MULTIPLE ERROR DIAGNOSIS BASED ON XLISTS

Vamsi Boppana, Rajarshi Mukherjee, Jawahar Jain, Masahiro Fujita
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway
Sunnyvale, CA 94086
{vboppana,rmukherj,jawahar,fujita}@fla.fujitsu.com

Pradeep Bollineni
Department of Computer Science
Iowa State University
Ames, IA 50014
pradeep@cs.iastate.edu

Abstract

In this paper, we present *multiple* error diagnosis algorithms to overcome two significant problems associated with current error diagnosis techniques targeting large circuits: their use of limited error models and a lack of solutions that scale well for multiple errors. Our solution is based on a non-enumerative analysis technique, based on logic simulation (3-valued and symbolic), for simultaneously analyzing *all* possible errors at *sets* of nodes in the circuit. Error models are introduced in order to address the "locality" aspect of error location and to identify sets of nodes that are "local" with respect to each other. Theoretical results are provided to guarantee the diagnosis of modeled errors and robust diagnosis approaches are shown to address the cases when errors do not correspond to the modeled types. Experimental results on benchmark circuits demonstrate accurate and extremely rapid location of errors of large multiplicity.

1 Introduction

Error diagnosis is typically invoked after a failed design verification step to determine possible causes for the failure. It is critical both in providing feedback to designers about potential error sites and in providing a limited set of candidates where automatic correction could be performed.

Previous research in combinational error diagnosis has primarily yielded solutions practical only for diagnosing single errors in circuits. The solutions can be broadly classified into two categories: simulation-based or simulation-like analysis-based approaches [1–10] and synthesis-like formulations to error diagnosis and correction using symbolic functional manipulation [11–13]. Exact synthesis formulations for the error correction problem based on the use of BDDs typically do not rely heavily on the error models and can be extended for the general case of multiple errors relatively easily. However, they typically do not scale up for large circuits because of excessive run-time and memory requirements.

In contrast, simulation-based approaches offer a viable alternative for handling large circuits because they are efficient both in terms of their run-time and memory performance. In this class of techniques, by simulating each input vector on a set of assumed error models and by determining the modeled errors whose behavior conforms to the observed erroneous behavior, potential error candidates are gradually pruned. Diagnosis techniques based on this kind of analysis where modeled causes are examined for their effects are grouped under cause-effect techniques [10]. However, significant problems with applying these techniques to practical circuits result from their use of *limited error models* and a lack of solutions that scale well for *multiple errors*.

In this paper, we provide solutions to the above problems. Our solution derives its motivation from previous work to alleviate the problems with restrictive error models [2, 9, 14, 15]. These methods suggested the use of models more abstract than conventional ones in the applications of fault simulation, fault and error detection and diagnosis. However, even these approaches were limited to *single* node analysis. The reason for the popularity of the single node analysis-based techniques is that they provide excellent run-time performance. However, the accuracy of the diagnosis procedure suffers when the type of the error becomes more complicated. It is becoming increasingly important, however, to consider multiple design errors because even a single node error made at one stage in the design flow may result in multiple errors at a subsequent stage.

Our Contributions

In this paper, we develop efficient diagnosis algorithms that permit a departure from the notion of analyzing single nodes in the circuit for their likelihood as error sites. Our algorithms are easily integrated into a design simulation environment, are applicable to multiple levels of design hierarchy (RTL/gate/transistor) and are easily extended to handle sequential circuits and practical circuit configurations (with libraries). In addition, our algorithms overcome restrictions imposed by the use of overly restrictive error models. These capabilities make our algorithm suitable for handling real-world designs. In spite of targeting the analysis of multiple sets of nodes simultaneously, we show that it is indeed possible to achieve run-times comparable to single node-analysis based techniques. Since the problem of error *location* suggests proximity amongst the multiple errors, we develop models to identify and analyze groups of nodes that are *near* each other. We achieve our objectives by the fol-

lowing contributions:

- Development of an efficient, *non-enumerative (without explicitly considering each possible error)*, analysis technique (that helps us achieve small run-times) based on logic simulation (3-valued and symbolic) for simultaneously analyzing all possible errors at *sets of nodes*.
- Development of two error models (topologically bounded, region-based) that enable the identification of sets of nodes that are *near* each other.
- Development of theoretical results that **guarantee** the exact diagnosis of modeled errors.
- Development of robust diagnosis algorithms that perform well even in the presence of unmodeled errors.

