Some Common Aspects of Design Validation, Debug and Diagnosis

Talal Arnaout, Günter Bartsch, Hans-Joachim Wunderlich
Institut für Technische Informatik,
Universität Stuttgart
Pfaffenwaldring 47; D-70569 Stuttgart, Germany.
email: {talal.arnaout, guenter.bartsch, wu}@informatik.uni-stuttgart.de

***Abstract*— Design, Verification and Test of integrated circuits with millions of gates put strong requirements on design time, test volume, test application time, test speed and diagnostic resolution. In this paper, an overview is given on the common aspects of these tasks and how they interact. Diagnosis techniques may be used after manufacturing, for chip characterization and field return analysis, and even for rapid prototyping.**

I. Introduction

Traditionally, design, verification and diagnosis of microelectronic circuits have been viewed as separate tasks with individual challenges and techniques. However, in recent years more and more attention has been paid to the interaction of individual design steps in verification, diagnosis of prototypes, and field return analysis. These are tasks for quality control and improvement during the complete lifecycle of the system and tackle faults, occurring during design, manufacturing and operation.

As Systems on Chip (SoC) design complexity increases, the verification process is turning into a critical bottleneck in the design process. Estimates today are that more than 70% of the total design time is on verification [1], [2]. Despite the efforts spent from the academia and the industry on developing functional verification tools, logical and functional flaws remain the main cause of today's design spins. Between the years 2002 and 2004, the percentage of designs with functional errors has actually increased [3].

Design validation tries to rule out design faults, basic methodologies are simulation, emulation and formal verification. Modeling and test bench construction are the main tasks to be performed.

Debug is the time-consuming task of identifying faulty modules and structures within the design. While some methods of formal verification are constructive and able to find the cause of malfunctions, simulation and emulation usually require additional design means for fault location, so called assertions.

Test is not any more dealing with the design but with the produced chip. Test is performed all over the system's lifecycle. Not only the first prototypes but all the chips manufactured during mass production are subject of testing as we cannot account on 100% yield. Even in the field, testing is performed during start up or maintenance.

Online testing performs the test without stopping system operation. If just erroneous behavior has to be found, self-checking circuitry has to be employed. If von-volatile soft-errors or even permanent faults have to be detected in advance, concurrent structural testing is used as well.

While testing is just detecting the presence of a fault, *fault diagnosis* is the process of both, detecting and locating the fault at the various levels down to the real defect. Usually, the logic behavior of the design has been validated during simulation, verification and rapid prototyping. But numerous parasitic and timing effects may show up in the first silicon, identifying them is part of silicon debug. Hence, diagnosis is more related to defects and debug is closer to design errors, i. e. errors of the designer. However, there is a large overlap in between dealing with yield ramping and design for manufacturability.

For the rest of this paper we try to point out common aspects of the tasks mentioned above.

II. Test bench generation and virtual testing

During validation, the *test bench* is an environment for generating input stimuli and evaluating the response of the design under test (figure 1). High coverage of possible design faults requires that the design is exercised in all of its operation modes by a sufficient amount of input patterns. Usually, test benches are generated concurrently with the design under test in early design phases.

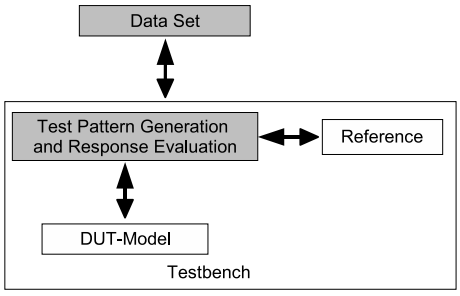

Fig. 1. General test bench

Virtual test provides a model of the automatic test equipment (figure 2). This model is used in late design phases or

even after design for test program generation. The ATE test program can be generated, simulated and debugged without the presence of an ATE. Development time is saved by concurrent engineering, and expensive ATE time is saved as well.

