What is the Path to Fast Fault Simulation?

Miron Abramovici
AT&T Information Systems

Balaji Krishnamurthy
Tektronix Laboratories

Rob Mathews
Zycad

Bill Rogers
University of Texas

Michael Schulz
Technical University of Munich

See next page for additional authors

Proceedings. New Frontiers in Testing, International Test Conference, 1988.
doi: 10.1109/TEST.1988.207796

What is the Path to Fast Fault Simulation? (A Panel Discussion)

Miron Abramovici – AT&T Information Systems[1]
Balaji Krishnamurthy – Tektronix Laboratories[2]
Rob Mathews – Zycad[3]
Bill Rogers – University of Texas[4]
Michael Schulz – Technical University of Munich[5]
Sharad Seth – University of Nebraska[6]
John Waicukauski – IBM Corporation[7]

ABSTRACT

Motivated by the recent advances in fast fault simulation techniques for large combinational circuits, a panel discussion has been organized for the 1988 International Test Conference. This paper is a collective account of the position statements offered by the panelists.

1. Introduction (Balaji Krishnamurthy)

1.1 Preamble: Fault simulation has attracted considerable attention among test engineers over the past few years. This increased focus on fault simulation can be attributed, in part, to the increasing complexity of designs and to the critical and CPU intensive role that testing plays in the design process. However, much of the increased attention to fault simulation stems from the *expanding* role that fault simulation has begun to play in VLSI testing. Fault simulation has been advocated as a tool for addressing a spectrum of issues ranging from analyzing testability [10] to evaluating the effectiveness of random patterns in deterministic test generation [38].

One of the advantages of such intense focus is that a variety of new and distinctly different approaches to the problem have been uncovered. In the past five years, the primary goal of these efforts has been to achieve fast fault simulation of large combinational circuits.

Motivated by this high-toned activity in the area, a panel discussion has been organized for the 1988 International Test Conference to discuss: *''What is the Path to Fast Fault Simulation?''* A group of eminent panelists have been recruited to represent each of several different approaches to the problem. This paper is a collection of position statements offered by each of the panelists together with this introduction written by the moderator.

1.2 Background on Fault Simulation: Informally, fault simulation is the task of determining the set of faults detected by a sequence of test patterns for a given circuit. However, there are a some nuances of this problem that are worth noting. First, there is the perennial distinction between combinational and sequential logic. Although the panel discussion will be oriented towards combinational logic (whose applications include scan-path designs), there is something to be said about algorithms that can be lifted to sequential circuits. Secondly, there is the issue of fault dropping, an acceptable approach when evaluating the test coverage, but not acceptable for developing fault dictionaries. Finally, there is the distinction between identifying the detected faults versus tagging the faults with the output(s) where they are detected – a subtlety of importance in BIST applications.

[1] 1100 East Warrenville Rd., Naperville, IL 60566
[2] PO Box 500, MS 50-662, Beaverton, OR 97077
[3] 1380 Willow Rd., Menlo Park, CA 94025
[4] Dept. of Electrical & Computer Eng., Austin, TX 78712
[5] Institute of CAD, Dept. of Electrical Eng., D-8000 Munich 2, West Germany
[6] Dept. of Computer Science, Lincoln, NE 68588
[7] Route 52, Zip 5A1, Hopewell Junction, NY 12533

The basic approach to fault simulation, known as *single-fault propagation*, consists of injecting the fault and performing the required logic simulation. Note that when the fault is injected, logic simulation need be performed only to the extent that the fault propagates. Consequently, a routine logic simulation algorithm would perform a large amount of unnecessary computations. At the risk of oversimplification, it might be argued that most of the more sophisticated algorithms reported in the early 70's concentrated on weeding out the unnecessary computations. Both the concurrent [37] and deductive [7] fault simulation algorithms fall in this category. (A notable exception, of course, is the parallel-fault fault simulation algorithm [36].) A more detailed account of the earlier approaches to the fault simulation problem can be found in any of the following texts: [9, 15, 28, 31].

