. Rozenberg (Eds.), Graph Transformations. XII, 451 pages. 2004.

Vol. 3255: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Advances in Databases and Information Systems. XI, 423 pages. 2004.

Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. XVI, 910 pages. 2004.

Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. X, 397 pages. 2004.

Vol. 3252: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004 Workshops. XVIII, 785 pages. 2004.

Vol. 3251: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004. XXII, 1025 pages. 2004.

Vol. 3250: L.-J. (LJ) Zhang, M. Jeckle (Eds.), Web Services. X, 301 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004. (Subseries LNAI).

Vol. 3246: A. Apostolico, M. Melucci (Eds.), String Processing and Information Retrieval. XIV, 332 pages. 2004.

Vol. 3245: E. Suzuki, S. Arikawa (Eds.), Discovery Science. XIV, 430 pages. 2004. (Subseries LNAI).

Vol. 3244: S. Ben-David, J. Case, A. Maruoka (Eds.), Algorithmic Learning Theory. XIV, 505 pages. 2004. (Subseries LNAI).

Vol. 3243: S. Leonardi (Ed.), Algorithms and Models for the Web-Graph. VIII, 189 pages. 2004.

Vol. 3242: X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII. XX, 1185 pages. 2004.

Vol. 3241: D. Kranzlmüller, P. Kacsuk, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XIII, 452 pages. 2004.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004. (Subseries LNBI).

Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 444 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004. (Subseries LNAI).

Vol. 3236: M. Núñez, Z. Maamar, F.L. Pelayo, K. Pousttchi, F. Rubio (Eds.), Applying Formal Methods: Testing, Performance, and M/E-Commerce. XI, 381 pages. 2004.

Vol. 3235: D. de Frutos-Escrig, M. Nunez (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2004. X, 377 pages. 2004.

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3231: H.-A. Jacobsen (Ed.), Middleware 2004. XV, 514 pages. 2004.

Vol. 3230: J.L. Vicedo, P. Martínez-Barco, R. Muñoz, M. Saiz Noeda (Eds.), Advances in Natural Language Processing. XII, 488 pages. 2004. (Subseries LNAI).

Vol. 3229: J.J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence. XIV, 744 pages. 2004. (Subseries LNAI).

Vol. 3226: M. Bouzeghoub, C. Goble, V. Kashyap, S. Spaccapietra (Eds.), Semantics of a Networked World. XIII, 326 pages. 2004.

Vol. 3225: K. Zhang, Y. Zheng (Eds.), Information Security. XII, 442 pages. 2004.

Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), Recent Advances in Intrusion Detection. XII, 315 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logics. VIII, 337 pages. 2004.

Vol. 3222: H. Jin, G.R. Gao, Z. Xu, H. Chen (Eds.), Network and Parallel Computing. XX, 694 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3219: M. Heisel, P. Liggesmeyer, S. Wittmann (Eds.), Computer Safety, Reliability, and Security. XI, 339 pages. 2004.

Vol. 3217: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 1114 pages. 2004.

Vol. 3216: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 930 pages. 2004.

Vol. 3215: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVII, 906 pages. 2004. (Subseries LNAI).

Vol. 3214: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1302 pages. 2004. (Subseries LNAI).

Vol. 3213: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1280 pages. 2004. (Subseries LNAI).

Vol. 3212: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 862 pages. 2004.

Vol. 3211: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 880 pages. 2004.

Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic. XI, 520 pages. 2004.

Vol. 3209: B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (Eds.), Web Mining: From Web to Semantic Web. IX, 201 pages. 2004. (Subseries LNAI).

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3204: C.A. Peña Reyes, Coevolutionary Fuzzy Modeling. XIII, 129 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Knowledge Discovery in Databases: PKDD 2004. XIX, 560 pages. 2004. (Subseries LNAI).

Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004. XVIII, 580 pages. 2004. (Subseries LNAI).

Vol. 3199: H. Schepers (Ed.), Software and Compilers for Embedded Systems. X, 259 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3196: C. Stary, C. Stephanidis (Eds.), User-Centered Interaction Paradigms for Universal Access in the Information Society. XII, 488 pages. 2004.

