Enhancing Diagnosis Resolution For Delay Defects Based Upon Statistical Timing and Statistical Fault Models*

A. Krstic, L.-C. Wang, K.-T. Cheng
University of California
Santa Barbara, CA
{angela,licwang,timcheng}@ece.ucsb.edu

J.-J. Liou
Tsing-Hua University
Taiwan
jjliou@slugger.ee.nthu.edu.tw

T. M. Mak
Intel Corporation
Santa Clara, CA
t.m.mak@intel.com

ABSTRACT

In this paper, we propose a new methodology for diagnosis of delay defects in the deep sub-micron domain. The key difference between our diagnosis framework and other traditional diagnosis methods lies in our assumptions of the statistical circuit timing and the statistical delay defect size. Due to the statistical nature of the problem, achieving 100% diagnosis resolution cannot be guaranteed. To enhance diagnosis resolution, we propose a 3-phase diagnosis methodology. In the first phase, our goal is to quickly identify a set of candidate suspect faults that are most likely to cause the failing behavior based on logic constraints. In the second phase, we obtain a much smaller suspect fault set by applying a novel diagnosis algorithm that can effectively utilize the statistical timing information based upon a single defect assumption. In the third phase, our goal is to apply additional fine-tuned patterns to successfully narrow down to more exact suspect defect locations. Using a statistical timing analysis framework, we demonstrate the effectiveness of the proposed methodology for delay defect diagnosis, and discuss experimental results based on benchmark circuits.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance Analysis and Design Aids

General Terms

Algorithm, Performance, Reliability

Keywords

Delay fault diagnosis, Statistical timing models, Delay ATPG

1. INTRODUCTION

Process variations, manufacturing defects and noise are major factors to affect timing of deep sub-micron (DSM) designs [1, 2]. These DSM delay effects are often continuous in nature [3, 4], and the traditional assumptions of discrete timing and delay models in analysis and simulations become less applicable. Instead, these factors should better be captured and simulated using statistical models and methods [5].

Historically, the diagnosis problem was defined over the logic domain and no timing information was involved. In today's industry, the single stuck-at fault model remains one of the most affordable and effective models for defect diagnosis. Stuck-at based diagnosis algorithms are often classified into two types: an *effect-cause* approach and a *cause-effect* approach [6]. In an effect-cause approach, the stuck-at fault model allows an ATPG to determine, from the failing behavior, if a particular line should be the stuck-at. Then, by searching backward from the failing outputs and matching to the input patterns, probable faults can be identified. In a cause-effect approach, diagnosis relies on the construction of a *fault dictionary* that contains information to differentiate the good and faulty behavior in the presence of each stuck-at fault. Then, for a given failing chip, the failing behavior is compared to the information in the fault dictionary, and the most probable fault is selected as the candidate for the defect source [6]. If we assume that the defects are from the single stuck-at faults, then it might be possible to identify the exact fault causing the problem depending on the existence of a test pattern set that can achieve the *maximal fault resolution* [6]. In other words, we can say that if faults and defects are the same, then in logic diagnosis the *diagnosis resolution* is the same as the *fault resolution*.

In delay defect diagnosis, the problem is fundamentally different in two ways: First, the exact delay configuration of a given failing chip instance is usually unknown. Second, even with a single defect assumption, the size of each possible delay defect is still a random variable. These two aspects prevent us from applying a traditional logic diagnosis algorithm to delay fault diagnosis and moreover, the diagnosis resolution is very different from the fault resolution defined in the logic domain.

In the past, much of the diagnosis research focused on two directions. One was to improve the efficiency of diagnosis by avoiding the computational expense of creating a large fault dictionary. The other was to extend the basic diagnosis algorithm for the single stuck-at fault model to other defect types [8, 9] or to multiple faults [10]. Even for diagnosing gate delay and path delay faults, most of the previous work was based purely on logic conditions for sensitizing the faults [11, 12]. A statistical diagnosis framework for delay defects was proposed in [13]. However, due to complexity reasons, it is not practical.

In this paper, we propose a novel methodology for diagnosing delay defects based upon statistical timing models and delay simulation methods. It consists of three phases:

1. **Effect-cause phase:** In this phase, a set of suspect faults are identified based purely on logic conditions.
2. **Cause-effect phase:** We apply a novel diagnosis algorithm operating on the probabilistic space, instead of the logic space to obtain a much smaller set of candidate faults.
3. **Fine-tuning phase:** Based upon the results in phase 2, we select a few target faults and produce additional patterns for them in order to further narrow down to more exact fault location(s).