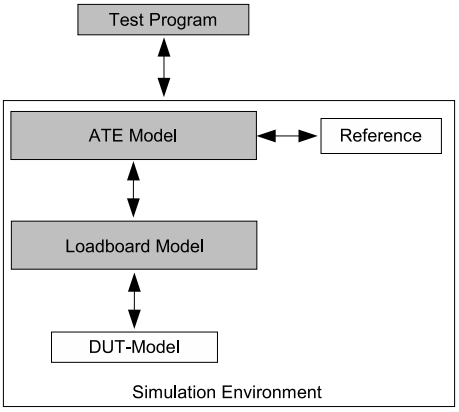

Fig. 2. Virtual Testing

Writing test benches for system verification is a very tedious task and error prone. Traditionally, verification engineers generate these test benches manually. However, with today's complex designs, it is almost impossible to cover all possible corner cases successfully. This motivated research to facilitate and improve the process of test bench generation.

One approach is based on a divide-and-conquer algorithm to automatically generate the test bench [4]. The key issue in this approach is identifying the partitioning boundaries where interactions among divided components are minimized.

Another approach is based on random simulation-vectors generation [5]. The key issue in this algorithm is simplifying conjunctive Boolean constraints defined over state and input variables, and applying it to constrained random simulation vector generation using binary decision diagrams (BDDs). The authors present a method to generate a relatively small sized BDD representation of the constraints, which in turn is reflected on reduced time to generate the simulation vectors.

Industrial products took over these ideas. Questa [3] of Mentor Graphics and Vera [6] of Synopsys, for example, use the constrained random simulation vector generation approach. The designer describes stimulus scenarios in terms of constraints, and the simulators generate random stimuli vectors based on the given constraints.

Other approaches [2], [7], [8], [9], [10], [11] have considered accelerating the test benches, by mapping them partially or fully into hardware, as a means to improve the efficiency of test benches and speedup the verification process. This step could be considered complementary to the above approaches. The authors in [7] propose a mechanism to partition a behavioral test bench into two parts: one part residing on hardware, and the other on software. This is beneficial when using the co-emulation approach discussed above. The proposed algorithm results in an optimal partition to reduce the communication overhead between the two components.

From the industry's side, Infineon has developed an adaptive approach to generate test benches that can be applied to simulation, test, and emulation depending on the abstraction level the verification task is being conducted at [11].

As shown in figure 3, each unit is connected to a compatible test bench element, depending on the abstraction level, and all the elements communicate via a bus to a main test bench kernel. This adaptive architecture for structuring test benches allows verifying the complete system with the same test bench, even if the system is described using different abstraction levels.

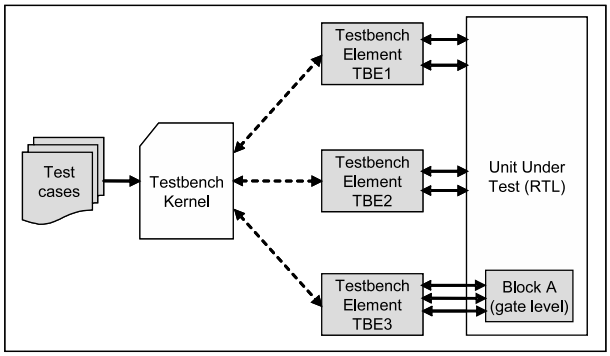

Fig. 3. Adaptive approach for structuring test benches

Reuse of test benches for virtual test is subject of recent research mainly in the mixed-signal area [12], up to now a considerable amount of double work is done by the design and test community.

III. Transaction Level Models and Embedded Test

Software simulators are the most common means for validation. However, the performance of these simulators is inadequate to handle today's and future design complexity. These simulators run on general purpose workstations, where they execute tasks sequentially.

To keep up with today's complexity, research and development efforts have been put up together to develop new and more efficient means for validation. One of these was hardware emulators, which provided a gigantic acceleration to the traditional simulation process. Hardware emulators run the circuit's model in real time on configurable hardware. Although emulation provides high verification performance, it is unable on its own to run a behavioral test bench. This reduces its flexibility in comparison to software simulators, which can execute system calls, such as terminal display and file I/O, and high-level test benches, such as Vera and C++. Therefore, to achieve a more efficient verification process, software simulators are coupled with hardware emulators [13], [14], [15].