Much of the recent work on fault simulation has focused more on the commonalities amongst computations. Notice that when you inject each of two neighboring faults (under a fixed test pattern) and perform the corresponding logic simulation tasks, there is likely to be a lot of commonality in the two computations. Likewise, if you inject the same fault under two different test patterns, once again there is likely to be much topological similarity in the logic simulations. The newer approaches exploit this similarity and identify common computations that do not need to be performed repeatedly. The first approach in this line of reasoning is the (now attributed to folklore) linear-time algorithm for fanout-free circuits [22].

In pursuing this line of reasoning in the 80's reconvergent fanout has been a nemesis of the fault simulation problem. While a number of new techniques for handling reconvergent fanout have been proposed, a linear time solution to the general fault simulation problem is by no means solved. In fact, there seems to be little hope for a provably linear solution to the problem [19]. Nevertheless, many new approaches to the fault simulation problem have been adequately justified empirically.

A comprehensive compilation of these recent attacks on the problem is conspicuously absent in the literature. On the other hand, at this time any such compilation will fast be rendered obsolete. This void was the primary motivation for this panel discussion.

1.3 The Panel: The panel will consist of six members each representing a different approach to fast fault simulation. The panelists were asked to take a somewhat extreme position, maybe even a more extreme position than they would normally advocate. The positions of the six panelists can be grouped into three pairs.

The first pair advocates parallel pattern fault simulation with the distinction being that Waicukauski prefers a clean and simple algorithm while Schulz advocates a judicious collection of intelligent heuristics for increased speed. Observe that both solutions attempt to exploit the commonality of the computations for a single fault under a set of patterns.

The second pair exploits the commonality of computations for neighboring faults under a single pattern. This leads them to analyze the circuit graph and consider the topological constraints. The distinction between the two approaches in this pair is that while Seth conducts an extensive graph-theoretic analysis of the circuit graph, Abramovici opts for a less thorough analysis and in turn, settles for an approximate solution.

The last pair offers drastically different solutions from the previous ones. Rogers points out that in the long run, one must exploit the inherent hierarchy in digital designs. Mathews, on the other hand, advocates the effective use of parallel and special purpose hardware to obtain fast fault simulation. (Note that these two solutions are the only ones that can accommodate sequential logic.)

It should be pointed out that the six camps represented here are not intended to be an exhaustive list of the recent approaches. Further, each panelist represents a school of thought rather than an individual's work. Consequently the work of many other authors is represented here. Nevertheless, there are some omissions which include the statistical approaches to fault simulation [24] (that yield a far less accurate result, albeit for considerably smaller CPU resources) and to the recent symbolic approach to fault simulation stemming from the work of [11].

What follows is a statement of position from each of the six panelists with their individual identity duly noted. It is the hope of this moderator that ensuing from these statements of position will arise a fruitful and lively discussion on the merits and drawbacks of the different approaches, and that this discussion will pave the way for a comprehensive and objective account of the current state of fast fault simulation.

2. Parallel Pattern Fault Simulation (John Waicukauski)

The parallel simulation of patterns is widely recognized as the fastest software fault simulation technique that is currently available. There are many forms of parallel pattern simulation that are practiced [8,13,23,26,27,32], but the method [38] where it is combined with single fault propagation (PPSFP) is particularly attractive and will be the focus of this paper. The results of this fault simulator are exact and are obtained with minimal CPU time without the expense of special purpose hardware or complex software.

The PPSFP Fault Simulator is totally optimized for designs that can be treated as combinational (i.e. LSSD). It is a 2-value, zero delay, 256 pattern-per-pass simulator. Its high performance is due to simulating many patterns in parallel and then performing single fault propagation to determine if a fault is detectable. Single fault propagation minimizes the number of gate calculations needed to determine if a given fault is detectable for a set of 256 patterns. Fault values are calculated beginning at the point of the fault and continue forward only for gates that continue to propagate differences for any of the 256 patterns. This typically reduces the average number of gate calculations per fault per set of 256 patterns to about 10 and is relatively independent of the circuit size.