In phases 2 and 3, statistical timing analyzer serves as a predictor for the delay configuration of a given failing chip instance. Because of this, how to match the failing behavior to the probabilistic information contained in the fault dictionary becomes an interesting question. Since the delay defect size is a random variable, the criteria to determine the maximal fault resolution for a given pattern set become

*This work was supported in part by the MARCO/DARPA Gigascale Silicon Research Center (http://www.gigascale.org). Their support is gratefully acknowledged.

DAC 2003, June 2–6, 2003, Anaheim, California, USA

probabilistic as well. As a result, we can no longer rely merely on the logic conditions to decide if a test can differentiate two faults.

To measure the accuracy of matching the probabilistic fault information in the fault dictionary to the failing behavior, we introduce a new concept called *diagnosis error function*. In the second phase, we utilize an Euclidean-distance-based diagnosis error function to decide the fault suspects. To obtain a good diagnostic pattern set, in the second phase, we use a path delay fault ATPG without considering timing (for a set of longest paths). Then, in the third phase we use a Genetic Algorithm based *timed* ATPG to derive additional fine-tuning patterns.

By separating phase 2 from phase 1, we avoid the construction of the fault dictionary for the faults that can be excluded as a cause of the failing behavior using only logic criteria. Hence, the effectiveness of our phase 2 diagnosis algorithm can be better observed. By separating phase 3 from phase 2, we avoid the application of a more complex (timed) ATPG to a large number of faults. Therefore, in our 3-phase methodology, we apply a more complicated algorithm to solve a problem aspect only when it cannot be solved by an easier approach.

While deciding which suspect faults should be kept after phase 1 is deterministic (because this decision is based purely on logic criteria), deciding which suspect faults should be used in phase 3 can only be probabilistic. Because of this, we re-define the concept of *diagnosis resolution* and discuss heuristics to separate faults in phases 2 and 3.

2. PROBABILISTIC FAULT DICTIONARY

In logic diagnosis, the circuit model used in the simulation is assumed to logically match to the chip instance. In delay diagnosis, this is not true due to the inclusion of statistical delay information. Each chip represents only a single instance of all possible delay configurations intended to be modeled statistically by the CAD tools.

Suppose the single stuck-at fault model is used in logic diagnosis. Let $\{f_1, \ldots, f_n\}$ be the n faults that belong to n different fault equivalence classes. Suppose a pattern set is available to achieve the maximal fault resolution, i.e., for any pair of faults f_i, f_j, there exists a pattern in the set to differentiate these two faults (detect one but not the other). Then, in theory, given the failing behavior resulting from a single stuck-at defect, the diagnosis algorithm can conclude exactly which fault is the cause. On the other hand, if the pattern set does not achieve the maximal fault resolution, then depending on the resolution, an algorithm can conclude a subset of the faults as the potential causes. Exactly which one is unknown. Based upon these observations, we can say that the diagnosis resolution in the logic domain is the same as the fault resolution *if* defects are the same as faults.

Take the single transition fault model as an example. If no delay information is involved, then the above statement still holds. However, if delay information is involved, then the diagnosis resolution is not the same as the fault resolution in the logic domain. Figure 1 illustrates the reasons.

In the figure, output arrival times are characterized as probability distributions. When a clock is given, from each probability distribution we can calculate the *critical probability* that represents the chance of an output delay exceeding the given clock [5]. In the figure, the critical probability at output *o1* is illustrated as the shaded area.

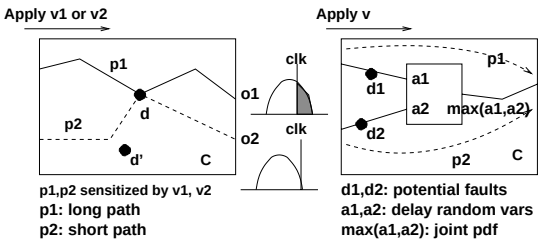

Figure 1: The Impact of Delays in Diagnosis

In the first case, for a fault d, suppose two patterns v_1, v_2 are available. In logic domain, both patterns detect d and can differentiate between d and d'. However, depending on the timing length of the sensitized path ($p1$ or $p2$), the critical probability (shaded area) resulting from each pattern can be different. If a pattern detects a fault through a short path (like v_2), then it is possible that with a small delay defect size, the pattern does not detect the defect at all. Consequently, v_2 can differentiate two faults in the logic domain but cannot do so by considering the delays (it may detect none).

In the second case, a pattern v detects both faults d_1, d_2, logically through sensitized paths p_1, p_2, respectively. Suppose the two paths merge at a 2-input cell and the arrival time random variables at the two inputs are denoted as a_1, a_2. The output arrival time random variable of the cell is the joint pdf random variable $\max(a_1, a_2)$. Suppose $Prob(a_1 > a_2) = 1$. Then, it is possible that p_1 always dominates the output delay (or vice verse depending on the transition type). Hence, pattern v can differentiate the two faults. As it can be seen, even though logically, pattern v does not differentiate the two faults, timing-wise it may.

