ErrorTracer: Design Error Diagnosis Based on Fault Simulation Techniques

Shi-Yu Huang and Kwang-Ting Cheng, *Senior Member, IEEE*

***Abstract*—This paper addresses the problem of locating error sources in an erroneous combinational or sequential circuit. We use a fault simulation-based technique to approximate each internal signal's correcting power. The correcting power of a particular signal is measured in terms of the signal's *correctable set,* namely, the maximum set of erroneous input vectors or sequences that can be corrected by resynthesizing the signal. Only the signals that can correct every given erroneous input vector or sequence are considered as a potential error source. Our algorithm offers three major advantages over existing methods. First, unlike symbolic approaches, it is applicable for large circuits. Second, it delivers more accurate results than other simulation-based approaches because it is based on a more stringent condition for identifying potential error sources. Third, it can be generalized to identify multiple errors theoretically. Experimental results on diagnosing combinational and sequential circuits with one and two random errors are presented to show the effectiveness and efficiency of this new approach.**

***Index Terms*—Design automation, error correction, fault diagnosis, simulation.**

I. Introduction

DURING the very large scale integration design process, functional mismatches between a given specification and the final implementation often occur. Once a functional mismatch is found by the verification tool, the designer faces the taunting task of design error diagnosis—a process that identifies or narrows down the error sources in the implementation, so as to assist the subsequent error correction process [2], [8], [11], [18], [20]–[22]. Due to the difficulty of diagnosing a sequential circuit, most previous approaches have focused on the combinational diagnosis. Most of them also assume the circuit under diagnosis is *single-signal correctable,* i.e., the circuit can be completely corrected by resynthesizing a particular signal f as shown in Fig. 1. Such a signal f is called a single-fix signal hereafter.

Most approaches to error diagnosis can be classified into two categories: 1) simulation-based approaches and 2) symbolic approaches. The simulation-based approaches first derive a number of input vectors that can differentiate the implementation and the specification. These binary or three-valued input vectors are called *erroneous vectors* in the sequel. By simulating each erroneous vector, the potential error region can then be trimmed down gradually. The heuristic for eliminating those signals that *cannot* be error sources vary from one to another [14], [16], [21], [22], [25]–[27], [29].

Manuscript received February 6, 1998; revised July 16, 1998. This work was supported in part by the National Science Foundation (NSF) under Grant MIP-9503651 and in part by the California MICRO program through Fujitsu/Rockwell. This paper was recommended by Associate Editor A. Saldanha.

S.-Y. Huang was with the University of California, Santa Barbara, CA 93106 USA. He is now with Worldwide Semiconductor Manufacturing Corp., HsinChu, Taiwan, ROC (e-mail: sy.huang@wsmc.com.tw).

K.-T. Cheng is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: timcheng@ece.ucsb.edu).

Publisher Item Identifier S 0278-0070(99)06630-0.

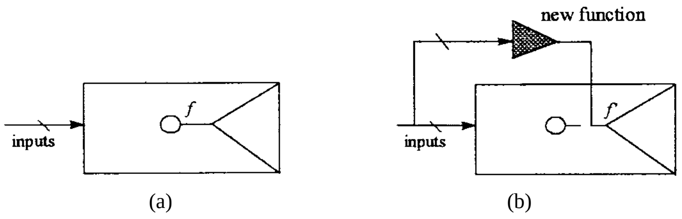


Fig. 1. Resynthesis of an internal signal. (a) Original implementation and (b) implementation after f is resynthesized.

Pomeranz and Reddy proposed a filter [21] for locating the error sites of an erroneous combinational circuit. The idea is based on the observation that if v cannot sensitize a discrepancy from a signal f to a primary output I_k, then the erroneous output response of I_k with respect to v cannot be corrected by changing the function of f. In other words, f is *not* responsible for the erroneous I_k with respect to vector v if $v \notin (dI_k/df)$. As will be discussed later, our approach could be viewed as an enhancement of this method as diagnosing a combinational circuit.

Kuehlmann *et al.* proposed another heuristic [14], referred to as *back propagation* here. Similar to the *critical path tracing* techniques used in fault simulation [4], it traces back from each erroneous primary output toward the primary inputs to find candidate error locations. This approach is more general in the sense that it does not rely on an error model that consists of most frequently occurred error types as defined in [1], (e.g., an inverter is missing). Relatively speaking, this approach is more efficient than the one in [21], while less accurate. The accuracy of back propagation can be further improved without sacrificing the efficiency through a technique called *observability measure* proposed by Veneris *et al.* [27].

On the other hand, the symbolic approaches do not enumerate any erroneous vector [8], [17]–[20], [24]. They primarily rely on ordered binary decision diagram (OBDD) [6] to formulate the necessary and sufficient condition of a single-fix signal. Based on this formulation, the signals that are most responsible for the incorrect output functions can be directly identified. In comparison, the symbolic approaches are more accurate than the simulation-based approaches and also extendible to multiple errors [19]. However, constructing the required BDD representations may cause memory explosion for large circuits. As will be discussed later, our diagnosis algorithm for combinational circuits is quite similar to the symbolic approaches in the concept of what signals should be regarded as error signals. In some sense, it can be viewed as an approximation of the symbolic approaches that achieves the same accuracy if given infinite time.

Given an erroneous implementation and a specification described at a higher level of abstraction, e.g., the register-transfer (RT) level, the first step of diagnosis is usually performing the fast-pass synthesis on the specification to derive a gate-level representation of the specification. After that, the gate-to-gate error diagnosis can be conducted. It is likely that one-to-one flip-flop (FF) correspondence between the implementation and the gate-level specification does not exist and, thus, the combinational diagnosis approaches cannot be applied. A sequential error diagnosis approach is needed for this kind of situations. However, due to the even higher difficulty and complexity, only few papers in the literature have addressed the problem of diagnosing design errors in sequential circuits [9], [23], [30]. The method discussed in [23], modeling the error in the state transition table, only targets small controllers. The approach proposed in [9] is not very general in the sense that it only focuses on small feedback-free circuits, or finite state machines that have one-to-one state correspondence with their specifications. Another approach, extending a combinational backward error tracing heuristic [29] to the iterative array model, was proposed in [30]. In this approach, a restricted error hypothesis (containing three types of wrong-gate errors) is used. It has two major limitations. First, it relies on a restricted error hypothesis to reduce the complexity of the diagnosis process, and thus, it may fail when the design error is not modeled in their hypothesis. Second, their approach cannot deal with multiple errors.

In this paper, we perform error diagnosis through a fault simulation process taking a number of erroneous vectors as the inputs. A set of erroneous vectors are generated in advance during the functional simulation process [15] or by verification tools. Our approach is based on a notion called *correctable set.* A signal's correctable set is defined as the set of erroneous vectors that can be corrected by resynthesizing the signal. Let v be an erroneous vector and f be a signal in the erroneous implementation. We show that whether v is correctable by resynthesizing f can be *precisely* determined by simulating input vector v for stuck-at faults at f. Like most simulation-based approaches, our algorithm is a monotone filtering process. Initially every signal is considered as a candidate of single-fix signals. We simulate every erroneous vector in the given erroneous vector set for the stuck-at faults at each candidate signal. According to the fault simulation results, if a signal is proven unable to correct the erroneous vector under simulation, then it is a false candidate. False candidates are removed immediately from the candidate list before simulating the next erroneous vector.

Our approach does not incorporate an error model and, thus, is suitable for general types of design errors. The accuracy of our approach is related to the input vectors simulated. The larger the number of vectors simulated, the higher the accuracy. Theoretically speaking, if we can afford to simulate the complete set of erroneous vectors, then our approach is as accurate as the symbolic methods. The experimental results show that this method is very effective for single-signal correctable circuits. On average, 99% of the signals can be filtered out from the original candidate list after simulating only 32 erroneous vectors.

Another advantage of this approach over the other simulation-based approaches is that: this approach can be generalized for circuits with multiple errors. For diagnosing multiple errors, we search for a set of signals that can *jointly* fix the erroneous circuit. Also, a two-stage fault simulation procedure is proposed to speed up the multiple error diagnosis process. This two-stage procedure, taking advantage of the topological dominance relation between signals, does not cause any loss of accuracy. In this paper, we present the results of diagnosing one and double errors for every ISCAS'85 combinational benchmark circuit. The larger circuits in this benchmark set can not be handled by the BDD-based symbolic approaches.