An important point to note is that while vectors differentiating the specification and the implementation can be generated to increase the accuracy of diagnosis, our focus in this work is on developing diagnosis algorithms that produce results of the highest quality, given a set of test vectors. Vector generation was not considered in this work.

2 Efficient Analysis of Errors at Sets of Nodes

Definition 1 (Xlist) *A set of nodes whose actual values would be replaced during simulation by the value X, and the X-values propagated to subsequent logic levels by 3-valued logic simulation is called an Xlist.*

Diagnosis With Xlists
The illustration of Figure 1 demonstrates the utility of Xlists for achieving powerful diagnostic results even without any enumeration on the possibility or kinds of errors. In this figure we have shown simulation results from the specification and three simulation results on the faulty implementation. The first result is from normal simulation on the faulty implementation. The other two simulation results are with the presence of Xlists indicated by the shaded regions A and B. The output responses produced by the simulation results on one erroneous primary output are also shown (specification produces 0, implementation produces 1, simulation with Xlist A produces X and simulation with Xlist B produces 1).

Claim: No multiple error consisting entirely of nodes in the region B, can explain the erroneous behavior.

The result is obvious from the fact that if there were indeed some error at the nodes in B, such that changing those nodes would enable the implementation to conform to the specification, then that changed functionality would still be *covered* by the X-values introduced at those nodes because of the Xlist simulation. Of course, it is obvious that since **all** possible error types are being targeted at these nodes, the analysis is pessimistic. This is illustrated by the result on Xlist A where we do not have a definitive diagnostic inference.

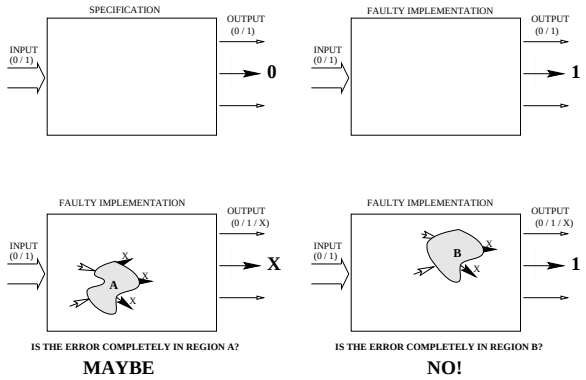

Figure 1: Using Xlists for Diagnosis

3 Choosing the right Xlists - Error Models

In this section, we describe techniques to choose Xlists to derive accurate diagnosis results. The aim here is to model the locality of the errors (because we are performing error *location*!) and derive candidate Xlists using this information. Two error models are introduced to model the locality; one based on a topological sorting of the nodes in the circuit and the other based on growing a region around each node in the circuit. We first define the error models and then based on the error model under consideration, Xlists can be constructed to include all the sets of nodes that are candidates as suggested by the error model. In our discussion, an error refers to a node or a set of nodes whose logic functions could be altered to attain functional equivalence with the specification.

3.1 Topologically Bounded Errors

Let the circuit for the implementation under consideration have n nodes. Let $T = (1,\ 2,\ \ldots,\ n)$ be a topological order on the nodes of the circuit.