Due to the above two reasons, in general, whether or not a test pattern can differentiate two given faults *should be characterized as a probability value that depends on the given clock period* clk. Therefore, in delay defect diagnosis, given a pattern v, our first task is to compute the probability that v detects a particular fault. This information is used to build the *probabilistic* fault dictionary, and our algorithm will use the dictionary to guess which fault is the most probable one to be the cause of failure.

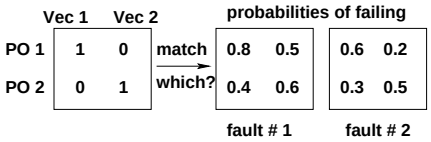

Figure 2: Illustration of The Matching Problem

The probabilistic nature of the fault dictionary raises an interesting question. Consider the example in Figure 2. Suppose the failing behavior of a chip instance is characterized as a 0-1 matrix (1 means that an error is observed). Suppose we have a way to calculate (in the simulation), for each candidate suspect fault, a probability matrix P where p_{ij} represents the chance that a failure is observed at primary output i during the application of test vector j. Then, in the example, the underlying question to ask is: which probability matrix is a better match to the failing behavior?

If we focus on matching the "1" entries in the 0-1 matrix, we would say that fault # 1 is a better match. However, if we focus on matching the "0" entries, fault # 2 would be a better match. In general, depending on our view of what do we mean by a "better match" the diagnosis answer can be different. Hence, in order to develop an accurate diagnosis algorithm, our first task is to define carefully how to match the information in the probabilistic fault dictionary to the failing behavior. We call such functions the *diagnosis error functions*.

The concept of probabilistic fault dictionary implies that an optimal test set considering only the logical conditions may not be optimal for delay defect diagnosis. A possible solution to obtain a good set of diagnostic delay patterns could be to use a *timed ATPG* [14, 15]. In our work, we do not consider a *deterministic* timed ATPG due to its high complexity. Instead, we use a path delay fault ATPG (based on logic sensitization conditions) as an approximation in phase 2, and we use a Genetic Algorithm based timed ATPG in phase 3.

3. AN ERROR-FUNCTION-DRIVEN DIAGNOSIS ALGORITHM

Given a failing n-output chip instance C_{in} and a set TP of m pat-

terns, suppose that the failing behavior is characterized in an $n \times m$-matrix $\mathcal{B}$. For $1 \leq i \leq n$, $1 \leq j \leq m$, each b_{ij} is "1" if an error is observed at output i while applying pattern j. Otherwise, $b_{ij} = 0$.

To diagnose the defect, we utilize a single-defect assumption. We assume that the defect location and defect size are independent random variables. In our current work, we do not consider that these random variables are correlated. Moreover, we assume that only one defect can occur for a given failing chip instance. This delay defect can happen on any one of the signals in the circuit.

3.1 Phase 1: Effect-Cause Analysis

In the phase 1, for each failing pattern we perform backward analysis from each failing output. We collect faults that fall on the logically sensitized paths (to the failing output) given by the patterns. At the end of this phase, we obtain a suspect fault set $F = \{f_1, \ldots, f_l\}$. Each fault f_k (for $1 \leq k \leq l$) falls on at least one sensitized path to one failing output during the sensitized path to one failing output during the application of at least one pattern.

3.2 Phase 2: Cause-Effect Diagnosis

Given F, for each fault f_k, we construct an $n \times m$- probability matrix $\mathcal{E}_k$. Each e_{ij}^k represents the probability that an error can be observed at output i while applying pattern j for a given clock period *if* fault f_k is present. Suppose we can calculate these probability matrices for all faults in F and obtain $\mathcal{E}_1, \ldots, \mathcal{E}_l$ [7]. Then, the underlying question to ask is: which $\mathcal{E}_k$ (for $1 \leq k \leq l$) is a better match to the failing behavior matrix $\mathcal{B}$? To measure the accuracy of this "match," we introduce the concept of *diagnosis error function* Err. In essence, $Err(\mathcal{B}, \mathcal{E}_k)$ measures the diagnosis error if fault f_k is selected as the answer of diagnosis.

3.2.1 An Error Function Based on Euclidean "Distance"

To simplify the problem, we first assume that $n = 1$. Hence, $\mathcal{B} = [b_1, \ldots, b_m]$ and $\mathcal{E}_k = [e_1^k, \ldots, e_m^k]$ for fault f_k. Then, the Euclidean *distance* between the expected results $\mathcal{E}_k$ when fault f_k is assumed to be the defect, and the observed behavior $\mathcal{B}$ can be measured by

$$Err(\mathcal{B}, \mathcal{E}_k) = Err_k = || \mathcal{E}_k - \mathcal{B} ||^2 = \Sigma_{i=1}^{m} (e_i^k - b_i)^2 \qquad (1)$$

Next, we compute Err_k for all f_k, $1 \leq k \leq l$, and we pick the minimum, i.e., pick the fault that minimizes this diagnosis error function $Err()$.