The fault simulator has truly become the most powerful tool that is used in today's testing. It can evaluate the test coverage of a set of patterns and by identifying undetected faults allowing enhancement of the patterns. Furthermore, whenever a device fails a test, the fault simulator can be used to diagnose the cause of the failure. Finally, an efficient fault simulator may be used as a kind of test generator by identifying patterns that detect faults from a set of random (or weighted random) patterns.

The PPSFP Fault Simulator has been shown to perform all these functions very efficiently. The fault simulation results on the ISCAS benchwork designs have been published [38]. The largest ISCAS design containing 3827 logic gates required only 83 CPU seconds (IBM 3081) to calculate the test coverage of 500 thousand random patterns. Using the PPSFP Fault Simulator to perform failure diagnosis, an experiment was conducted on a structure that contained over 100 thousand logic gates [40]. In each of 10 cases, the diagnosis successfully identified the defect down to a single fault equivalence class in an average of 8 CPU seconds (IBM 3081). Finally, the weighted random pattern (WRP) test generator (currently used by IBM to create tests for its high performance chips) used the PPSFP Fault Simulator to test the bulk of the faults. The total WRP test generation time for the largest ISCAS design to test all non-redundant faults was 14 seconds (IBM 3081), of which only 3 seconds were used in actual fault simulation. Clearly, fault simulation has become cheap relative to the other costs of test.

There is nothing elegant or complex about simulation by parallel pattern single fault propagation. It is a simple, yet efficient solution to fault simulation which has long been considered prohibitively expensive. The implementation of the algorithm is extraordinarily easy and can be run on any kind of computer. Even the computer memory requirement is modest compared to other simulation techniques. However, it is limited to devices which are combinational in nature such as LSSD. But for devices that meet this condition, it is the most cost effective way to provide fault simulation that can precisely evaluate test coverage, perform failure diagnosis, and serve as an efficient test generator.

3. Intelligent Heuristics (Michael Schulz)

A diverse set of experiences and results have been reported in the literature [5,6,26,29,39] on speeding up fault simulation algorithms. Concluding from these results, the following global goals should be used as a guideline when searching for improvements of fast fault simulation in combinational circuits:

1. Apply parallel processing of patterns at all stages of the fault simulation procedure
2. Avoid all unnecessary operations
3. Reduce the number of signals for which explicit fault simulations have to be performed
4. Reduce the number of gate evaluations which are required for explicit fault simulations

With these global goals in mind, a highly efficient fault simulation approach (applying parallel processing of patterns at all stages of the calculation procedure) has been developed and presented very recently [5,6]. Basically, it represents a combination of the PPSFP method and the well-known concept of fanout-free regions (FFRs) [22], which offers the important advantage of restricting the expensive explicit fault simulations to the fanout stems (FOSs) and of determining the observabilities inside the FFRs of a combinational circuit with the aid of an inexpensive backward traversal procedure. In addition, significant gains in efficiency result from the application of numerous acceleration techniques, which can be viewed as intelligent heuristics and which are briefly described below:

1. **Check-up criterion.** In order to reduce the number of FOSs for which an explicit fault simulation has to be executed, a check-up of the FFRs establishes whether or not the explicit fault simulation of the corresponding FOS may lead to detecting a fault, which is still uncovered [5,6].
2. **Taking advantage of structural circuit characteristics.** While *independent fanout branches* [10] can be employed for further reducing the number of FOSs for which explicit fault simulations have to be carried out, a significant reduction of the number of gate evaluations, which are necessary for explicitly fault-simulating the FOSs, is achieved by exploiting the *dominance relationships* between FOSs [5,6]. Note that the independent fanout branches and the dominators of FOSs are very similar to the exit lines referred to in [29].
3. **FFR-dropping.** If all faults in a certain FFR have been detected, this FFR is totally neglected during the further fault simulation process.
4. **Improved fault injection.** The applied method of fault injection guarantees that the forward propagation process of the fault effects can be stopped as soon as possible.
5. **Dynamic update of the check-up criterion.** Whenever the forward propagation of the fault effects reaches a primary output and new faults are marked as detected, the check-up criterion is dynamically updated in order to avoid unnecessary operations.