Definition 2 (Topologically Bounded Error) *An error is called a topologically bounded error if the logic functions at the nodes in set $E = \{e_1,\ e_2,\ \ldots,\ e_k\}$ are erroneous and satisfy the following:*

$$\exists\ i,\ j,\ (1 \leq i \leq j \leq n)\ such\ that$$
$$\{i \leq e_l \leq j,\ \forall\ l \mid 1 \leq l \leq k\}.$$

The integers i, j are the lower and upper bounds within which the error lies and $j - i + 1$ is called the bound on the error.

3.2 Errors in Regions of the Circuit

We now define another model for local design errors. In this model we define regions in the circuit that consist of nodes that are located "near" each other. Regions are defined

```
// Each L is an Xlist of nodes
// MATCHCOUNT, PARTIALMATCHCOUNT,
// MISMATCHPENALTY
// and DROPTHRESHOLD are input parameters
// Match means a (0, 0) or (1, 1) combination
// Mismatch means a (0, 1) or (1, 0) combination
// Partialmatch means a (X, 0) (X, 1) combination
for each Vector
// Let CurrVec be the current vector
 for each Xlist L do
  if (Score[L] > DROPTHRESHOLD) do
   for each primary output do
   // Let CurrPo be the current primary output
    if (Match(Impl(L, CurrVec, CurrPo), Spec(CurrVec, CurrPo)))
     Score[L] += MATCHCOUNT;
    elseif (PartialMatch (Impl(L, CurrVec, CurrPo),
          Spec(CurrVec, CurrPo)))
     Score[L] += PARTIALMATCHCOUNT;
    else // MISMATCH
     Score[L] -= MISMATCHPENALTY;
    endif
  endfor
 endfor
endfor
```

Figure 2: Xlist ranking procedure

for each node in the circuit and are defined with respect to the structural distance of nodes in the circuit. Structural distance between two nodes refers to the minimum number of wires (connections) needed to traverse from one node to the other.

Definition 3 (Region-based Error) *An error is called a region-based error of radius r centered at node p if the logic functions at the nodes in set $E = \{e_1, e_2, \ldots, e_k\}$ are erroneous and satisfy the following:*

$$\forall\, l\,,\ 1 \le l \le k,$$
$$Structural\ Distance(e_l,\ p) \le r$$

4 Diagnosis Algorithms

Our diagnosis algorithm based on the simulation of Xlists is presented in Figure 2. Diagnosis proceeds for each given vector, by considering each candidate Xlist (generation of candidate Xlists to *guarantee* exact diagnosis will be discussed later in the paper) and observing the number of matches {(0,0), (1,1) value combinations}, mismatches {(0,1), (1,0) value combinations} and partial matches {(X,0), (X,1) value combinations } on the primary outputs. If an Xlist produces a mismatch (match, partial match), the "potential" of that Xlist to contain the error nodes is reduced (increased, increased slightly). The "potential" of that Xlist for either explaining or not explaining the erroneous behavior depends on the type of error and the error models that the Xlists are created to handle. Confidence in the modeling process is provided to the diagnosis procedure through three parameters: MATCHCOUNT, PARTIALMATCHCOUNT, MISMATCHPENALTY, representing the confidence in a match result, partial match result and a mismatch result, respectively. Assigning MATCHCOUNT and MISMATCHPENALTY to large values supports diagnosis with high confidence in the error models while assigning them to low values helps applications where the kinds of errors are uncertain. In particular, if the errors are exactly of the modeled type and Xlist construction has been performed to cover all the modeled errors, then MISMATCHPENALTY can be set to ∞. These input parameters are provided to the diagnosis algorithm in order to prevent an over-reliance on the error models.

4.1 Guaranteed Diagnosis

In this subsection, we present the results that guarantee exact diagnosis of modeled errors. It is worth noting that our guaranteed diagnosis results are not restricted to the error models introduced. In fact, it can be shown that whenever there exists an Xlist that contains every single node of the actual error sites, it can be guaranteed to obtain the highest score amongst all the Xlists.

