Post-Silicon Validation
Opportunities, Challenges and Recent Advances

Subhasish Mitra
Dept. of EE and Dept. of CS
Stanford University
Stanford, CA, USA
subh@stanford.edu

Sanjit A. Seshia
Dept. of EECS
University of California
Berkeley, CA, USA
sseshia@eecs.berkeley.edu

Nicola Nicolici
Dept. of ECE
McMaster University
Hamilton, ON, Canada
nicola@ece.mcmaster.ca

ABSTRACT

Post-silicon validation is used to detect and fix bugs in integrated circuits and systems *after manufacture*. Due to sheer design complexity, it is nearly impossible to detect and fix all bugs before manufacture. Post-silicon validation is a major challenge for future systems. Today, it is largely viewed as an art with very few systematic solutions. As a result, post-silicon validation is an emerging research topic with several exciting opportunities for major innovations in electronic design automation. In this paper, we provide an overview of the post-silicon validation problem and how it differs from traditional pre-silicon verification and manufacturing testing. We also discuss major post-silicon validation challenges and recent advances.

Categories and Subject Descriptors
B 7.2 Design Aids – Verification, B8.1 Reliability, Testing and Fault-Tolerance, B 8.2 Performance Analysis and Design Aids

General Terms
Reliability, Verification.

Keywords
Post-silicon validation.

1. Opportunities

Ensuring correct operation despite rising levels of design complexity has been a major focus of research and development since the dawn of digital system design. These efforts led to remarkable advances in the theory and practice of design verification and manufacturing testing of digital systems over several decades. Post-silicon validation of overwhelmingly complex systems of the future is an emerging field of research with exciting opportunities for major innovations.

Post-silicon validation involves operating one or more manufactured chips in actual application environments to validate correct behaviors over specified operating conditions. The objective is to ensure that no bugs escape to the field. According to several industry reports, post-silicon validation is becoming significantly difficult and prohibitively expensive because existing techniques cannot cope with the sheer complexity of future systems [Abramovici 06, Patra 07, Yeramilli 06]. Post-silicon validation involves four major steps:

1. *Detecting a problem* by running test programs ranging from random instruction sequences to end-user applications, e.g., operating systems, games, and scientific applications, until a system failure occurs (e.g., system crash, segmentation fault or exceptions).

2. *Localizing the problem* to a small region from the system failure. For example, a program crash on a complex microprocessor might be localized to errors produced by an instruction scheduler under a certain application workload. The stimulus that exposes the bug, e.g., particular 10 lines of code from some application, is also important.

3. *Identifying the root cause* of the problem. For example, a bug may be caused by power-supply noise slowing down some circuit paths resulting in errors at the outputs of some design block only for a certain input sequence.

4. *Fixing or bypassing the problem* by patching, circuit editing or, as a last resort, re-spinning using a new mask.

Post-silicon validation has significant overlap with pre-silicon design verification and manufacturing (or production) testing. Traditionally, most hardware design bugs are detected during pre-silicon verification, and manufacturing defects are targeted by manufacturing testing. While both manufacturing testing and pre-silicon verification continue to be essential, post-silicon validation is becoming extremely important because of several unique aspects (Table 1.1):

1. We cannot rely on pre-silicon design verification alone to detect all design bugs. Simulation is several orders of magnitude slower than actual silicon. Formal verification is very useful for certain situations, such as the verification of individual arithmetic units or protocols, it faces scalability challenges for full chip-level verification. Hence, bugs that escape pre-silicon design verification are often detected during post-silicon validation.

2. In advanced technologies, several interactions between a design and the electrical state of a system are becoming significant, e.g., signal integrity (cross-talk and power-supply noise), thermal effects, and process variations. Such interactions can result in incorrect behaviors and are often referred to as *electrical bugs*. Accurate modeling of all these physical effects is usually very difficult during pre-silicon design verification.

3. Unlike manufacturing defects, post-silicon bugs may be caused by subtle interactions between a design and physical effects (the so-called electrical bugs) or by design errors (the so-called *logic bugs*). It may be very difficult to create accurate and effective fault models for such bugs.

4. A primary reason behind the success of today's manufacturing testing techniques is the existence of test metrics such as single-stuck-at coverage, transition fault coverage and N-detect coverage, and experimental demonstration of the effectiveness of such metrics using actual chips, e.g., work by [Ma 95] and others. Such metrics enable automatic test pattern generation and fault simulation. For pre-silicon design verification, such metrics are far less standardized. Syntactic metrics, e.g., code coverage, and semantic metrics, e.g., covering assertion goals, constitute an integral part of the verification sign-off process. There exist new opportunities for establishing coverage metrics for post-silicon validation.