This approach is further generalized for sequential circuits. We first derive the necessary and sufficient condition of whether an erroneous input sequence can be corrected by changing the function of a particular internal signal or not. Similar to the combinational cases, we then search for the potential error signals based on this condition through a modified sequential fault simulation process. Experimental results on some of ISCAS'89 benchmark circuits injected with one and two errors will also be presented.

The rest of this paper is organized as follows. Section II gives the basic assumptions and definitions. Section III describes our combinational diagnosis algorithm for single-signal correctable circuits. In Section IV, we generalize this algorithm for multiple errors. In Section V, we generalize this technique for sequential circuits. We present the experimental results in Section VI, and conclude in Section VII.

II. Basic Definitions

We assume that the specification and the erroneous implementation are given as gate-level circuits. Both share the same set of primary inputs, denoted as $\{x_1, x_2, \ldots, x_n\}$. The primary outputs of the specification and the implementation are denoted as $\{S_1, S_2, \ldots, S_m\}$ and $\{I_1, I_2, \ldots, I_m\}$, respectively.

Definition 1: (S_i, I_i) is called the ith *primary output pair.*

Definition 2: Joint network is a network obtained by connecting the primary inputs of the specification and the implementation together as shown in Fig. 2. For the rest of this paper, we also refer to the specification as C_1 and the implementation as C_2.

Definition 3: Erroneous vector is a binary input vector that can differentiate at least one primary output pair.

Definition 4—Erroneous Output: If an erroneous vector v can differentiate the ith primary output pair, then I_i is an erroneous output; otherwise, I_i is a correct output with respect to v. Given an erroneous vector v, we can then partition the

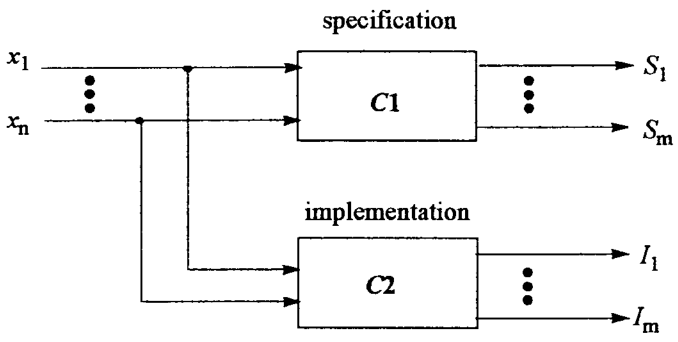


Fig. 2. Joint network.

primary outputs of C_2 into two groups with respect to this vector: 1) erroneous output group, Error_$PO(v)$ and 2) correct output group, Correct_$PO(v)$.

Definition 5—Sensitization Set: For a signal f and a primary output I_i in C_2, the sensitization set, denoted as $SEN_i(f)$, is the set of input vectors that can sensitize a discrepancy from f to I_i. Boolean difference dI_i/df is the characteristic function of the sensitization set $SEN_i(f)$. $SEN_i(f)$ represents those input vectors for which signal f determines the value at I_i.

III. Single Error Diagnosis for Combinational Circuits

In this section we begin with the introduction of the notion of correctable vector. This is followed by the necessary and sufficient condition for a single-fix signal from a slightly different point of view than the ones in [18]–[20], and [24]. After that, we present our overall algorithm.

A. Correctability

Definition 6—Correctable Vector: An erroneous vector v is *correctable* by a signal f in C_2 if there exists a new function for signal f such that v is not an erroneous vector for the resulting new circuit.

Proposition 1: Let v be an erroneous vector and f be a signal in C_2. Then v is correctable by f if and only if the following two conditions are satisfied:

- v *can* sensitize a discrepancy from f to every erroneous primary output of C_2, i.e., for every primary output I_i in Error_$PO(v)$, $v \in SEN_i(f)$;
- v *cannot* sensitize a discrepancy from f to any correct primary output of C_2, i.e., for every primary output I_i in Correct_$PO(v)$, $v \notin SEN_i(f)$.

Proof: Let f^{new} be a Boolean function that disagrees with the original function of f only on the input vector v. Then after replacing signal f with the new function f^{new}, a discrepancy is injected at f. If the above two conditions are satisfied, then the response of every erroneous output toggles and thus becomes correct (because v *can* sensitize a discrepancy from f to every one of them). At the same time, every originally correct output remains correct (because vector v *cannot* sensitize a discrepancy from f to any one of them). On the other hand, it can be shown that if either of the above two conditions are not satisfied, then at least one erroneous primary output will remain erroneous, or one originally correct output will become erroneous, regardless of what the new function is. (Q.E.D.)

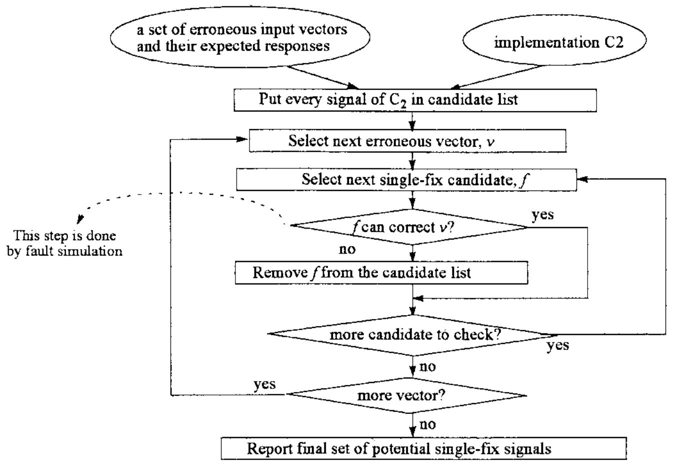

a set of erroneous input vectors and their expected responses
implementation C2
Put every signal of C2 in candidate list
Select next erroneous vector, v
Select next single-fix candidate, f
This step is done by fault simulation
f can correct v?
yes
no
Remove f from the candidate list
more candidate to check?
yes
no
yes
more vector?
no
Report final set of potential single-fix signals

Fig. 3. Flow for single-signal correctable circuits.

Definition 7—Single-Signal Correctable: If the implementation C_2 can be completely corrected by resynthesizing a signal f, then C_2 is single-signal correctable, and the signal f is called a *single-fix signal.*

Proposition 2—Necessary and Sufficient Condition for Signal-Fix Signal: A signal f is a single-fix for C_2 if and only if every erroneous vector is correctable by f.

Proof: Here, we make no distinction between a signal and its function. If a signal f is a single-fix signal, then by definition every erroneous vector is correctable by f. In the following we show that f is a single-fix signal if every erroneous vector is correctable by f. Let $f^{\text{new}} = f \cdot E' = f' \cdot E$, where E is the characteristic function of the entire erroneous vector set. Intuitively, f^{new} can be interpreted as a function that *agrees* with f on all nonerroneous vectors, while *disagrees* on all erroneous vectors. It can be shown that f^{new} is a fix function at f by Proposition 1 and, thus, we conclude that f is a single-fix signal if every erroneous vector is correctable by f. (Q.E.D.)

B. Algorithm for Single Error Diagnosis

In this subsection, we describe the general flow for diagnosing a single-signal correctable circuit. In order to handle large circuits, we do not attempt to identify the single-fix signals using BDD. Instead, we employ an iterative filtering process which reduces the number of single-fix candidate signals gradually. The overall flow is shown in Fig. 3.

This process takes as inputs the erroneous implementation, and a set of erroneous vectors and their expected output responses. At the beginning, we assume every signal in C_2 is a single-fix candidate. Then our algorithm starts a two-level loop. The outer loop enumerates every erroneous vector. The inner loop iterates through every single-fix candidate signal. For each erroneous vector v and a target single-fix candidate f, we examine if v is correctable by f by fault simulation (will be explained later). If signal f cannot correct vector v, then f cannot be a single-fix signal. Therefore, it is safe to remove

f from the candidate list. After we have examined every erroneous vector and eliminated false single-fix candidates, a set of potential single-fix signals is derived.

C. Correctability Check via Fault Simulation

Given an erroneous vector v and a signal f, *correctability check* is to decide whether v is correctable by f. It is an operation to verify the following two conditions: 1) if v *can* sensitize a discrepancy from f to *every* erroneous output in response to v and 2) if v *cannot* sensitize a discrepancy from f to any correct output in response to v. Both conditions should be satisfied to assure that f can correct v. Proposition 3 shows that this can be checked by simulating vector v for stuck-at-0 and stuck-at-1 faults at f. In the following discussion, we use $C_{2|f=0}$ to represent the faulty implementation with f-stuck-at-0 fault. Similarly, $C_{2|f=1}$ represents the faulty implementation with f-stuck-at-1 fault.

