Feature based similarity search with application to speedpath analysis *

Nicholas Callegari[1], Li-C. Wang[1], Pouria Bastani[2]
[1]University of California - Santa Barbara
[2]Intel Corporation

Abstract

In test and diagnosis, one often runs into the situation that after analyzing a set of samples, a few of these samples are identified as being "special". Then, in a large population of samples one desires to identify all samples that are "similar" to the special samples. The process is called a similarity search. This paper presents a feature based similarity search approach and discusses three potential methods to implement this approach. These methods are (1) building a model to capture the characteristics of the non-special samples, (2) building a model to capture the characteristics of the special samples, and (3) searching for the hypotheses to explain individually why each sample is special. We apply similarity search to the speedpath analysis problem where special samples are special paths that limit the performance of silicon chips. The goal is to identify more paths in the design with similar characteristics to the speedpaths. The effectivenesses of the three methods are analyzed based on speedpath data collected from a high-performance microprocessor.

1 Introduction

Similarity search is the process of identifying from a large population the samples similar in characteristics to a few given special examples. In test and diagnosis, similarity search can be applied in different scenarios.

One example is in the identification of potential speedpaths. A *speedpath* is a path that limits the performance of a chip. In high-performance design, silicon information is often analyzed carefully to drive further speed and power improvements through multiple silicon steppings. In between subsequent steppings, tremendous efforts are spent on detecting and improving the speedpaths. Because identifying and verifying silicon speedpaths demand a huge amount of engineering effort, the number of speedpaths found at each silicon steppings is limited. Once this small set of speedpaths is collected, they should be utilized as much as possible for improving performance before the next silicon tapeout. One way to expand the utilization is to find potential speedpaths so that fixing these potential speedpaths can lead to additional performance improvements [3].

*This work is supported in part by National Science Foundation, Grant No. 0541192, Semiconductor Research Corporation, project task 1585.001 and CA Micro/Intel project No. 08-44

In another example, a few chips are found as special samples because they did not fail functional test before burn-in, but fail functional test after burn-in. To minimize the use of burn-in test, one desires to identify from a large population of chips the ones similar in characteristics to the special samples. These identified chips can then be screened out without using the burn-in process.

In the third example, silicon debug on a handful of failing chips may root-cause the failure to be the result of a few problematic layout patterns. To avoid similar problems in the future, one desires to search the entire design layout for any patterns that are similar to the problematic patterns.

In the first example, each sample corresponds to a path and a special sample corresponds to a speedpath. In the second example, each sample corresponds to a chip and a special sample corresponds to a problematic chip where its problem only gets manifested through burn-in. In the third example, each sample corresponds to a layout pattern and a special sample corresponds to a problematic layout pattern. In all three examples, similarity search is the process of taking the special samples and identifying other special examples from a large population.

To search for samples with similar characteristics, one approach is to use *features* [2] to describe these characteristics. Figure 1 illustrates the feature based similarity search approach. A set of features are used to encode each sample into a characteristic vector (of some numerical values). A similarity function $k(x,z)$ is defined to calculate a similarity measure on a pair of sample vectors x,z. Then, a model $M()$ is built based on the set of given samples. This model is applied on every sample s in a large population to calculate a similarity score $M(s)$ for s. Based on the scores and a threshold, samples with high similarity show up above the threshold and become the results of the search process.

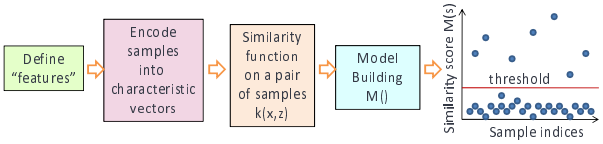


Figure 1. Feature based similarity search approach

Figure 2 illustrates the three potential methods to implement the feature based similarity search approach. Usually, a small set of m special samples are collected by separating them from a much larger set of K non-special samples. Hence, in a typical

978-1-4244-4867-8/09/$25.00

problem setting, not only the m special samples are available but also the K non-special samples can be utilized if needed.

With n features, each sample is encoded as a characteristic vector as shown in the figure. Then, the three potential methods can be the following: (1) building a model A for all the non-special samples, (2) building a model S for all the special samples, and (3) building a hypothesis h_i to explain each special sample s_i individually and use h_i to search for samples similar only to s_i.