To conclude, all heuristics employed for accelerating fast fault simulation algorithms, must definitely be

(1) of linear complexity with respect to the number signals in the circuit, and

(2) well-suited to parallel processing of patterns.

Unless a distinct heuristic fulfills both of those conditions, no speed-up will result from its application. For example, if the check-up for the FFRs cited above could not be performed by parallel processing of patterns, the effort required for evaluating the check-up criterion would presumably be larger than the achievable savings in the explicit fault simulation of FOSs. Moreover, since the use of any heuristics, even those satisfying both conditions (1) and (2), induces some computational overhead in terms of CPU-time, all heuristics should be applied very carefully and flexibly.

4. Graph Theoretic Approaches (Sharad Seth)

4.1 Introduction: Fault simulation, as a tool, is useful in a wide range of applications: fault grading, testability analysis, test generation, compilation of fault dictionaries, analyzing aliasing errors or fault coverage in BIST structures, etc. The approach chosen for (fast) fault simulation is not entirely independent of the target application. For example, an approximate approach may be acceptable in fault grading; the fault-dropping technique, though generally useful in gaining speed, cannot be used in the fault-dictionary and the BIST applications. Thus, the pungent definite article in the title of this panel discussion is surely more rhetorical than real! To equivocate further, not all the approaches to be defended by the various panelists appear to be mutually exclusive. For example, hardware solutions are not incompatible with most fault simulation methods and the graph theoretic approach can be applied to single-fault propagation which is the basis for the parallel-pattern method. With these disclaimers, I will proceed to describe what the salient features of the graph-theoretic approaches are and why they might be superior to other competing methods. Only combinational circuits are assumed to be of concern to this panel.

The circuit graph, representing connectivity of elements, is a natural data structure for a circuit and as such, all methods of fault simulation are trivially graph-theoretic. However, very few methods process this graph structure explicitly and base fault simulation heavily on the information derived from such processing. Thus, in our view, the traditional fault simulation methods – parallel, deductive, and concurrent – are not graph-theoretic. Neither are some others which may be understood in terms of graph ideas but do not involve much explicit processing of the graph structure. An example would be the critical path tracing method [1] which is explained in terms of the graph theoretic notions of *fanout-free regions* and *capture lines* found implicitly by the algorithm.

4.2 Why graph based fault simulation? Graph theoretic approaches have appeared for both parallel and single pattern fault simulation. A clue to the efficiency of graph based algorithms comes from considering the basic sources of fault simulation complexity, namely *self masking* (that is, cancellation of the effect of a stem fault propagating along multiple paths at a reconvergent gate) and *multiple path sensitization.* In circuits without reconvergent fanouts such effects can not be present and known techniques, e.g. critical path tracing, can be adapted for exact linear time fault simulation. In other circuits (unfortunately, most real-world circuits), self masking and multiple path sensitization preclude inference of a stem's detectability directly from its fanout branches. Any exact fault simulation method must explicitly or implicitly analyze reconvergences of sensitized paths emanating from a stem and its efficiency is essentially determined by how quickly it can carry out such graph-based processing.

4.3 Some Recent Work: Graph based algorithms may involve static or dynamic processing of the circuit graph. Static processing is independent of the input patterns and represents a one-time cost in a preprocessing phase. Dynamic processing, on the other hand, is pattern dependent and must be repeated for each input. Several recent proposals involve only static processing. For example, the *dominators* in the transformed circuit graph with an auxiliary output are coupled with a priority queue data structure to develop an efficient single fault propagation algorithm in [20]. In another proposal, *stem region* and its *exit lines* of each reconvergent stem in a circuit are found in the preprocessing phase and shown to provide fairly reliable estimates of fault simulation complexity of a circuit [29]. An exact and efficient fault simulation method based on stem regions has also been reported [30] and is said to perform favorably compared to the fastest reported fault simulator [5] on the ISCAS benchmark circuits.