Proposition 3: A signal f can correct an erroneous vector v if and only if the faulty circuit $C_{2|f=0}$ or $C_{2|f=1}$ has the same output responses as the specification C_1 with respect to the input vector v.

Proof: It is obvious that if $C_{2|f=0}$ or $C_{2|f=1}$ is equivalent to C_1 with respect to v, then v is correctable. On the other hand, if neither $C_{2|f=0}$ nor $C_{2|f=1}$ is equivalent to C_1 with respect to v, then it can be shown that at least one of the two criteria of Proposition 1 cannot be satisfied, and thus, there does not exist a new function for signal f to correct v. (Q.E.D.)

Based on Proposition 3, we can simulate the generated erroneous the input vector set for the stuck-at faults at each candidate signal to gradually prune out the false single-fix candidates. Based on this formulation, any kind of efficient fault simulation techniques, e.g., differential fault simulation [7], can be applied to improve the efficiency. This process can be further sped up by exploring the topological dominance relation between signals. Let f and *dom* be two signals, and *dom* be a topological dominator of f. In other words, every path originated from f to any primary output passes through signal *dom*. In [17], it has been proven that if *dom* cannot correct an erroneous vector v, then f cannot correct it, either. Therefore, once a false single-fix candidate is found, we can immediately remove its dominated signals from the candidate list as well. Fig. 4 shows the revised routine of simulating one erroneous vector for correctability check.

First, we sort the candidate list in a fanout-first order, (i.e., every signal is after its transitive fanout signals). Given an erroneous vector, we examine each signal according to this order. Then differential fault simulation [7] is performed for the stuck-at-0 and stuck-at-1 faults for each target signal, say f. The simulation results are compared with prestored output responses of the specification to decide the correctability. If the target signal f fails to correct v, then we drop not only f but also every signal dominated by f from the candidate list. The correctability check iterates until every signal remaining in the candidate list has been checked. In this revised routine, some candidate signals may be dropped without fault simulation because one of their dominators has been proven unable to correct the given erroneous vector.

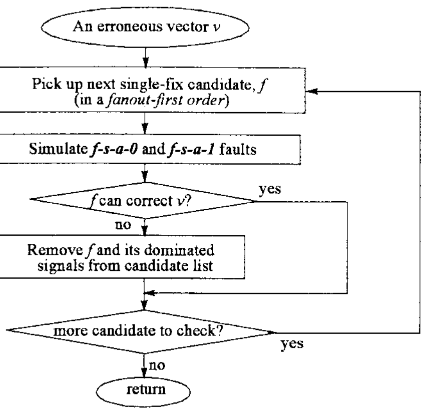

An erroneous vector v
Pick up next single-fix candidate, f (in a *fanout-first order*)
Simulate ***f-s-a-0*** and ***f-s-a-1*** faults
f can correct v?
yes
no
Remove f and its dominated signals from candidate list
more candidate to check?
yes
no
return

Fig. 4. Checking correctability for an erroneous vector.

IV. Multiple Error Diagnosis for Combinational Circuits

In this section we generalize the approach for dealing with circuits with multiple errors. In general, the number of errors introduced in the erroneous implementation is not known during the diagnosis process. Therefore, we first try to find a potential single-fix signal. If there does not exist such signals, then we continue to search for multiple signals that can jointly correct the implementation completely. First, we define k-correctable vector and k-correctable circuit. Then we show how to perform k-correctability check via fault simulation and then present a two-stage algorithm.

A. k-Correctability

Definition 8—k-Correctable Vector: An erroneous vector v is k-correctable by a set of k signals in C_2, $\sigma = \{f_1, f_2, \ldots, f_k\}$, if there exists a new function for each signal f_i in σ such that v is not an erroneous vector for the resulting new circuit.

Definition 9—k-Correctable Circuit: If the implementation C_2 can be completely corrected by resynthesizing a set of k signals, $\sigma = \{f_1, f_2, \ldots, f_k\}$, then C_2 is k-correctable and σ is called a *fix set.*

For a given signal set σ, an enumeration over σ is an assignment that assigns a binary value to each signal in σ. For a signal set with k signals, there will be 2^k different enumerations, and each enumeration corresponds to a faulty circuit with multiple stuck-at faults. For example, consider a set of two signals $\sigma = \{f_1, f_2\}$. There will be four different enumerations, namely, $\{f_1 = 0, f_2 = 0\}$, $\{f_1 = 0, f_2 = 1\}$, $\{f_1 = 1, f_2 = 0\}$, $\{f_1 = 1, f_2 = 1\}$. Among them, $\pi = \{f_1 = 0, f_2 = 0\}$ defines a faulty circuit with a double-fault (f_1 stuck-at-0 and f_2 stuck-at-0). Proposition 4 shows that in order to decide whether an erroneous vector is k-correctable by a set of signals σ, fault simulation needs to be performed on every one of the 2^k faulty circuits defined over σ.

Proposition 4: Let σ be a set of k signals and π_i be one of 2^k enumerations defined over σ, $i = [1, \ldots, 2^k]$. An erroneous input vector v is k-correctable by σ if and only if there exists a faulty circuit $C_{2|\pi i}$ that has the same output responses as the specification C_1 with respect to v, where $C_{2|\pi i}$ denotes the faulty implementation defined by the enumeration π_i.

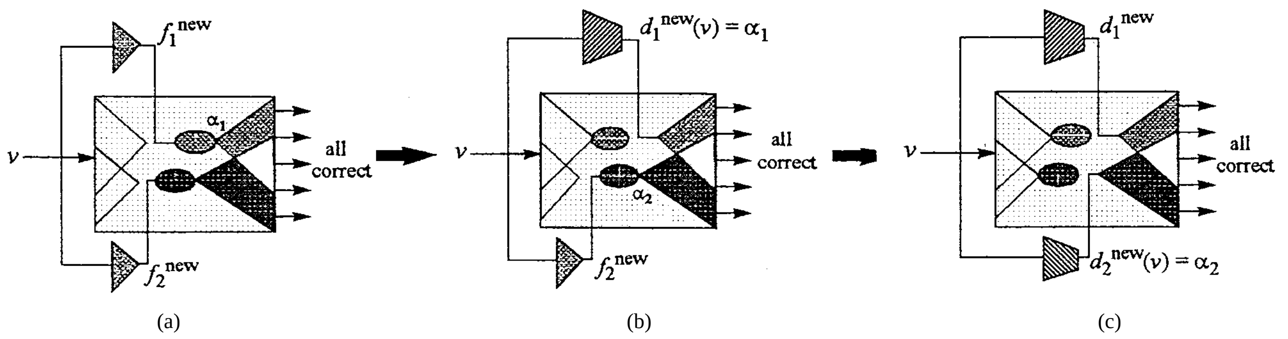


Fig. 5. Illustrations for the proof of Proposition 5.

Proof: ($\Rightarrow$) Assume that v is k-correctable by σ. Then, by definition, there exists a new Boolean function at each signal in σ such that the output responses of the resulting circuit are all correct with respect to v. Let these k Boolean functions evaluate to $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ with respect to v, where α_i is a binary value. Let f-s-a-α denote signal f stuck-at-α fault. Then circuit C_2 with k stuck-at faults $\{f_1$-s-a-α_1, f_2-s-a-α_2, …, f_k-s-a-$\alpha_k\}$ also produces all correct output responses with respect to v. Thus, we conclude that there exists a $C_{2|\pi i}$ that has the same output responses as the specification C_1 with respect to v.

($\Leftarrow$) Assume that there exists a $C_{2|\pi i}$ that has the same output responses as the specification C_1 with respect to v and $\pi_i = \{f_1 = \alpha_1, f_2 = \alpha_2, \ldots, f_k = \alpha_k\}$. Suppose f_i^{new} $(1 \leq i \leq k)$ is a Boolean function such that $f_i^{\text{new}}(v) = \alpha_i$. Then the replacement of each signal f_i in σ with the new function f_i^{new} will result in a new implementation that produces all correct output responses with respect to v. Thus, by definition, we conclude that v is k-correctable by σ. (Q.E.D.)