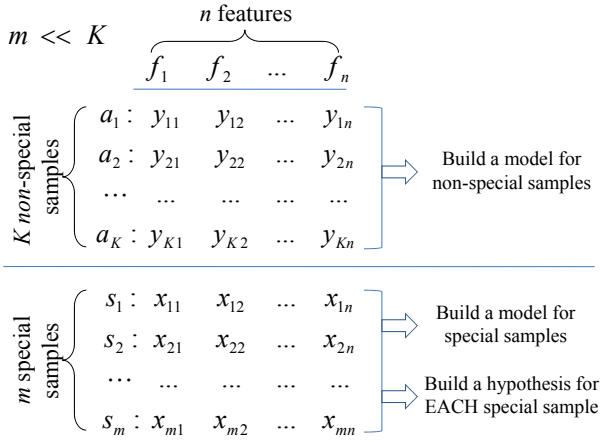


Figure 2. Three methods to implement similarity search

When applying the model A in search, it intends to identify samples in the population similar to the non-special samples. Hence, those not similar to the non-special samples become similar to the special samples. The method can be seen as a "modeling for the good" method. Therefore, building and applying model S can be seen as the "modeling for the bad" method where a model for "being special" is developed and used to scan the population to find special samples. While model S intends to find a single model for the special samples, building individual hypotheses accounts for the diversity where each special samples can be due to totally different reasons.

In this paper, we study the similarity search problem in the context of speedpath analysis. We compare the three methods discussed above based on speedpath data collected from silicon chips of a high-performance microprocessor design. We will reach three conclusive findings: (1) In general, it would be difficult to use the "modeling for the good" method for similarity search if the search is for bad examples. (2) The "modeling for the bad" method is much more efficient to run than the method of building hypotheses while results of the two method can overlap significantly. (3) The hypothesis building method, although takes more time to run, can find special samples that are unlikely to be found by other methods and also gives insight into why that sample is special.

The rest of the paper is organized as the following: Section 2 gives the background on speedpath analysis. Section 3 explains feature selection for speedpath analysis. Section 4 presents building a single model for speedpath or non-speedpath modeling and how the model can be used to find potential speedpaths. Section 5 introduces Hypothesis Pruning and Ranking and how it is utilized for a similarity search. Experiments are presented in section 6. Section 7 concludes.

2 Speedpath analysis

In high-performance chip design, timing analysis and optimization does not stop at first silicon. Silicon information is often analyzed carefully to drive further speed and power improvements. This process is called *silicon steppings*, that involves detection and improving of speed limiting paths (*speedpaths*). A speedpath limits the performance of a chip where the performance can be defined by observing the result of applying (functional) legacy tests. Because speedpaths can be observed at different cycles of a (functional) test sequence where different parts of the chip are exercised, a single chip can have multiple speedpaths [1].

Due to the high-performance nature of the design, speedpaths are usually not well-predicted by the timing flows. The deficiency of these CAD flows are due to a multitude of process, design and environmental effects that are either unknown or too difficult to model and simulate accurately for millions of paths. Therefore, for high-performance high-volume microprocessor designs, it is more (cost) effective to uncover speedpaths using actual silicon samples. Performance can be further improved in each silicon stepping by pushing the delays on those paths. Simultaneously, if causes can be established to explain why some paths become speedpaths, this information can be fed back to the timing flows and other potential speedpaths can be identified and fixed before they are detected in the subsequent silicon steppings. These iterations continue until the performance is pushed to a satisfactory level, at which time design changes are frozen and mass production begins.

Each silicon stepping has significant associated manufacturing cost. Moreover, identifying and verifying silicon speedpaths demand a huge amount of engineering effort. Because of this, the number of speedpaths found at each silicon steppings is small, usually in the 10's to 100's. Hence, once this small set of speedpaths are collected, they need to be utilized as much as possible to improve performance before the next silicon tapeout. This may be done by enhancing the CAD for timing prediction flows with the information presented by the speedpaths. However, it is not obvious what the most effective approach is to best uncover and utilize the information on a few identified speedpaths.

Once a set of speedpaths are found, there are two fundamental ways the information on the speedpaths can be utilized to find additional speedpaths. In root-cause analysis, one tries to get to the root-cause(s) of speedpath mis-prediction. A root-cause may lead to improvements in delay modeling or in timing methodology or both. Based on the improved model and/or the improved methodology, more accurate design optimization can be carried out before the next stepping. This includes finding additional paths that are potential speedpaths.

While root-cause analysis is intuitive, it may not be the most desirable because of the usual high cost associated. When time-to-market is critical, one desires a more efficient approach that

can help identify potential speedpaths without knowing the reasons behind the identified speedpaths. The basic idea is that, if the goal is simply to isolate potential speedpaths one need not improve the modeling and methodology first. Instead, a direct approach can be employed, which can take place in parallel to the root-cause analysis, that predicts other potential speedpaths based only on comparing the *characteristics* of the speedpaths to the characteristics on all other paths.