In a purely dynamic approach Ke et al. [25] propose creating an auxiliary graph to reflect self masking and multiple sensitization constraints between lines of the circuit. These constraints are propagated in a single backward pass avoiding propagation of individual stem faults. Another notable recent contribution [21] involves repeated steps of graph compaction reminiscent of the flow graph manipulations used in code optimization algorithms. The method is shown to outperform all previously published fault simulation algorithms on a *hard* and an *easy* family of circuits. This method is also notable for solving more than the standard fault simulation problem: it can tag each detected fault with the output(s) at which it will be detected by the input pattern thus being useful in applications involving fault dictionaries and BIST.

4.4 Conclusion: A family of graph based algorithms are now available for single and parallel pattern fault simulation of combinational logic circuits. Their performance directly reflects the basic complexity of fault simulation arising from reconvergent fault propagation paths. Indeed, experimental data on early implementations indicates these algorithms to be some of the fastest available.

5. Approximate Solutions (Miron Abramovici)

Conventional fault simulation is a very expensive computational process. The question "What is the Path to Fast Fault Simulation?" has several valid answers, depending on the environment where fault simulation is used and on the application it supports.

Even the fastest general-purpose fault simulation algorithm – concurrent simulation – requires lots of CPU time. Software implementations can be accelerated using hierarchical modeling [35]. Significant speed-ups are realized by implementing the algorithms with special-purpose hardware or by distributing the fault list among the processing units of a general-purpose multiprocessor [17].

Several approximate fault simulation techniques have been developed with the goal of trading off some accuracy in results for a substantial reduction in the computation cost. The most important aspect of an approximate method is the nature – pessimistic or optimistic – of its approximations. If the only objective of fault simulation is to obtain an estimate of the fault coverage, then any approximate method is acceptable, provided that the extent of the approximation is small and can be bounded. Fault sampling [4] satisfies these requirements. However, for the other main applications of fault simulation – diagnosis and test generation – the use of optimistic approximations is detrimental [2].

For circuits incorporating BIST hardware, the fault simulation problem is compounded by the large number of vectors that have to be evaluated. If BIST circuits can be treated as combinational during testing, one can use techniques specialized for combinational circuits. The best methods for this application rely on parallel evaluation of groups of vectors [5, 39].

The best fault simulation method to support a test generation system for combinational circuits is Critical Path Tracing (CRIPT) [1]. CRIPT is fast and its approximations occur seldom and are guaranteed to be pessimistic. Additional features of CRIPT allow test generation algorithms to be guided by fault simulation results [3].

6. Hierarchical Fault Simulation (Bill Rogers)

Hierarchical fault simulation reduces the space time product of fault simulation without using simplified models or constrained design techniques [35]. The computational complexity of the hierarchical approach is much more nearly linear than non-hierarchical [16] fault simulation, so the demand for more CPU cycles increases less explosively as circuit designs get larger.

The hierarchical approach is based on using the design hierarchy and functional models (from design verification) to create alternative circuit representations which contain roughly $log(n)$ elements instead of the original n elements [33, 34]. The representation can be reduced further by wrapping collections of lower level primitives into more complex primitives . [18] Since the simulator has fewer elements to simulate it goes faster. The smaller circuit representation takes less memory so both the space and time requirements are reduced. Since the hierarchical approach affects the circuit representation and not the simulation algorithm, practically any simulator can be enhanced to exploit hierarchy. The adaptation is simple and the only requirement is that the simulator must support mixed mode simulation.

There are other benefits of using hierarchy. The hierarchy provides a set of natural boundaries for partitioning and reconfiguration to create multiple different representations of the circuit. Each of the $log(n)$ representations models a different part of the circuit at the fault modeling level and the remainder of the circuit is modeled functionally. Each circuit representation is complete and can be simulated separately. This is perfect for distributed simulation [14]. The number of faults in each representation can be chosen to optimize throughput and the distributed simulation is so loosely coupled that it can be expanded indefinitely.

The hierarchical approach also encourages overlapped design and test generation. As soon as a complete representation of the circuit is available fault simulation and test generation can begin by using functional fault models [12]. These can be mixed with more detailed fault models as the design becomes more complete. A preliminary test can be developed and then refined along with the design. Mixed fault modeling is natural in this environment. For example, stuck-at faults can be modeled in gates while missing and extra crosspoints are modeled in PLAs. Hierarchical simulation couples nicely with the design process and supports mixed fault modeling which in turn also supports mixed technology designs.