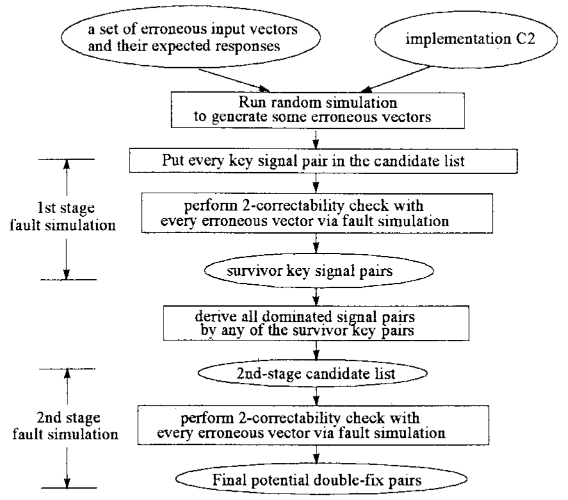
Like in the case of finding single-fix signals, the topological dominance relation is useful in reducing the complexity. To explain this speedup technique, we first introduce a dominance relation defined between two sets of signals.

Definition 10—Set Dominance Relation: Let $\sigma_1 = \{f_1, f_2, \ldots, f_k\}$ and $\sigma_2 = \{d_1, d_2, \ldots, d_k\}$. If f_i is topologically dominated by d_i for $1 \leq i \leq k$, then σ_1 is dominated by σ_2, denoted as $\sigma_1 < \sigma_2$. We refer to σ_1 as a subordinate set of σ_2.

Proposition 5: Let σ_1 and σ_1 be two set of signals with the same cardinality, and $\sigma_1 < \sigma_2$. If σ_2 cannot correct an erroneous vector v, then σ_1 cannot correct v either.

Proof: For simplicity without losing generality, we assume that σ_1 and σ_2 both have only two signals, $\sigma_1 = \{f_1, f_2\}$ and $\sigma_2 = \{d_1, d_2\}$, where d_1 dominates f_1, and d_2 dominates f_2. We will show in the following that if σ_1 can correct v, then σ_2 can correct v as well. Fig. 5(a) shows an implementation where signals f_1 and f_2 are replaced by two new functions, denoted as f_1^{new} and f_2^{new}, so that every output response with respect to v is correct. Suppose the signal response at d_1 in the circuit in Fig. 5(a) is α_1, then we can create the circuit shown in Fig. 5(b) from the circuit in Fig. 5(a) by the following two steps.

1) Remove the new function at f_1 and reconnect signal f_1.
2) Resynthesize signal d_1 with a new function that maps v to α_1, i.e., $d_1^{\text{new}}(v) = \alpha_1$.



a set of erroneous input vectors and their expected responses
implementation C2
Run random simulation to generate some erroneous vectors
Put every key signal pair in the candidate list
1st stage fault simulation
perform 2-correctability check with every erroneous vector via fault simulation
survivor key signal pairs
derive all dominated signal pairs by any of the survivor key pairs
2nd-stage candidate list
2nd stage fault simulation
perform 2-correctability check with every erroneous vector via fault simulation
Final potential double-fix pairs

Fig. 6. Two-stage algorithm for diagnosing double errors.

The output responses of this new circuit with respect to v are still all correct, because the above transformations do not affect any of the side inputs to the fanout cone of d_1. Suppose the signal response at d_2 in the circuit in Fig. 5(b) is α_2. Similarly, we can create the circuit shown in Fig. 5(c) from the circuit in Fig. 5(b) by the following two steps.

1) Remove the new function at f_2 and reconnect signal f_2.
2) Resynthesize signal d_2 with a new function that maps v to α_2, i.e., $d_2^{\text{new}}(v) = \alpha_2$.

Again, the output responses of this new circuit with respect to v are still all correct. Therefore, we conclude that there exists new functions at d_1 and d_2 to fix v. (Q.E.D.)

B. Two-Stage Algorithm for Multiple Errors

Based on Proposition 5, we propose a two-stage fault simulation algorithm for diagnosing multiple-errors. For the sake of simplicity, we discuss the case of $k = 2$. That is, we assume that the implementation C_2 is double-signal correctable. The overall algorithm is shown in Fig. 6.

A signal d in C_2 is referred to as a *key signal* if d is not dominated by any other signal. Similarly, if d_1 and d_2 are both key signals in C_2, then $\{d_1, d_2\}$ is called a key signal pair. In the first stage, we consider only key signal pairs as candidates. For an erroneous vector v and a candidate key signal pair $\sigma = \{d_1, d_2\}$, we perform simulation on each of the four possible faulty circuits $C_{2|\{d_1=0, d_2=0\}}$, $C_{2|\{d_1=0, d_2=1\}}$,

TABLE I
RESULTS OF DIAGNOSING OPTIMIZED ISCAS'85 BENCHMARK CIRCUITS INJECTED WITH ONE ERROR

circuit	# signals in C2	**# potential fix signals**	pessimistic lower bound	hit ratio	**suspect ratio**	time (seconds)		
						random simulation	fault simulation	**total**
C432	175	**3.4**	2.3	100%	**1.94%**	4	1	**5**
C499	410	**4.7**	2.3	100%	**1.15%**	11	1	**12**
C880	292	**5.2**	2.7	100%	**1.78%**	6	1	**7**
C1355	410	**5.1**	2.5	100%	**1.32%**	15	1	**16**
C1908	357	**4.3**	2.4	100%	**1.20%**	15	2	**17**
C2670	803	**10.6**	4.3	100%	**1.32%**	38	5	**43**
C3540	1189	**7.3**	2.6	100%	**0.61%**	54	6	**60**
C5315	1248	**4.7**	2.7	100%	**0.38%**	76	8	**84**
C6288	2339	**2.9**	1.8	100%	**0.13%**	27	33	**60**
C7552	2187	**18.0**	3.4	100%	**0.82%**	185	20	**205**
Average	941	**6.6**	2.7	100%	**1.06%**	43	8	**51**

$C_{2|\{d_1=1, d_2=0\}}$, and $C_{2|\{d_1=1, d_2=1\}}$. If any one of these four faulty circuits have the same output response as the specification with respect to v, then v is two-correctable by σ. Otherwise, σ is a false candidate pair and should be removed from the candidate list. After the fault simulation process has iterated through every erroneous vector and candidate pair, we obtain a set of survivor key signal pairs that are potential to be fix pairs. However, these pairs are only a subset of the potential fix pairs. Every subordinate pair of each of them is also a possible fix pair. For example, suppose $\sigma = \{d_1, d_2\}$ is a potential fix pair after the first stage. Let $\{f\}$ and $\{g_1, g_2\}$ be the sets of signals dominated by d_1 and d_2, respectively. Then $\{(d_1, g_1), (d_1, g_2), (f, g_1), (f, g_2), (f, d_2)\}$ are possible fix pairs, too. However, they are not examined in the first stage. Note that only the subordinate pairs of those surviving key signal pairs need to be further checked in the second stage. The subordinate pairs of those false key signal pairs filtered out in the first stage are guaranteed not fix pairs. Usually, the number of subordinate pairs that need to be checked in the second stage is very small as will be shown in Section V.

C. Complexity Analysis

In general, the number of errors introduced in the implementation is not known prior to the diagnosis process. The complexity of the search for multiple-fix signals may grow rapidly. For example, for a circuit that can only be fixed by resynthesizing at least k signals, we may need to check every set of signals with cardinality less than or equal to k. Also the complexity of each correctability check may go exponentially in terms of the cardinality of the candidate set under consideration. Thus, the overall complexity of multiple-error diagnosis in the worst case is

$$\begin{aligned}&\mathrm{Comb}(n, 1) \cdot O(2^1) + \mathrm{Comb}(n, 2) \cdot O(2^2) + \cdots \\ &\quad + \mathrm{Comb}(n, k) \cdot O(2^k)\end{aligned}$$

where n is the number of signals in the implementation C_2, and $\mathrm{Comb}(n, k)$ represents the combination number of choosing k signals from n signals. In Section V, we will show the experimental results of diagnosing circuits injected with two random errors. However, in practice, the identification of the multiple-fix signals may become too time-consuming for large circuits with more than two errors. The approach proposed in [14], approximating each signal's probability of being an error source using heuristics, may become more appropriate to allow the designer to locate-and-fix one error at a time.

V. EXPERIMENTAL RESULTS OF COMBINATIONAL DIAGNOSIS

We have implemented our algorithm in C language in the environment of SIS [28]. The program is named ErrorTracer, which incorporates a differential fault simulator as described in [7]. Our experiments for combinational diagnosis are performed on every ISCAS benchmark circuit. For each circuit, we first optimize it by the optimization script, *script.rugged*, to obtain the implementation. Then we decompose it into AND/OR gates using SIS command "*tech_decomp -a* 5 *-o* 5." For generating erroneous implementation, gate type errors are injected using an error injection program [14]. This program randomly selects a logic gate and then randomly scrambles its truth-table. Note that, similar to symbolic approaches, our approach is based on the notion of resynthesis, and thus, is not restricted to the types of errors introduced in the implementation.