Vol. 3195: C.G. Puntonet, A. Prieto (Eds.), Independent Component Analysis and Blind Signal Separation. XXIII, 1266 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Table of Contents

Games for Formal Design and Verification of Reactive Systems

Rajeev Alur

University of Pennsylvania, USA

Abstract. With recent advances in algorithms for state-space traversal and in techniques for automatic abstraction of source code, model checking has emerged as a key tool for analyzing and debugging software systems. This talk discusses the role of games in modeling and analysis of software systems. Games are useful in modeling open systems where the distinction among the choices controlled by different components is made explicit. We first describe the model checker Mocha that supports a game-based temporal logic for writing requirements, and its applications to analysis of multi-party security protocols. Then, we describe how to automatically extract dynamic interfaces for Java classes using predicate abstraction for extracting a boolean model from a class file, and learning algorithms for constructing the most general strategy for invoking the methods of the model. We discuss an implementation in the tool JIST—Java Interface Synthesis Tool, and demonstrate that the tool can construct interfaces, accurately and efficiently, for sample Java2SDK library classes.

F. Wang (Ed.): ATVA 2004, LNCS 3299, p. 1, 2004.

Evolution of Model Checking into the EDA Industry

Robert P. Kurshan

Cadence Design Systems, USA

Today, the Electronic Design Automation (EDA) industry is making its second attempt to commercialize model checking tools for hardware verification. Its first attempt started about 6 years ago, in 1998. While this first attempt was only barely successful commercially, it resonated well enough with customers of the model checking tool vendors to motivate a second round of commercial offerings in 2004.

Why has it taken almost a quarter century for model checking to surface in a commercial venue? Why did not the great academic tools of the '80s and '90s translate more directly into useful commercial tools?

In retrospect, there are three clear answers to these questions:

1. application of model checking in commercial flows requires a significant change in design methodology, advancing verification from post-development to the early design stage, onto the shoulders of developers; historically, developers have been considered "too valuable" to burden with testing ("verification" and "test" are not distinguished in EDA);
2. a commercial-quality tool is expensive to deploy, requiring verification experts to program the core algorithms with close attention to performance, and beyond this requiring significant efforts in developing use models, the integrated tool architecture, user interfaces, documentation, product quality validation (product testing), marketing, customer support and – very critically – convincing the Sales Team (who work on commission and/or bonuses based on sales volume) that they should put in a lot of effort to learn and sell a new tool when they already have an established customer base for the commoditized tools like simulators that they can sell in million dollar batches with a phone call;
3. given 1., the market for these tools was hard to estimate, so it was hard or impossible to calculate the expected return on investment of the daunting costs in 2.

Nonetheless, in the '90s, the growing inadequacy of existing functional verification methods was reaching crisis proportions on account of the inability of simulation test to keep up with exponentially growing design complexity. There was growing pressure on the EDA industry to provide better support for weeding out of circuit designs an increasing number of disastrous functional bugs, before those designs hit the marketplace.

Previously, proof-of-concept demonstrations of the value of formal verification, at least in the hands of experts, had become ever more persuasive, with many demonstration projects in academia and industry that showed the potential of formal verification to solve the looming test crisis.

Around 1998, the EDA industry responded timidly to these pressures by releasing under-funded and thus inadequate answers to these industry needs. Lack of funding resulted in short-changing one or more of the requirements cited in 2. above, and/or failing to adequately address the issue 1. The result was a lot of sparks of interest,

F. Wang (Ed.): ATVA 2004, LNCS 3299, pp. 2–6, 2004.

even some flames of satisfaction from well-positioned users, but no broadly sustainable verification product that could be fanned out widely in EDA.

However, the sparks and flames did catch the attention of EDA management enough to fund a second round. The first focus of the second round was to evaluate the failures of the first round and devise solutions for these failures.

The first issue to address was 1. Designers are widely supported like "prima donnas", whereas "product verification" is often considered to be an entry level job from which one seeks to advance to more glamorous work like design. Therefore, to ask the designer to support verification was largely considered by management to be a non-starter.