In [3], the authors present an implementation of the direct approach using one-class *support vector machine* (SVM) [6] to learn from the characteristics of speedpaths. The result is a one-class model that characterizes the speedpaths. This model can then be used to scan all other paths and rank them according to their similarities to the speedpaths. From this ranking and a proper selection criterion, potential speedpaths can be derived.

The direct approach is attractive because it avoids the difficulty of finding the root-causes. However, it may not be as effective as the root-causing approach for finding all potential speedpaths. In other words, the direct inference may fail to identify a potential speedpath that can be identified if one knows the root-cause. For example, two paths (a speed path and a potential speedpath) may share the same root-cause and hence, they are similar if one compares the two paths based on only the characteristics associated with the cause. However, these two paths may be deemed dissimilar if one compares the overall characteristics of the paths. From this perspective, we see that the direct approach, although much more efficient, may only find a subset of the potential speedpaths or identify non-speedpaths as potential speedpaths.

The limitation of the direct inference approach and the high cost of the root-cause analysis approach, motivate the development of a third approach, *hypothesis pruning and ranking* (HPR) [5]. In the HPR approach, we assume two sets of paths are given, a small set of speedpaths and a large set of non-speedpaths, where non-speedpaths can be paths that were sensitized but were not speed limiting. While a speedpath set may contain tens of paths, a non-speedpath set may contain millions.

The strategy of the HPR approach is to analyze each speedpath individually against the information presented in the non-speedpath set. Conceptually, this is achieved by two steps: (1) forming a hypothesis space that contains all the hypotheses of interest, (2) pruning the unlikely hypotheses based on the non-speedpaths and if possible further ranking the remaining hypotheses. The result of HPR approach is therefore a rank of hypotheses and this rank may be based on a partial ordering. Top hypotheses (that describe certain characteristics of the speedpaths) can then be used for finding potential speedpaths.

In [5] the work focuses on the development and comparison of implementation techniques. Validation of the approach is determined by root-cause evaluation of an unknown population.

The primary goal of this work is to compare four approaches in terms of their effectiveness in feature based similarity search on a known dataset. The first approach is "modeling for the good" or non-speedpath modeling. The second and third approach is "modeling for the bad" or speedpath modeling, where we utilize the approach presented in [3], and present an improved approach, individual speedpath modeling. The fourth approach is the newly developed HPR. We applied the approaches on a speedpath dataset recently collected from a high-performance microprocessor.

3 Feature selection

Feature based similarity search begins with a definition on the set of features to encode the search space (see Figure 1). A feature can be seen as an aspect to describe some characteristic of a path. Usually, a feature corresponds to some aspect of a concern that a path may limit timing.

There can be two types of features, *occurrence based* and *description based*. An occurrence based feature has a binary value. For example, a cell by itself can be a feature. Then, given a path, if the path contains the cell, the value of that feature is 1. Otherwise, the value is 0. A description based feature has a numerical value. For example, the capacitive load associated with a specific cell on a path can be a description based feature.

Usually, a speedpath is due to effects that are not accounted for in the timing flow. This can be because their rarity of occurrence along with the large additional cost to model them. Selecting a proper set of features to begin undoubtedly requires some knowledge about these unaccounted effects. Although this is an important step for the proposed approach to be effective, it should not be seen as a severe limitation for the following reasons: First, the approach does not require to select a minimal set and hence, if one is not sure about some features, the solution is simply to include them. Second, a feature is a single-order effect that can be part of the concern. Usually, it is not too difficult for an experienced designer (or a process engineer) to point out such single-order effects. For example, a design can list a set of cells that she/he feel less confident on their timing modeling. Third, a missing feature does not necessarily degrade the effectiveness of the approach significantly. For example, suppose the root-cause is a combination of three features $\{A,B,C\}$ and suppose the feature set only includes A and B. This does not imply that potential speedpaths cannot be identified based on only $\{A,B\}$. From the viewpoint of fixing the potential speedpaths, fixing all paths with $\{A,B\}$ would have included the paths with $\{A,B,C\}$. This may lead to over-fixing, but the effectiveness with respect to the performance improvement does not suffer.

Authors of [2, 3] group features into five categories: **Topological effects** that are layout dependent, **Dynamic effects** that are test pattern dependent, **Static effects** that are independent of test patterns, **Statistical effects** that are process dependent, and **Random effects** that do not belong to the previous four.