Table I shows the results of single-error diagnosis. The program is run 20 times for each benchmark circuit; each of them uses a different single-error implementation. In the preprocessing stage, we run random simulation until 32 erroneous vectors are collected, or 16 000 random patterns have been simulated. For very few cases, no erroneous vector is found after simulating 16 000 random patterns. For each of these cases, our formal equivalence checker, AQUILA [10], successfully proves that the error-injected implementation is actually functionally equivalent to the specification. Table I shows the average results for those real erroneous implementations. The meaning of some columns are explained as follows.

1) Number of Potential Fix Signals: This is the number of potential single-fix signals delivered by our program. On average, our algorithm outputs 6.6 potential single-fix signals for ISCAS'85 benchmark circuits. Among these single-fix signals, a signal that does not dominate any other single-fix signals is even more likely to be the real location where the error occurs. Although this heuristic is not always true in our experiments, it is helpful in most cases in predicting the real error sources among the reported potential single-fix signals. Fig. 7 shows the curve of the number of potential single-fix signals versus the number of erroneous vector simulated during the diagnosis process for a single-error implementation of C6288.

TABLE II
RESULTS OF DIAGNOSING OPTIMIZED ISCAS'85 BENCHMARK CIRCUITS INJECTED WITH TWO ERRORS

circuit	# signals in C2	# candidate pairs checked			speedup	**potential fix pairs**	pessimis. lower bound	hit ratio	time (seconds)		
		stage1	stage2	total					random simulation	fault simulation	**total**
C432	175	1035	13	1048	14.5	**4**	2	100%	1	4	**5**
C499	410	4753	81	4834	17.3	**12**	4	100%	1	12	**13**
C880	292	3916	32	3948	10.8	**6**	2	100%	1	7	**8**
C1355	410	4753	36	4789	17.5	**8**	6	100%	1	5	**7**
C1908	357	4371	361	4732	-13.4	**56**	12	100%	1	15	**16**
C2670	803	2701	135	2836	113.5	**20**	3	100%	2	19	**21**
C3540	1189	20100	1176	21276	33.2	**72**	8	100%	3	134	**137**
C5315	1248	34191	5	34196	22.8	**2**	2	100%	3	138	**141**
C6288	2339	264628	8	264632	10.3	**4**	3	100%	4	3240	**3244**
C7552	2187	41401	453	41494	57.6	**9**	3	100%	15	1153	**1168**
Average	941	38148	230	38378	21.1	**19.3**	4.5	100%	3	473	**476**

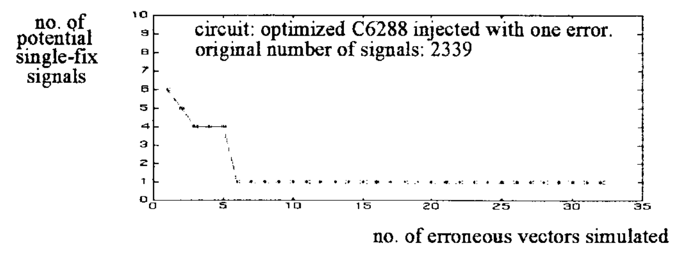


Fig. 7. The curve of the accuracy versus the number of erroneous vectors that are fault simulated.

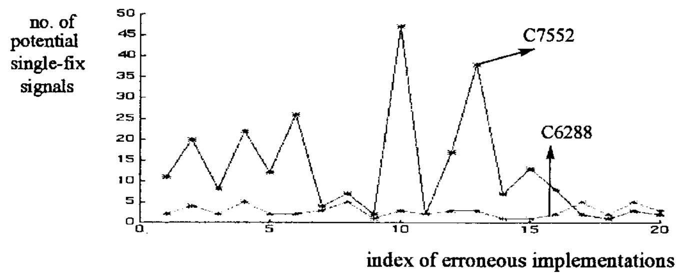


Fig. 8. The number of potential single-fix signals for C6288 and C7552 with single error.

Initially, every signal is regarded as a potential single-fix signal, so there are totally 2339 candidates. Only simulating one erroneous vector, our approach narrows down the error region to only six signals. After simulating six erroneous vectors, we precisely pin-point the location of the injected error. This curve indicates that our criterion for a signal to be a potential single-fix signal is very stringent, and thus, is able to filter out most false candidates rapidly. In our experience, the numbers of potential single-fix signals for most cases saturate quickly after simulating less than ten erroneous vectors.

Fig. 8 shows the final numbers of potential single-fix signals after simulating 32 erroneous vectors for 20 single-error implementations of C6288 and C7552. It can be seen that, C7552 has a large variation from one erroneous implementation to another. On the other hand, the potential single-fix signals of C6288 are always small. This may indicate that C6288, as a 16-b multiplier, is easier to diagnose.

2) Lower Bound: This is a *pessimistic* lower bound on the total number of single-fix signals. It is obtained by counting the number of dominators of the injected error signal plus one. This number is pessimistic because some single-fix signals may not dominate the injected error signal and, thus, could be ignored in this calculation.

3) Hit Ratio: This indicates the probability that the injected error signal is included in the delivered set of potential fix signals. Our program will not overlook the real error signal, so the hit ratio is always 100%.

4) Suspect Ratio: This is ratio of the number of potential fix singles to the total number of signals in C_2. The average obtained by our program is 1.06%, which means almost 99% signals are disqualified as a single-fix signal.

5) CPU Time: The CPU time on 150 MHz Sparc20 consists of the random simulation time and the fault simulation time. The random simulation time for generating 32 erroneous vectors (the average is 40 s) is a more dominating factor than the fault simulation time (the average is 8 s). For large circuits, the heuristic proposed in [14] can be used as a fast pass before ErrorTracer to further reduce the CPU time.

Table II shows the results of diagnosing circuits injected with two random errors. Since this is a more time-consuming process, we only diagnose one erroneous implementation for each benchmark circuit. Each of these erroneous implementation is proven not single-signal correctable by our program (i.e., every signal is disqualified as a single-fix). Note that if an implementation is single-signal correctable, then the number of potential double-fix pairs would be huge. For example, if f is a single-fix, then the signal pair (f, x), where x is any signal in C_2, would be a double-fix pair. This leads to an even more timing-consuming double-error diagnosis process because our false-candidate dropping technique is not effective for such cases. Several columns of Table II are discussed as follows.

6) Number of Candidate Pairs Checked: This is the total number of candidate pairs checked by two stages of fault simulation. Usually the number of candidates in the second stage is negligible compared to the one in the first stage. This number also implies the speedup factor by exploring the set dominance relation. Consider C432 for example. The number of signals in C_2 is 175, hence, the total number of candidate pairs without using dominance relation is $(175)(175-1)/2 = 15\,225$. In our algorithm, the total number of candidate pairs is reduced to only $1035 + 13 = 1048$. Therefore, the dominance relation reduces the number of candidate pairs that need to checked from $15\,225$ to 1048, and the speedup is 14.5 times.

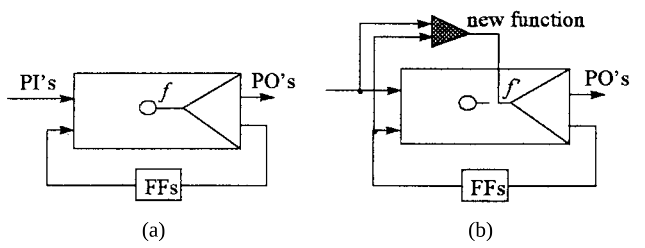


Fig. 9. A correctable input sequence E by resynthesizing a signal. (a) E is erroneous and (b) E is no longer erroneous.

The average speed up factor for the entire set of benchmark circuits is 21.

7) Hit Ratio: This is also 100% for double-error diagnosis using our program.

8) Number of Potential Fix Pairs: This number is found to be larger than the number of potential single-fix signals in the single-error implementations.

9) CPU Time: In these cases, the CPU times are mostly dominated by the fault simulation times here. It is proportional to the number of candidate pairs. C6288 is particularly time-consuming because a high percentage of its signals have multiple fanout branches, and thus, the dominance relation does not reduce the number of candidate pairs substantially.

VI. Generalization for Sequential Circuits

In this section we extend the above idea to diagnose a sequential circuit. First, we show the necessary and sufficient condition of whether an erroneous input sequence can be corrected by resynthesizing a particular signal. Then, we discuss how to check this condition via fault simulation. The approach is then generalized for multiple errors.