Since the hierarchical approach is an enhancement to existing algorithms there is minimal impact on accuracy. If the underlying algorithm supports sequential circuits or transistor based designs so does the hierarchically enhanced algorithm. The hierarchical enhancements do not add any design constraints except that the circuit design must be hierarchical. This is not an issue because large circuits cannot be designed without using a hierarchical methodology.

Due to the reduced space and time requirements of hierarchical simulation, flexible fault modeling techniques, and suitability for large distributed simulation, hierarchical fault simulation is the most promising approach to fault simulation of tomorrow's multimillion transistor integrated circuits. Because hierarchical techniques provide so many benefits and can be applied to most algorithms, research and development should first concentrate on exploiting hierarchy and later on enhancements to individual algorithms.

7. Hardware Solutions (Rob Mathews)

If you need fast fault simulation, adopt the hardware approach: run known, proven algorithms on fast hardware. Your simulation results will have a known relationship to test quality, and others will know what you mean when you give coverage numbers. With a proper choice of hardware and software, you can keep implementation costs down while achieving practical turnaround. For example, a hardware accelerator running concurrent fault simulation provides a cost-effective, low-risk, practical solution for fault simulation needs today.

7.1 What is a good solution? There are four factors to consider in choosing a path to fault simulation:

1) *Acceptance of results.* If it is not well known and widely accepted how your results relate to test quality, you don't have a solution.

2) *Costs.* A good solution must fit into existing design and test practices, have a reasonable price, and require reasonable setup and maintenance.

3) *Speed.* A solution must deliver results quickly enough.

4) *Timeliness.* If you need a solution soon, that solution had better be available soon.

A hardware solution measures up along each of these dimensions.

7.2 Acceptance The hardware approach is a proven solution. When people specify fault coverage, the coverage numbers it computes are the numbers they mean. These numbers have an established, empirical correlation to defect levels, given gate-level modeling of input and output faults.

Similarly, the hardware approach employs the algorithms that define fault simulation. The crucial coverage/quality relationship has proven to be sensitive even to apparently small changes in fault models and algorithms e.g., the unlying logic simulation algorithm. Thus, the MIL 38510 slash sheet specifies 95% gate-level coverage, but the military is going to considerable additional trouble to correlate results among various commercial implementations of the concurrent algorithm. A new algorithm is suspect until it is known to be equivalent to the accepted ones; non-equivalent or approximate solutions face an even larger burden of proof. 95 ±5% is just not good enough.

7.3 Costs The hardware approach is independent of design style. Tools and techniques for design and test vary widely. The hardware approach does not require a shop to face the real costs and risks associated with changing how it does business. Rather, it fits into the existing design flow.

It works for general, sequential circuits with or without scan, BIST, etc. It works for high-impedance technologies, including sequential behavior associated with MOS faults. It allows behavioral models to be included in the fault simulation. You needn't redefine the problem to get a fast solution.

The hardware approach requires no special setup. Since you can fault simulate with or without timing, you can proceed directly from logic simulation to fault simulation without having to debug your circuit first. Also, you needn't support two libraries: one for logic and another for fault – one library can do it all.

7.4 Speed The hardware solution is practical today. Given enough memory, fast hardware can exploit the available parallelism in the computation. The concurrent algorithm provides between one and two orders of magnitude speedup over serial fault simulation. Full grading of a typical 10K-gate ASIC need only take a few hours with commercially available hardware solutions. Moreover, a full, batch fault run is a worst case, since you are typically focusing on a portion of a design at a time as you develop tests; run time for such a partial run is proportionally less.

The solution extends even to large problems. Hardware accelerators are in use today grading single circuits of 200K-gate complexity.

7.5 Timeliness The hardware solution is in wide use today. Hundreds of fault simulators are in regular use on accelerators, mainframes, and supercomputers. Depending on your needs, you can choose from a half dozen commercial alternatives available from vendors today.