5. Unlike manufacturing testing where the primary objective is to detect defects, post-silicon validation involves localizing, root-causing and fixing bugs. Bug localization generally dominates post-

silicon validation effort and costs [Josephson 06]. While defect diagnosis techniques may be somewhat applicable for bug localization, most defect diagnosis techniques rely on scan design for testability (DFT) which enables a sequential circuit to be treated as a combinational circuit in test mode. Such opportunities may not be available for bug localization during post-silicon validation. Moreover, unlike defect diagnosis which targets manufacturing yield improvement, the presence of one or more bugs may prevent detection of errors caused by other bugs. Such bugs are also referred to as *blocking bugs* because they may slow down the design / validation cycle [Anis 08, Keshava 10].

Table 1.1: Qualitative comparison of pre-silicon verification vs. manufacturing testing vs. post-silicon validation.

Pre-silicon verification	**Manufacturing testing**	**Post-silicon validation**
Excellent controllability and observability because any signal can be accessed	Controllability and observability primarily through scan DFT	Insufficient controllability and observability due to limited access to internal signals. Scan DFT useful for certain cases when a failure caused by a bug is repeatable.
Complex physical effects difficult to model	Several defect models exist	Accounts for signal-integrity, process variations, non-determinism
Simulation of full-chip designs very slow; formal verification only selectively applicable	Generally very fast (few seconds to minutes per chip)	Silicon speed orders of magnitude faster than simulation
Some metrics exist (e.g., code coverage, assertion coverage, mutation coverage)	Test Metrics (e.g., stuck-at, transition, N-detect coverage) widely used	Coverage metrics for post-silicon validation: Open research question
Bug fixing inexpensive	Bug fixing not the primary objective	Bug fixing can be expensive

In spite of its unique challenges, post-silicon validation research can be inspired by the successes in manufacturing testing and design verification, as discussed next.

Manufacturing Testing

Examples of major successes in manufacturing testing include:

1. The concept of structural testing [Eldred 59] and scan design for testability [Eichelberger 77, Williams 73] that are used in almost all integrated circuits today.

2. Quick adoption of test compression techniques, e.g., Illinois scan [Hamzaoglu 99], reseeding techniques [Koenemann 91], and X-Compact [Mitra 02, 04], pointing to the success of systematic and structured methods in overcoming manufacturing testing challenges for complex systems.

3. Various flavors of self-test techniques, including Built-In Self-Test (BIST) (pseudo-random and several enhancements) for general designs ([Bardell 87] and several others), and native-mode testing / software-based self-test [Krstic 02, Parvathala 02, Shen 98] targeting microprocessors.

4. A wide variety of test metrics (and test experiments demonstrating their effectiveness) that enabled automatic test pattern generation (ATPG), fault simulation, and coverage estimation (as discussed earlier).

Design Verification

Since the 1970s, several fundamental ideas in formal and semi-formal verification have made their way from theoretical concept to industrial practice. In the context of this paper, the arguably most important idea is that of formal specifications – *assertions*. Temporal logic [Pnueli 77] and similar automata-based specification languages have made their way into standardized industrial use [Vardi 08]. Assertions are central to post-silicon validation as they provide a precise method of specifying application requirements and system functionality that could be adversely affected by post-silicon bugs.

Algorithmic verification techniques, based on advances in Boolean reasoning such as *Boolean Satisfiability* (SAT) [Malik 09] and *Binary Decision Diagrams* (BDDs) [Bryant 86], are also central to post-silicon validation. Perhaps one of the biggest industrial successes of formal hardware verification is *combinational equivalence checking*. In addition, recent advances in SAT-based algorithms and circuit simplification are scaling up *sequential equivalence checking* greatly, as demonstrated by the ABC system [Brayton 10]. Going beyond equivalence checking, *model checking* [Clarke 00], i.e., deciding whether a system satisfies a property specified usually in temporal logic, is now a key component of all industrial formal verification tools. Formal techniques are often combined with traditional random simulation methods. Finally, *theorem proving* methods are also standard industrial practice today, particularly in microprocessor design and verification (e.g., [Moore 98]). One of the major drivers for formal verification going forward, both using theorem proving and model checking, will be *satisfiability modulo theories* (SMT) solvers [Barrett 09].