Exact diagnosis of errors is possible when the error being diagnosed is known to be of the modeled type. In order to achieve the exact diagnosis, we need to configure the input parameters of the diagnosis algorithm to reflect the confidence in the error model. Specifically, we have to set the parameters MATCHCOUNT = 0, PARTIALMATCHCOUNT = 0 and MISMATCHPENALTY = ∞. With this configuration, our next two results demonstrate that exact diagnosis of the modeled errors can be guaranteed.

Theorem 1 (Diagnosing topologically bounded errors) *If there exists an error (multiple) topologically bounded by k, then by choosing overlapping Xlists as defined below, it can be guaranteed that there exists an Xlist containing all the nodes that are error sites and that its rank would be the highest of any of the candidate Xlists.*
$(1,\ 2,\ \ldots,\ 2k),\ (k+1,\ k+2,\ \ldots,\ 3k),$
$(2k+1,\ \ldots,\ 4k)\ \ldots\ ((\lceil n/k \rceil - 2)k + 1,\ \ldots,\ n)$

Proof: It is easy to show that there is at least one Xlist that completely covers each node of the error. The proof follows by noting that setting the above-mentioned input parameters for the algorithm guarantees that only mismatches are penalized (by the definition of Xlists, it is guaranteed that no mismatches would occur for any Xlist containing all the error nodes).

Theorem 2 (Diagnosing region-based errors) *If there exists a region-based error (multiple) of radius r centered at node i, then by choosing a region-based Xlist at each node in the circuit that includes every node within a radius of r*

from that node, it can be guaranteed that the Xlist at node i would achieve the highest score of all the Xlists.

Proof: Follows from the fact that we have a guaranteed inclusion of all the error nodes in the Xlist at node i and by observing again that it is possible to guarantee no mismatches for any Xlist that includes all the error nodes. □

Having provided the above two results, we now present a practical problem and solution in the application of these results. When the diagnostic resolution of the diagnostic test vectors is not sufficiently high, it is possible that the number of Xlists with the highest score (including the correct Xlist) may be high for practical usage if the diagnosis algorithm were run with the parameters MATCHCOUNT = 0, PARTIALMATCHCOUNT = 0 and MISMATCHPENALTY = ∞. In order to overcome this problem, we again resort to the flexibility provided by the input parameters in the diagnosis algorithm that matches may indeed be better than partial matches. Of course, this may then make the diagnosis algorithm slightly inexact, but it does provide an intuitive and practical approach to rank candidate Xlists that do not provide any mismatches. Our experiments show that this is indeed a practical approach. A typical run of the diagnosis algorithm with modeled error types would proceed with a MATCHCOUNT of 10, PARTIALMATCHCOUNT of 5 and a MISMATCHPENALTY of 1000.

4.2 Improved diagnosis using symbolic variables

In order to improve the accuracy of the diagnosis procedure, we introduce symbolic variables at the outputs of the Xlist regions to compute the primary output functions in terms of these variables and use them to prune the set of diagnosis candidates obtained after application of the Xlist-based diagnosis algorithm. This method improves the accuracy of diagnosis due to removal of any losses in accuracy due to the use of 3-valued logic and by ensuring that a function can be realized at the outputs of any Xlist region in terms of its inputs in order that the implementation be made functionally equivalent to the specification. Such a function may be absent because conflicting values may be required at the outputs of an Xlist region for the same set of input values. The absence of such a function at the outputs of an Xlist clearly indicates the inability of that Xlist to explain the observed error and can be used to reduce the number of candidate Xlists. An algorithm for the construction of these functions is outlined next. Details of the complete diagnosis algorithm are omitted due to lack of space.

