Critical Path Selection for Delay Fault Testing Based Upon a Statistical Timing Model

Li-C. Wang, *Member, IEEE*, Jing-Jia Liou, *Member, IEEE*, and Kwang-Ting Cheng, *Fellow, IEEE*

***Abstract*—Critical path selection is an indispensable step for testing of small-size delay defects. Historically, this step relies on the construction of a set of worst-case paths, where the timing lengths of the paths are calculated based upon discrete-valued timing models. The assumption of discrete-valued timing models may become invalid for modeling delay effects in the deep submicron domain, where the effects of timing defects and process variations are often statistical in nature. This paper studies the problem of critical path selection for testing small-size delay defects, assuming that circuit delays are statistical. We provide theoretical analysis to demonstrate that the new path-selection problem consists of two computationally intractable subproblems. Then, we discuss practical heuristics and their performance with respect to each subproblem. Using a statistical defect injection and timing-simulation framework, we present experimental results to support our theoretical analysis.**

***Index Terms*—Path selection, process variations, statistical timing, testing.**

I. Introduction

PROCESS variations, manufacturing defects, and noise are major sources affecting the timing characteristics of deep submicron (DSM) designs [1]. Process variations may result in a wide range of possible device parameters, causing variations in timing. Delay faults, due to interconnect defects and noise sources, can be hard to predict in terms of their actual delay sizes [2]. For these DSM timing effects, the traditional assumption of discrete-valued timing models may become invalid [3], [4]. These DSM timing effects can better be captured and simulated using statistical models and methods [5].

In today's industry, the single transition fault model remains one of the most affordable and effective models for at-speed testing. The transition fault model contains no timing information. Hence, it is often thought that transition fault tests are good for capturing large-size defects. For capturing small-size defects, it is a common practice to test a set of *critical paths*. Critical paths are defined by their *timing lengths*. Traditionally, the timing length of a path is calculated based on a discrete-valued model. As the timing model changes from a discrete-valued model to a statistical model, we need to restudy the problem of critical-path selection.

Manuscript received May 12, 2003; revised September 27, 2003. This work was supported in part by the National Science Foundation under Grant CCR-0312701 and in part by the Semiconductor Research Corporation under Grant 1173.001. This paper was recommended by Associate Editor F. N. Najm.

L.-C. Wang and K.-T. Cheng are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: licwang@ece.ucsb.edu; timcheng@ece.ucsb.edu).

J.-J. Liou is with the Electrical Engineering Department, National Tsing-Hua University, Taiwan (e-mail: jjliou@ee.nthu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2004.835137

The primary goal of this paper is to understand how the new path-selection problem is different from those formulated based on discrete-valued timing models. We formulate the new path-selection problem as an optimization problem consisting of two objectives. The first objective is to maximize the topological coverage of selected paths. The second objective is to maximize the *return of testing a path* by considering all paths that have been tested before the path. The second objective is formulated using the concept of *path correlation*. Considering path correlation in path selection makes our problem different from those studied before [6], [7].

To understand the complexity associated with the two objectives, we provide theoretical analysis to show that optimizing each objective is computationally intractable. We then analyze several heuristics for optimizing each objective individually. For optimizing both objectives together, we derive three heuristics and suggest that one heuristic (called H-Opt) should be better than the other two (called H-Timing and H-Segment).

To validate our theoretical results, we developed a statistical timing simulation framework capable of performing random defect injection and simulation. We introduce a quality evaluation scheme based upon the framework and present consistent experimental results to support our theoretical analysis.

This paper is organized into three parts. In Section II, we give a brief introduction of prior work. Section III introduces the path-selection problem based on a statistical timing model. We discuss the concepts of *path correlation* and *path independence* and their roles to our path-selection problem.

The second part consists of Sections IV–VIII. This part analyzes the complexity of the new path-selection problem. The goal of Section IV is to define the path-selection problem. In Section V, we show that for optimizing path selection with the statistical model, it is required to simultaneously optimize two independent objectives (call them $\wp_{\mathrm{obj1}}$ and $\wp_{\mathrm{obj2}}$). In Section VI, the problem of optimizing the first objective $\wp_{\mathrm{obj1}}$ is analyzed in detail. Then, in Section VII, we analyze the second objective $\wp_{\mathrm{obj2}}$. In Section VIII, we combine results from Sections VII and VIII to estimate the performance of three path-selection heuristics.

Experimental methods are explained in Section IX. This section also includes experimental results to compare the three heuristics. The last section concludes the paper and suggests future research directions.

II. Prior Work

Historically, the definition of a critical path is based upon nominal or worst-case timing analysis [6]–[11]. In traditional critical-path analysis, delays are often bounded by ranges. In the industry, timing analysis relies on cell characterization, where the earliest, latest, and average signal arrival times are estimated for the pin-to-pin delay of each cell [8]. With these discrete timing values, the delay of a path can be defined as the accumulated delay on the path. The set of critical paths can then be constructed by selecting either a fixed number of the longest paths, or all paths that fall into a predefined time range. If circuit-segment coverage is considered, then the set of critical paths can include, for each signal segment, the longest timing path that covers the segment [6], [7].

When critical paths are calculated based upon a fixed threshold, the number of selected paths can be very large [6], [12]. In practice, this is not feasible due to the limitation on test length and the possible high cost associated with path delay fault ATPG and fault simulation. Usually, a pattern generated from a given path can fortuitously sensitize only a few or no other paths. Hence, the size of a pattern set based on a path set can be close to the size of the path set.

The authors in [13] proposed a test generation method where estimated path delays were calculated repeatedly based on process parameters after each path was selected. By doing so, they demonstrated that it was possible to dramatically reduce the number of selected paths for a given threshold value. The authors in [9] generalized the path selection concept in [13]. They proposed a delay model that could capture intradie (within-the-chip) variations as well as interdie (chip-to-chip) variations. Based upon their variation model, the authors presented a path selection algorithm to reduce the sizes of path sets without sacrificing their test quality.

The path selection methods described above are all based upon discrete-valued timing models. While the goal in [9] and [13] was to reduce the number of selected paths by taking process variations into account, our goal is to further optimize the critical path set by using a statistical model that can better capture the entire spectrum of the variations.

III. Considerations in Path Selection

Consider the example shown in Fig. 1. Suppose cell characterization gives the mean and standard deviation of the delay random variable for each pin-to-pin delay. For example, the pin-to-pin delay $a \rightarrow e$ is a random variable with mean 15 and standard deviation 1. By assuming 3σ bound, the minimal and maximal delays are $15-3\times1$ and $15+3\times1$, respectively. Hence, in a discrete delay model, the pin-to-pin delay can be denoted as $[12, 15, 18]$ which represents the earliest, the average, and the latest signal arrival delays.

Based on the worst-case scenario using the 3σ bounds, path P4 is most critical because the worst-case delay is $12+3\times3+9+3\times3=39$. The worst-case delays for path P1, P2, and P3 are 31, 32, and 38, respectively. If we are to test two paths, paths P3 and P4 will be selected based on the 3σ worst-case analysis.

If we statistically stimulate 1000 sample instances of the example circuit according to the delay distributions (assuming that they are normal), then among these instances, roughly 436 of them will have P1 as the longest path. P3 will appear to be the longest path on 236 instances. P4 will only be the longest path on 137 instances. From this perspective, one may argue that

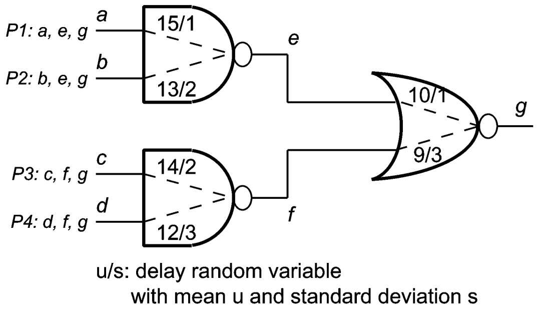

Fig. 1. Illustrative example.

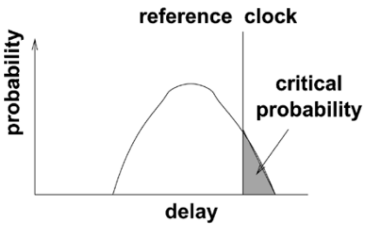

Fig. 2. Critical probability.

in order to maximize the chance of capturing a defective chip, testing P1 and P3 will be better than testing P3 and P4.

Whether to test $\{\mathrm{P1}, \mathrm{P3}\}$ or to test $\{\mathrm{P3}, \mathrm{P4}\}$ depends on the test clock selected and the target quality level. One way to answer this question is to statistically simulate a large number of sample chips and then analyze those chips whose delays exceed the given test clock.

A statistical timing model utilizes probability distributions to replace the delay bounds in traditional discrete-valued timing analysis. Using delay distributions facilitates more detailed analysis in order to better answer the questions mentioned above. This motivates us to study the critical path-selection problem from a statistical perspective.

We note that if pin-to-pin delays are characterized as random variables, then the delay of each path can be characterized as the summation of all pin-to-pin random variables on the path. The summation can be done by the convolution of the two random variables. For example, the delay of path P1 in the example is also a random variable whose probability distribution is the convolution of the two pin-to-pin random variables from "$a \rightarrow e$" $(15/1)$ and $e \rightarrow g(10/1)$. Note that the calculation of this summation depends on whether the two random variables are correlated or independent.

A. Critical Probability

If the timing model becomes statistical, the definition of a critical path can no longer be deterministic. In this paper, we define a critical path based on a given reference clock. We define the *critical probability* of a path as the probability that it exceeds the given clock (Fig. 2).

Suppose that a statistical timing model is given. From the model, we derive n sample instances each with a different delay configuration. Suppose that a path p has a critical probability crt. Intuitively, this crt estimates the percentage of the n sample instances where p is greater than the clk. If clk is used as the test clock, then by testing p (checking if the delay of p is $\leq clk$ on

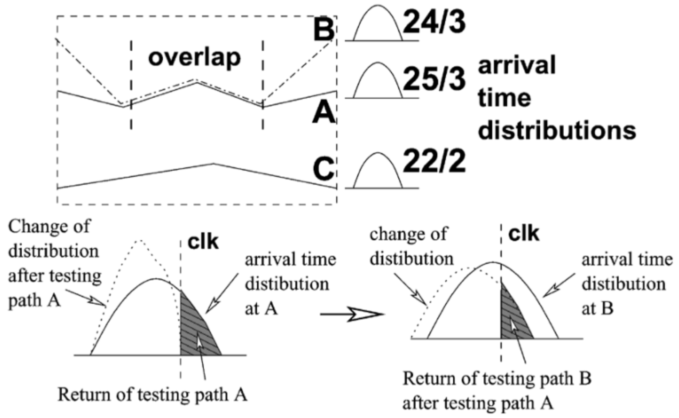
overlap
B
A
C
24/3
25/3
22/2
arrival time distributions
Change of distribution after testing path A
clk
arrival time distibution at A
Return of testing path A
change of distibution
clk
arrival time distibution at B
Return of testing path B after testing path A

Fig. 3. Path correlation and diminishing test return.

every sample instance), roughly $crt \times n$ samples would fail the clock. Hence, the critical probability can be thought as a way to measure, if p is tested based on clk, the probability of a chip failing. From another perspective, the critical probability can be thought as a way to measure the *return of testing* the path. For example, testing p would remove the $crt \times n$ failing samples. Afterwards, we only need to continue testing of the remaining $(1 - crt) \times n$ samples.

The most critical path can be defined as the one that has the largest critical probability. The critical probabilities allow us to rank paths. This probabilistic view of path timing suggests that more sophisticated analysis is required for selecting critical paths for delay testing.

B. Path Correlation in Delay Fault Testing

Statistically, if two paths have a substantial overlap, then the return of testing the second path after testing the first path, should be reduced and is not the same as by testing of the second path alone. We call this statistical correlation between two paths as *path correlation*. Here, the return of testing can be thought as the the probability of capturing *additional* bad chips that have not been classified as defective chips so far.