For multiple-output circuits ($n > 1$), Figure 3 demonstrates a simple view about the meaning of an error in the diagnosis. Under the equivalence checking model, an error in the diagnosis for a given pattern, is defined as *at least one output* produces a difference. In the figure, the delay configuration of the failing chip instance C_{in} is also unknown, and is modeled in the simulation with statistical timing in the circuit model C. What we know is the failing behavior matrix $\mathcal{B}$.

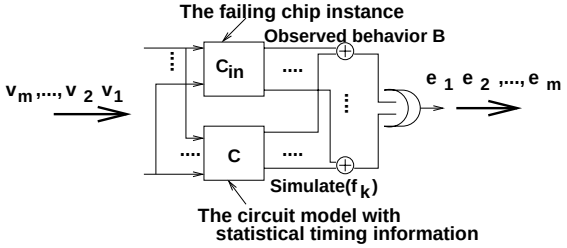

Figure 3: Error Under Equivalence Checking Model

What would be the ideal case? The ideal case, where no mismatch occurs, would be $e_1 = e_2 = \cdots = e_n = 0$. However, this is impossible even though we have correctly guessed the fault f_k in the diagnosis process. The reason is that we still do not know the exact delay configuration of the failing chip instance C_{in}.

Now, suppose we have an algorithm to compute, for each suspect fault f_k, the probability p_j^k that e_j (the "e_j" in the figure) is 1 ($1 \leq j \leq m$). In other words, the algorithm outputs a probability vector for each f_k as $P_k = [p_1^k, p_2^k, \ldots, p_m^k]$. Then, since the ideal outcome we want to see is $\mathbf{0} = [0, 0, \ldots, 0]$, we can measure the diagnosis error between the probability vector P_k and the ideal solution $\mathbf{0}$ as simply

$$Err_k = \Sigma_{j=1}^{m} (p_j^k)^2 \qquad (2)$$

Equation (2) follows the same spirit as equation (1), both of them use the Euclidean distance to measure the diagnosis error. Hence, we can use equation (2) to pick a fault whose error is the minimum.

ALGORITHM 3.1. *(The Diagnosis Algorithm)*

Inputs A circuit with statistical timing model; a pattern set TP; a failing behavior matrix $\mathcal{B}$; and a set of suspect faults F given from phase 1 analysis; Moreover, a user-defined number K called the *diagnosis resolution*.

Outputs A set of *ranked* diagnosis error values $Err_1 < \cdots < Err_K$ which indicate the K most likely faults causing the failing behavior.

Steps The clock period clk is used to observe the failing behavior matrix $\mathcal{B}$.

1. For each fault f_k in F, calculate the probability matrix $\mathcal{E}_k$. The tools and methodologies used in this calculation will be summarized in Section 4. From $\mathcal{E}_k$ and $\mathcal{B}$, we use the method described in [7] to calculate the probability vector $P_k = [p_1^k, p_2^k, \ldots, p_m^k]$, where each p_j^k is the probability that e_j is 1 (a mismatch between the observed and simulated results is at least at one output) if f_k is present as shown in Figure 3.
2. Calculate $Err_k = \Sigma_{j=1}^{m} (p_j^k)^2$ as described above to measure the diagnosis error.
3. After we finish the calculation for all faults in F, we have $\{Err_1, Err_2, \ldots, Err_l\}$. We rank them in such a way that $Err_{j_1} \leq Err_{j_2} \leq \cdots \leq Err_{j_l}$ and output the first K faults as the diagnosis answer.

END OF ALGORITHM

4. TOOLS AND METHODOLOGIES FOR THE EXPERIMENTS

The key tools to realize the proposed diagnosis algorithm include a statistical timing analysis tool and a dynamic timing simulator. Moreover, to measure the effectiveness of our diagnosis method, we need to perform statistical defect injection and fault simulation.

4.1 Statistical Timing Analysis

In statistical timing analysis framework, the delays of cells/interconnects are modeled as correlated random variables with known probability density functions (pdf's). These pdf's can be obtained using a Monte-Carlo-based SPICE simulator. Given cell/interconnect delay functions and a cell-based netlist, the statistical framework can derive the pdf's of signal arrival times for both internal signals and primary outputs using Monte-Carlo based simulation technique.