In conclusion the path to fast fault simulation exists today: combine powerful hardware with known, proven algorithms.

References

1. Abramovici, M., Menon, P.R., and Miller, D.T., "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Comput.*, vol. 1, no. 1, pp. 83-93, February 1984.
2. Abramovici, M., "Low-Cost Fault Simulation: Why, When and How"," *Proc. Intl. Test Conf.*, p. 795, November 1985.
3. Abramovici, M., Kulikowski, J.J., Menon, P.R., and Miller, D.T., "SMART and FAST: Test Generation for VLSI Scan-Design Circuits," *IEEE Design & Test of Comput.*, vol. 3, no. 4, pp. 43-54, August 1986.
4. Agrawal, V.D., "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *Journal of Digital Systems*, vol. 5, no. 3, pp. 189-202, Fall 1981.
5. Antreich, K.J. and Schulz, M.H., "Accelerated Fault Simulation and Fault Grading in Combinational Circuits," *IEEE Trans. on CAD*, vol. CAD-6, pp. 704-712, September 1987.
6. Antreich, K.J. and Schulz, M.H., "Fast Fault Simulation for Scan-Based VLSI Logic," *Proc. European Conf. on Circuit Theory and Design*, pp. 101-106, September 1987.
7. Armstrong, D.B., "A Deductive Method for Simulating Faults in Logic Circuits," *IEEE Trans. Comput.*, vol. C-21, no. 5, pp. 464-471, 1972.
8. Barzilai, Z., Carter, J.L., Rosen, B.K., and Rutledge, J.D., "HSS – A High-Speed Simulator," *IEEE Trans. on CAD*, vol. 6, no. 4, pp. 601-617, July 1987.
9. Breuer, M.A. and Friedman, A.D., *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, 1976.
10. Brglez, F., "A Fast Fault Grader: Analysis and Applications," *Proc. of Intl. Test Conf.*, pp. 785-794, 1985.
11. Bryant, R.E., "Symbolic Verification of MOS Circuits," in *1985 Chapel Hill Conf. on Very Large Scale Integration*, ed. Henry Fuchs, pp. 419-438, Computer Science Press, Rockville, MD, 1985.
12. Chang, H.P., Rogers, W.A., and Abraham, J.A., "Structured Functional Level Test Generation Using Binary Decision Diagrams," *Proc. of the IEEE Intl. Test Conf.*, pp. 97-104, 1985.
13. Daehn, W. and Geilert, M., "Fast Fault Simulation for Combinational Circuits by Compiler Drive Single Fault Propagation," *Proc. IEEE Intl. Test Conf.*, pp. 286-292, September 1987.

14. Duba, P., Roy, R., Abraham, J.A., and Rogers, W., "Fault Simulation in a Distributed Environment," *Proc. of the 25th Design Automation Conf.*, pp. 686-691, 1988.

15. Fujiwara, H., *Logic Testing and Design for Testability,* MIT Press, Cambridge, MA, 1985.

16. Goel, P., "Test Generation Cost Analysis and Projections," *Proc. of 17th Design Automation Conf.*, pp. 77-84, 1980.

17. Goel, P., Huang, C., and Blauth, R.E., "Application of Parallel Processing to Fault Simulation," *Proc. Intl. Conf. on Parallel Processing*, pp. 785-788, August 1986.

18. Guzolek, J.F., Rogers, W.A., and Abraham, J.A., "WRAP: An Algorithm for Hierarchical Compression of Fault Simulation Primitives," *Proc. of the IEEE Intl. Conf. on Computer-Aided Design*, pp. 338-341, 1986.

19. Harel, D. and Krishnamurthy, B., "Is There Hope for Linear Time Fault Simulation," *Proc. 17th FTCS*, pp. 28-33, July 1987.

20. Harel, D., Sheng, R., and Udell, J., "Efficient Single Fault Propagation in Combinational Circuits," *Proc. of ICCAD-87*, November 1987.

21. Harel, D. and Krishnamurthy, B., "A Graph Compaction Approach to Fault Simulation," *Proc. 25th Design Automation Conf.*, pp. 601-604, June 1988.