This has given rise to a new verification methodology known as Transaction-based verification (TBV) [16], which poses a new bottleneck to the process: communication overhead.

In an attempt to reduce the communication overhead, transaction-level interface was introduced by Accellera as

Standard Co-Emulation Modeling Interface (SCE-MI) [17], which defines application program interfaces (APIs) for test bench and hardware interfaces for transactor description. For this method, the transactor should be described in a synthesizable form and the test bench should be written in a high level language. The test pattern program communicates with the transactors on the hardware side in a compressed form, while the transactors generate comprehensive stimuli for the DUT and evaluate the responses (figure 4).

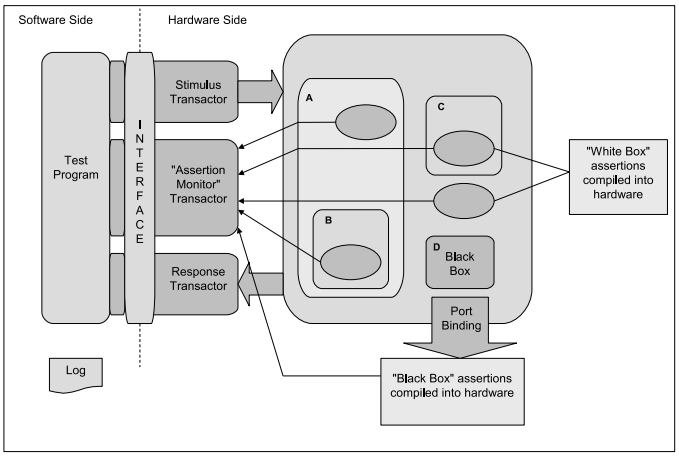

Fig. 4. Transaction level modeling according to [18]

Several attempts try to improve the functional coverage of TLV [3], [19], [20], [21], [22]. For example, the authors of [21] propose fine-grained transaction-level verification, where high signal level coverage is achieved by systematically modifying the operation of the transactors and rerunning the simulation rather than changing transaction-level scenarios. According to the authors, this should reveal more bugs in a shorter time.

While TLM supports the early design phases, embedded testing is a means for enabling an economic test after manufacturing. Test resources are partitioned between the external ATE and the chip.

The control of the test hardware is not generated on chip but provided by a low cost tester. Commercial schemes like EDT or OPMISR implement this type of test resource partitioning [23], [24]. Figure 5 illustrates this principle on the basis of embedded deterministic test (EDT).

It is immediately seen that the basic architectures for TLM and ET are identical, test data reduction may be considered as a special phase of transactions. Reuse of transactions in both domains is an emerging field of research.

IV. Debug and diagnosis

A. Assertion-based debug

The so far mentioned techniques may help in verifying the functionality of a design. If the verification process detects a mismatch in the result, it might be unable to pinpoint the cause of this mismatch. To overcome this weakness, assertions are introduced into the process to monitor the activity of the

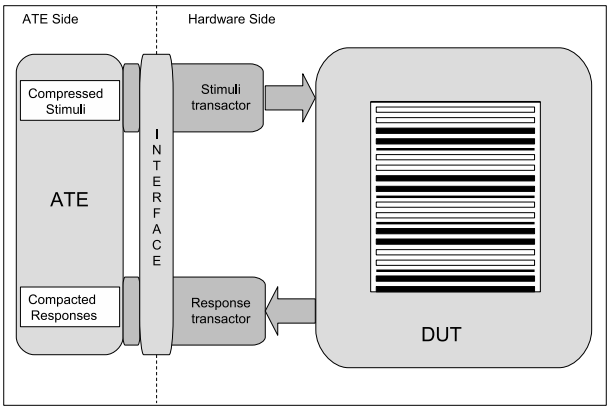

Fig. 5. Principle of embedded test

circuit, and report any present bugs. This process is known as assertion-based verification (ABV)(see figure 4).