In our experiments, we use a cell-based statistical timing analysis framework [5]. It requires pre-characterization of cells, i.e., building libraries of pin-pin cell delays and output transition times (as random variables). We use a Monte-Carlo-based SPICE (ELDO) [17] to extract the statistical delays of cells for a 0.25μm, 2.5V CMOS technology. The input transition time and output loading of the cells are used as indices for building/accessing these libraries. Each interconnect delay is also modeled as a random variable and is pre-characterized once the RCs are extracted.

4.2 Dynamic Timing Simulation

With a given set of test patterns the statistical timing analysis framework can be used to perform statistical dynamic timing simulations to obtain the pdf's of internal signals and primary outputs for the given set of test patterns. These pdf's are obtained by simulating a large number of circuit instances with different cell/interconnect delay assignments.

4.3 Defect Injection and Simulation

To measure the accuracy of our diagnosis method in the cause-effect phase (phase 2), we apply it with single as well as multiple defect models. For both models, we adopt an exponential delay size distribution function.

Single Defect Model. This model can be used to represent small delay faults resulting from manufacturing defects, resistive opens and shorts, bridging faults. We use exponential distribution for defect size $\lambda e^{-\lambda x}$ where x is the defect size and λ is a constant. We use $\lambda = 0.04$ in our experiments. Other defect distributions could be used as well and using other distributions in general should not invalidate the trends observed in our work [7].

Multiple Defect Model. In this model, several single defects are simultaneously injected into the design. It can represent delay faults from a defect localized to a certain area of the chip.

5. INITIAL EXPERIMENTAL RESULTS

For each circuit model C, we produce N circuit instances with different delay configurations. On each instance, we inject a delay defect of which both location and size are drawn randomly according to the defect model (single or multiple). These instances model the faulty chips. We then apply our diagnosis method to each instance. The accuracy of diagnosis for single defects is measured in two ways: 1) In the algorithm, if the user-defined diagnosis resolution number K value is 1 (refer to Algorithm 3.1 above), then the accuracy is a binary value *success* and *failure* depending on whether the answer matches the injected defect or not. 2) If the user-defined $K > 1$, then if the injected defect is *contained* in the potential defect set answered by the algorithm, it is counted as a *success*; otherwise, it fails. Then, we calculate the success rate as the accuracy measurement by averaging over the results from all N instances. Clearly, the larger the K value is, the higher the success rate will be. For multiple faults, in the current implementation, we evaluate the algorithm assuming that the defect can be diagnosed if at least one of the single faults contained in the multiple fault can be diagnosed.

For single and multiple defect models, for each injected fault, we find a set of statistically "long" paths through the fault site and generate path delay tests for them *without* considering timing. These long paths are derived using the false-path aware static statistical timing analysis tool [16]. Then, robust or non-robust patterns for testing these paths are produced.

Table 1 shows results on the accuracy of diagnosis. As expected, the rates of success increase for larger K. Values S_1, S_2 and S_3 represent the average number of suspect faults per injected fault for single, double and triple faults, respectively after the phase 1 effect-cause analysis. The number of applied diagnostic patterns is in the range of few tens of patterns, depending on the fault model and the circuit. The results in the table can be interpreted in the following way: For example for s15850, for the case of single faults, 38% of the failing chip instances can be diagnosed successfully with a diagnosis resolution $K = 2$, 57% can be diagnosed successfully with a diagnosis resolution $K = 6$, etc.

These numbers for K should be compared to the number of potential suspect faults 224 given at the end of the phase 1. In other words, at the start of phase 2 where we apply our main diagnosis algorithm, we have already excluded the faults that are impossible to cause the faulty behavior by considering the logic sensitization conditions of the given pattern set. Hence, the effectiveness of our algorithm can be clearly observed by the small K relative to the large suspect set in each case.

Ckt; average number of suspect faults after Phase 1	K	Single (%)	Double (%)	Triple (%)
s1196	1	23	38	32
$S_{single} = 123$	2	48	56	44
$S_{double} = 189$	3	60	67	70
$S_{triple} = 239$	5	80	86	82
s1423	1	7	22	50
$S_{single} = 230$	2	29	32	50
$S_{double} = 209$	4	52	63	68
$S_{triple} = 414$	9	75	71	82
s5378	1	29	20	27
$S_{single} = 137$	4	58	42	54
$S_{double} = 100$	6	86	63	72
$S_{triple} = 189$	9	86	80	81
s9234	1	15	45	17
$S_{single} = 118$	3	35	72	56
$S_{double} = 116$	5	64	89	88
$S_{triple} = 170$	8	73	89	93
s15850	2	38	54	43
$S_{single} = 224$	6	57	90	64
$S_{double} = 258$	10	76	93	71
$S_{triple} = 289$	13	95	97	94

Table 1: Diagnosis Accuracy on Benchmarks

From comparing the results for double and triple faults with the results for single faults, it can be seen that our single defect-based diagnosis algorithm performs very well for the case of multiple defects as well. In the cases of double and triple faults, a success is declared if at least one of the faults is diagnosed correctly. Note that this does not imply that the diagnosis problems become easier because the effect resulting from multiple faulty delay random variables is statistical and remains hard to predict.