A. Correctable Input Sequence

Definition 11—Correctable Sequence: An erroneous input sequence E is called *correctable* by signal f in C_2 if there exists a new function for signal f in terms of the primary inputs and the present state lines of C_2 such that E is not an erroneous sequence for the resulting new circuit (as illustrated in Fig. 9).

Based on this definition, we assume that the errors only affect combinational logic. Similar to the combinational cases, we assume that an error source should be able to correct every erroneous input sequence. Once a signal is found unable to correct any erroneous sequence, it can be excluded from the candidate list of potential error sources. It is worth mentioning that, for combinational circuits, if a signal can correct every erroneous input vector, then it is a single-fix signal. However, this statement is not true for sequential circuits. The reason is as follows. For a combinational circuit, every erroneous vector can be fixed *independently,* i.e., the requirement to fix every erroneous vector can be satisfied at the same time as shown earlier in Proposition 3. But for sequential circuits, some conflict may occur as deriving the fix function that corrects every erroneous sequence, even though each of them can be fixed individually.

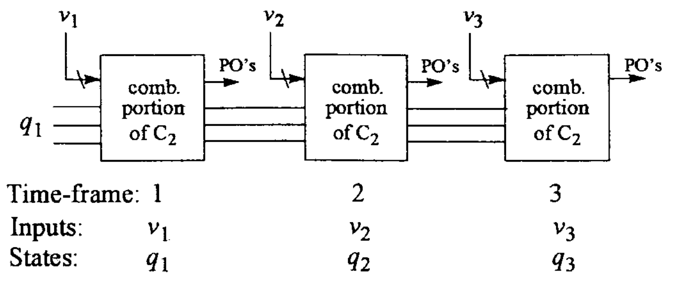


Fig. 10. An example of the iterative array model.

B. Necessary and Sufficient Condition

In the following discussion, we assume that both C_1 and C_2 have a known reset state, p_1 and q_1, respectively. The implementation C_2 is represented by the iterative array model as shown in Fig. 10.

The number of the copies of the combinational portion duplicated in the time-frame expansion model equals the length of the input sequence under consideration. Consider an erroneous input sequence with three input vectors, $E = \{v_1, v_2, v_3\}$. Suppose E brings C_2 through a sequence of states $\{q_1, q_2, q_3\}$, where q_1 is the initial state of C_2. Then we call $(v_1|q_1)$ the *pseudoinput vector* for the first time-frame. Similarly, $(v_2|q_2)$ and $(v_3|q_3)$ are the pseudoinput vectors for the second and the third time-frames, respectively. Based on this model, it follows directly that if signal f can correct an erroneous input sequence E, then there exists a new function of f such that every primary output at every time-frame becomes error free, (i.e., every primary output has the same response as their corresponding primary output of C_1 with respect to the input sequence E). We define a term called *injection* before we derive the necessary and sufficient condition of correcting an erroneous input sequence.

Definition 12—Injection: Given a signal f in C_2, a *t time-frame injection* at f is a set of value assignments to the signal f for the first t time-frames. For example, $J = \{f^{(1)} = 0, f^{(2)} = 0, f^{(3)} = 0\}$ represents a three time-frame injection that injects value "0" at f for all three time-frames, where the superscript denotes the index of a time-frame.

An injection defines a new circuit. The output responses of the resulting new circuit are computed by treating the injected signal as an independent pseudo primary input line, taking the injected value as the input. Since we can inject either "0" or "1" to a signal at each time-frame, there are 2^t different combinations for a t time-frame injection. The number of injections grows exponentially with the number of time-frames. Based on the above definition, we have the following proposition.

Proposition 6—Cure Injection: Let E be an erroneous input sequence with t input vectors, $E = \{v_1, v_2, \cdots, v_t\}$. A signal f in C_2 can correct E *only if* there exists a t time-frame injection at f such that E is not an erroneous input sequence for the resulting new circuit. (Such an injection is called a *cure injection* at f for E.)

Explanation: This proposition states that if there does not exist a cure injection at f, then there does not exist a *new function* for f to correct the erroneous input sequence E, but

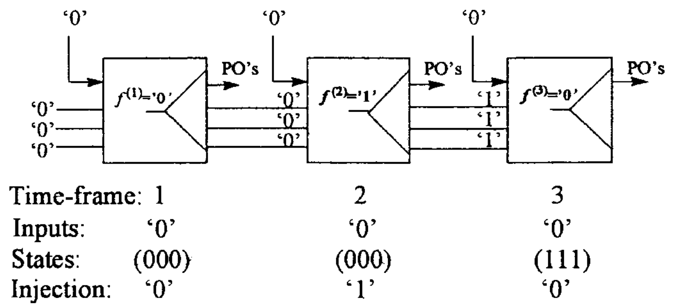


Fig. 11. Unrealizable injection.

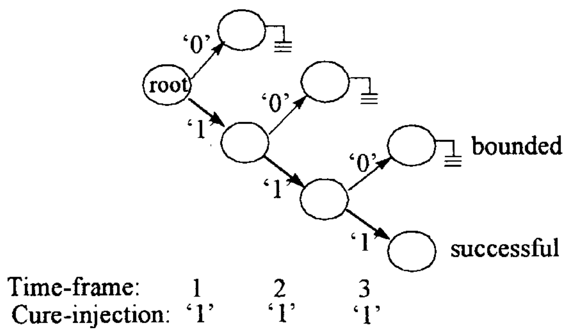


Fig. 12. Injection tree.

not vice versa. In other words, to correct an erroneous input sequence, it is necessary to find a cure injection. However, a cure injection is not sufficient to assure that the input sequence is indeed correctable. This is due to a fact that, for every fix function at f, there always exists a cure injection. Conversely, not every cure injection can be realized by a function.

Given a fix function at f that can correct an erroneous input sequence E, the corresponding cure injection can be derived as follows: Let the response of f with respect to E in the resulting new circuit is $\{\alpha_1, \alpha_2, \ldots, \alpha_t\}$, where α_i, $i \leq i \leq t$, is a binary value. Then $J = \{f^{(1)} = \alpha_1, f^{(2)} = \alpha_2, \ldots, f^{(t)} = \alpha_t\}$ is a cure injection. On the other hand, there exists some injection that is not realizable. Fig. 11 shows an example.

In this example, the pseudoinput vectors for the first and second time-frames are the same: $(0|000)$. But the injected values at f for these two time frames are different ("0" and "1," respectively). A function realizing this injection needs to map the same pseudoinput vector, $(0|000)$, to "0" and "1" at the same time, which is impossible. It follows that there does not exist a new function for f in terms of primary inputs and present state lines to realize this injection. Whether an injection is realizable or not can be checked easily by simulating the input vector for the resulting circuit with the injection. After collecting the sequence of states encountered in the resulting circuit, the pseudoinput vector for each time-frame and the injected value can then be derived. If no conflict exists, then the injection is realizable.

Proposition 7—Necessary and Sufficient Condition: Let E be an erroneous input sequence with t input vectors, $E = \{v_1, v_2, \ldots, v_t\}$. A signal f in C_2 can correct E if and only if there exists a *realizable t time-frame cure injection.*

Proof: As described above.

C. Correctability Check via Fault Simulation

Based on the above proposition, we can determine if an erroneous input sequence is correctable by a signal by a two-step checking: 1) check if there exists a cure injection and 2) check if it is realizable. Given an injection, determining whether it is a cure injection can be done via a modified sequential fault simulation process. Traditionally, sequential fault simulation assumes that the target signal is stuck at the same binary value for every time-frame. In our application, we need to modify the fault simulation algorithm, so that a signal is allowed to be stuck at different binary values at different time-frames to account for the fact that an injection may inject different values at different time-frames.

Theoretically, we need to enumerate every possible 2^t injections in the worst case to find a cure injection, or to conclude that there does exist one. However, like most branch-and-bound procedures, some criterion can be used to cut down the search space. In our application, the search space can be represented as a binary injection tree shown in Fig. 12.

The meaning of this tree is explained as follows: 1) Each node corresponds to a resulting circuit with a partial injection (i.e., t' time-frame injection where $t' < t$). The root node (level 0) corresponds to the original implementation C_2. 2) The level of each node corresponds to the current time-frame being considered for value injection. 3) The upper (lower) branch of each node represents injecting value "0" ("1") to the signal under consideration in the current time-frame and 4) a path from the root node to a leaf node represents a complete t time-frame injection.