Take Fig. 3 as an example. The arrival times of the paths are characterized in terms of their means and standard deviations at circuit primary outputs (POs). Three paths are shown in the example with the probability density functions (pdfs) denoted as 25/3 (path A), 24/3 (path B), and 22/2 (path C). If two paths are to be selected (based on a given clock clk, for example 25.5), simply choosing the two most critical paths would include paths A and B.

Suppose path A is first tested, and it is ensured that its output arrival time is within the given clock period clk. Then, after testing path A, we need to calculate the conditional timing pdf for path A. This is because if n chips are tested and i chips are classified as bad chips by testing path A, the timing pdf of path A on the remaining $n-i$ chips should be different. The new pdf is the conditional probability distribution given that the arrival time of path A is less than clk.

The change of the pdf on path A (from its original pdf to the conditional pdf) suggests that implicitly, testing path A has already tested a significant part of path B due to their overlap. Hence, by knowing the fact that path A is less than clk, the chance of path B exceeding clk (on the remaining $n - i$ chips) becomes smaller. This is reflected in the reduced return of testing path B afterwards, as shown in the figure.

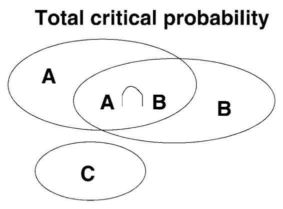

Fig. 4. Path correlation illustrated by a Venn diagram.

Consider path C that is topologically independent from path A. Since statistically path C is slightly "shorter than" path B, testing path C after path A may give a better return than testing path B. We note that topological independence does not directly imply that path C is a better choice after path A. To answer this question, we need to recalculate the pdf of path B based on the change of the pdf on path A.

C. Notes on Path Correlation

Fig. 4 illustrates the path correlation concept in terms of a Venn diagram. The *total critical probability* is the probability that there exists a path whose delay is greater than the given clock. The area $A \cap B$ denotes the *shared* critical probability by path A and path B, which characterize how much the two paths correlate. From the Venn diagram, it is clear that by considering this path correlation, we would want to select k paths to maximize the coverage of the area in the total critical probability space. If we change the total critical probability space to be a space with n discrete elements, then it becomes a traditional *Maximum Coverage* problem that has been well-studied in the literature [14], [15]. Later in Section VII, we will discuss this problem in more detail.

We note that for a given circuit, the total critical probability based on all paths of the circuit is the probability that the circuit delay is greater than the reference clock used to derive the critical probabilities (We do not consider path sensitization in this work). If three paths p_1, p_2, p_3 and a reference clock clk are given, then the total critical probability of these three paths is:

$$\begin{aligned}
&\mathrm{Prob}(p_1 > clk \vee p_2 > clk \vee p_3 > clk)\\
&\quad = \mathrm{Prob}(p_1 > clk) + \mathrm{Prob}(p_2 > clk) + \mathrm{Prob}(p_3 > clk)\\
&\quad\quad - \mathrm{Prob}(p_1 > clk \wedge p_2 > clk)\\
&\quad\quad - \mathrm{Prob}(p_2 > clk \wedge p_3 > clk)\\
&\quad\quad - \mathrm{Prob}(p_1 > clk \wedge p_3 > clk)\\
&\quad\quad + \mathrm{Prob}(p_1 > clk \wedge p_2 > clk \wedge p_3 > clk)
\end{aligned}$$

where $\wedge$ means AND and $\vee$ means OR.

$\mathrm{Prob}(p_1 > clk \wedge p_2 > clk)$ is the shared probability between path p_1 and path p_2. Moreover, we have

$$\begin{aligned}
\mathrm{Prob}(p_1 > clk \wedge p_2 > clk) = {}& \mathrm{Prob}(p_1 > clk)\\
&+ \mathrm{Prob}(p_2 > clk)\\
&- \mathrm{Prob}(p_1 > clk \vee p_2 > clk).
\end{aligned}$$

Also note that our definition of the path correlation is based on a given reference clock. For a given path whose critical probability is zero, it correlates to none of other paths even though it may topologically overlap with many paths. From the Venn diagram example, we can see that our definition of the path correlation is from a delay-testing point of view.

D. Path Correlation From Nonoverlapping Paths

Path overlap is only one possible reason that paths can have shared critical probabilities. Another reason can be due to *intradie* process variations. Hence, even though paths C and A are topologically independent, they can still have a shared critical probability. Testing A would have *implicitly* tested some part of C. In this work, we assume that process variations (intradie or interdie) have already been modeled in the statistical timing model in use. Hence, by defining the path correlation as the shared critical probability among paths, path overlap implies path correlation for two paths whose critical probabilities are not zero. We note that the reverse is not true.

In a statistical timing model, if we assume that the delay random variables on path A and those on path C are independent, then testing path A should not change the critical probability of path C. This means that, out of n samples, if testing A can identify i failing samples, then for the remaining $n-i$ samples, the critical probability of path C stays the same. In other words, suppose that the critical probability of path C is crt_c. Then, testing C on the n samples would identify $crt_c \times n$ failing chips. Testing C on the remaining $n-i$ samples would identify $crt_c \times (n-i)$ failing chips. Hence, even though the numbers of failing chips captured by testing path C in these two cases are different, the crt_c remains the same.

E. Path Independence

If a (spot) defect that falls beyond the *topological coverage* of a selected path set, then this defect has no chance of being detected. Hence, the selection of critical path also needs to consider *path independence* [6], [7] so that selected paths can cover as many circuit segments as possible. Here, the path independence is defined strictly from the topological point of view. Hence, if two paths overlap, then they are not independent. Otherwise, they are. Later in Section VI, we will discuss heuristics to solve this problem.

In summary, path selection for delay fault testing in the statistical domain should consider two objectives. First, we need to consider path independence in order to maximize the topological coverage. Second, we need to select paths with high critical probabilities where path correlation is taken into account. Due to the inclusion of path correlation, it would be difficult to develop a good heuristic to *simultaneously* select k paths as those algorithms proposed in [6], [7]. Hence, in this paper, we focus on heuristics that follows a path-by-path selection process.

IV. Problem Formulation

A circuit C is a graph with five-tuple (V, E, I, O, f), where V is a set of vertices, E is a set of arcs, and I, O are two subsets of V where $I \cap O = \phi$. f is a function on E. $\forall e_i \in E, f(e_i)$ is a random variable defined over $(0, +\infty]$.

This view of the circuit is consistent with the statistical timing model defined based upon cell-based pin-to-pin delay random variables proposed in [5]. We note that in the model, delays on both pin-to-pin of a cell and on the interconnects are modeled as random variables. In essence, the f function characterizes the random variables while each vertex corresponds to an input or output point of a cell. We note that in this circuit model, the delay random variables can be correlated, i.e., $f(e_i)$ can be correlated with $f(e_j)$ for any $i \neq j$. This circuit model is supported in our false-path-aware statistical timing analysis framework [16].

For simplicity, the circuit model is simpler than that actually simulated in the statistical timing analysis framework. To simplify the presentation, we do not differentiate between rising and falling transition delays. However, in the actual experiments, rising and falling transition delays are separated.

A path p on C is defined as a path starting from a vertex in I and ending at a vertex in O. Let $p = \{e_1, \ldots, e_i\}$. The *timing length* of p, denoted as $\mathrm{TL}(p)$ is a random variable characterized by the joint distribution $\mathrm{TL}(p) = f(e_1) + \cdots + f(e_i)$. For each vertex $o_i \in O$, the *arrival time* denoted as $\mathrm{Ar}(o_i)$ is a random variable characterized by the joint distribution $\max\{\mathrm{TL}(p_1), \ldots, \mathrm{TL}(p_j)\}$, where each $p_l, 1 \leq l \leq j$, ending at o_i. The *circuit delay* of C is defined as a random variable characterized by the distribution $\Delta(C) = \max\{\mathrm{Ar}(o_1), \ldots, \mathrm{Ar}(o_r)\}$, where $r = |O|$.

Given a path set P, the induced circuit of P on C, denoted as $\mathrm{induced}(P)$, is a subcircuit C' where any edge segment not on a path in P is removed from C. Hence, $\mathrm{induced}(P)$ may contain paths that are not in P.

Let D be a *defect distribution function* defined on C, which adds delay random variables to some circuit segments. We use $D(C)$ to denote the resulting circuit model after defects are injected on the circuit model C, and $D(p)$ to denote the resulting path after defects are injected on a specific path p.

Definition 1 (Problem Definition): Given a circuit C, a defect distribution D, and an integer k, find a path set P of k paths such that the following conditional probability is minimized:

$$\wp_{\mathrm{miss}} = \mathrm{Prob}\Big(\Delta\big(D(C)\big) > clk \,|\, \forall p_i \in P, \mathrm{TL}\big(D(p_i)\big) \leq clk\Big).$$

In other words, $\wp_{\mathrm{miss}}$ is defined as the conditional probability that the circuit delay $(\Delta(D(C)))$ is greater than the clock clk after all paths in P have been ensured to be $\leq clk$.

A. Defect Distribution

Problem 1 above is not well-defined unless a specific defect distribution D is given. There can be two ways to define D.

Definition 2 (Segment-Oriented): D can be a function defined on E, where $D(e_i) = (\delta_i, \gamma_i), \gamma_i$ is a random variable characterizing the probability of a defect occurrence on e_i, and δ_i is a random variable characterizing the delay defect size. Usually, we can assume that δ_i and γ_i are independent.

Definition 3 (Path-Oriented): Similarly, D can be a function defined on all paths, where each $D(p_i) = (\delta_i', \gamma_i')$.

In general, since γ_i' may depend on the length of p_i, and so does δ_i', the two random variables are not independent. Moreover, due to path overlap, for two paths p_i and p_j, δ_i', and δ_j' may not be independent. Similarly, γ_i' and γ_j' may not be independent. Because of these aspects, the optimization problem associated with a path-oriented D function could be much harder to solve than that with a segment-oriented D function. Moreover, measuring defect distribution using a path-oriented model can be difficult in reality.

In this paper, we consider a segment-oriented D function. For simplicity, we further assume that δ_i and δ_j are independent for $i \neq j$. We also assume that γ_i and γ_j are independent.

Example 1: With single-site uniform delay defect assumption, $\mathrm{Prob}(\gamma_1 = 1) = \cdots = \mathrm{Prob}(\gamma_m = 1) = (1/m)$, where $m = |E|$.

V. Optimization Objectives

Given a segment-oriented defect distribution D, we reformulate the problem slightly. To minimize $\wp_{\mathrm{miss}}$, we will try to maximize $\wp_{\mathrm{capture}}$, where $\wp_{\mathrm{capture}}$ is defined as below. For a selected set of paths P, let X denote the event "defects fall on P." Let Y denote the event "there is a path $p_i \in P$ such that $\mathrm{TL}(D(p_i)) > clk$" for a given test clock clk

$$\wp_{\mathrm{capture}} = \wp_{\mathrm{obj1}} * \wp_{\mathrm{obj2}} \tag{1}$$

$$\wp_{\mathrm{obj1}} = \mathrm{Prob}(X) \tag{2}$$

$$\wp_{\mathrm{obj2}} = \mathrm{Prob}(Y \mid X). \tag{3}$$

We note that $\mathrm{Prob}(Y)$ is the probability that by testing P, any defect on P can be captured. This probability is $\mathrm{Prob}(\mathrm{TL}(D(p_1)) > clk \vee \mathrm{TL}(D(p_2)) > clk \vee \cdots \vee \mathrm{TL}(D(p_k)) > clk)$ for $k = |P|$ ($\vee$ represents OR) and can be thought as the *total critical probability* of paths $\{p_1, \ldots, p_k\}$ after the defect function D is applied on those paths.

In contrast, $\mathrm{Prob}(Y \mid X)$ is the probability that by testing P, defects can be captured given that these defects fall on P. Further, note that given a path $p, \mathrm{TL}(D(p))$ can be rewritten as $\mathrm{TL}(p) + d$, where d is the random variable denoting the additional delay caused by the defect distribution function D.