5.1 Questions Left After Phase 2

Two fundamental questions remain at the end of the phase 2. First, although the experimental results indicate that a small diagnosis resolution K can usually give us good results, how do we know that our selection of K is good enough? Second, suppose we want to further improve the diagnosis resolution by adding more patterns, how can we produce the additional good diagnosis patterns? Moreover, what faults should be targeted for producing these additional patterns? As it can be seen, the answer to the second question partly depends on the answer to the first question. In the following section, we will present our methodology in phase 3 to answer these questions.

6. PHASE 3: FINE-TUNING

In phase 3, we use a Genetic Algorithm (GA) based ATPG to produce additional fine-tuning patterns. The ATPG process is guided using a fitness function based on the timing information. Since including timing can significantly increase the cost of test generation, we are interested in finding a small set of target faults. The additional tests for these faults should produce the greatest impact on the diagnosis accuracy.

6.1 Selection of Target Suspects

After phase 2, we can rank the set of suspects for a given fault according to the value of the diagnosis error. Then, using the ranking and user-specified K value, we can pick a set of most likely faults as the target faults in phase 3 for generating additional patterns. However, what if a user does not know what K value to pick?

It is obvious that selection of K affects the diagnosis results. A larger K provides a higher confidence that the defect is contained in

the final set of suspects, while a smaller K provides a better diagnosis resolution. For the purpose of selecting the target faults in phase 3, a larger K implies more ATPG effort. Given the ranking of the suspects at the end of phase 2, the underlying question becomes how to choose K so that we strike a good balance among all these concerns. From the experimental results shown above, in general we expect that a small K value should be good enough.

To help us answer this question, we focus again on the diagnosis error values for the suspects after phase 2. Figures 4 and 5 show plots of the diagnosis error values for every suspect in 4 faulty chips for s5378 and s15850. These chip instances are randomly chosen from the set of defects being diagnosed in the experiment in Section 5. The suspects in the x-axis are the ones remaining after eliminating all the impossible faults based on phase 1.

In these plots, we can observe a clear trend showing that after a certain fault index *there is a rapid increase of the diagnosis error values*. This trend suggests that there is a small set of faults for which the failing patterns result in a much better match between the observed and the statistically simulated behavior (as defined by the diagnosis error). Therefore, we can select K based upon when the rapid increase happens. Our experimental results in Section 5 support this easy approach to be a good heuristic for selecting the K value.

For example, next to each curve in Figure 4, we show the K value that results in successful diagnosis, i.e. the injected defect is one of the first K faults (as validated by our experiments). Hence, for the curve denoted as "fault #9" this curve was obtained based upon the injection of fault #9 in the experiments. And, with a selection of $K = 3$, the fault # 9 is contained in the first three suspects being diagnosed in phase 2. We note that for fault #9, the diagnosis error values increase rapidly after the first 5 suspects. Similar trends can be observed for other faults and the other design as well.

The shape of these diagnosis error curves reveals the obvious heuristic of selecting the first K suspects based upon the rapid increase of the error values. Based upon this heuristic, we select a small set of K suspect faults for test generation of the additional fine-tuning patterns.

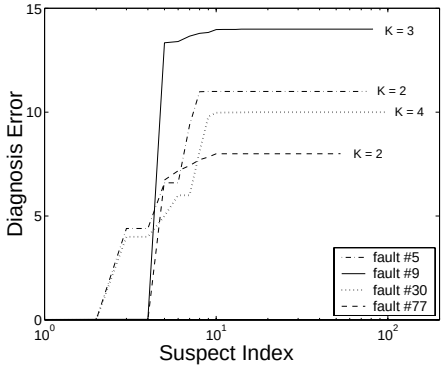

Figure 4: Diagnosis Errors For Four Defects, s5378

6.2 Pattern Generation for Diagnosis

Generating "good" diagnostic patterns for delay faults is a complex task. Since the ATPG has the burden of ensuring that small defects are not missed due to poor quality patterns, in general, it means that timing information needs to be involved.

A given path can be sensitized with many different patterns resulting in different path delays. "Good" patterns are defined as those that sensitize the fault through long paths and produce a long delay on it. It is the task of the pattern generator to generate patterns resulting in longer path delays.