Based on this injection tree, it follows that if a node's corresponding partial t' time-frame injection cannot produce the correct responses for the first t' time-frames, then the subtree of this node need not be explored. This simple bounding criterion is useful to speed up the search for a cure injection. Once a cure injection is found, the pseudoinput vectors of the resulting circuit with respect to this injection can be derived. The realizability check can then be followed to determine if the erroneous input sequence is indeed correctable by the target signal.

D. Diagnosing Multiple Errors

For diagnosing circuits with multiple errors, our algorithm searches for multiple signals that can jointly correct every generated erroneous input sequence. Fig. 13 shows an example of a three time-frame injection defined over a set of signals $\{f, g\}$. Similar to the case of single-error diagnosis, if this injection is a realizable cure injection for the applied erroneous sequence $E = \{(0), (0), (0)\}$, then E is correctable by this set of signals. Again, this condition can be checked primarily via a modified fault simulation process. Given a set of k signals, σ, and an erroneous input sequence with t input vectors, E, the worst case complexity of deciding if E can correct σ is proportional to the number of possible injections. There are 2^k possible value combinations for each time-frame and, thus, the total number of possible t time-frame injections defined over σ is $2^{(k \cdot t)}$.

TABLE III
RESULTS OF DIAGNOSING OPTIMIZED ISCAS'89 BENCHMARK CIRCUITS INJECTED WITH ONE ERROR

circuit	# signals in C2	E-length (min.)	**# potential fix signals**	pessimistic lower bound	hit error?	time (seconds)		
						random simulation	fault simulation	**total**
s298	65	10	**3**	**2**	Yes	36	13	**49**
s344	102	4	**4**	**2**	Yes	1	12	**13**
s349	101	3	**2**	**1**	Yes	1	10	**11**
s386	60	1	**1**	**1**	Yes	1	2	**3**
s510	146	5	**7**	**4**	Yes	1	12	**13**
s641	109	1	**2**	**2**	Yes	1	1	**2**
s713	112	1	**2**	**2**	Yes	1	5	**6**
s820	171	5	**7**	**3**	Yes	9	12	**21**
s832	174	2	**2**	**2**	Yes	1	8	**9**
s1196	327	1	**6**	**5**	Yes	1	13	**14**
s1238	348	1	**1**	**1**	Yes	1	8	**9**
s1488	387	7	**18**	**2**	Yes	205	40	**245**
s1494	375	2	**11**	**2**	Yes	2	27	**29**
s1423	422	10	**42**	**2**	Yes	209	1020	**1229**
s5378	920	2	**3**	**2**	Yes	13	53	**66**
s9234	1075	6	**6**	**2**	Yes	38	1050	**1089**
s13207	1325	2	**4**	**2**	Yes	11	205	**216**
Average	631	3.7	**7.1**	**2.2**	Yes	31	146	178

Intractable circuits:
(1) s208, s400, s444: due to long erroneous sequences.
(2) s15850, s35932, s38417, s38584: due to difficulty of generating erroneous sequences.

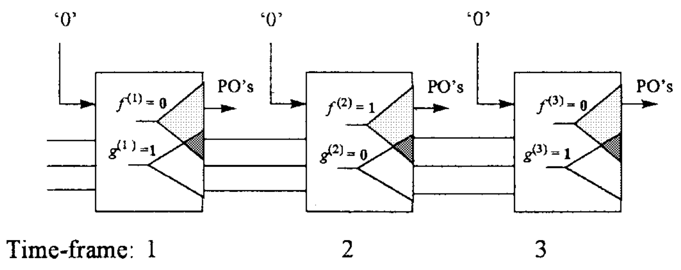


Fig. 13. A three-time-frame injection defined over a set of signals $\{f, g\}$.

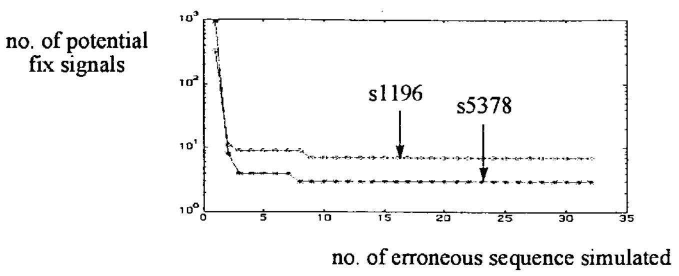


Fig. 14. The number of potential single-fix signals versus the number of fault simulated erroneous input sequences.

VII. EXPERIMENTAL RESULTS OF SEQUENTIAL DIAGNOSIS

Our experiments of sequential diagnosis are performed on ISCAS'89 sequential benchmark circuits. The erroneous implementations are generated as in the combinational cases. For sequential diagnosis, our algorithm does not require the knowledge of the number of FF's or the state encoding of the specification. Only the input/output functional behavior of the specification is needed.

A. Results of Diagnosing Single-Error Circuits

Table III shows the results of single-error diagnosis. In the preprocessing stage, we run random simulation to collect erroneous input sequences. We set ten as the maximum limit on the length of the sequences. The random simulation terminates when 32 erroneous input sequences have been collected, or 32 000 sequences have been simulated. The meanings of some columns are as follows.

1) *E-length (min):* The minimal length of the erroneous input sequences found in our preprocessing step.

2) *Number of Potential Fix Signals:* The number of potential single-fix signals delivered by our program. On average, the number of potential single-fix signals produced by our program is 7.1 for those ISCAS'89 benchmark circuits listed in Table III. The curves of the number of potential single-fix signals versus the number of simulated erroneous input sequences for s1196 and s5378 are shown in Fig. 14. It is very common in our experience that only a small number of erroneous input sequences are enough to drop most false single-fix candidates. Again, our algorithm will not overlook the real injected error signal. Our program fails on seven circuits: s208, s400, s444, s15850, s35932, s38417, and s38584. The reasons will be discussed later. 3) *Lower Bound*: There is a *pessimistic* lower bound on the total number of single-fix signals. It is obtained by counting the number of dominators of the injected error signal plus one. For ISCAS'89 benchmark circuits listed in Table III, the average is 2.2.

B. Results of Diagnosing Double-Error Circuits

Table IV shows the results of diagnosing implementation injected with two random errors. Our program first searches for single-fix signals. If there does not exist such signals, then

TABLE IV
RESULTS OF DIAGNOSING OPTIMIZED ISCAS'89 BENCHMARK CIRCUITS INJECTED WITH TWO ERRORS

circuit	# signals in C2	E-length (min.)	# potential single-fix	# potential double-fix	pessimistic lower bound	hit error?	time (seconds)		
							random simulation	fault simulation	total
s298	65	2	**0**	**36**	4	Yes	1	212	**213**
s344	102	3	**0**	**16**	6	Yes	1	611	**612**
s349	101	2	**0**	**16**	6	Yes	1	60	**61**
s386	60	1	**3**	-	-	-	2	42	**44**
s510	146	1	**0**	**5**	2	Yes	1	100	**101**
s641	109	1	**0**	**15**	6	Yes	1	60	**61**
s713	112	1	**0**	**6**	4	Yes	1	26	**27**
s820	171	2	**1**	-	-	-	1	10	**11**
s832	174	2	**3**	-	-	-	4	6	**10**
s1196	327	2	**0**	**10**	3	Yes	1	86	**87**
s1238	348	3	**0**	**10**	4	Yes	1	156	**157**
s1488	387	1	**0**	**2**	1	Yes	1	268	**269**
s1494	375	2	**0**	**18**	1	Yes	2	312	**314**
s1423	422	3	**5**	-	-	-	7	109	**116**
s5378	920	1	**0**	**14**	4	Yes	10	3873	**3883**

we search for signal pairs that can jointly correct the implementation. Among the 15 circuits in Table IV, four of them are classified as single-signal correctable. For these circuits, there exists signals that can fix every erroneous sequence. But it may not guarantee that implementations are *indeed* single-signal correctable because we do not exhaustively simulate every erroneous sequence. On the other hand, the other 11 are proven *not* single-signal correctable by our program (the number of potential single-fix signals is zero). For these circuits, we report the number of the potential double-fix pairs. The run time is longer than the case of diagnosing single error due to the rapid growth of the number of candidate signal pairs and the number of possible injections that need to be checked for correctability.

C. Future Work

Based on this approach, there are several issues that need to be further addressed in the future.