Let $V = \{v_1, v_2, \cdots, v_m\}$ be the vector set being used for diagnosis. Let B_{ijk} be the function (BDD) built for the ith output, for the jth vector in V, and for the kth Xlist, X_k. Let sets, $V_{s_{j1}}, \cdots, V_{s_{jn}}$, each contain vectors that produce same values on the inputs of a particular Xlist. For example, if vectors v_1, v_3 and v_6 produce same values at the inputs of Xlist X_p, then one of $V_{s_{j1}}, \cdots, V_{s_{jn}}$ will contain these three vectors. Functions can be computed to store the values at the outputs of Xlist X_k such that the outputs of the specification and the implementation are consistent for all the vectors that produce the same input values at the input of X_k. Construction of these functions is helped by constructing functions to store the values at the outputs of Xlist X_k such that the outputs of the specification and the implementation are consistent for each individual vector v_j. An intuitive explanation of these functions is that they represent the **correction functions** at the outputs of Xlists for the respective vector/vectors.

5 Experimental Results

We evaluate the performance of our new diagnosis algorithms in this section. Experiments were performed on the ISCAS 85 benchmark circuits. Diagnosis experiments were repeated for 10 different errors for each benchmark circuit and error type considered. Experiments were performed on a SPARCstation20 workstation with 64MB of main memory. The test vectors used in the diagnosis were adapted from the single stuck-at vectors generated by the HITEC test generator [16]. We also note that the introduction of errors is accompanied by checking whether the error was detected by the given vectors. If the error was not detected, then it was immediately discarded and an alternative error was generated. This was done primarily because test generation to help the diagnosis was not considered in this work.

Improvements due to region-based analysis over single node-based analysis (Table 1)

The diagnosis algorithm as described in Figure 2 was used for the diagnosis. It is first shown that while single node analysis-based diagnoses are unable to track multiple design errors, the new approaches produce remarkable improvements in the accuracy of the diagnosis. These results are summarized in Table 1. Ten instances of region-based errors within a *radius* of 2 were introduced into each benchmark circuit and diagnosis results are analyzed for a single node analysis-based approach and for a region-based Xlist approach with a radius of 2. The center of the error and the logic function changes were randomly chosen. The columns in the table represent for each of the circuits, the number of nodes in the circuit, the number of vectors used in the diagnosis, average number of logic changes introduced in the circuit, average rank of the center of the error set in the candidate list (intuitively, since the error nodes are all centered at the center node, it should have the best chance of tracking the error), time taken in seconds for the diagnosis, average rank of the region-based Xlist centered at the center of the error and the time taken in seconds for the region-based diagnosis algorithm. The lesser the rank for the faulty node, the better is the accuracy of the diagnosis. Therefore, it is

Table 1: Diagnosis results comparing point-based vs. region-based analysis

Circuit	Nodes	Vectors	#Errors	Point-Based		Region-Based	
				Rank	Time(sec)	Rank	Time(sec)
c432	275	55	7.40	113.10	0.25	7.90	0.88
c499	276	55	9.50	47.90	1.98	18.80	0.99
c880	469	75	5.90	7.00	3.26	5.40	0.53
c1355	619	88	11.40	121.70	15.07	18.40	5.70
c1908	938	280	8.90	116.50	63.98	5.40	5.59
c2670	1414	102	8.60	189.50	28.40	31.10	8.21
c3540	1741	350	10.70	200.50	165.58	17.30	42.45
c5315	2608	264	10.00	52.80	150.01	30.00	22.61
c6288	2480	46	9.90	197.60	105.54	96.90	135.46
c7552	3827	450	7.60	134.60	751.92	16.00	62.12

clear from the data in this table, that our new diagnosis algorithm is significantly more effective. An interesting result is the surprisingly large average rank produced for the circuit c6288 even with the improved diagnosis algorithm. The most likely cause for this is the much lesser number of test vectors used for this circuit. This, however, does not prevent us from drawing our expected conclusion that the accuracy is indeed significantly better than a single node analysis-based diagnosis approach.