Assertions are statements that describe how a design's logic is intended to behave. They are embedded into the hardware description language (VHDL or Verilog) code that describes the IP block's logic. When this code is fed into a design verification program, the program can check the assertions to determine whether the block behaves as it should. For example, assertions check that two signals never contend a certain bus at the same time, as shown in figure 6.

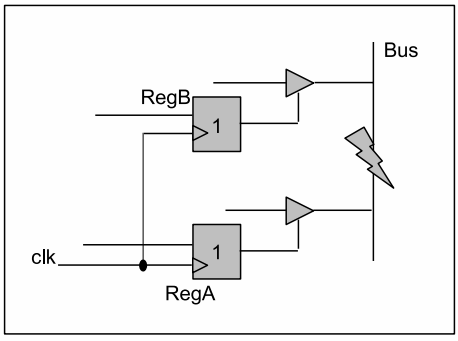

Fig. 6. Assertion check to prevent the hazard

The benefits of assertions are as follows:

- Improve observability and controllability of the design,
- Capture design specification in a formal way,
- Improve the verification efficiency by detecting more inherent bugs,
- Improve integration and reuse of third party designs,
- Improve communication through documentation.

Today, there are a number of tools to describe assertions with, such as Superlog, Property Specification Language (PSL), OpenVera Assertions (OVA), Open Verification Library (OVL) [25]. To ease the design and verification process, Accellera has taken action to provide a unified Hardware Description and Verification Language (HDVL) that can serve the dual purpose of designing as well as "assertion-based"

verifying the design by releasing SystemVerilog as a new standard [26].

Once inserted into a code, assertions can be used as monitors to detect incorrect behavior. And the same assertions can also be used for formal verification, such as in model checking. Model checkers see assertions either as properties inside a discrete block of code or as a constraint that describe legal inputs to the IP block.

Recently, ABV has been the center of attention of a number of researchers and developers. Some researchers have made the attempt to upgrade existing modeling description languages, such as SystemC, by adding new features that support assertion-based design [27], [28], while others have brought up the necessary concepts, developed new strategies and even modified existing ones to support this powerful process [1], [25], [29], [30], [31], [32], [33].

After describing and synthesizing the assertions, the logical following step would be generating the correct stimuli vectors that would prove (or falsify) them. This field also has seen recently some activity [32], [34], [35], [36], where algorithm developers have proposed some practical solutions. The basis for developing such solutions is quite diversified. Some are based on interval analysis [32], [35], where the range of applicable inputs that trigger the assertion is analyzed to generate the best stimuli vectors. Other methods are based on SAT solvers [34], [36], which try to generate the stimuli vectors by satisfying the constraints set by the assertions. [32] even utilizes genetic algorithms to even further optimize the test generation to get more efficient coverage of the assertions. While assertions help in identifying regions and modules with a faulty behavior the target of diagnosis is an even more fine-grained analysis.

B. Fault models and dictionaries

For test and diagnosis, fault models play an important role as it is not feasible to generate tests for an arbitrary faulty behavior. Fault models at the various abstraction levels restrict the complexity of test generation, on the other hand, they also reduce resolution. Some of these faults may not have a counterpart at lower levels or in reality. For instance, there may be a gate level model of a design with a corresponding fault, but actually, the layout was generated by a single pass synthesis directly from register transfer level. On the other hand, some defects may not be modeled at all.

In nanometer technology, one must be aware that the parametric and functional properties of each single gate, transistor or line may vary within a large range of values [37]. Specification compliance testing tries to select a module and to check whether its properties are still within the allowed range despite the variations of its devices. If such a module is too large, compliance testing will not be feasible, but if it is too small, the test outcome may be invalidated into a false accept or false reject by the variations of the environment.

Path delay fault testing may be considered as a special case of compliance testing, where all the gates of a certain path are varying within the allowed range, but the path is still slow [38], [39]. In order to deal with the increasing variations, statistical methods for both, fault and circuit modeling are required to obtain diagnostic test patterns [40], [41], [42], [43], [44], [45].