1) Erroneous Input Sequence Generation: For combinational circuits, random simulation [15] or advanced automatic test pattern generation (ATPG) based techniques [2] have provided satisfactory solutions to generate erroneous vectors even for fairly large circuits. However, these techniques may not be adequate to generate erroneous input sequences for some sequential designs. Random simulation cannot find any erroneous input sequences for single-error circuits s15850, s35932, s38417, and s38584 in our experiments. For these large designs, if manually crafted functional sequences are available for simulation-based design validation, then most design errors are likely to be exposed and the erroneous sequences can be generated as a by-product. In that case, our approach is then applicable. Another possible solution to this problem is to explore the FF correspondence or internal structural similarity between the specification and the implementation. If a large number of corresponding FF's exists, then the sequentiality of the circuit can be reduced by treating the inputs (outputs) of some FF's as pseudoprimary outputs (inputs). In this way, the difficulty of generating erroneous input sequences can be reduced.

2) High Complexity for Long Erroneous Input Sequences: If the errors occur in a highly sequential module (e.g., a counter) and cannot be detected by any input sequence with reasonable length (e.g., 30 vectors), then our approach may become too time-consuming. For s208, s400, and s444, our approach fails due to this reason. To address this problem, new techniques are under development to deal with long erroneous input sequences for circuits that are manageable by BDD techniques.

3) Difficulty for Circuits with Larger Number of Errors: In practice, the complexity of diagnosing circuits with more than two errors is prohibitively high. Some heuristics are under investigation to estimate each signal's error probability and help the designer to locate one error at a time.

VIII. CONCLUSION

We present a new approach to design error diagnosis. Our algorithm searches for the potential error sources that are most likely responsible for the incorrectness of the implementation. Unlike symbolic approaches, we do not rely on BDD to search for such signals. Instead, we prove that fault simulation can precisely decide if a signal can be held responsible for a particular erroneous vector. This formulation allows us to exclude most signals from being potential error sources efficiently by performing fault simulation with a number of erroneous input vectors. In order to speed up the process, we further propose a two-stage algorithm that can take advantage of the topological dominance relation between signals. Compared to other simulation-based approaches, our algorithm has two advantages. First, it is more accurate because it is based on a more stringent condition for identifying potential error sources. Second, it can be generalized to multiple errors. We also show how to generalize this idea to sequential diagnosis. Although the complexity of this generalization may still be high for some cases, it can serve as a basis for future improvement. The experimental results of diagnosing ISCAS benchmark circuits injected with one or two random errors are presented to demonstrate its effectiveness.

References

[1] M. S. Abadir, J. Ferguson, and T. E. Kirkland, "Logic design verification via test generation," *IEEE Trans. Comput.,* pp. 138–148, Jan. 1988.

[2] H. Al-Asaad and J. P. Hayes, "Design verification via simulation and automatic test pattern generation," in *Proc. Int. Conf. on Computer-Aided Design,* Nov. 1995, pp. 174–180.

[3] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* Piscataway, NJ: IEEE Press, 1990.

[4] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical path tracing-an alternative to fault simulation," in *Proc. Design Automation Conf.,* June 1983, pp. 214–220.

[5] D. Brand, A. Drumm, S. Kundu, and P. Narrain, "Incremental synthesis," in *Proc. Int. Conf. on Computer-Aided Design,* Nov. 1994, pp. 14–18.

[6] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.,* vol. C-35, pp. 667–691, Aug. 1986.

[7] W.-T. Cheng and M.-L. Yu, "Differential fault simulation—A fast method using minimal memory," in *Proc. Design Automation Conf.,* June 1989, pp. 424–428.

[8] 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.

[9] M. Fujita, "Methods for automatic design error correction in sequential circuits," in *Proc. European Conf. on Design Automation,* 1993, pp. 76–80.

[10] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen, "AQUILA: An equivalence verifier for large sequential circuits," in *Proc. Asia and South Pacific Design Automation Conf.,* Jan. 1997, pp. 455–460.

[11] S.-Y. Huang, K.-C. Chen, and K.-T. Cheng, "Incremental logic rectification," in *Proc. VLSI Test Symp.,* Apr. 1997, pp. 134–139.

[12] 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. Int. Test Conf.,* Nov. 1997, pp. 974–981.

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

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

[15] F. Krohm, A. Kuehlmann, and A. Mets, "The use of random simulation in formal verification," in *Proc. Int Conf. on Computer Design,* Oct. 1996.

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

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

[18] C.-C. Lin, K.-C. Chen, S.-C. Chang, M. Marek-Sadowska, and K.-T. Cheng, "Logic synthesis for engineering change," in *Proc. Design Automation Conf.,* June 1995, pp. 647–652.

[19] C.-C. Lin, K.-C. Chen, D. I. Cheng, and M. Marek-Sadowska, "Logic rectification and synthesis for engineering change," in *Proc. Asian and South Pacific Design Automation Conf.,* 1995, pp. 301–309.

[20] J. C. Madre, O. Coudert, and J. P. Billon, "Automating the diagnosis and the rectification of the design errors with PRIAM," in *Proc. Int. Conf. on Computer-Aided Design,* Nov. 1989, pp. 30–33.

[21] I. Pomeranz and S. M. Reddy, "On correction of multiple design errors," *IEEE Trans. Computer-Aided-Design,* vol. 14, pp. 255–264, Feb. 1995.

[22] ——, "On error correction in macro-based circuits," in *Proc. Int. Conf. on Computer-Aided Design,* Nov. 1994, pp. 568–575.

[23] ——, "A method for diagnosing implementation errors in synchronous sequential circuits and its implications on synthesis," in *Proc. European Design Automation Conf.,* Sept. 1993, pp. 252–258.

[24] S. Rudeanu, *Boolean Functions and Equations.* Amsterdam, the Netherlands: North-Holland, 1974.

[25] M. Tomita, H. H. Jiang, T. Tomamoto, and Y. Hayashi, "An algorithm for locating logic design errors," in *Proc. Int. Conf. on Computer-Aided Design,* Nov. 1990, pp. 468–471.

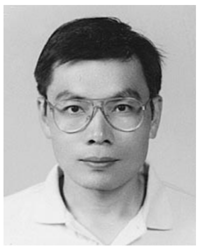
[26] K. A. Tamura, "Locating functional errors in logic circuits," in *Proc. Design Automation Conf.,* June 1989, pp. 185–191.

[27] A. G. Veneris and I. N. Hajj, "A fast algorithm for locating and correcting simple design errors in VLSI digital circuits," in *Proc. Great Lake Symp. on VLSI Design,* Mar. 1997, pp. 45–50.

[28] "SIS: A system for sequential circuit synthesis," Univ. of California, Berkeley, Tech. Rep. M92/41, 1992.

[29] A. M. Wahba and D. Borrione, "A method for automatic design error location and correction in combinational logic circuits," *J. Electron. Testing: Theory Applicat.,* vol. 8, no. 2, pp. 113–27, Apr. 1996.

[30] ——, "Design error diagnosis in sequential circuits," in *Proc. Correct Hardware Designs and Verification Methods; CHARME'95, Lecture Notes in Computer Science 987.* Berlin, Germany: Springer-Verlag, Oct. 1995, pp. 171–188.



Shi-Yu Huang received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taiwan, in 1988 and 1992, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1997.

From 1997 to 1998, he was a Software Engineer at National Semiconductor Corporation, Santa Clara, CA, investigating the system-on-a-chip design methodology. He is currently a Principal Engineer at Worldwide Semiconductor Manufacturing Corporation, HsinChu, Taiwan. His research interests are in the areas of computer-aided design for VLSI with an emphasis on logic verification and design for testability. He coauthored *Formal Equivalence Checking and Design Debugging* (Norwell, MA: Kluwer Academic, 1998).

Kwang-Ting Cheng (S'88–M'88–SM'98) received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, in 1983 and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1988.

He worked at AT&T Bell Laboratories in Murray Hill, NJ, from 1988 to 1993. He joined the faculty at the University of California, Santa Barbara, in 1993, where he is currently a Professor of Electrical and Computer Engineering. His current research interests include VLSI testing, design synthesis, and design verification. He has published more than 120 technical papers, coauthored three books, and holds six U.S. patents in these areas. He has also been working closely with U.S. industry for projects in these areas.

Dr. Cheng received the Best Paper Award at the 1994 Design Automation Conference and the Best Paper Award at the 1987 AT&T Conference on Electronic Testing. He currently serves on the Editorial Boards of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, IEEE DESIGN AND TEST OF COMPUTERS, and *Journal of Electronic Testing: Theory and Applications*. He has been General Chair and Program Chair of IEEE International Test Synthesis Workshop. He has also served on the technical program committees for several international conferences on CAD and testing.