Due to complexity reasons, most conventional path delay fault pattern generators do not take timing information into account and generate tests based purely on logic path sensitization conditions. Thus,

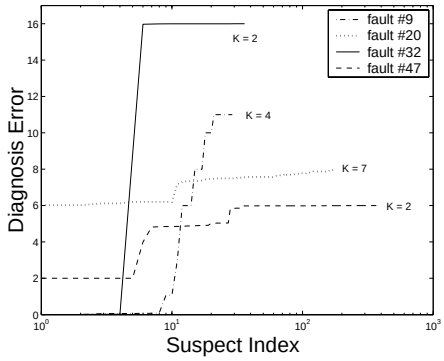

Figure 5: Diagnosis Errors For Four Defects, s15850

the patterns might not always exercise the worst-case timing scenarios and result in longest path delays. One possible solution for generating high quality patterns could be a timed ATPG technique in which the timing information is used in addition to the logic information to generate deterministic patterns [14]. However, due to its complexity, timed ATPG is not practical. In addition, due to the statistical nature of the timing information, patterns derived using nominal or worst-case delays might not be best for all circuit instances. On the other hand, considering statistical timing would just further hurt the efficiency of this method.

From previously published work [19], Genetic Algorithm based ATPG seems to be able to strike a good balance between the quality of produced test patterns and complexity. Therefore, we use this approach to generate fine-tuning patterns for enhancing the diagnosis resolution in phase 3.

6.2.1 *Genetic Algorithm based ATPG*

For each target fault, we select a small set of long paths based on statistical timing analysis. Next, for each path, we assign the mandatory logic assignments to sensitize it. After assigning the mandatory values to sensitize a given path, usually there are still many unspecified values at the primary inputs. Even though the path sensitization does not depend on the assignment of these values, the path delay does. Therefore, we use an iterative GA process to specify these PI values such that they result in longer path delays.

Genetic algorithms [18] are search algorithms based on the mechanics of natural selection and natural genetics. In our GA based delay fault ATPG, the solution space is represented by the set of all possible patterns satisfying the mandatory values for sensitizing the target path delay fault. Each pattern has an associated fitness value which is, in our case, given by the path delay under the current set of patterns. We use statistical dynamic timing simulations to evaluate the path delays [16]. In the initial generation of patterns, the unspecified PI values are randomly assigned. Next, the GA searches for the pattern(s) with the optimal solution using three processes: selection, crossover and mutation [19]. The objective of the GA is to evolve a population of patterns having high fitness values. This iterative process continues until the number of generations reaches a pre-defined value.

7. EXPERIMENTAL RESULTS

In this section, we present the final experimental results for all phases of our diagnosis framework. For each design and each injected defect, we first apply phase 1 and phase 2 of our diagnosis methodology. The results after phase 2 are given in Section 5. Next, for each defect, we select K highest ranking suspects to use them as target faults for generating the fine-tuning patterns. The value of K is determined based on when the rapid change of the diagnosis error is observed for the given defect, as described in Section 6.1.

Let the set of target faults be $F = \{f_1, \ldots f_K\}$. For each target

Ckt; average number of suspect faults S after phase 2	K	Phase 2 (%)	Phase 3 (%) Random	GA
s1196	1	23	27	35
$S = 8$	2	48	51	64
	3	60	64	77
	5	80	82	90
s1423	1	7	12	25
$S = 11$	2	29	37	62
	4	52	57	75
	9	75	78	87
s5378	1	29	36	58
$S = 12$	4	58	67	83
	6	86	87	91
	9	86	88	95
s9234	1	15	18	25
$S = 10$	3	35	41	56
	5	64	73	81
	8	73	81	93
s15850	2	38	45	52
$S = 16$	6	57	61	70
	10	76	79	83
	13	95	95	98

Table 2: Fine-Tuning The Diagnosis Accuracy

fault $f_i \in F$, we select 10 longest paths and use them in the GA based ATPG to generate patterns. The paths are selected using statistical timing analysis tool [16]. For each path, we set the size of the GA population to 12 and the number of generations as 5. At the end of the GA process, for each target fault f_i, we pick 4 patterns resulting in a longest path delay.The path delay is obtained using statistical dynamic timing simulations [16].

Next, we use our phase 2 diagnosis methodology again with these additional patterns. The experiment follows the same spirit as the one described in Section 5. To evaluate the contribution of generating high quality patterns as opposed to just generating additional patterns without timing information, for each target fault, we also derive 40 additional test patterns such that the unspecified values at the primary inputs after sensitizing the paths are assigned randomly (rather than generated using the GA based ATPG).

Table 2 shows the results for diagnosis resolution in the case of single defect. The average number of target faults for which additional test patterns are generated is given under the circuit name. The average is taken over all injected defects for the given design. The percentages in column 3 are the results after phase 2, i.e., they are repeated from Table 1. The results under column marked "Phase 3" are obtained with the two sets of additional patterns: random and GA generated. Even though both sets result in improved diagnosis resolution, the GA generated patterns have a clear advantage. Due to the relatively low cost of GA based ATPG as compared to deterministic timed ATPG, the results suggest that the extra cost is worth the effort.