A special type of delay faults is formed by so called cross talk faults. This fault model describes the capacitive coupling of two lines. If these lines are switched in parallel to opposite values, the aggressor line may finally slow down the signal change of the victim line. Here, a diagnostic pattern pair must first initialize two nodes and, second, observe the transition speed of the victim node. Most often these tests are required during chip characterization and are part of silicon debug, since these faults mainly exhibit a design flaw [46]. But also after manufacturing variations may cause cross talk effects on some dies to be diagnosed.

A more direct coupling of two lines is formed by bridging faults. Additional material or missing insulators connect two lines a and b, and a standard fault model maps this fault to wired-AND or wired-OR structures [47], [48]. To implement a more realistic model, resistive bridges are considered, too [49].

A fault dictionary is a matrix containing the information which pattern detects which faults. Once a fault dictionary is computed, a simple search algorithm is sufficient to deduce at least for single faults which fault caused the observed faulty circuit behavior.

The number of faults grows at least linearly with the size of the circuit. As the number of faults increases linearly, so does the number of necessary test patterns. This results in a $O(n^2)$ growth for the fault dictionary, which renders the straight forward approach impractical for larger circuits.

Some attempts keep fault dictionary sizes low [50] by generating them on the fly in order to handle the complexity.

Fault dictionaries are bound to a specific fault model. The larger the number of faults defined by the fault model for a given circuit, the bigger the dictionary becomes.

C. Diagnosis

The main goal of diagnostic techniques is to achieve the highest diagnostic resolution within a short application time [51].

Design for test techniques are reused and mandatory, but not all of these techniques really support diagnosis. There is a plethora of pattern compression and compaction techniques available which loose efficiency if diagnostic patterns are applied. The main reason is that these techniques work best for patterns with a large account of unspecified bits while high resolution usually increases the number of specified bits. Nearly all of the diagnosis algorithms rely on a scan design.

The traditional approach for diagnosis is using a pre-calculated fault dictionary for circuit response analysis. This is also called *static diagnosis*.

For example [52] describes Poirot, a tool for diagnosis of full-scan and partial-scan sequential circuits using this approach. This tool first does static (just circuit structure based) or dynamic (also including pattern simulation based) structural processing on the given circuit. In this step cones

originating from scan cells are extracted from the circuit for later deduction of fault locations from circuit responses.

The composite Fault Model is a diagnostic fault model which uses stuck-at fault information to model bridging faults, node faults (gates representing a wrong logic function) and stuck-open faults. Each complex fault is represented by a number of possible stuck-at fault configurations which are then matched to the observed stuck-at faults.

First defined in [53], adaptive diagnosis does not need precalculated fault dictionaries and test patterns and therefore reduces the complexity issues with the static diagnosis approaches.

[53] suggest an iterative algorithm which generates in each step a small number of test patterns for a small set of faults. Then this small set of patterns (which does not necessarily cover the whole fault list for the circuit) is applied and the responses are used for dropping those faults which have been covered but have not been observed in the circuit. This is repeated until no further faults can be removed from the list.

The test patterns used in this approach are generated in three phases: In the beginning the test patterns are randomly generated. In the next phase patterns are generated specifically for hard-to-control parts of the circuit and finally deterministically generated patterns for the remaining faults are generated.

One advantage of this approach is that existing tools for static diagnosis can be reused. The main disadvantage is, however, that a complete fault list for the circuit has to be generated which can become huge for complex fault models. Since the DfT measures mentioned above do not necessarily support diagnosis, additional means are rather helpful. For this reason, behavioral assertions are synthesized into structures on chip not only for design debug and removed afterwards but also for chip diagnosis and kept for manufacturing.

V. Debug and Online Test

As mentioned in the previous section, assertions support pattern generation for both debug and diagnosis. For this purpose in addition to describing assertions, mapping them efficiently to hardware has to be considered. The research and development community has spared no effort into developing optimized synthesis mechanisms to accommodate for assertions in the designs. [54] addresses the problem of HW/SW co-verification after system synthesis and presents a novel assertion synthesis technique, which converts system-level assertions to implementation-level assertions automatically during the synthesis process. This way, assertions defined in system-level can be reused in lower levels of abstraction for run-time system verification. On the other hand, [55] provides an algorithm that automates the synthesis process of assertion monitors from a high level visual specification language. [56] provides a hardware environment and protocol for using synthesized assertions effectively in the verification and debug processes.