Improved diagnosis using symbolic variables (Table 2)

Improvements resulting from the introduction of symbolic variables at the outputs of Xlist-regions to further reduce the size of candidates which produced no MISMATCHES are shown in Table 2. In this experiment, candidate Xlist regions that produced no differences from the specification were chosen and symbolic variables were introduced at their outputs. Analysis using symbolic variables and assuming modeled errors was performed in order to prune the set of candidates. Table 2 shows for each circuit, percentage reduction in the number of candidate Xlists, time taken by the procedure and the maximum size of the BDDs encountered averaged over 25 different error diagnoses per circuit. The data in the table clearly demonstrates that a significant improvement in the accuracy of diagnosis can be achieved with this technique.

Variation in the accuracy and run-time required for diagnosis with varying region sizes (Table 3)

In order to study the performance (both time and accuracy) of the region-based diagnosis algorithm with the parameter *radius*, the following experiment was performed. Errors were introduced with an increasingly larger radius (varied from 1 through 4) on one of the benchmark circuits, c432. Again, ten instances of errors were diagnosed for each radius and average numbers are reported. As the radius was increased, the number of errors introduced in the circuit also increased. These results are presented in Table 3. This data can also be used to study the performance of the algorithm in situations where confidence in the error model may not be high (hence, we would choose larger regions). The small average ranks of the correct Xlist (out of 275 total Xlists) show that the algorithm scales well to even gross errors.

Table 2: Diagnosis with symbolic variables

Circuit	% Reduction	Average Time(s)	Avg. Max. Bdd Nodes
c432	80.50	7.01	91.94
c499	20.85	1.81	358.40
c880	23.33	1.38	112.27
c1355	55.86	11.86	724.30
c1908	48.93	8.02	398.19
c2670	66.71	20.77	233.12
c3540	53.15	34.94	691.50
c5315	36.65	13.18	262.37
c6288	95.75	92.11	73.23
c7552	59.52	111.15	1122.82

Table 3: Variation of accuracy, run-times with error-radius

Circuit	Radius	#Errors	XlistRank	Time(s)
c432	1	3.00	7.70	0.47
c432	2	8.90	10.80	1.20
c432	3	18.80	25.20	2.57
c432	4	30.60	13.50	7.17

Diagnosis for topologically bounded errors (Table 4)

Finally, Table 4 presents the results for diagnosing errors that are topologically bounded. The columns represent for each of the circuits, the number of nodes in the circuit, the num-

Table 4: Diagnosis results based on topological Xlists

Circuit	Nodes	Vectors	#Errors	XlistRank	Time(s)
c432	275	55	7.30	1.80	0.24
c499	276	55	7.20	4.40	0.36
c880	469	75	7.50	1.50	0.35
c1355	619	88	8.00	9.30	2.41
c1908	938	280	7.90	4.80	5.39
c2670	1414	102	7.50	3.10	1.47
c3540	1741	350	7.80	1.30	5.97
c5315	2608	264	7.80	13.70	4.34
c6288	2480	46	7.90	16.30	23.81
c7552	3827	450	8.00	5.56	23.11

ber of vectors used in the diagnosis, average number of logic changes introduced in the circuit, average rank of the topological Xlist consisting of all the nodes at which the logic changes were introduced and the time taken in seconds for the topological Xlist-based diagnosis.

Errors with a topological bound of 8 were introduced into the circuit. The topological region where the error was introduced and the types of logic changes introduced were all chosen randomly. Ten different topological error sets were diagnosed for each benchmark circuit and the average results reported. The Xlists generated were with a topological bound of 20, i.e., each Xlist had a cardinality of 20 nodes. A topological bound of 16 would have also sufficed (according to our result from Theorem 1). However, we chose a bound of 20 in order to illustrate the case where confidence in the type of the error may not be exact; thus forcing a safer Xlist generation strategy. The small average ranks of the correct Xlists and the small run-times demonstrate the ability to diagnose topologically bounded multiple errors rapidly and accurately.