It is clear from the above discussions that fundamental research enabled significant progress in the adoption of structured methods for manufacturing testing and design verification. With growing complexity of post-silicon validation, our hope is that new ideas will emerge that will transform post-silicon validation from skilled art practiced by a few experienced engineers to a discipline with strong foundations enabled by structured approaches and design automation. Such new techniques can have far reaching impact on other fields related to post-silicon validation. Examples include reliable system design, embedded systems, and software test and verification.

2. Challenges

There are numerous challenges in post-silicon validation. We do not attempt to present a comprehensive survey here. Instead, we highlight a few important ones.

1. *Failure reproduction:* This step involves returning the hardware to an error-free state, and re-executing the failure-causing stimulus (including instruction sequences, interrupts, and operating conditions) to reproduce the same failure. Unfortunately, many electrical bugs are very hard to reproduce. The difficulty of bug reproduction is exacerbated by the presence of asynchronous I/Os,

and multiple clock domains. Techniques to make failures reproducible [Heath 04, Sarangi 06, Silas 03] are often intrusive to system operation, and may not expose bugs.

2. *System-level simulation:* In order to obtain "golden responses," i.e., correct signal values for every clock cycle for the entire system (e.g., processor and all peripheral devices on the board), one must run expensive system-level simulation. Running system-level simulation can be 7-8 orders of magnitude slower than actual silicon. In addition, expensive external logic analyzers are required to record all signals values that enter and exit the system through external pins [Silas 03].

Due to the above factors, a functional bug typically takes hours to days to be localized vs. electrical bugs that require days to weeks and more expensive equipment [Josephson 01].

3. *Coverage metrics:* As noted earlier, while coverage metrics are well-established in manufacturing test, they are less standard in pre-silicon design verification. Several methods exist for measuring pre-silicon verification coverage, including code coverage (e.g., statement, branch, path coverage in an RTL description), FSM coverage (e.g., transition or state coverage), assertion coverage, and mutation-based coverage (e.g., checking whether an injected bug is caught during verification). Quantifying coverage of post-silicon validation tests is very challenging due to limited controllability and observability (hence, it is harder to mutate a design or monitor an assertion failure).

4. *Test Patterns for post-silicon validation:* Several recent techniques for improving real-time observability during post-silicon validation, e.g., using assertion checkers and optimized embedded logic analyzers, enable monitors from pre-silicon verification testbenches to be reused during post-silicon validation. On-chip generation of post-silicon validation tests can potentially enable reuse of controllability of testbenches used for pre-silicon verification (structured and scalable, yet slow) during post-silicon validation (fast, yet ad-hoc and labor-intensive). Note that, there are key differences between such on-chip generation of post-silicon validation tests vs. logic BIST for manufacturing testing. This is because most logic BIST techniques rely on scan DFT and target manufacturing test metrics.

3. Recent Advances

In this section, we present a brief overview of recent results in the field of post-silicon validation. This survey is by no means exhaustive. Our goal is just to highlight some results that could provide a basis for further work in this field. We begin with challenges related to bug localization because it is typically the most expensive step in post-silicon validation.

Post-silicon bug localization for processor cores

A new technique called *IFRA* (Instruction Footprint Recording and Analysis) [Park 08, 09, 10a, 10b] has been demonstrated to be highly effective in overcoming major challenges associated with a very expensive step in post-silicon validation of processors – pinpointing a bug location and the instruction sequence that exposes the bug from a system failure, such as a crash.

Figure 1 presents an overview of IFRA. Special on-chip recorders, inserted in a processor during design, collect *instruction footprints* – special information about flows of instructions, and what the instructions did as they passed through various microarchitectural blocks of the processor. The recording is done concurrently during the normal operation of the processor in a post-silicon system validation setup. Upon detection of a system failure, the recorded information is scanned out and analyzed offline for bug localization. Special self-consistency-based program analysis techniques, together with the test program binary of the application executed during post-silicon validation, are used for this purpose.

Major benefits of using IFRA over traditional techniques for post-silicon bug localization are: 1. It does not require full system-level reproduction of bugs, and, 2. It does not require full system-level simulation. Hence, it can overcome major hurdles that limit the scalability of traditional post-silicon validation methodologies. Simulation results on a complex Alpha 21264-like superscalar processor demonstrate that IFRA is effective in localizing electrical bugs with 96% accuracy at 1% chip-level area impact.

A new Bug Localization Graph (BLoG) framework [Park 10b] enables systematic application of IFRA to new processor microarchitectures with reduced engineering time and less expert knowledge. Results obtained from an industrial microarchitectural simulator modeling a state-of-the-art complex commercial microarchitecture (Intel Nehalem, the foundation for the Intel Core™ i7 and Core™ i5 processor families) demonstrate the effectiveness of IFRA in localizing electrical bugs in complex microarchitectures with 90% accuracy (Fig. 2 taken from [Park 10b]).