Given a defect distribution function D, by assuming that $\forall i, j, \delta_i$ is independent of γ_j, we can remove the conditional event in (3) and obtain a simpler equation for $\wp_{\mathrm{obj2}}$

$$\wp_{\mathrm{obj2}} = \mathrm{Prob}(Y). \tag{4}$$

This is because if defect locations are independent of defect sizes, events X and Y become independent and $\mathrm{Prob}(Y \mid X) = \mathrm{Prob}(Y)$.

While the objective $\wp_{\mathrm{obj1}}$ tries to maximize the topological coverage of a path set P, the goal of the objective $\wp_{\mathrm{obj2}}$ will be to maximize the *total critical probability* resulting from P. Hence, the optimization objectives defined in (1) correspond to the two objectives discussed in Section III.

Theorem 1: $\wp_{\mathrm{capture}} = 1 - \wp_{\mathrm{miss}} - \wp_{\mathrm{no}}$, where $\wp_{\mathrm{no}}$ is the probability of defects having no faulty effect on circuit timing.

Proof: It suffices to show that the event spaces defined in the three probabilities are disjoint. Suppose that a circuit instance C_{in} is given (We note a circuit instance C_{in} is a sample instance randomly derived from the statistical circuit model C). Further, assume that some defects $\{d_1, \ldots, d_j\}$ occur on the circuit, with sizes $\{s_1, \ldots, s_j\}$, respectively. Then, there are three cases to be discussed.

- All d_i have no effect on the circuit performance. In this case, $(\Delta(D(C_{\mathrm{in}})) \leq clk)$, and this instance should be accounted for $\wp_{\mathrm{no}}$.
- At least one d_i affects circuit performance (i.e., $\Delta(D(C_{\mathrm{in}})) > clk$). For a defect affecting circuit performance, observe that if it falls beyond the topological coverage of P, it has zero probability of being detected. Then, for all defects on P, if any one causes a path p_i in P to have a delay greater than clk, then it is classified as a "capture" $(\mathrm{TL}(D(p_i)) > clk)$. Otherwise, the defect will be missed because we have $\Delta(D(C_{\mathrm{in}})) > clk$ after making sure that the event "$\forall p_i \in P, \mathrm{TL}(D(p_i)) \leq clk$" is true. Hence, the circuit instance will be classified as either a capture or a miss but not both.
- All defects affect circuit performance. In this case, the argument is similar to case 2.

Since the event spaces defined in the three probabilities are disjoint (and they form the total space), the theorem holds. □

Note that $\wp_{\mathrm{no}}$ depends only on the circuit C and the defect function D, and is independent of P. Therefore, to minimize $\wp_{\mathrm{miss}}$, we can try to maximize $\wp_{\mathrm{capture}}$.

A. Testing With and Without the Defect Function D

In the proof of Theorem 1 above, we assume that without a defect (or defects have no timing impact on a circuit), $\Delta(C) \leq clk$. In general, this assumption is not true. However, we can think that this is due to a two-stage process. In the first stage, the objective is to test against the timing model C without assuming a defect function D. In this case, we may use $\wp_{\mathrm{capture}} = \wp_{\mathrm{obj2}} = \mathrm{Prob}(B)$. $\wp_{\mathrm{obj1}}$ is excluded because we assume that no random defect is involved. The event B here should be "there is a path p_i such that $\mathrm{TL}(p_i) > clk$." Accordingly, we also change the miss probability as $\wp_{\mathrm{miss}} = \mathrm{Prob}(\Delta(C) > clk | \forall p_i \in P, \mathrm{TL}(p_i) \leq clk)$. After this stage, we assume that for a remaining chip C_{in} passing clk in the first stage, $\Delta(C_{\mathrm{in}}) \leq clk$ (with a very high probability) if no defect function D is involved. Then, in the second stage, a defect function D is introduced. In this paper we define timing validation as testing against C without D.

In delay testing, our goal is to capture *random* errors that may be caused by spot defects. In actual timing validation, the goal should be to correct *systematic* errors that may be caused by the difference between the intended design behavior and the actual design behavior. This involves diagnosis of systematic errors in the statistical timing model used to produce the design [17]. In our case, timing validation can be seen as a screening step to achieve $\Delta(C) \leq clk$.

B. Maximizing $\wp_{\mathrm{obj1}} * \wp_{\mathrm{obj2}}$

Given a circuit C, a defect function D, and a path set P, $\wp_{\mathrm{obj1}}$, and $\wp_{\mathrm{obj2}}$ can be estimated independently. However, it is important to note that to maximize $\wp_{\mathrm{capture}}$, it is not sufficient to maximize $\wp_{\mathrm{obj1}}$ and $\wp_{\mathrm{obj2}}$ independently. This is because these two objectives can be opposite to each other during the selection of P. In other words, the optimal P to maximize $\wp_{\mathrm{obj1}}$ may not be the same optimal P to maximize $\wp_{\mathrm{obj2}}$. However, we also note that if this can be done with the same set of P, then obviously the same P also maximizes $\wp_{\mathrm{capture}}$ as well.

Without knowing if a single optimal P set exists for both $\wp_{\text{obj1}}, \wp_{\text{obj2}}$ (and, hence, for $\wp_{\text{capture}}$), we consider the following three questions.

1) Independently, how do we optimize $\wp_{\text{obj1}}$?
2) Independently, how do we optimize $\wp_{\text{obj2}}$?
3) Can we combine the answers for question 1 and question 2 to optimize $\wp_{\text{capture}}$?

VI. Optimizing $\wp_{\text{obj1}}$

Given a circuit C, a path set P, and a segment-oriented defect distribution D, let $\{e_1, \ldots, e_k\}$ be the set of segments covered by P. Then, we have

$$\wp_{\text{obj1}} = \text{Prob(defects fall on P)} = \text{Prob}(\gamma_1, \cdots, \gamma_k) \quad (5)$$

where $\text{Prob}(\gamma_1, \cdots, \gamma_k)$ is the joint probability distribution of all random variables $\gamma_1, \ldots, \gamma_k$. If $\gamma_1, \ldots, \gamma_k$ are mutually independent, then we have

$$\wp_{\text{obj1}} \propto \frac{\sum_{\forall e_i \in P} \text{Prob}(\gamma_i)}{\sum_{\forall e_i \in C} \text{Prob}(\gamma_i)} = \frac{\sum_{\forall e_i \in P} \text{Prob}(\gamma_i)}{\text{some constant}}. \quad (6)$$

Hence, to maximize $\wp_{\text{obj1}}$ in (6), we will focus the discussion on the following optimization problem.

Definition 4 [Maximum Path Cover (MPC)]: Given a circuit graph $G = (V, E, I, O)$, a weight assignment function W defined on E such that $W(e_i) = w_i$, a path candidate set U, and an integer k, the problem is to find a subset P of k paths from U such that the total weight cover $(\sum_{e_i \in P} w_i)$ is maximized.

In the MPC problem, we use the weight w_i to denote the probability of defect occurrence $\text{Prob}(\gamma_i)$. The path set U is called the *Universal Path Candidate Set*. The construction of a U set will be discussed in Section IX-B later. In our path-selection methodology, a *path filtering* step is first applied to eliminate false as well as timing noncritical paths. The remaining paths after the path-filtering step form the U candidate set that serves as the basis for critical-path selection [16], [18].

A. Strategy to Study the MPC Problem

Fig. 5 illustrates the strategy to study the complexity of the MPC problem. This type of strategy is quite typical in theoretical computer science for studying the complexity of a newly-defined problem.

To show that the MPC is intractable, we identify a known intractable problem and construct a *reduction scheme* so that every problem instance in the intractable problem can be efficiently *translated* into a problem instance of MPC. The intractable problem that we identified was the weighted maximum vertex cover (WMVC) problem (as explained later). The reduction from the WMVC problem to the MPC problem implies that solving the MPC problem is as hard as solving the WMVC problem.

To utilize existing heuristics to solve the MPC problem, we identify another problem called Max-C and construct a reduction scheme to reduce the MPC problem to the Max-C problem. Through this reduction, any known heuristic for solving the Max-C problem can be applied to solve the MPC problem.

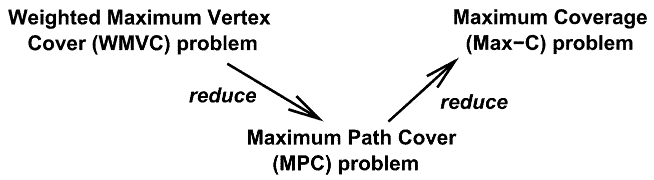
Weighted Maximum Vertex Cover (WMVC) problem
Maximum Coverage (Max−C) problem
reduce
reduce
Maximum Path Cover (MPC) problem

Fig. 5. Illustration of our strategy to study the MPC problem.

In the following, we discuss the first reduction scheme to show that the MPC problem is intractable. After that, we will discuss the second reduction scheme so that known heuristics can be applied to solve the MPC problem.

B. Intractability of the MPC Problem

One problem related to MPC is the minimum vertex cover (Min-VC) problem discussed in [14]. Given an undirected graph $G = (V, E)$, the Min-VC problem is to find the minimum set of vertices that cover all edges. It is shown in [14] that Min-VC is a problem in the MAX-SNP class, meaning that finding a $(1 + \epsilon)$ polynomial time approximation algorithm is NP-hard [14]. That is, if the optimal size of vertex cover to the problem is OPT, it is NP-hard to guarantee a vertex cover with size $\leq (1+\epsilon)\text{OPT}$ for any ϵ, where $0 < \epsilon \leq 1$.

There is a slightly different version of the Min-VC problem called the Maximum Vector Cover (Max-VC) problem. The problem is, given an integer k, find a set of k vertices that cover the maximum number of edges. Petrank [15] shows that it is also NP-hard to find a $(1 - \epsilon)$-approximation algorithm for the Max-VC problem.

The generalized version of the Max-VC problem is called the maximum coverage (Max-C) Problem. Given a set $I = \{1, \ldots, n\}$. Let J denote the indices of all nonempty subsets of I, and S_j denote the jth subset with index $j \in J$. Given a set $\mathcal{F} = \{S_i \,|\, i \in J\}$, a nonnegative weight w_i for each S_i, and a positive integer p, the problem is to find a subset $X \subseteq I$ where $p = |X|$ such that the total weight of all S_k which have nonempty overlap with X is maximized.

To give an example of the Max-C problem, consider a set $I = \{1, 2, 3\}$. All nonempty subsets of I would be $S_1 = \{1\}, S_2 = \{2\}, S_3 = \{3\}, S_4 = \{1, 2\}, S_5 = \{1, 3\}, S_6 = \{2, 3\}, S_7 = \{1, 2, 3\}$. To construct a Max-C problem instance, we select $F = \{S_1, S_4, S_5, S_6\}$ with a weight assignment $\{0.1, 0.2, 0.3, 0.4\}$, respectively. If we are allowed to choose two numbers to maximize the weight coverage, we would select 2 and 3 from I so that the total weight is 0.9 (including weights from S_4, S_5, S_6 because $\{2, 3\} \cap S_4 \neq \phi, \{2, 3\} \cap S_5 \neq \phi$, and $\{2, 3\} \cap S_6 \neq \phi$).

Let $d = \max\{|S_j| : j \in J\}$. The Max-C problem is a generalized version of the Max-VC problem because in the Max-C problem, if $d = 2$ and all $w_i = 1$, then the Max-C problem is reduced to the Max-VC problem. If we allow any weight assignments, but keep the constraint $d = 2$, the Max-C problem is reduced to the weighted version of the Max-VC problem (that is WMVC). We note that for $d = 2$, each subset can be thought as an edge connecting two vertices. Hence, the Max-C problem becomes a graph covering problem. For $d > 2$, it is the same as solving the WMVC problem on a hypergraph.

Lemma 1: MPC problem is intractable.