Again, if these assertions are kept in the circuit structure for manufacturing, they are a useful means for online test.

VI. Conclusions

Today's complex designs require special structures to allow validation and verification. These structures, called synthesizable assertions, are also useful later in the development and system lifecycle, for debug, diagnosis, online and offline testing. Moreover, not only the structures but also the validation and debug patterns generated for these structures support test and diagnosis.

Acknowledgment

The authors would like to thank Stefan Holst for contributing parts of his diploma work to this paper.

Part of the work has been funded by the DFG under contract WU 245/3-2.

References

[1] K. Chen, "Assertion-based verification for soc designs," in *Proceedings. 5th International Conference on ASIC, Vol. 1*, 2003, pp. 12–15.

[2] R. Klein and T. Piekarz, "Accelerating functional simulation for processor based designs," *Mentor Graphics Corporation*, 2005, white paper.

[3] T. Fitzpatrick, "Realizing advanced functional verification with questa," *Mentor Graphics Corporation*, 2005, white paper.

[4] C.-C. Yen, J.-Y. Jou, and K.-C. Chen, "A divide-and-conquer-based algorithm for automatic simulation vector generation," *IEEE Design and Test of Computers*, vol. 21, no. 2, pp. 111–120, 2004.

[5] J. Yuan, K. Albin, A. Aziz, and C. Pixley, "Simplifying boolean constraint solving for random simulation-vector generation." in *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, 2002, San Jose, California, USA, November 10-14, 2002*, 2002, pp. 123–127.

[6] Synopsys, "Constrained-random test generation and functional coverage with vera," *Synopsys, Inc.*, 2003, white paper.

[7] Y.-I. Kim and C.-M. Kyung, "Tpartition: Testbench partitioning for hardware-accelerated functional verification," *IEEE Design and Test of Computers*, vol. 21, no. 6, pp. 484–493, November/December 2004.

[8] B. L. Hutchings and B. E. Nelson, "Unifying simulation and execution in a design environment for fpga systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 1, pp. 201–205, 2001.

[9] W. Ecker and R. Henftling, "Accelerated testbenches," *IEEE Proceedings of 5th International Conference on ASIC*, vol. 1, p. 11, October 2003.

[10] R. Henftling, A. Zinn, M. Bauer, W. Ecker, and M. Zambaldi, "Platform-based testbench generation." in *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, 2003, pp. 11 038–11 043.

[11] M. Zambaldi, W. Ecker, R. Henftling, and M. Bauer, "A layered adaptive verification platform for simulation, test, and emulation," *IEEE Design and Test of Computers*, vol. 21, no. 6, pp. 464–471, November/December 2004.

[12] E. Sax, G. Krampl, M. Miegler, and S. Sattler, "A common mixed-signal testbench for design verification and test," in *Proc. 5th International Mixed-Signal Testing Workshop*, June 1999.

[13] Cadence, "Incisive palladium family with incisive xe software," *Cadence Design Systems, Inc.*, 2005, datasheet.

[14] Prodesign, "Chipit platinum edition," *Prodesign GmbH*, 2005, datasheet.

[15] Mentor, "Vstationpro high performance system verification," *Mentor Graphics Corporation*, 2003, datasheet.

[16] D. S. Brahme, S. Cox, J. Gallo, M. Glasser, W. Grundmann, C. N. Ip, W. Paulsen, J. L. Pierce, J. Rose, D. Shea, and K. Whiting, "The transaction-based verification methodology," *Cadence Design Systems, Inc.*, August 2000, technical Report Num. CDNL-TR-2000-0825.

[17] Accellera, "Standard co-emulation modeling interface (sce-mi) reference manual," *Accellera*, 2003, version 1.0.

[18] R. Newell and R. Schutten, "Bringing assertions into hardware-assisted verification," in *EETimes*, April 2004.