6 Conclusions

Our experiments demonstrate the ability of Xlists to efficiently diagnose errors of large multitude. The only assumption on the error types is one of locality i.e., they are either topologically bounded or are in some region. If there is reason to believe in any alternative multiple error model, then Xlists may be designed even there to capture the effects of arbitrary errors on those sets of nodes. The efficiency of our diagnosis algorithms results from the fact that they are based on a simple application of logic simulation, using 3-valued and symbolic logic to model arbitrary errors by single Xlists (without enumeration). Although not covered in this paper, an important advantage of the analysis based on Xlists lies in the simple extensions they offer to diagnose sequential circuits based on logic simulation. Extensive experimentation with the design and use of Xlists for RTL circuits is also currently being performed.

References

[1] M. Tomita and Hong-Hai Jiang, "An algorithm for locating logic design errors", in *Proc. Intl. Conf. Computer-Aided Design*, Nov. 1990, pp. 468–471.

[2] S.-Y. Kuo, "Locating logic design errors via test generation and don't-care propagation", in *Proc. European Design Automation Conf.*, 1992, pp. 466–471.

[3] I. Pomeranz and S. M. Reddy, "On diagnosis and correction of design errors", in *Proc. Intl. Conf. Computer-Aided Design*, Nov. 1993, pp. 500–507.

[4] A. Srinivasan A. Kuehlmann, D. I. Cheng and D. P. LaPotin, "Error diagnosis for transistor-level verification", in *Proc. Design Automation Conf.*, June 1994, pp. 218–224.

[5] M. Tomita, T. Yamamoto, Sumikawa F, and K. Hirano, "Rectification of multiple logic design errors in multiple output circuits", in *Proc. Design Automation Conf.*, June 1994, pp. 212–217.

[6] I. Pomeranz and S. M. Reddy, "On error correction in macro-based circuits", in *Proc. Intl. Conf. Computer-Aided Design*, Nov. 1994, pp. 568–575.

[7] S. Y. Huang, K. T. Cheng, K. C. Chen, and D. I. Cheng, "Errortracer: A fault simulation based approach to design error diagnosis", in *Proc. Intl. Test Conf.*, Nov. 1997, pp. 974–981.

[8] S. Y. Huang, K. T. Cheng, K. C. Chen, and J. J. Lu, "Fault-simulation based design error diagnosis for sequential circuits", in *Proc. Design Automation Conf.*, June 1998, pp. 632–637.

[9] V. Boppana and M. Fujita, "Modeling the unknown! towards model-independent fault and error diagnosis", in *Proc. Intl. Test Conf.*, Oct. 1998, pp. 1094–1101.

[10] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital System Testing and Testable Design*, New York, NY: Computer Science Press, 1990.

[11] P-Y. Chung, Y-M. Wang, and I. N. Hajj, "Diagnosis and correction of logic design errors in digital circuits", in *Proc. Design Automation Conf.*, June 1993, pp. 503–508.

[12] H-T. Liaw, J-H. Tsaih, and C-S. Lin, "Efficient automatic diagnosis of digital circuits", in *Proc. Intl. Conf. Computer-Aided Design*, Nov. 1990, pp. 464–467.

[13] P-Y. Chung and I. N. Hajj, "Accord: Automatic catching and correction of logic design errors in combinational circuits", in *Proc. Intl. Test Conf.*, Sept. 1992, pp. 742–751.

[14] S. B. Akers, B. Krishnamurthy, S. Park, and A. Swaminathan, "Why is less information from logic simulation more useful in fault simulation?", in *Proc. Intl. Test Conf.*, Sept. 1990, pp. 786–800.

[15] R. C. Aitken and P. C. Maxwell, "Better models or better algorithms? Techniques to improve fault diagnosis", *Hewlett-Packard Journal*, pp. 110–116, Feb. 1995.

[16] T. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits", in *Proc. European Design Automation Conf.*, Feb. 1991, pp. 214–218.