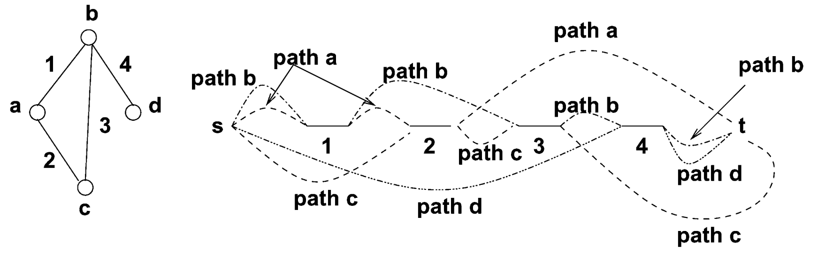

Fig. 6. Example of WMVC-to-MPC reduction scheme.

Proof: To demonstrate that MPC is intractable, it suffices to develop an efficient (polynomial time) reduction scheme to translate any WMVC problem instance into an MPC problem instance. Here we explicitly state the WMVC problem. Given an undirected graph $G=(V,E)$, a weight assignment $W(e_i)=w_i$ for all $e_i \in E$, and a positive integer p, find a p-vertex cover $X \subseteq V$ whose total covered weight is the maximum. Given a problem instance of WMVC, we will reduce it into an MPC problem instance using the following polynomial time reduction.

1) Create two nodes s and t.
2) Order all edges as $e_1,\ldots,e_m$ where $|E|=m$.
3) Pick a vertex $v_j \in V$, create a path p_j where p_j starts from s and ends at t, and contains all ordered edges $\{e_{j_1},\ldots,e_{j_k}\}$ connecting v_j. Add p_j to the U path set. The following "pseudoedges" with weight assignments equal to some fixed small number close to zero are added to connect $\{e_{j_1},\ldots,e_{j_k}\}$ in order to form a path.
   - Add a pseudoedge from s to e_{j_1}.
   - Add a pseudoedge from e_{j_k} to t.
   - For any adjacent edges $e_{j_l}, e_{j_{l+1}}$, add a pseudoedge from e_{j_l} to $e_{j_{l+1}}$.
4) $V=V-\{v_j\}$, if V is empty, stop; otherwise, go to step 3.

Fig. 6 illustrates the reduction scheme with a simple example. The graph contains four vertices $\{a,b,c,d\}$ and four edges ordered as $\{1,2,3,4\}$. In the transformed path problem instance, for example, edges 1, 3, and 4 have b as their end point and, hence, "path b" passes through edges 1, 3, and 4 and skip 2 with a pseudoedge (denoted as a dashed edge). The resulting MPC instance contains four paths $\{\text{path } a, \text{path } b, \text{path } c, \text{path } d\}$ in the U set, where each path corresponds to a vertex.

It is obvious that the above reduction is an $O(m|V|)$ algorithm. By keeping the weight assignment for each original edge, the MPC problem is to find p paths from U to cover the maximum total weight. If we ensure that the total weight given by all pseudoedges is far less than the minimum edge weight assigned in the original WMVC problem, then those pseudo edges will have no impact on the total weight calculation for the optimal solution.

Let $T(P)$ denote the total weight covered by a solution P in MPC and $T(X)$ denote the total weight covered by a solution X in WMVC. Then, for any two solutions P_1, P_2 in MPC, where $|P_1|=|P_2|=p$, there exist two corresponding solutions X_1, X_2 in WMVC (just map a path back to its corresponding vertex) such that $T(P_1) < T(P_2) \Leftrightarrow T(X_1) < T(X_2)$. The same ordering in the solution spaces of the WMVC and WPC instances implies that if the optimal solution is unique, it is the same in both instances. Moreover, given an $\epsilon, 0 < \epsilon \leq 1$, if there exists a polynomial time $(1-\epsilon)$-approximation algorithm for MPC, then it implies that there exists a polynomial time $(1-\epsilon)$-approximation algorithm for WMVC. Hence, the MPC problem is intractable. □

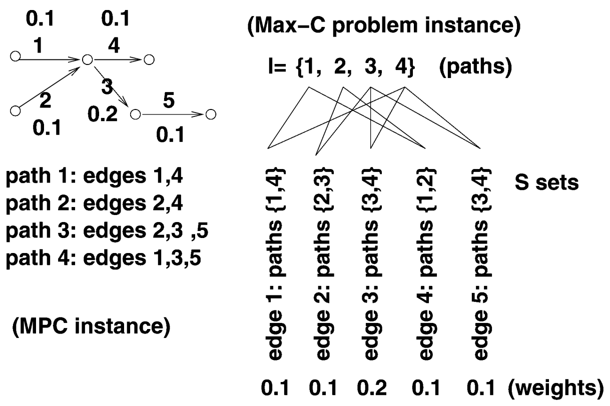

Fig. 7. Example of MPC-to-Max-C reduction scheme.

C. Heuristics to Find Approximate Solutions for MPC

In this section, we will discuss heuristics to solve the MPC problem. Most of these heuristics have been analyzed for the Max-C problem. Therefore, to facilitate the discussion, we will first show a polynomial time reduction from MPC to Max-C.

Lemma 2: Given that the size of U path set is of polynomial size in terms of $|E|$, MPC is polynomial-time reducible to Max-C such that if there exists a polynomial time $(1-\epsilon)$-approximation algorithm for Max-C, the algorithm is a $(1-\epsilon)$-approximation for MPC.

Proof: The mapping from MPC to Max-C is straightforward. Fig. 7 illustrates the idea with a simple example. Let $U=\{p_1,\ldots,p_n\}$ be the path candidate set. We simply let $I=\{1,\ldots,n\}$ in the Max-C. Each S_i in the Max-C problem corresponds to an edge e_i. Hence, the weight of each $e_i(w_i)$ is also the weight for S_i. We also have $j \in S_i$ if p_j contains the edge e_i. Essentially, the MPC problem is the same as the WMVC problem on a hypergraph. □

The above reduction further implies that if there exists a heuristic that can give a good approximate solution for the Max-C problem, then the heuristic can also provide a good approximate solution for the MPC problem.

1) Linear Program Relaxation (LPR) Heuristic: Authors in [19] utilize LPR to solve the Max-C problem. They demonstrate that the LPR heuristic is a $[1-(1-(1/d))^d]$-approximation algorithm, where $d=\max\{|S_j| : j \in J\}$. For WMVC, $d=2$ and hence, LPR heuristic is a $(3/4)$-approximation algorithm. With Lemma 2, the LPR heuristic is also a $[1-(1-(1/l))^l]$-approximation algorithm for the MPC problem, where l is the maximum path length measured by the total number of edges on a path. It is interesting to note that if a circuit is shallow (e.g., the path length is small), the LPR heuristic will perform better.

The LPR heuristic requires solving LP problem for maximizing $\wp_{\text{obj1}}$ alone. If we adopt this heuristic, it is hard to see how to combine it with any other heuristic(s) used to maximize $\wp_{\text{obj2}}$ later. Since our final goal is to maximize $\wp_{\text{capture}}$, not just $\wp_{\text{obj1}}$, the LPR heuristic is not a suitable candidate even though it is the best approximation algorithm known for the MAX-C problem.

2) Greedy Heuristics: In the following, we discuss simple "greedy" type of heuristics. Our first greedy heuristic is a typical and widely-used one in many optimization applications.

Heuristic 1: In each step, select the path that results in maximum additional weight coverage.

Theorem 2: The greedy heuristic in Heuristic 1 is a $[1-(1-(1/k))^k]$-approximation algorithm for MPC problem, where k is the number of paths allowed in the problem.

Proof: It is well known as shown in [20] that the same greedy heuristic is a $[1-(1-(1/p))^p]$-approximation algorithm for the Max-C problem, where p is the number of selected points (vertices) allowed in the problem. Hence, by Lemma 2, the theorem holds. □

As k becomes large, the greedy heuristic approaches to $(1-(1/e))$-approximation, where e is the natural number.

Heuristic 2: Sort all paths according to their total weights covered. Select the largest k paths.

Let L be the number of total paths in U in an MPC instance (vertices in WMVC). Authors in [21] shows that the above heuristic for WMVC problem is a (k/L)-approximation algorithm. However, the same argument used in [21] does not hold for WMVC on a hypergraph. This is because an edge on a hypergraph can connect more than two vertices.

Actually, one can construct an MPC instance to make the performance of Heuristic 2 as bad as possible. In fact, let $p_1,\ldots,p_n$ as the sorted paths with total covered weights $t_1 \geq \cdots \geq t_n$. It is easy to construct an instance to "fool" the heuristic by making the first k paths $p_1,\ldots,p_k$ exactly the same except for the last edge segment. And for each last edge, we associate a very small weight ϵ. On the other hand, for all $p_{k+1},\ldots,p_n$, we make them all independent with each total covered weight all equal to "$t_1-\epsilon$." It is easy to see that the only bound we can have by Heuristic 2 is then, $(1/k)$. That is, the heuristic guarantees the selection of the first maximum-weight path, but nothing more.

Theorem 3: The Heuristic 2 is a $(1/k)$-approximation algorithm for MPC problem.

The best know heuristic is the $[1-(1-(1/p))^p]$-approximation by LPR, and the Heuristic 2 essentially is an unbounded algorithm. That is, as k becomes sufficiently large, this simple heuristic can perform poorly.

Heuristic 3: Sort all edges according to their weights. In each step, select a path that covers an uncovered edge whose weight is the maximum.

Let $m=|E|$. The following theorem is straightforward.

Theorem 4: Heuristic 3 is a (k/m)-approximation algorithm for MPC problem, where m is the number of edges covered by all paths in U.

Proof: Let the sorted edge weights be $w_1 \geq \cdots \geq w_m$. Let Sol be the solution weight given by the heuristic. It is clear that OPT $\leq \sum_{1\leq i\leq m} w_i$. Also, Sol $\geq (k/m)(\sum_{1\leq i\leq m} w_i)$ because Sol contains the largest-weighted k edges. Hence, the theorem holds. □

Since we usually expect that $m \ll L$, this heuristic provides a much better bound than the second heuristic.

VII. Optimizing $\wp_{\text{obj2}}$

Given a path $p_j=\{e_1,\ldots,e_i\}$. Let $A_j=\text{TL}(p_j)=a_1+\cdots+a_i$. We denote the *critical probability* of p_j as $crt_j=\text{Prob}(A_j>clk)$ for a given reference clock clk.

Given two paths p_i,p_j with initial critical probabilities crt_i,crt_j, respectively, the $\text{Prob}(A_i>clk \mid A_j \leq clk)$ may not be the same as $\text{Prob}(A_i>clk)$. We define $\text{Cor}(A_i,A_j)=\text{Prob}(A_i>clk \wedge A_j>clk)=\text{Prob}(A_i>clk)+\text{Prob}(A_j>clk)-\text{Prob}(A_i>clk \vee A_j>clk)$. Hence, $\text{Cor}(A_i,A_j)=\text{Prob}(A_i>clk)-\text{Prob}(A_i>clk \mid A_j \leq clk)$. Similarly, $\text{Cor}(A_i,A_j)=\text{Prob}(A_j>clk)-\text{Prob}(A_j>clk \mid A_i \leq clk)$. We observe that the correlation is symmetric between A_i and A_j.

$\text{Cor}(A_i,A_j)$ characterizes the *correlation factor* (or shared critical probability) between paths p_i,p_j (as described in Section III before).

Suppose that we rank all paths $p_1,\ldots,p_L$ in U according to their critical probabilities $crt_1 \geq \cdots \geq crt_L$. If we select $P_k=\{p_1,\ldots,p_k\}$, following the same spirit in traditional path selection where the k longest paths are usually selected, how well will this perform with respect to maximizing $\wp_{\text{obj2}}$? We analyze this question below.