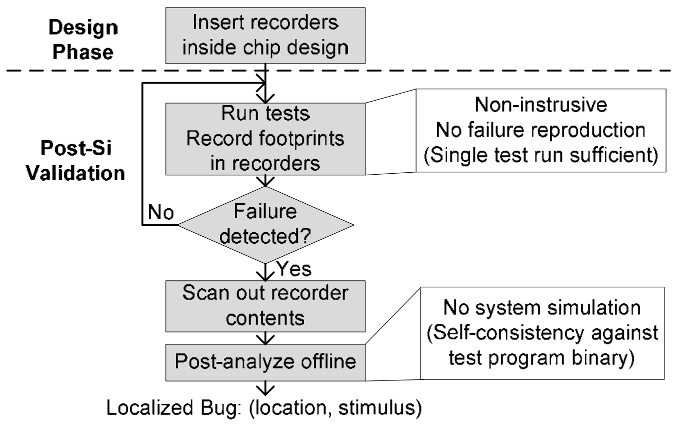
Design Phase
Insert recorders inside chip design
Post-Si Validation
Run tests Record footprints in recorders
Non-instrusive No failure reproduction (Single test run sufficient)
No
Failure detected?
Yes
Scan out recorder contents
Post-analyze offline
No system simulation (Self-consistency against test program binary)
Localized Bug: (location, stimulus)

Figure 1. Post-silicon bug localization flow using IFRA.

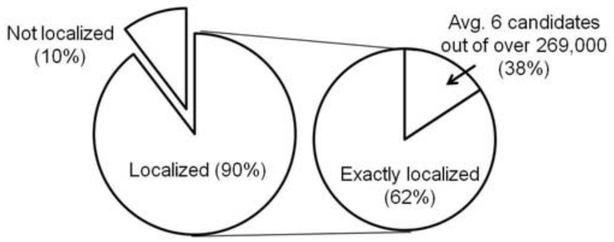

Figure 2. BLoG-assisted IFRA results obtained from an industrial simulator modeling the Intel Nehalem microarchitecture (details in [Park 10b]).

Observability enhancement for post-silicon validation

The IFRA approach discussed earlier utilizes the structured architecture of a processor to perform bug localization without requiring system-level simulation or failure reproduction. For post-silicon bug localization in general designs, the debug infrastructure must collect real-time response data in an effective manner for efficient failure reproduction and root-cause analysis. The amount of data that can be collected during a single post-silicon validation run is ultimately limited by the capacity of on-chip trace buffers. Data compression techniques can increase the amount of data that may be collected in a single post-silicon validation run by 20-30% [Anis 07]. Moreover, special state restoration (or data expansion or visibility enhancement) techniques can further enhance the amount of information made available to the post-silicon validation engineer [Ko 08a, Liu 09].

These techniques are currently applicable for logic bugs, and understanding how to extend these approaches for electrical bugs is a key open challenge. Another technique for enhancing observability is to use multiple trace buffers with programmable priorities. [Ko 08b] presents a case study of a video decoder demonstrating how dynamic sharing (i.e., during post-silicon validation) of trace buffers can double the amount of acquired data when compared to static trace buffer allocation.

Formal methods for error localization

Assertions can play an important role in localizing post-silicon bugs, especially as we move from processor cores to general designs. An obvious approach is to write assertions on module interfaces, and then synthesize on-chip monitors to check whether those assertions are satisfied or violated. This output may be made available in the event of a failure. The failing assertions can then help localize the failure to module boundaries. The challenge with this approach, though, is to obtain *relevant assertions* in the first place. Very recent work [Li 10] has shown how formal assertions can be mined automatically simulation traces in a scalable manner and can be effectively used in localizing errors (caused by bugs) to module boundaries. The overall idea, illustrated in Fig. 3, is to find assertions that are satisfied by correct system traces (e.g., recorded pre-silicon), but not by the error trace. Such distinguishing assertions are then ranked according to some metric, such as time of first failure. The location of the bug is then hypothesized to be the modules with the highest-ranked distinguishing assertions.