[19] K. Ara and K. Suzuki, "A proposal for transaction-level verification with component wrapper language." in *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, 2003, pp. 20 082–20 087.

[20] W. Z. Hai and Y. Y. Zheng, "The improvement for transaction level verification functional coverage," in *IEEE Proceedings of International Symposium On Circuits and Systems*, 2005, pp. 5850–5853.

[21] K. Ara and K. Suzuki, "Fine-grained transaction-level verification: Using a variable transactor for improved coverage at the signal level," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 24, no. 8, pp. 1234–1240, August 2005.

[22] S. Regimbal, Y. Savaria, and G. Bois, "Verification strategy determination using dependence analysis of transaction-level models." in *Proceedings of the 4th IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC'04), 19-21 July 2004, Banff, Alberta, Canada*, 2004, pp. 87–92.

[23] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded deterministic test for low cost manufacturing test," in *Proc. of IEEE International Test Conference*, October 2002, pp. 301–310.

[24] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Scott, and B. Koenemann, "Extending opmisr beyond 10x scan test efficiency," *IEEE Design and Test*, vol. 19, no. 5, pp. 65–73, October 2002.

[25] K. Datta and P. P. Das, "Assertion based verification using hdvl." in *17th International Conference on VLSI Design (VLSI Design 2004), with the 3rd International Conference on Embedded Systems Design, 5-9 January 2004, Mumbai, India*, 2004, pp. 319–325.

[26] Accellera, "Systemverilog 3.1a language reference manual," *Accellera Organization*, 2004.

[27] S. Bailey, E. Marschner, J. Bhasker, J. Lewis, and P. J. Ashenden, "Improving design and verification productivity with vhdl-200x." in *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, 2004, pp. 332–335.

[28] A. Habibi and S. Tahar, "On the extension of systemc by systemverilog assertions," in *IEEE Proceedings of Canadian Conference on Electrical and Computer Engineering*, May 2004, pp. 1869–1872.

[29] Synopsys, "Assertion-based verification," *Synopsys, Inc*., March 2003, white paper.

[30] F. De Paula, C. Coelho, H. Foster, J. Nacif, J. Tompkins, A. Fernandes, and D. da Silva, "Refactoring digital hardware designs with assertion libraries," in *IEEE Proceedings of 8th International Workshop on High Level Design Validation and Test*, 2003, pp. 37–42.

[31] P. Yeung, "The four pillars of assertion-based verification," *Mentor Graphics Corporation*, 2004, white paper.

[32] A. Habibi and S. Tahar, "Towards an efficient assertion based verification of systemc designs," in *IEEE Proceedings of 9th International Workshop on High Level Design Validation and Test*, November 2004, pp. 19–22.

[33] A. Dahan, D. Geist, L. Gluhovsky, D. Pidan, G. Shapir, Y. Wolfsthal, L. Benalycherif, R. Kamdem, and Y. Lahbib, "Combining system level modeling with assertion based verification," in *IEEE Proceedings of 6th International Symposium on Quality of Electronic Design*, March 2005, pp. 310–315.

[34] F. Fallah, P. Ashar, and S. Devadas, "Simulation vector generation from hdl descriptions for observability-enhanced statement coverage." in *DAC*, 1999, pp. 666–671.

[35] I. Ugarte and P. Sanchez, "Functional vector generation for assertion-based verification at behavioral level using interval analysis," in *IEEE Proceedings of 8th International Workshop on High Level Design Validation and Test*, 2003, pp. 102–107.

[36] D. Wang and J. Levitt, "Automatic assume guarantee analysis for assertion-based formal verification," in *IEEE Proceedings of the Asia and South Pacific Design Automation Conference*, January 2005, pp. 561–566.

[37] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture." in *Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003*, 2003, pp. 338–342.

[38] A. K. Majhi, G. Gronthoud, C. Hora, M. Lousberg, P. Valer, and S. Eichenberger, "Improving diagnostic resolution of delay faults using path delay fault model," in *Proc. 21st IEEE VLSI Test Symposium*, May 2003, p. 345.