A. Change the Objective of Maximizing $\wp_{\text{obj2}}$

To maximize $\wp_{\text{obj2}}$, we would select a path set P such that the probability of capturing any defect on P can be maximized $(\text{Prob}(Y))$. In Section V, we have explained that this probability is $\text{Prob}(\text{TL}(D(p_1))>clk \vee \text{TL}(D(p_2))>clk \vee \cdots \vee \text{TL}(D(p_k))>clk)$ for $k=|P|$.

Given two paths p_i,p_j, we have $\text{TL}(D(p_i))=\text{TL}(p_i)+d_i$ and $\text{TL}(D(p_j))=\text{TL}(p_j)+d_j$, where d_i,d_j are the random variables denoting the additional delays caused by the defect distribution function D. If we adopt the single-defect assumption, then $d_i=d_j=d$. This is because there is only one defect occurrence and hence, both paths have the same defect size. In this case, if $\text{Prob}(\text{TL}(p_i)>clk)>\text{Prob}(\text{TL}(p_j)>clk)$ then $\text{Prob}(\text{TL}(D(p_i))>clk)>\text{Prob}(\text{TL}(D(p_j))>clk)$, and vice versa. Moreover, $\text{Prob}(\text{TL}(D(p_1))>clk \vee \text{TL}(D(p_2))>clk \vee \cdots \vee \text{TL}(D(p_k))>clk)=\text{Prob}(\text{TL}(p_1)>clk-d \vee \text{TL}(p_2)>clk-d \vee \cdots \vee \text{TL}(p_k)>clk-d)$. If d is a fixed value, to maximize $\wp_{\text{obj2}}$, we would maximize the probability $\text{Prob}(\text{TL}(p_1)>clk' \vee \text{TL}(p_2)>clk' \vee \cdots \vee \text{TL}(p_k)>clk')$ where $clk'=clk-d$ is the new reference clock.

In practice, d is a random variable whose distribution may not be known in advance. Hence, to maximize $\wp_{\text{obj2}}$, we would simply maximize the probability $\wp'_{\text{obj2}}=\text{Prob}(\text{TL}(p_1)>clk$

$\vee\ \mathrm{TL}(p_2) > clk\ \vee \cdots \vee\ \mathrm{TL}(p_k) > clk)$ for a given clock clk. Strictly speaking, maximizing $\wp'_{\mathrm{obj2}}$ is not the same as maximizing $\wp_{\mathrm{obj2}}$, unless the defect size is fixed.

Without the single-defect assumption, we would maximize the probability $\mathrm{Prob}(\mathrm{TL}(p_1) > clk - d_1 \vee\ \mathrm{TL}(p_2) > clk - d_2 \vee \cdots \vee\ \mathrm{TL}(p_k) > clk - d_k)$, where $d_1, \ldots, d_k$ are the random variables for the defect sizes on these k paths. Similarly, without knowing what $d_1, d_2, \ldots, d_k$ are, the logical approach is again to maximize the probability $\wp'_{\mathrm{obj2}} = \mathrm{Prob}(\mathrm{TL}(p_1) > clk\ \vee\ \mathrm{TL}(p_2) > clk\ \vee \cdots \vee\ \mathrm{TL}(p_k) > clk)$.

Furthermore, we mentioned before that in timing validation, $\wp'_{\mathrm{obj2}}$ is the objective to be maximized. Hence, instead of maximizing $\wp_{\mathrm{obj2}}$ which may not be a well-defined problem, we can try to maximize $\wp'_{\mathrm{obj2}}$. Maximizing $\wp'_{\mathrm{obj2}}$ provides another advantage: the result does not depend on the defect function. This is an important feature because in reality, perhaps no one can 100% correctly characterize the true defect function.

We note that the optimization of $\wp'_{\mathrm{obj2}}$ depends on the reference clock clk. In this paper, we do not study the selection of this reference clock. However, it will become clear later that the selection of this clock does impact the selected path set. The intuition is that with a larger clock period, critical probabilities are more important than path independence during path selection. In other words, with a larger clock period we would favor the selection of long timing paths. On the other hand, with a short clock period, we would favor the selection of more independent paths.

B. Analogy to the Optimization of $\wp'_{\mathrm{obj2}}$

Given a circuit $G = (V, E, I, O, f)$, we first consider optimizing $\wp'_{\mathrm{obj2}}$ based on a fixed-delay model where $f = f_{\mathrm{fixed}}(e_i) = c_i$ for some fixed constant c_i.

Lemma 3: With the fixed-delay circuit model above, the optimal solution path set P for maximizing $\wp'_{\mathrm{obj2}}$ is to select the k longest paths.

Proof: It can be observed that if P consists of the k longest paths, then for an edge on the induced subcircuit $\mathrm{Induced}(P)$, the longest path that covers the edge in G must be included in P. Hence, $\wp'_{\mathrm{obj2}}$ is maximized for all the edges on the subcircuit $\mathrm{Induced}(P)$. □

We note that with a fixed-delay model, the concept of path correlation does not apply. However, the concept of path independence does apply. Lemma 3 does not imply that selecting the k longest paths would result in an optimal path set for testing. In the lemma, the path independence is not considered.

Next, we consider the case for $f = f_{\mathrm{random}}(e_i) = a_i$ where a_i is a random variable characterizing the delay on edge e_i. The Venn diagram illustration in Fig. 4, provides a good perspective to derive an analogy to the problem of maximizing $\wp'_{\mathrm{obj2}}$. Consider the following problem. Given a set of n elements $S = \{c_1, c_2, \ldots, c_n\}$. Given a set of subsets $S_1, S_2, \ldots, S_m$ derived from S, our goal is to select k subsets such that they cover the maximum number of elements in S. This is the un-weighted version of the maximum coverage problem (MCP). In the weighted version of the MCP problem, a nonnegative weight w_i is associated with each element c_i. Then, the objective is to maximize the total weight covered.

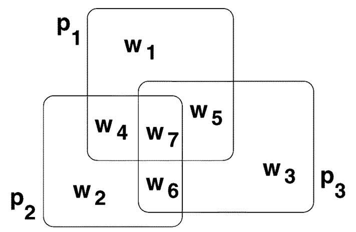

Fig. 8. Difficulty of mapping the probability space to MCP.

MCP is a different version of the Max-C problem discussed above. The authors in [20] shows that the simple greedy heuristic as Heuristic 1 is a $(1 - (1/e))$-approximation algorithm for the MCP problem. Hence, in the unweighted version, each time we would select a subset that has the maximum number of uncovered elements. In the weighted version, we would select a subset to maximum the additional weight coverage.

C. Heuristic to Maximize $\wp'_{\mathrm{obj2}}$

As illustrated in Fig. 4 before, maximizing $\wp'_{\mathrm{obj2}}$ is to maximize the *coverage* of critical probability in the total critical probability space. If we can translate the total critical probability space into the weighted n-element set S in the MCP problem, then we can show that the greedy algorithm is also a $(1 - (1/e))$-approximation algorithm for maximizing $\wp'_{\mathrm{obj2}}$. Unfortunately, this translation may not be easy.

Fig. 8 shows three paths $\{p_1, p_2, p_3\}$. In order to convert the total probability space into a weighted n-element set S as that in MCP, we need to consider all the partitions resulting from the intersection of individual probability subspaces. In this case, there are three paths and hence, the maximum possible number of partitions is $2^3 - 1 = 7$. We then sum up the probability in each partition to obtain w_i for $1 \leq i \leq 7$. Hence, in the resulting MCP problem instance, we have $S = \{c_1, \ldots, c_7\}$. Path p_1 corresponds to $S_1 = \{c_1, c_4, c_5, c_7\}$ covers total weight $w_1 + w_4 + w_5 + w_7$. Hence, selecting a subset S_j for $1 \leq j \leq 3$ is like selecting the path p_j.

Although the above reduction scheme works, it is not a polynomial time reduction, i.e., the size of the S set can be exponential. Given a path set $\{p_1, \ldots, p_n\}$, in general, translation from its total critical probability space to an MCP instance may result in a MCP problem with an exponential size in terms of n. However, if we can know in advance that no more than h paths can have a shared probability, for a constant h, then the translation can be done in polynomial time.

Even though we do not know that such an h exists, if we can still try to find a small h such that the partitioned subspaces can be approximated well, then the greedy algorithm can perform like a $(1 - (1/e))$-approximation algorithm.

The translation from the total probability space to the MCP problem space is difficult in theory. However, it is important to note that if the critical probabilities are calculated based only on N sample instances, then the maximum number of partitioned subspaces in the total critical probability space is $\leq N$. In this case, the translation can be done in polynomial time in terms of

$N \times n$. In other words, if N is the number of simulated samples in Monte Carlo simulation, then it is possible to derive the weighted set S in the simulation.

In summary, our heuristic to optimize $\wp_{\text{obj2}}$ follows the same spirit as the greedy heuristic for the MCP problem.

Heuristic 4 (Greedy by Considering Path Correlation): Each time, we select the path p with the largest critical probability. After the selection, we recalculate the critical probabilities for all unselected paths based on their correlations with p. In short, we select the path that can provide the maximum additional coverage of the total probability space.

We state the following observation to summarize our discussion above.

Observation 1: Let Sol be the total critical probability output by using Heuristic 4 to maximize $\wp'_{\text{obj2}}$. Let OPT be the true optimal value. We can have $(1-(1/e))(\text{OPT}) \leq \text{Sol}$, depending on how complex the path correlation is.

VIII. Heuristics to Optimize $\wp_{\text{capture}}$

Recall that $\wp_{\text{capture}} = \wp_{\text{obj1}} * \wp_{\text{obj2}}$. In the previous sections, we discuss heuristics to maximize $\wp_{\text{obj1}}$ and $\wp_{\text{obj2}}$ individually. Based up those results, in this section, we discuss three heuristics to maximize $\wp_{\text{capture}}$.

- **H-Timing**: Traditionally, the most natural way is to select the k longest paths. Under a fixed-delay model, this heuristic optimize $\wp_{\text{obj2}}$ (Lemma 3) but has little guarantee for $\wp_{\text{obj1}}$. With a statistical timing model, this heuristic becomes the selection of paths with the k largest critical probabilities, and is similar to Heuristic 2. This heuristic offers little guarantee for optimizing either $\wp_{\text{obj1}}$ or $\wp_{\text{obj2}}$.
- **H-Segment**: With this heuristic, optimizing $\wp_{\text{obj1}}$ has a higher priority than $\wp_{\text{obj2}}$. Given a circuit instance $G = (V, E, I, O, f)$ and a defect function $D(e_i) = (\delta_i, \gamma_i)$, at each step we select a path that covers the maximum additional probability of defect occurrence. If there are multiple such paths, we then select the one with the largest critical probability. Path correlation is not considered here. H-Segment follows the Heuristic 1 above and hence, is an $(1-(1/e))$-approximation algorithm for maximizing $\wp_{\text{obj1}}$. However, it provides no guarantee for optimizing $\wp_{\text{obj2}}$. Therefore, the performance can be unsatisfactory.
- **H-Opt**: This is Heuristic 4. The main question here is how H-Opt would perform with respect to $\wp_{\text{obj1}}$. If every selected path by H-Opt can include an additional uncovered circuit segment, then H-Opt is a (k/m)-approximation algorithm for optimizing $\wp_{\text{obj1}}$, under the condition that the defect occurrence probabilities are the same for all segments (ex. uniform distribution).

Consider an ordered path set $P = \{p_1, \ldots, p_i\}$ selected by using H-Opt in that order. Let $C' = \text{Induced}(P)$. For any $p' \in C'$ and $p' \notin P$, we want to have $\text{Prob}(\text{TL}(p') > clk \,|\, \forall p \in P, \text{TL}(p) \leq clk) = 0$ so that p' will not be selected after selecting p_i. This means that in the critical probability space, the *union* of all the critical probabilities from paths in P is equal to the critical probability of C'. In other words, $\text{Prob}(\Delta(C') > clk)) = \text{Prob}(\text{TL}(p_1) > clk \vee \cdots \vee \text{TL}(p_i) > clk)$. If this is true, then for the original circuit C, we can have $\text{Prob}(\Delta(C) > clk)) = \text{Prob}(\text{TL}(p_1) > clk \vee \cdots \vee \text{TL}(p_m) > clk)$ for some path set $\{p_1, \ldots, p_m\}$, where $m = |E|$ and $\text{Induced}(\{p_1, \ldots, p_m\}) = C$. This is unlikely to be always true.