22. Hong, S.J., "Fault Simulation Strategy for Combinational Logic Networks," *Proc. of 8th Intl. Symp. of Fault Tolerant Comp.*, pp. 96-99, 1978.

23. Ishiura, N., Yasuura, H., Kawata, T., and Yajima, S., "High-Speed Logic Simulation Using a Vector Processor," ER 85-02, Kyoto University, Yajima Res. Lab, Kyoto Japan, May 1985.

24. Jain, S.K. and Agrawal, V.D., "STAFAN: An Alternative to Fault Simulation," *Proc. 21st Design Automation Conf.*, pp. 18-23, 1984.

25. Ke, W., Seth, S., and Bhattacharya, B.B., "A Fast Fault Simulation Algorithm for Combinational Circuits," Department of Computer Science Report Series #69, May 1988. (To be presented at ICCAD, November 1988)

26. Koeppe, S. and Starke, C.W., "Logiksimulation Komplexer Schaltungen fuer Sehr Grosse Testlaengen," in *Vortraege der NTG-Fachtagung*, pp. 73-80, Baden-Baden, March 1985.

27. Koeppe, S., "Modeling and Simulation of Delay Faults in CMOS Logic Circuits," *Proc. IEEE Intl. Test Conf.*, pp. 530-536, September 1986.

28. Levendel, Y. and Menon, P.R., "Fault Simulation," in *Fault Tolerant Computing: Theory and Techniques*, ed. D.K. Pradhan, vol. 1, pp. 184-264, Prentice-Hall, Englewood Cliffs, NJ, 1986.

29. Maamari, F. and Rajski, J., "A Reconvergent Fanout Analysis and Fault Simulation Complexity of Combinational Circuits," *Proc. FTCS-18*, June 1988.

30. Maamari, F. and Rajski, J., "An Exact Fault Simulator Based on Stem Region Analysis," *Presented at the 11th Annual Workshop on DFT*, Vail, Colorado, April 1988.

31. Miczo, A., *Digital Logic Testing and Simulation,* Harper and Row Publishers, New York, 1986.

32. Nagamine, M., "An Automated Method for Designing Logic Circuit Diagnostic Programs," *Proc. 8th ACM-IEEE Design Automation Conf.*, pp. 236-241, June 1971.

33. Rogers, W.A. and Abraham, J.A., *CHIEFS User's Manual.* unpublished

34. Rogers, W.A. and Abraham, J.A., "High-Level Hierarchical Fault Simulation Techniques," *Proc. of the ACM Spring Comp. Sci. Conf.*, pp. 89-97, March, 1985.

35. Rogers, W.A., Guzolek, J.F., and Abraham, J., "Concurrent Hierarchical Fault Simulation," *IEEE Trans. on CAD*, vol. 6, no. 5, pp. 848-862, September 1987.

36. Seshu, S., ''On an Improved Diagnosis Program,'' *IEEE Trans. Elect. Comput.*, vol. EC-12, no. 2, pp. 76-79, 1965.

37. Ulrich, E.G. and Baker, T., ''The Concurrent Simulation of Nearly Identical Digital Networks,'' *Proc. of 10th Design Automation Workshop*, vol. 6, pp. 145-150, 1973.

38. Waicukauski, J.A., Eichelberger, E.B., Forlenza, D.O., Lindbloom, E., and McCarthy, T., ''A Statistical Calculation of Fault Detection Probabilities by Fast Fault Simulation,'' *Proc. Intl. Test Conf.*, pp. 779-784, November 1985.

39. Waicukauski, J.A., Eichelberger, E.B., Forlenza, D.O., Lindbloom, E., and McCarthy, T., ''Fault Simulation for Structured VLSI,'' *VLSI Systems Design*, vol. 6, no. 12, pp. 20-32, December 1985.

40. Waicukauski, J.A., et al, ''Diagnosis of BIST Failures by PPSFP Simulation,'' *Proc. IEEE Intl. Test Conf.*, pp. 480-484, September 1987.