8. CONCLUSIONS AND FUTURE WORK

In this work, we study the problem of delay defect diagnosis based upon statistical timing model. We propose a 3-phase diagnosis methodology that gradually improves the diagnosis resolution. While most of the previous delay diagnosis approaches stop after faults can be distinguished using logic conditions in phase 1, our main contribution is a novel delay diagnosis methodology based on statistical timing information used to further distinguish faults in phase 2 and phase 3. To do this, we propose an error-function-driven diagnosis algorithm based upon the single defect assumption, and demonstrate its effectiveness under different defect assumptions. We note that this algorithm was the best after experimenting with several other different approaches. In phase 3, we propose a novel methodology for deriving fine-tuning diagnostic patterns to further enhance the diagnostic resolution. Our experimental results indicate that using our diagnosis framework with extra effort of generating additional good patterns results in improved diagnosis resolution.

Future research includes many possible directions, including 1) development of better diagnosis error functions and new diagnosis algorithms accordingly, 2) development of methods to reduce the expense of computing and storing the probablistic fault dictionary, 3) the improvement of dynamic statistical timing simulator for more accurate delay fault simulation.

9. REFERENCES

[1] K. Baker, *et al.* Defect-Based Delay Testing of Resistive Vias-Contacts, A Critical Evaluation. *Int'l Test Conf.*, 1999.

[2] M. A. Breuer, C. Gleason, S. Gupta. New Validation and Test Problems for High Performance Deep Sub-Micron VLSI Circuits. *Tutorial Notes, VLSI Test Symp.*, 1997.

[3] R. C. Aitken. Nanometer Technology Effects on Fault Models for IC Testing, *Computer*, Nov. 1999.

[4] K-T Cheng, S. Dey, M. Rodgers, K. Roy. Test Challenges for Deep Sub-Micron Technologies. *Proc DAC*, 2000.

[5] J.-J. Liou, A. Krstić, K.-T. Cheng, D. Mukherjee, S. Kundu. Performance Sensitivity Analysis Using Statistical Methods and Its Applications to Delay Testing. *ASP DAC*, 2000.

[6] M. Abramovici, M. A. Breuer, A. D. Friedman. *Digital Systems Testing and Testable Design*, W.H.Freeman, 1990.

[7] A. Krstic, L.-C. Wang, K.-T. Cheng, J.-J. Liou. Diagnosis of Delay Defects Based Upon Statistical Timing Models — The First Step. *Design Automation and Test in Europe*, 2003.

[8] D. B. Lavo, B. Chess, T. Larrabee, F. J. Ferguson. Diagnosing Realistic Bridging Faults with Single Stuck-At Information. *IEEE Trans. on CAD*, Mar. 1998.

[9] J. Ghosh-Dastidar, N. A. Touba. Diagnosing Resistive Bridges Using Adaptive Techniques. *Custom Integr. Circ. Conf.*, 2000.

[10] H. Takahashi, K. O. Boateng, Y. Takamatsu. A New Method for Diagnosing Multiple Stuck-at Faults using Multiple and Single Fault Simulations. *VLSI Test Symp.*, 1999.

[11] P. Girard, C. Landrault, S. Pravosssudovitch. A Novel Approach to Delay-Fault Diagnosis. *Design Automation Conf.*, 1992.

[12] P. Pant, A. Chatterjee. Path-Delay Fault Diagnosis in Non-Scan Sequential Circuits with At-Speed Test Application. *ITC*, 2000.

[13] M. Sivaraman, A. J. Strojwas. Path Delay Fault Diagnosis and Coverage - A Metric and an Estimation Technique. *IEEE Trans. on CAD*, Mar. 2001.

[14] W.-Y. Chen, S. K. Gupta, M. A. Breuer. Test Generation for Crosstalk-Induced Delay in Integrated Circuits. *ITC*, 1999.

[15] Y-M. Jiang, A. Krstic, K.-T. Cheng, Estimation for Maximum Instantaneous Current Through Supply Lines for CMOS Circuits. *IEEE Trans. on VLSI*, Feb, 2000.

[16] J.-J. Liou, A. Krstic, L.-C. Wang, K.-T. Cheng. False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation. *Design Automation Conf.*, 2002.

[17] Anacad. Eldo v4.4.x User's Manual. 1996.

[18] D. E. Goldberg, R. Burch. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[19] A. Krstic, Y.-M. Jiang, K.-T. Cheng. Pattern Generation for Delay Testing and Dynamic Timing Analysis. *IEEE Trans. on CAD*, Mar. 2001.