Nevertheless, notice that the critical probabilities are defined based on the reference clock clk. For the purpose of path selection, this reference clock does not need to be the same as the test clock. We observe that by setting a short reference clock period, each selection of H-Opt would tend to include a path that contains uncovered circuit segments. In fact, if we set the reference clock period short enough, we can always make H-Opt to behave as Heuristic 3. In the extreme case, suppose that we set the reference clock at 0. Then, after testing a path, all segments on the path would have been ensured to have delay 0. Hence, it is obvious that $\text{Prob}(\text{TL}(p') > 0 \,|\, \forall p \in P, \text{TL}(p) \leq 0) = 0$.

Observation 2: Between a given test clock and 0, we can find a reference clock clk so that when using H-Opt to obtain a path set $P = \{p_1, \ldots, p_i\}$, we can have $\text{Prob}(\text{TL}(p') > clk \,|\, \forall p \in P, \text{TL}(p) \leq clk) = 0$ for $p' \in \text{Induced}(P)$ and $p' \notin P$. In this case, for every path $p_j, 1 \leq j \leq i$, there exists a segment edge e such that $e \in p_i$ and $e \notin \text{Induced}(P - \{p_i\})$.

Observation 3: Suppose that defect occurrence probabilities are all the same. Then, it is possible to find a short enough reference clock period clk such that H-Opt behave as Heuristic 3 for maximizing $\wp_{\text{obj1}}$. In this case, suppose that the optimal value is $\wp_{\text{capture}} = \text{OPT1} * \text{OPT2}$ (based on clk). H-Opt computes a solution value Sol for maximizing $\wp_{\text{capture}}$. Then, we can have $((k/m))\text{OPT1}(1-(1/e))(\text{OPT2}) \leq \text{Sol}$, depending on how complex the path correlation is.

Observation 3 suggests that in order for H-Opt to achieve a good topological coverage, we may want to use a short enough reference clock period. The selection of the reference clock represents a tradeoff between optimizing the topological coverage and optimizing the critical probability coverage. Because H-Opt is the only heuristic that can simultaneously address the optimization of both objectives, we expect that H-Opt should be better than H-Timing and H-Segment. In this following section, we will present experimental results to support this conjecture.

IX. Experimental Results

Our goal in this section is to provide experimental results in order to support our theoretical analysis in the previous three sections.

A. Approximate Path Correlations

The statistical method described in [16] provides a feasible approach to calculate the critical probability for a given path. In order to implement Heuristic H-Opt, we need a method to compute the shared critical probabilities among paths.

For the recalculation of critical probability after i paths are selected, $\forall i, 1 \leq i \leq k$, we use a Monte Carlo sampling approach to approximate the desired results. At first, we sample n chip instances, each with a different delay configuration derived from the statistical timing model. Suppose path A is selected. Then, we remove all instances whose delays of path A is greater than the reference clock. Suppose this leaves us

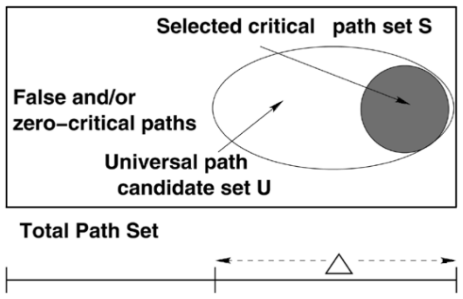
Selected critical path set S
False and/or zero-critical paths
Universal path candidate set U
Total Path Set
Δ

Fig. 9. Universal path candidate set.

$n-i$ instances. Based on this $n-i$ instances, we recalculate the critical probabilities. In other words, for another path B, we simply count the number of instances whose delays of path B is greater than the clock. Let this number be j. Then, the new critical probability of path B is $(j)/(n-i)$.

We note that even though paths A, B have no path correlation, it does not mean that there cannot be a circuit instance where the delay of path A and the delay of path B are both greater than the clock. Hence, among the i instances removed due to the selection of path A, some of them may have a delay of path B greater than the clock. However, because A and B have no correlation, in theory, removing these i instances should not change B's critical probability (as discussed in Section III-D).

Further note that our method to account for path correlations is only a quick approximation. Our goal is to obtain experimental results in order to assess the performance of H-Opt. Hence, the method is by no mean an optimal approach to implement H-Opt. In fact, after each step of path selection, the number of instances is continuously reduced. Therefore, the critical probabilities calculated at a later stage in the path selection process will be less accurate than those calculated at the beginning.

B. Universal Path Candidate Set

One key assumption during the theoretical analysis is that the number of paths being considered during path selection is $O(m)$, where $m = |E|$ in a given circuit model C. Without preprocessing, this is an unrealistic assumption because a circuit can easily have an exponential number of paths. In this section, we discuss a simple path-filtering scheme as the preprocessing step in the path-selection optimization process. During this preprocessing step, the goal is to quickly cut down the size of total path population.

In our methodology, we construct the *universal path candidate set* (U) [22] as illustrated in Fig. 9. The size of U is much smaller than the number of all paths and hence, coverage of U can be calculated much faster. We also ensure that by covering U, the probability of circuit delay greater than the clock is very small. That is, $\mathrm{Prob}(\Delta(C) > clk \,|\, \forall p_i \in U, \mathrm{TL}(p_i) \leq clk) \approx 0$. Then, the U set serves as the basis for later path selection.

If in our statistical framework a path has a very low probability of being a "long path," then in reality, we assume that it is unlikely for a small delay defect or variation on the path to cause a timing problem. In other words, we assume that the statistical timing model can correctly model the delay variations in reality. With this idea in mind, construction of U are based on two given parameters: a reference clock clk and a cutoff period T where $T \leq clk$.

TABLE I
EXAMPLES OF U PATH SET SIZES AND RUNTIMES (IN SECONDS)

Circuit	size	T	cpu	size	T	cpu	size	T	cpu
s5378	8093	170	163.44	4148	180	126.35	2342	190	84.53
s9234	5193	160	104.03	1618	170	45.03	595	175	24.13
s38417	8244	395	289.11	6444	397	246.93	4416	400	228.41

TABLE II
RUNTIMES FROM THE PATH SELECTION USING H-Opt (s5378)

# of paths	5	20	40	80	200
cpu seconds	1.98	3.78	6.48	6.91	8.4

The U consists of every path whose critical probabilities are greater than zero based on the cut-off period T. In other words, if all paths in U are covered, then any faulty behavior resulted from delay defect and variation of a delay size smaller than $\Delta = clk - T$ can be captured (small-size defects). This assumption is consistent with our goal to capture small-size delay defects. In other words, we can assume that transition fault testing is applied before testing of the critical paths. Hence, large-size delay defects (whose delay sizes $>\Delta$) will have been captured by transition fault testing.

After an initial U set is established, we can further prune the size of U by removing those functionally unsensitizable paths [23] using the new methodology developed in [16]. This can further reduce the size of the U path set.

C. Experimental Setup

Our experimental flow consists of three major phases, *U path set construction*, *path selection*, and *quality evaluation* as described below.

1) U Path Set Construction Phase: A cell-based statistical timing analysis framework was developed [5]. It requires precharacterization of cells, i.e., building libraries of pin-pin cell delays and output transition times (as random variables). For our experiments, we utilize a Monte-Carlo-based SPICE (ELDO) [24] to extract the statistical delays of cells for a 0.25 μm, 2.5 V CMOS technology. Each interconnect delay is also modeled as a random variable and is precharacterized once the RCs are extracted. This framework uses Monte Carlo simulation techniques to approximate the critical probabilities.

To construct the U path set, a cut-off period T is selected. At first, all paths with nonzero critical probabilities are included. Then, techniques proposed in [16] are applied to remove logic and timing false paths.

The selection of T affects the size of the U path set and limits the size of defect guaranteed to be captured. Although the logical approach to decide T should be based on manufacturing data with characterization of delay defect sizes, for experimental purpose, we took a simpler approach.

Given a circuit model C, for each circuit segment e_i, we randomly select a path to cover e_i. If the total number of circuit segments is m, this would give us m paths $\{p_1, p_2, \ldots, p_m\}$. Let their 3σ worst-case delays be $\{a_1, a_2, \ldots, a_m\}$, respectively. We would set $T = \min(a_1, a_2, \ldots, a_m)$. This process is to

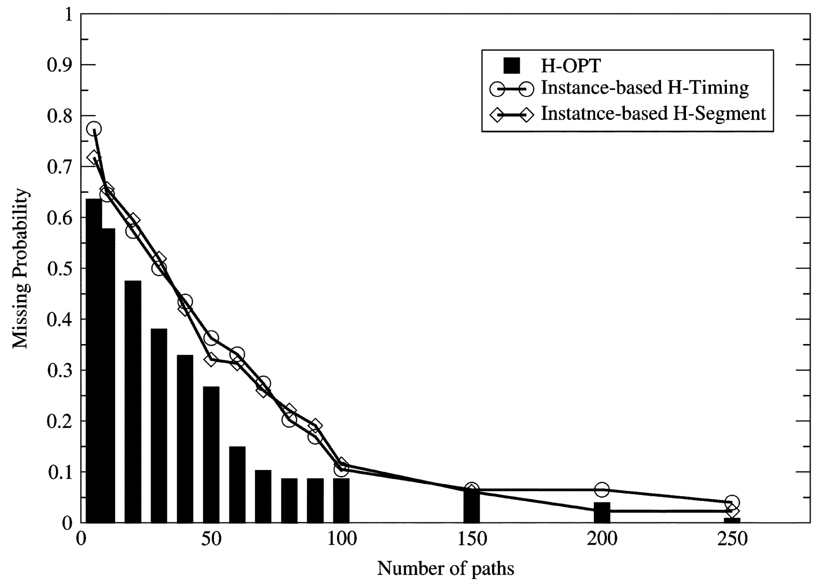

Fig. 10. Using sample-instance-based approach by assuming $e^{-0.1x}$.

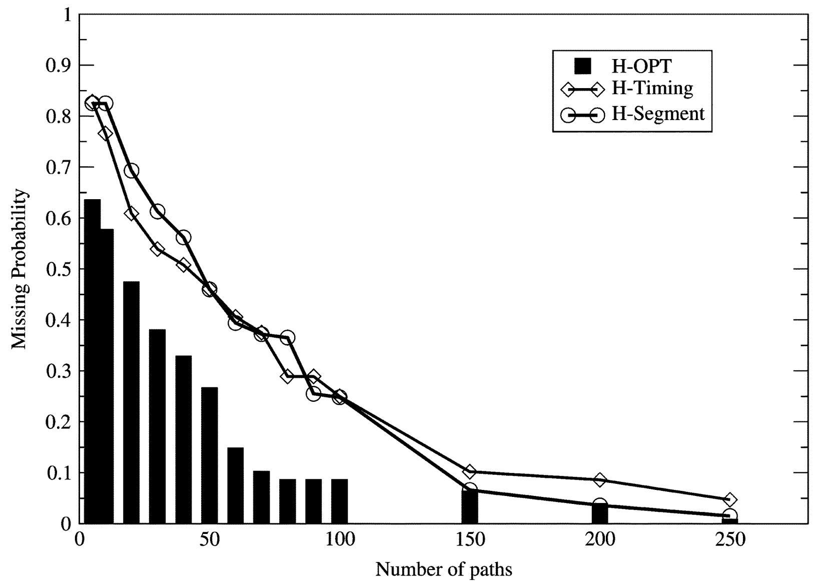

Fig. 11. Comparison results by assuming $e^{-0.1x}$.

approximate the results of transition fault testing. Therefore, our selection of T ensures that the critical path selection step focuses on detection of only small-size defects.