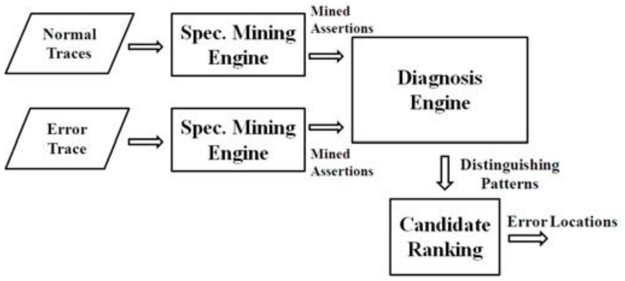

Figure 3. Assertion mining for failure diagnosis.

The following representative result from [Li 10] shows the promise of this assertion-based bug localization approach. For an experimental MIPS core with over 20,000 signals, the approach correctly localized single-bit and multiple-bit flips in the logic to the single faulty module out of a total of 278 modules. While this work is only a first step, it can be very effective especially when combined with reconfigurable on-chip infrastructure, e.g., [Abramovici 06], for application-specific error monitors (that may be software-controlled).

Formal methods can also be useful for system *replay*. Specifically, algorithms developed for model checking can be useful during the root-cause step as well. For example, one challenge during the root-cause step is to reconstruct an error trace backward from a failing state. The problem is similar to that of performing backward reachability analysis in model checking, where, starting from an error state, we seek to find a trace that begins in some valid initial state and which can end in that error state. This analogy has been explored recently in the Backspace technique [de Paula 08]. In essence, the Backspace approach involves recording a crash state, computing its expected predecessor state (based on RTL), setting that state as a breakpoint, and then repeating this procedure so as to reconstruct the error trace backwards. This approach has shown promise, generating diagnostic traces for two open-source processor cores: a Motorola 68HC05 and an Intel 8051. In both cases, the Backspace system could reconstruct the error traces for hundreds of cycles backwards from a crash state.

Detecting a problem

For error detection, the following two questions must be answered: (i) what input stimulus should be applied to the circuit? and, (ii) how to detect if a failure has occurred?

While the first problem is still largely open, *online in-hardware assertion checking* has shown promise in answering the second. In recent years assertions have been proposed for in-system hardware validation and debugging [Bayazit 05] and [Boule 05]. The basic idea is to reuse the assertions written in a software environment during the pre-silicon phase, map them onto hardware and let them run in the background to detect errors at runtime. However, two challenges arise. The first has to do with signal locality. Pre-silicon assertions can reference signals that are not spatially nearby in actual silicon. Due to this, if one must use pre-silicon assertions for monitoring, the assertions must be suitably decomposed into localized assertions. The second challenge lies in minimizing the area investment. The number of assertions to be placed in hardware, as well as the locations where they may be placed to enable effective sharing between different hardware blocks, are open questions. An equally important problem is determining compact input sequences that can improve bug detection and coverage using hardware assertion checkers.

Root cause identification

Once a bug is localized to a design block from a system failure, scan chains are extensively used for finding the root cause. System states are scanned out at pre-programmed clock cycles and compared against golden signatures in consecutive clock cycles in order to identify the failing flip-flops and failing cycles. Subsequently, logic cone analysis is used to identify the erroneous signals. While such an approach can benefit from existing fault diagnosis techniques, the sequential nature of the problem (unlike scan-based manufacturing testing) creates several complications. A key challenge is that system states are not necessarily deterministic, i.e., it may not be possible to determine expected logic values of all state bits by simulation. This is due to the presence of several sources of unknown logic values (X's). Examples include multiple clock domains, uninitialized embedded memory blocks or mixed-signal blocks. X's are known to complicate manufacturing testing as well [Mitra 02, 04]. A technique known as latch divergence analysis [Dahlgren 03] may be used to "filter-out" state deviations during post-silicon validation. The information obtained from these "filtered" states can be subsequently processed using modified backtrace procedures for logic cone analysis [Caty 05].

When debugging speedpaths, it is common to collect responses at multiple clock frequencies where the clock signal is selectively stretched/shrunk only for certain logic cones or selected paths. This can be achieved using special clock tuning elements that inserted in the design [Naffziger 06]. Accelerating speedpath debugging through automated insertion and configuration of clock tuning elements, as well as the analysis of failure traces are open challenges in this context.

Bug fixing

After a bug has been localized, analyzed, and root-caused, one can attempt to patch the problem in the design itself without a re-spin. For microprocessors, one way to achieve this goal is through microcode patches or special design techniques such as the Field-Repairable Control Logic (FRCL) approach [Wagner 08]. State sequences that trigger a bug are recorded during detection and localization phases and subsequently used for in-field patches. State matcher circuitry is employed to monitor the states that are known to trigger the bug. When the matcher reports a match, the processor pipeline is flushed and instruction execution is resumed in a low-performance serial mode (i.e., with only one instruction at a time in the pipeline). After the bug is bypassed, the microprocessor resumes execution in its normal high-performance mode (i.e., with multiple instructions at a time in the pipeline). The assumption is that most bugs are introduced by complex interactions between instructions and a serial low-performance serial execution mode may be generally expected to be bug-free.