Since the classical test flow consisted of handing the completed design together with a system specification to a testing group, it was natural at some level for management to presume that by analogy they should hand off a completed design to a model checking team. Since there was little available expertise in model checking, they looked to academia for this expertise. In the first model checking flows, newly hired formal verification Ph.d's augmented with summer students served as the first model checking teams.

The trouble with this setup was that whereas classical test teams could infer system tests from a system specification with which they were provided, the intrinsic computational capacity limitations on model checking required that model checking be applied at the design block level. There are generally no design specifications for individual blocks beyond rough engineering notes that are rarely up to date and often hard to understand.

These model checking teams were thus forced to spend an inordinate amount of time studying block designs in order to come up with suitable properties to check. Often, this process included quizzing the designers, which some designers resented as an unwelcome intrusion on their time, or else management feared that it would be that. Moreover, it was insufficient to learn only the blocks to be tested. It was also required to learn the environment of those blocks, in order to design an "environment model" or constraints for the blocks to be verified, in order to preclude false failures. Getting the environment model right was often the hardest part of the process, as it required learning a large part of the design, far beyond the portion to be checked.

In summary, using a dedicated model checking team was not a solution that would scale to a general widely deployed practice. The team was too far from the design to be able to easily understand it as required, and extracting the required information from the designers was considered too disruptive to the designers.

For the second round the clear priorities, in order, were these:

1. FIRST, focus on USABILITY to break into the current development flow;
2. then, focus on capacity: how to scale the use model to the same size designs to which simulation test applies;
3. finally, focus on PERFORMANCE in order to get results fast enough to augment and keep up with the normal test flow.

The syllogism went like this: formal verification must be applied to design blocks, on account of capacity limitations; but, only the designer understands a design at the granularity of its blocks; therefore, it must be the designer who facilitates formal verification.

On the one hand, advancing verification in the development flow to the earlier design phase offered a big potential advantage. It could reduce development costs by finding bugs earlier, thereby saving more costly fixes later. But this left unanswered how break through the cultural barrier: designers dont do test!

The answer to this puzzle came through the evolution of *assertion languages*. An assertion language is a formal language that is used to specify properties to be checked in a design. The most common assertion languages (although they were not called that) were the logics LTL and CTL, in use in academia for two decades. These were hard to assimilate (even for the experts) and only meek attempts were made to introduce them to designers. In 1995, along with one of the first commercial model checkers, FormalCheck from Lucent Technologies, came a very simple and intuitive assertion language: the FormalCheck Query Language (FQL). FQL was strictly more expressive than LTL, being able to express any ω-regular language. It was expressed through templates like

$$\text{After}(e)\ \text{Always}(f)\ \text{Unless}(d)$$

$$\text{After}(e)\ \text{Eventually}(d)$$

where e, f and d are Boolean expressions in design variables and the template expressions imply universal quantification over design states. FQL made it harder to write complex logic expressions, but simpler and more transparent to write simple common expressions. By conjuncting such simple templates, any ω-regular property could be expressed.

IBM also saw a need to make the assertion language more accessible to designers, but took another approach. They implemented a textual version of CTL in their model checker RuleBase. Intel, Motorola and others also found solutions to the the problem of making assertion languages more palatable to designers, in some cases by greatly restricting expressiveness. One example in this direction was Verplex's OVL, a template-based assertion language like FQL, but significantly less expressive although possibly even simpler to understand.

Designers were encouraged to use an assertion language to write "comments" that described the correct behavior of their blocks. This was not the same as asking the designer to participate in testing (verification) – it was only asking the designer to document precisely the functional requirements of a block's design. While some designers also shy away from commenting their code, requiring a designer to write a precise functional specification of a design is something that development managers have long thought to be important, and now with a good concrete justification (to facilitate better verification), managers bought into this requirement on their designers.

With assertions in place, the plan was to use a verification team to check the assertions. This strategy became know as *Assertion-Based Verification*. Since the designers wrote the assertions, there was no need for the verification team to understand the design. Moreover, with assertions in every block, there was no need for the verification team to write an environment model: the assertions from adjacent blocks served as the environment model.

There was one practical problem with this approach. Managers felt uneasy to invest considerable resources in a proprietary assertion language: what if the tools that supported that assertion language were not the best tools? Design managers wanted to evaluate the various options in the market place and then select the best one. But to