Table I presents results to demonstrate the impact of T on the size of U path set by setting T at different values. All experiments were obtained on a Pentium III 733-MHz machine running Linux mdk version 2.2.17. The runtimes depend on the desired accuracy in the converging criteria to stop the Monte Carlo simulation. Hence, the runtimes depend on the number of instances simulated. The results in Table I are based on simulation of 1000 sample instances. We note that from our experience, simulating a few thousands of samples is usually enough to stabilize the results in our experiments.

For the cut-off period T, the number 170 is roughly 17 ns. As it can be seen, depending on the selection of T, the size of a U path set can change. In general, the U path set sizes remain to be manageable [25].

The construction of a proper U path set to optimize its size and effective coverage will continue to be an interesting research topic. This paper does not focus on the optimization of the path set U.

2) Path-Selection Phase: In this phase, we apply each of the three heuristics (H-Timing, H-Segment, and H-Opt described in Section VIII) to derive a path set S where $|S| = k$.

Normally, the path selection is done following a path-by-path basis. However, for comparison purpose, we also experimented H-Timing and H-Segment where the path selection follows a sample-by-sample approach.

3) Sample-Based Path Selection: In these experiments, we first produce n sample circuit instances. Then, a path selection heuristic is applied to one sample instance after another. This process will produce a sequence of path sets $\{P_1, \ldots, P_n\}$. Then, the final path set is formed by collecting the k paths with the highest frequencies to appear in $\{P_1, \ldots, P_n\}$.

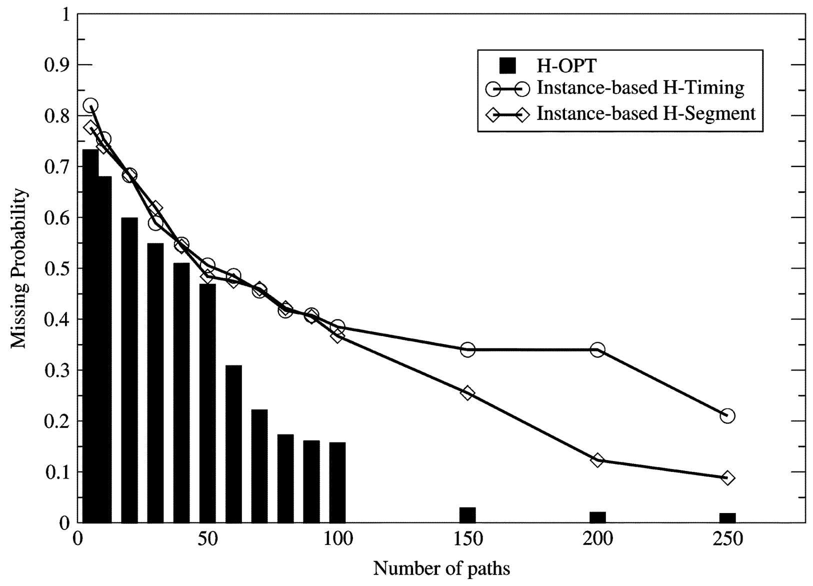

Fig. 12. Using sample-instance-based approach by assuming $e^{-0.04x}$.

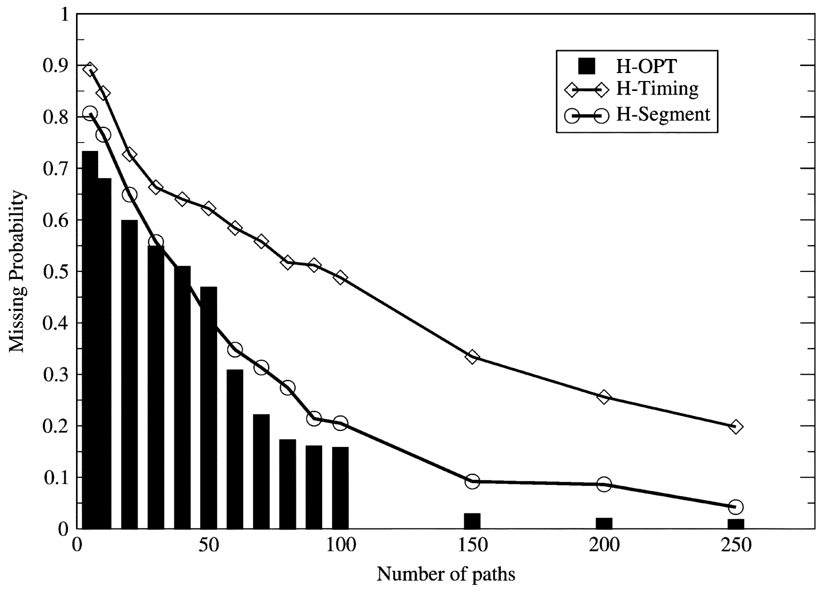

Fig. 13. Comparison results by assuming $e^{-0.04x}$.

It is important to note that when applying a heuristic to each particular sample instance, the circuit instance has a fixed delay configuration. Hence, individually the problem associated with each sample instance is easier to solve. In our experiments, we implemented both H-Timing and H-Segment using the sample-based approach.

4) Evaluation Phase: In our study, we estimate the quality of selected paths in terms of the *miss probabilities*. This estimation is calculated based upon paths alone, instead of the quality of tests generated for those paths [26]. Hence, our metric involves only static analysis and is pattern independent.

Assume that a path set S is given. In each Monte Carlo sampling run, first a circuit instance is derived from the statistical circuit model C. A random-location defect is injected on the circuit instance. The size of this defect is derived randomly from an exponential distribution explained below. This instance will then be evaluated by two steps: "analysis of S" and "analysis of $U - S$". The "analysis of S" is to check if there is any path in S longer than the test clock clk. If there is, then this instance is said to be faulty and covered by S(Covered). If the instance is not covered by S, the "analysis of $U - S$" step performs a similar analysis on the set of paths $U - S$. If there is any path in $U - S$ longer than the test clock, then this instance is faulty but is not covered by S(Noncovered). After simulating n instances (say 5000), we calculate the miss probability as the following:

$$\wp_{\text{miss}} = \frac{\text{Noncovered}}{\text{Covered} + \text{Noncovered}}.$$

In other words, the miss probability $\wp_{\text{miss}}$ is the conditional probability that a delay defect is not covered by S given that the delay defect will affect the circuit performance.

5) Defect Distribution: In the experiments, the evaluations are based on the assumption of a defect size distribution function $e^{-\lambda x}$ where x is the defect size and λ is a constant. We use $\lambda = 0.1$ and 0.04 in the experiments.

This exponential distribution for defect size (given that defects occur) has been studied in many publications [27], [28] and is a practical assumption to be used. Note that it is also possible to adopt other distributions. However, using other distributions in general does not invalidate the trends observed in our work [22]. As mentioned before, H-Opt only tries to optimize $\wp'_{\text{obj2}}$ and hence, a selected path set does not depend on the defect size distribution.

D. Results

We first utilize the results from circuit s5378 for detailed discussion. Results for other circuits are similar and hence, those results are not shown in detail. Instead, at the end of this section, Table III will present additional comparison results between H-Timing and H-Opt.

For s5378, the cut-off period T is set at 202. This results in a U path set with $|U| = 1328$. Table II shows the runtimes of the path selection step when using H-Opt. The reference clock is set at 208 which is also the test clock.

The following plots show the evaluation results for different heuristics. These plots demonstrate the trends of miss probabilities versus the number of selected paths. Results in Figs. 10 and 11 are based on the defect distribution $e^{-0.1x}$. For comparison, we also derive results for the defect distribution $e^{-0.04x}$ in Figs. 12 and 13. Random delay samples from $e^{-0.1x}$ range roughly from 0 to 40, while those from $e^{-0.04x}$ may extend to 100. By using two different defect size models, we can see how defect size may affect the results of different heuristics.

By comparing all four plots (Figs. 10–13), we see that H-Opt outperforms all other heuristics as predicted by the theoretical analysis (with smaller miss probabilities).

In Fig. 10, both sample-based H-Timing and sample-based H-Segment heuristics have similar levels of miss probabilities. The two curves converge when the number of selected paths increases. The sample-based H-Timing curve shows a slightly higher miss probability than the sample-based H-Segment curve at 250 paths. When the two heuristics are applied directly in the statistical domain (Fig. 11), H-Segment is better after 100 paths. From our theoretical analysis, we know that H-Segment aims to optimize $\wp_{\text{obj1}}$ while H-Timing optimizes none.

More interesting observations can be made when the model of $e^{-0.04x}$ is used (Figs. 12 and 13). Since the range of defect size spreads out, more edges have to be covered in order to maintain a low miss probability. As shown in the figures, the H-Opt still converges quickly as the number of paths increases. For sample-based results in Fig. 12, both H-Timing and H-Segment clearly fail to cover enough edge segments as required. Notice that H-Timing performs (relatively) much worse for larger number of paths. This is because selecting longest paths does not guarantee to cover independent segments. In Fig. 13, H-Segment can have a similar level of coverage as H-Opt while the number of selected paths is small. On the contrary, H-Timing becomes very ineffective.

TABLE III
COMPARISON RESULTS ON LARGE ISCAS'89 BENCHMARKS WITH TEN SELECTED PATHS

Circuit	U	H-Timing	H-Opt	T	clk
s5378	1238	0.485	0.129	202	208
s9234	842	0.0	0.0	172	189
s15850	1728	0.043	0.0	375	382
s35932	4702	0.266	0.0	192	198
s38417	3972	0.809	0.474	402	405
s38584	136	0.0	0.0	550	572

TABLE IV
RUNTIMES FROM H-Opt PATH SELECTION AND QUALITY EVALUATION

Circuit	s5378	s9234	s15850	s35932	s38417	s38584
H-Opt (Secs)	2.34	5.56	26.57	121.06	298.1	68.45
Evaluation (Secs)	0.29	1.37	5.23	11.7	6.95	0.26

TABLE V
COMPARISON RESULTS WITH 20, 30, AND 40 SELECTED PATHS

	H-Timing Vs. H-Opt		
Circuit	20 paths selected	30 paths selected	40 paths selected
s5378	0.38 Vs. 0.04	0.21 Vs. 0.013	0.16 Vs. 0.002
s15850	0 Vs. 0	0 Vs. 0	0 Vs. 0
s35932	0.02 Vs. 0	0.001 Vs. 0	0.001 Vs. 0
s38417	0.8 Vs. 0.3	0.8 Vs. 0.21	0.8 Vs. 0.14

Table III presents comparison results between H-Timing and H-Opt for several benchmark circuits using the defect-free simulation. With defect-free simulation, a faulty instance is the result of statistical variations. In these experiments, clk is both the test clock and the reference clock. These additional results further confirm the superiority of H-Opt.

Table IV shows runtimes from the H-Opt path selection step and from the quality evaluation step. Since the sizes of the U path sets are small in our experiments, the quality evaluation step can be done quite efficiently.

Table V shows additional comparison results based on 20, 30, and 40 selected paths using the defect-free simulation. It is interesting to observe that for s15850, H-Opt and H-Timing perform the same.

Fig. 14 shows the path delay profile from the circuit s15850. This profile is based on the average path delays. Notice that there are a few long paths whose delays are much greater than others. If a design contains only a few paths that are much longer than others, then both H-Timing and H-Opt will first focus their selection on these paths. In this case, a small number of selected paths is sufficient to achieve the complete coverage. This may partially explain the results for s15850 in Table V.

It is also interesting to see that for s38417, while H-Opt continues to improve its results as more paths are selected, H-Timing provides not much help by selecting more paths. This represents an extreme situation where H-Timing can be very ineffective and may have selected the wrong paths.