Another approach called FogClear [Chang 08] enables post-silicon metal fixes allowed by engineering change order (ECO) routing and spare cell insertion. [Chang 08] show that by pre-placing spare cells in 70% of the unused regions in the layout, FogClear can repair 70% of the injected functional errors.

Error-Resilient System Design

While mainstream post-silicon validation activities focus on detecting, localizing and fixing post-silicon bugs before product shipment, the ultimate objective may be to design robust systems that are resilient to design errors. There exists a large body of literature on robust system design and fault-tolerant computing techniques in this context (e.g., [Iyer 05, Li 09, Mitra 10, Siewiorek 98] and several others). For such resilience techniques to be cost-effective, it is essential to perform pre-silicon analysis to identify the most vulnerable components of a system. Formal methods offer a promising approach for this purpose for two reasons. First, formal assertions can precisely capture "critical" application-level requirements on the design that must hold at all times. Second, formal verification techniques can perform exhaustive search over the faulty behavior space as well as the input space. In this context, the *verification-guided error resilience* (VGER) approach [Seshia 07] has demonstrated how formal specifications and model checking can be used to perform such analysis for transient errors. VGER can pinpoint components (e.g., latches or gates) that are most vulnerable to soft errors in the sense that a soft error in those components will cause the system/application-level requirements to fail severely. Such analysis can be useful for diagnosis – for pinpointing the most likely areas in which a post-silicon error might occur. It is also very useful for synthesizing a low power robust design: in experiments on an implementation of the SpaceWire protocol, the VGER approach showed how a robust design could be synthesized for one-fifth the power overhead of a traditional fault-tolerance technique.

4. Conclusion

Post-silicon validation is an emerging research topic with several outstanding challenges that can create opportunities for exciting research contributions. Systematic techniques for post-silicon validation together with coverage metrics for quantifying the effectiveness of various post-silicon validation approaches can transform this field from an art requiring considerable skill and experience to a discipline with strong foundations enabled by structured approaches and design automation.

Acknowledgment

This work is supported in part by the Semiconductor Research Corporation (SRC), the Gigascale Systems Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), an SRC entity, the National Science Foundation (NSF), and the National Science and Engineering Research Council (NSERC) of Canada.

References

[Abramovici 06] Abramovici, M., P. Bradley, K. N. Dwarakanath, P. Levin, G. Memmi and D. Miller, "A Reconfigurable Design-for-Debug Infrastructure for SoCs," *Proc. Design Automation Conf.*, pp. 7-12, 2006.

[Anis 07] Anis, E., and N. Nicolici, "On Using Lossless Compression of Debug Data in Embedded Logic Analysis," *Proc. Intl. Test Conf.*, 2007.

[Anis 08] Anis E., and N. Nicolici, "On By-passing Blocking Bugs during Post-silicon Validation," *Proc. European Test Symp.*, pp. 69-74, 2008.

[Bardell 87] Bardell, P.H., W.H. McAnney and J. Savir, *Built-In Test for VLSI: Pseudo-random Techniques*, John Wiley & Sons, 1987.

[Barrett 09] Barrett, C., R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability Modulo Theories," *Handbook of Satisfiability*, IOS Press, 2009.

[Bayazit 05] Bayazit, A.A., and S. Malik, "Complementary Use of Runtime Validation and Model Checking," *Proc. Intl. Conf. CAD*, pp. 1052-1059, 2005.

[Boule 05] Boule, M., and Z. Zilic, "Incorporating Efficient Assertion Checkers into Hardware Emulation," *Proc. Intl. Conf. Computer Design*, pp. 221-228, 2005.

[Brayton 10] Brayton, R.K., *et al.*, Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/

[Bryant 86] Bryant, R.E., "Graph-based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, Vol. C-35, No. 8, pp. 677-691, Aug. 1986.

[Caty 05] Caty, O., P. Dahlgren and I. Bayraktaroglu, "Microprocessor Silicon Debug Based on Failure Propagation Tracing," *Proc. Intl. Test Conf.*, pp. 293-302, 2005.