[39] S. Padmanaban and S. Tragoudas, "Non-enumerative path delay fault diagnosis," in *Proc. Design, Automation and Test in Europe*, March 2003, pp. 322–327.

[40] A. Krstic, L. Wang, K. Cheng, J. Liou, and M. Abadir, "Delay defect diagnosis based upon statistical timing models - the first step," in *Proc. Design, Automation and Test in Europe*, March 2003, p. 10328.

[41] A. Krstic, L.-C. Wang, K.-T. Cheng, and T. M. Mak, "Diagnosis-based post-silicon timing validation using statistical tools and methodologies." in *Proceedings 2003 International Test Conference (ITC 2003), Breaking Test Interface Bottlenecks, 28 September - 3 October 2003, Charlotte, NC, USA*, 2003, pp. 339–348.

[42] A. Krstic, L. C. Wang, K. Cheng, J. J. Liou, and T. M. Mak, "Enhancing diagnosis resolution for delay defects based upon statistical timing and statistical fault models," in *ACM/IEEE Proceedings Design Automation Conference*, June 2003, pp. 668–673.

[43] Y. Huang, W. Cheng, C. Hsieh, H. Tseng, A. Huang, and Y. Hung, "Intermittent scan chain fault diagnosis based on signal probability analysis," in *Proc. Design, Automation and Test in Europe*, February 2004, p. 21072.

[44] C. Hora, R. Segers, S. Eichenberger, and M. Lousberg, "An effective diagnosis method to support yield improvement," in *Proc. of IEEE International Test Conference*, October 2002, pp. 260–269.

[45] ——, "On a statistical fault diagnosis approach enabling fast yield ramp-up," in *IEEE European Test Workshop*, May 2002, p. 193.

[46] C. Metra, M. Favalli, and B. Ricco, "Self-checking detection and diagnosis of transient, delay, and crosstalk faults affecting bus lines," *IEEE Transactions on Computers*, vol. 49, no. 6, pp. 560–574, June 2000.

[47] B. Chess, B. Lavo, F. Ferguson, and T. Larrabee, "Diagnosis of realistic bridging faults with single stuck-at information," in *1995 International Conference on Computer-Aided Design (ICCAD '95)*, 1995, pp. 185–192.

[48] S. Venkataraman and W. K. Fuchs, "A deductive technique for diagnosis of bridging faults," in *Proc. IEEE International Conference on Computer-Aided Design*, November 1997, pp. 562–567.

[49] M. Renovell, P. Huc, and Y. Bertrand, "The concept of resistance interval: A new parametric model for realistic resistive bridging fault," in *Proc. 13th IEEE VLSI Test Symposium*, May 1995, pp. 184–189.

[50] Chess, B. Larrabee, and T., "Creating small fault dictionaries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 3, pp. 346–356, March 1999.

[51] H.-J. Wunderlich, "From embedded test to embedded diagnosis," in *Proceedings of the European Test Symposium 2005, Tallinn, Estonia*, 2005.

[52] S. Venkataraman and S. B. Drummonds, "Poirot: a logic fault diagnosis tool and its applications." in *Proceedings IEEE International Test Conference 2000, Atlantic City, NJ, USA, October 2000*, 2000, pp. 253–262.

[53] Y. Gong and S. Chakravarty, "On adaptive diagnostic test generation," in *Proc. IEEE International Conference on Computer-Aided Design*, November 1995, p. 181.

[54] S. Hessabi, A. M. Gharehbaghi, B. H. Yaran, and M. Goudarzi, "Integrating assertion-based verification into system-level synthesis methodology," in *IEEE Proceedings of 16th International Conference on Microelectronics*, 2004, pp. 232–235.

[55] A. A. Gadkari and S. Ramesh, "Automated synthesis of assertion monitors using visual specifications." in *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, 2005, pp. 390–395.

[56] K. Peterson and Y. Savaria, "Assertion-based on-line verification and debug environment for complex hardware systems," in *IEEE Proceedings of International Symposium on Circuits and Systems*, 2004, pp. 685–688.