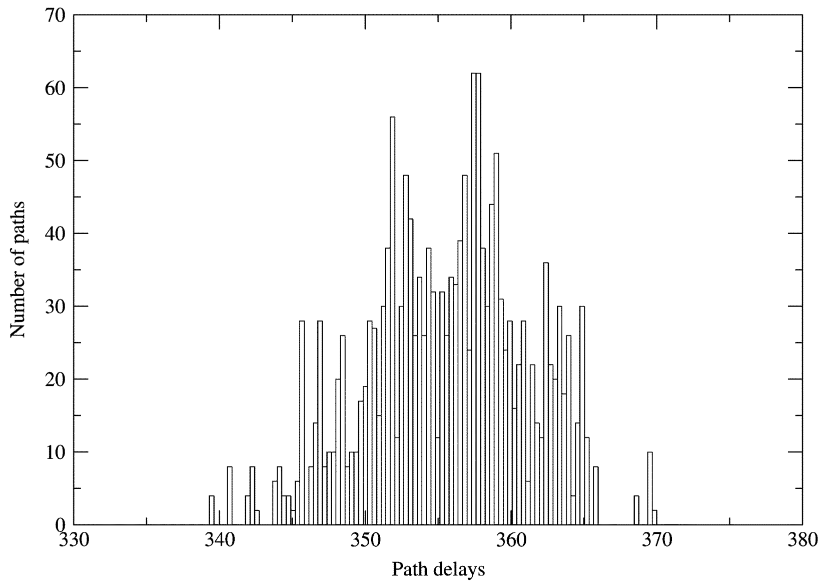

Fig. 14. Path profile's average delays (s15850)

X. Conclusion and Future Work

In this paper, we study the problem of critical path selection for delay fault testing based on a statistical timing model. We formulate the problem as an optimization problem that consists of two theoretically intractable subproblems. We first analyze various heuristics for solving each subproblem individually. Then, we analyze these heuristics for solving the original optimization problem.

We demonstrate that for path selection with a statistical timing model, it is important to consider path correlation. The concept of path correlation is not well-defined with a traditional discrete-valued timing model and hence, was not discussed before in traditional path-selection methods. The main contribution of this paper is the introduction of the path-correlation concept into path selection. We show that by considering path correlation, we can implicitly ensure a certain degree of topological coverage. The result is the H-Opt heuristic. Based on our analysis, we suggest that the H-Opt heuristic is better than the H-Timing heuristic and the H-Segment heuristic, both of which were derived following traditional thinking of the path-selection problem.

To support our theoretical analysis, we develop an experimental framework based on statistical timing and defect simulation. With this framework, we introduce the concept of using a U path set to facilitate our experiments. The introduction of the U path set suggests that our path selection methodology actually consists of two phases: *path filtering* and *path selection*. This paper focuses on the second phase.

Our formulation of the path-selection problem may inspire many interesting theoretical developments in the area of delay fault testing. The statistical timing simulation and evaluation framework provides a feasible approach to facilitate future research in the area. Future research can include the following three directions.

1) An important assumption in this work is that the statistical timing model in use can accurately model the reality. In practice, this is hardly true. It will be interesting to study the path-selection problem by assuming that the statistical timing model does not model certain aspects of the real silicon timing behavior. In this case, the value of the H-Opt heuristic need to be reevaluated.
2) More research effort should be devoted to study the construction of an effective U path set. The intention of constructing a U path set is to include all paths such that testing these paths can guarantee timing performance. Hence, a U path set can be thought as a *superset* of potential critical path sets. The effectiveness of a U path set can be measured by its size (that affects the runtimes of any programs based on it) and its quality. U path set will play an important role to study the path-selection problem, especially when we assume that the timing model can never be 100% correct. In this case, the superset U can provide a certain degree of tolerance for errors in the statistical timing model.
3) Like most of traditional path-selection research, this paper does not include pattern generation into the discussion. For delay testing, the goal of path selection is to produce high-quality test set. In this work, we show that H-Opt is better than H-Timing using our quality evaluation scheme that involves no pattern. In the future, it is important to reevaluate all the path-selection heuristics based on their resulting pattern sets.

References

[1] M. A. Breuer, C. Gleason, and S. Gupta, "New validation and test problems for high performance deep sub-micron VLSI circuits," in *Tutorial Notes, IEEE VLSI Test Symp.*, Apr. 1999.
[2] K. Baker, G. Gronthoud, M. Lousberg, I. Schanstra, and C. Hawkins, "Defect-based delay testing of resistive vias-contacts, a critical evaluation," in *Proc. Int. Test Conf.*, Sept. 1999, pp. 467–476.
[3] R. C. Aitken, "Nanometer technology effects on fault models for IC testing," *IEEE Computer*, vol. 32, pp. 46–51, Nov. 1999.
[4] K.-T. Cheng, S. Dey, M. Rodgers, and K. Roy, "Test challenges for deep sub-micron technologies," in *Proc. ACM/IEEE Design Automation Conf.*, June 2000, pp. 142–149.

[5] J.-J. Liou, A. Krstić, K.-T. Cheng, D. Mukherjee, and S. Kundu, "Performance sensitivity analysis using statistical methods and its applications to delay testing," in *Proc. Asian South Pacific Design Automation Conf.*, Jan. 2000, pp. 587–592.

[6] W.-N. Li, S. M. Reddy, and S. K. Sahni, "On path selection in combinational logic circuits," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 56–63, Jan. 1989.

[7] ——, "Long and short covering edges in combinational logic circuits," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1245–1253, Dec. 1990.

[8] R. B. Hitchcock Sr., "Timing verification and the timing analysis program," in *Proc. ACM/IEEE Design Automation Conf.*, 1982, pp. 235–248.

[9] S. Tani, M. Teramoto, T. Fukazawa, and K. Matsuhiro, "Efficient path selection for delay testing based on partial path evaluation," in *Proc. IEEE VLSI Test Symp.*, May 1998, pp. 188–193.

[10] C. P. Ravikumar, N. Agrawal, and P. Agrawal, "Hierarchical delay test generation," *J. Electron. Testing: Theory Applicat.*, vol. 10, pp. 188–193, June 1997.

[11] K. Antreich, A. Ganz, and P. Tafertshofer, "Statistical analysis of delay faults—Theory and efficient computation," *AEU Int. J. Electron. Commun.*, vol. 51, no. 3, pp. 117–130, 1997.

[12] T. W. Williams, B. Underwood, and M. R. Mercer, "The interdependence between delay-optimization of synthesized networks and testing," in *Proc. ACM/IEEE Design Automation Conf.*, June 1991, pp. 87–92.

[13] G. M. Luong and D. M. H. Walker, "Test generation for global delay faults," in *Proc. Int. Test Conf.*, 1996, pp. 433–442.

[14] C. H. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *J. Comput. Syst. Sci.*, vol. 43, pp. 425–440, 1991.

[15] E. Petrank, "The hardness of approximation: Gap location," *Computat. Complexity*, vol. 4, pp. 133–157, 1994.

[16] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng, "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation," in *Proc. ACM/IEEE Design Automation Conf.*, June 2002, pp. 566–569.

[17] A. Krstic, L.-C. Wang, K.-T. Cheng, and T. M. Mak, "Diagnosis-based post-silicon timing validation using statistical tools and methodologies," in *Proc. Int. Test Conf.*, 2003, pp. 339–348.

[18] J.-J. Liou, K.-T. Cheng, and D. Mukherjee, "Path selection for delay testing of deep sub-micron devices using statistical performance sensitivity analysis," in *Proc. IEEE VLSI Test Symp.*, Apr. 2000, pp. 97–104.

[19] A. Ageev and M. Sviridenko, "Approximation algorithms for maximum coverage and max cut with given sizes of parts," in *Proc. 7th Inc. Conf. Integer Program. Combinator. Optimization (IPCO), Lecture Notes Comput. Sci.*, G. Cornuejols, R. Burkard, and G. Woeginger, Eds., 1999, pp. 17–30.

[20] G. Cornuejols, M. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Manage. Sci.*, vol. 23, no. 8, pp. 789–810, 1977.

[21] Q. Han, Y. Ye, H. Zhang, and J. Zhang, "On approximation of max-vertex-cover," *Eur. J. Oper. Res.*, 2001.

[22] J.-J. Liou, L.-C. Wang, A. Krstic, and K.-T. Cheng, "Experience in critical path selection for deep sub-micron delay test and timing validation," in *Proc. ACM/IEEE ASP Design Automation Conf.*, Jan. 2003, pp. 751–756.

[23] A. Krstić and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*. Norwell, MA: Kluwer, 1998.

[24] *Eldo v4.4.x User's Manual*, 1996. Anacad.

[25] J.-J. Liou, L.-C. Wang, K.-T. Cheng, J. Dworak, R. Mercer, R. Kapur, and T. W. Williams, "Analysis of delay test effectiveness with a multiple-clock scheme," in *Proc. Int. Test Conf.*, Oct. 2002, pp. 407–416.

[26] M. Sivaraman and A. Strojwas, "Path delay fault diagnosis and coverage—A metric and an estimation technique," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 440–457, Mar. 2001.

[27] N. N. Tendolkar, "Analysis of timing failures due to random ac defects in VLSI moduels," in *Proc. Design Automation Conf.*, June 1985, pp. 709–714.

[28] J. P. de Gyvez, *Integrated Circuits Defect-Sensitivity: Theory and Computational Models*. Norwell, MA: Kluwer, 1993.

Li-C. Wang (M'96) received the B.S. degree in computer engineering from the National Chiao-Tung University, Taiwan, in 1986 and the M.S. degree in computer science and the Ph.D. degree in electrical and computer engineering, University of Texas, Austin, in 1991 and 1996, respectively.

He is a tenured Faculty Member in the Electrical and Computer Engineering Department, University of California, Santa Barbara. He was a Senior CAD Software Technical Staff Member at the Somerset PowerPC Design Center, Motorola, from 1996 to 2000. His research interests include microprocessor test and verification, defect-oriented testing, SAT solver development, and statistical timing analysis and validation.

Prof. Wang cofounded the IEEE Microprocessor Test and Verification (MTV) Workshop and is currently its program chair. He is a Steering Committee Member of the IEEE Test Synthesis Workshop and Test Economic Workshop. He serves as the DFVT Track Chair for ISQED'2003 and is a PC Member of the IEEE VTS'03/04, HLDVT'03, and IEEE ATS'04. He received best paper awards from DATE-1998, IEEE VTS-1999, and from DATE-2003 for his work on verification of PowerPC, on commercial experiments of novel ATPG method, and on delay defect diagnosis, respectively.

Jing-Jia Liou (M'03) received the B.S. and M.S. degrees in electrical engineering from National Tsing Hua University, Taiwan, in 1993 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 2002.

He is currently an Assistant Professor in the Electrical Engineering Department, National Tsing Hua University, Taiwan. His research interests include testing and diagnosis for delay defects of deep submicron designs, and statistical timing analysis and performance validation of system chips.

Prof. Liou is a member of ACM. He received Best Paper Awards at the IEEE Conference of Design, Automation, and Test in Europe (DATE 2003).

Kwang-Ting (Tim) Cheng (S'88–M'88–SM'98–F'00) received the B.S. degree in electrical engineering from National Taiwan University in 1983 and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1988.

He worked at 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 research interests include VLSI testing, design verification, and multimedia computing. He has published over 200 technical papers, coauthored three books, and holds nine U.S. patents in these areas. He has also been working closely in U.S. industry for projects in these areas.

Dr. Cheng received the Best Paper award at the 1994 Design Automation Conference and the 1999 Design Automation Conference, the 2001 Annual Best Paper Award from the *Journal of Information Science and Engineering*, the Best Paper Award at the 2003 Conference of Design Automation and Test in Europe (DATE 2003), and the Best Paper Award at 1987 AT&T Conference on Electronic Testing. He currently serves as Associate Editor-in-Chief for IEEE Design and Test of Computers. He had also served on the Editorial Boards of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and the *Journal of Electronic Testing: Theory and Applications*. He has been General Chair and Program Chair of IEEE International Test Synthesis Workshop and served on the technical program committees for several international conferences on CAD and testing including DAC, ICCAD, ITC, and VTS.