[Chang 08] Chang, K.H., I.L. Markov and V. Bertacco, "Automating Post-silicon Debugging and Repair," *IEEE Computer*, Vol. 41, No. 7, pp.47-54, July 2008.

[Clarke 00] Clarke, E.M., O. Grumberg and D. Peled, *Model Checking*, MIT Press, 2000.

[Dahlgren 03] Dahlgren, P., P. Dickinson and I. Parulkar, "Latch Divergency in Microprocessor Failure Analysis in Microprocessor Failure Analysis," *Proc. Intl. Test Conf.*, pp. 755-763, 2003.

[de Paula 08] de Paula, F.M., M. Gort, A.J. Hu, S.E. Wilton and J. Yang, "BackSpace: Formal Analysis for Post-Silicon Debug," *Proc. Intl. Conf. Formal Methods in CAD*, 2008.

[Eichelberger 77] Eichelberger, E.B., and T.W. Williams, "A Logic Design Structure for LSI Testability," *Proc. Design Automation Conf.*, pp. 462-468, 1977.

[Eldred 59] Eldred, R.D., "Test Routines based on Symbolic Logic Statements," *Journal ACM*, Vol. 6, No. 1, pp. 33-37, Jan. 1959.

[Hamzaoglu 99] Hamzaoglu, I., and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," *Proc. Intl. Symp. Fault-Tolerant Computing*, pp. 260-267, 1999.

[Heath 04] Heath, M.W., W.P. Burleson and I.G. Harris, "Synchro-Tokens: Eliminating Nondeterminism to Enable Chip-Level Test of

Globally-Asynchronous Locally-Synchronous SoC's," *Proc. Design, Automation and Test in Europe*, pp. 1532-1546, 2004.

[Iyer 05] Iyer, R.K., N. Nakka, Z. Kalbarczyk and S. Mitra, "Recent Advances and New Avenues in Hardware-Level Reliability Support," *IEEE MICRO*, Vol. 25, No. 6, pp. 18-29, Nov.-Dec. 2005.

[Josephson 01] Josephson, D., S. Poehlman and V. Govan, "Debug Methodology for the McKinley Processor," *Proc. Intl. Test Conf.*, pp. 451-460, 2001.

[Josephson 06] Josephson, D., "The Good, the Bad, and the Ugly of Silicon Debug," *Proc. Design Automation Conf.*, pp. 3-6, 2006.

[Keshava 10] Keshava, K., N. Hakim and C. Prudvi, "Post-Silicon Validation Challenges: How EDA and Academia Can Help," *Proc. Design Automation Conf.*, 2010.

[Ko 08a] Ko, H.F., and N. Nicolici, "Automated Trace Signals Identification and State Restoration for Improving Observability in Post-silicon Validation," *Proc. Design Automation and Test in Europe*, pp. 1298-1303, 2008.

[Ko 08b] Ko, H.F., A.B. Kinsman and N. Nicolici, "Distributed Embedded Logic Analysis for Post-silicon Validation of SOCs," *Proc. Intl. Test Conf.*, 2008.

[Koenemann 91] Koenemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc. European Test Conf.*, pp. 237-242, 1991.

[Krstic 02] Krstic, A., W.C. Lai, K.T. Cheng, L. Chen and S. Dey, "Embedded Software-Based Self-Test for Programmable Core-Based Designs," *IEEE Design and Test of Computers*, Vol. 19, No. 4, pp. 18-27, July-Aug. 2002.

[Li 09] Li, Y., Y.M. Kim, E. Mintarno, D.S. Gardner and S. Mitra, "Overcoming Early-Life Failure and Aging Challenges for Robust System Design," *IEEE Design and Test of Computers,* Vol. 26, No. 6, pp. 28-39, Nov.-Dec. 2009.

[Li 10] Li, W., A. Forin and S. A. Seshia, "Scalable Specification Mining for Verification and Diagnosis," *Proc. Design Automation Conf.*, 2010.

[Liu 09] Liu, X., and Q. Xu, "Trace Signal Selection for Visibility Enhancement in Post-Silicon Validation," *Proc. Design Automation and Test in Europe*, pp. 1338-1343, 2009.

[Ma 95] Ma, S.C., P. Franco and E.J. McCluskey, "An Experimental Test Chip to Evaluate Test Techniques: Experimental Results," *Proc. Intl. Test Conf.*, pp. 663-672, 1995.

[Malik 09] Malik, S., and L. Zhang, "Boolean Satisfiability: From Theoretical Hardness to Practical Success," *Communications of the ACM*, Vol. 52, No. 8, pp. 76-82, Aug. 2009.

[Mitra 02] Mitra, S., and K.S. Kim, "X-Compact: An Efficient Response Compaction Technique for Test Cost Reduction," *Proc. Intl. Test Conf.*, pp. 311-320, 2002.

[Mitra 04] Mitra, S., and K.S. Kim, "X-Compact: An Efficient Response Compaction Technique," *IEEE Trans. CAD*, Vol. 23, issue 3, pp. 421-432, March 2004.

[Mitra 10] Mitra, S., "Robust System Design," *Proc. Intl. Conf. VLSI Design*, pp. 434-439, 2010.

[Moore 98] Moore, J., T. Lynch, and M. Kaufmann, "A mechanically checked proof of the AMD5 K86(TM) floating-point division program," *IEEE Trans. Computers*, Vol. 47, No. 9, pp. 913-926, Sept. 1998.

[Naffziger 06] Naffziger, S., B. Stackhouse, T. Grutkowski, D. Josephson, J. Desai, E. Alon and M. Horowitz, "The Implementation of a 2-core, Multi-threaded Itanium Family Processor," *IEEE Journal Solid-State Circuits*, Vol. 41, No. 1, pp. 197-209, Jan. 2006.

[Park 08] Park, S.B., and S. Mitra "IFRA: Instruction Footprint Recording and Analysis for Post-Silicon Bug Localization in Processors," *Proc. Design Automation Conf.*, 2008.

[Park 09] Park, S.B., T. Hong and S. Mitra, "Post-Silicon Bug Localization in Processors using Instruction Footprint Recording and Analysis (IFRA)," *IEEE Trans. CAD*, Vol. 28, Issue 10, pp. 1545-1558, Oct. 2009.

[Park 10a] Park, S.B., and S. Mitra, "Post-silicon Bug Localization for Processors using IFRA," *Communications of the ACM*, Vol. 53, No. 2, pp. 106-113, Feb. 2010.

[Park 10b] Park, S.B., A. Bracy, H. Wang and S. Mitra, "BLoG: Post-Silicon Bug Localization in Processors using Bug Localization Graphs," *Proc. Design Automation Conf*, 2010.

[Parvathala 02] Parvathala, P., K. Maneparambil and W. Lindsay, "FRITS – A Microprocessor Functional BIST Method," *Proc. Intl. Test Conf.*, pp. 590-598, 2002.

[Patra 07] Patra, P., "On the Cusp of a Validation Wall," *IEEE Design and. Test of Computers*, Vol.24, No.2, pp.193-196, March-April 2007.

[Pnueli 77] Pnueli, A., "The Temporal Logic of Programs," *Proc. Symp. Foundations of Computer Science*, pp. 46-57, 1977.

[Sarangi 06] Sarangi, S., B. Greskamp and J. Torrellas, "CADRE: Cycle-Accurate Deterministic Replay for Hardware Debugging," *Proc. Dependable Systems and Networks*, pp. 301-312, 2006.

[Seshia 07] Seshia, S.A., W. Li and S. Mitra, "Verification-Guided Soft Error Resilience," *Proc. Design Automation and Test in Europe*, pp. 1442-1447, 2007.

[Shen 98] Shen, J., and J. A. Abraham, "Native Mode Functional Test Generation for Processors with Applications to Self Test and Design Validation," *Proc. Intl. Test Conf.*, pp. 990-999, 1998.

[Silas 03] Silas, I., I. Frumkin, E. Hazan, E. Mor and G. Zobin, "System-Level Validation of the Intel Pentium M Processor," *Intel Technology Journal*, Vol. 7, No.2., pp. 37-43, May 2003.

[Siewiorek 98] Siewiorek, D.P., and R.S. Swarz, *Reliable Computer Systems: Design and Evaluation*, A.K. Peters, 1998.

[Vardi 08] Vardi, M., "From Church and Prior to PSL," *Proc. 25 Years of Model Checking*, pp. 150-171, 2008.

[Wagner 08] Wagner, I., V. Bertacco and T. Austin, "Using Field-Repairable Control Logic to Correct Design Errors in Microprocessors," *IEEE Trans. CAD*, Vol. 27, Issue 2, pp.380-393, Feb 2008.

[Williams 73] Williams, M.J.Y., and J.B. Angell, "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Trans. Computers*, Vol. C-22, Issue 1, pp. 46-60, Jan. 1973.

[Yerramilli 06] Yerramilli, S., "Addressing Post-Silicon Validation Challenge: Leverage Validation and Test Synergy," Keynote, *Intl. Test Conf.*, 2006.