In practice, an engineer develops a set of features that can potentially be used to differentiate non-speedpaths and speedpaths. Keep in mind that feature definitions are entirely up to the user and can depend on the application. Since features can be closely related to the concerns in modeling, design methodology, and/or process, they are treated as highly-sensitive pro-

prietary information.

In this work the following characteristics of paths were encoded into 21 features for our high performance design;

1. Layout factors such as path location on a die, lithographic characteristics, device, interconnect and process related.
2. Timing factors such as path, stage (cell in combination with interconnect), and clock timing.
3. Electrical factors such as voltage, capacitance, frequency features for a cell, interconnect and devices.

Note that most of the features are description based. These features encodes every path into a vector of feature values.

4 Speedpath and non-speedpath modeling

We examine two approaches for building a model with the goal of identify potential speedpaths. The first is to build a model for the non-speedpaths, and the second is to build a model for the speedpaths. In both approaches, we use the one-class Support Vector Machine (SVM) [6] to develop a model for the class of paths.

Suppose we obtain a model M_n for the non-speedpaths. A one-class model means that M_n tries to capture the characteristics of the non-speedpaths. When using M_n in the similarity search on a large population of paths, all paths that are deemed *not* similar to the non-speedpaths are classified as potential speedpaths. This is different from using a model M_s that tries to capture the characteristics of the speedpaths. In this case, all paths that are deemed similar to the speedpaths are classified as potential speedpaths. This difference is illustrated in Figure 3.

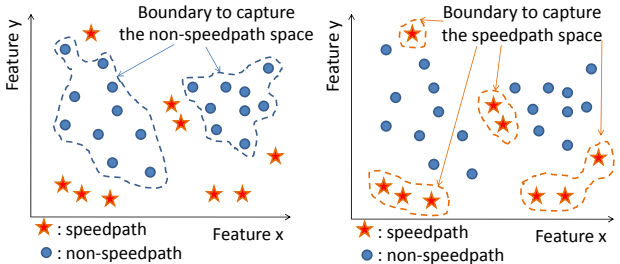


Figure 3. Two perspectives to apply one-class SVM

Figure 3 shows that modeling non-speedpaths and modeling speedpaths are two different things. If the model is for the non-speedpaths, then any paths falling outside the boundary are potential speedpaths. In contrast, if the model is for the speedpaths, then only paths falling inside the speedpath boundary are considered as potential speedpaths. Intuitively, since the objective of the similarity search is for finding paths similar to the speedpaths, modeling the speedpaths should be the preferred approach. This observation will be validated with experimental results in Section 6.1.1.

If the goal is to separate speedpaths from non-speedpaths, one may wonder why we did not consider using a binary classifier [6] as that in [9]. There are two fundamental reasons why a binary classifier is not suitable for similarity search.

The first reason is due to the extremely unbalanced dataset. The number of speedpath examples is usually much smaller than the number of non-speedpath examples. Hence, if we treat the two classes together in a single dataset, the dataset would be extremely biased toward the non-speedpath examples. As a result, the binary classifier would be biased to favor classifying a path as non-speedpaths because such a classifier is optimized to minimize the overall prediction error. For example, suppose there are 5 speedpaths and 1M non-speedpaths. When training a binary classifier on such an extremely unbalanced dataset, the classifier could simply classify all paths as non-speedpaths to achieve an misclassification error of $0.005\% = \frac{5}{1M}$.

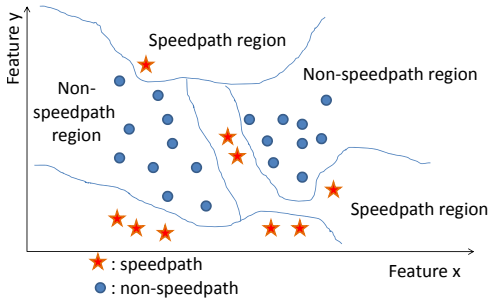


Figure 4. A binary classifier partitions the entire space into speedpath vs. non-speedpath regions

Even with a not so unbalanced dataset, a binary classifier is still not preferred. With a binary classifier, on a given path it has to decide between the path being more similar to speedpaths and being more similar to non-speedpaths (In contrast, similarity search is to determine if the path is similar to speedpaths or not, i.e. "not similar to speedpaths" does not imply it is similar to non-speedpaths). Therefore, a classifier partitions the entire input space into speedpath region and non-speedpath region as shown in Figure 4. This means that for the sub-spaces lacking coverage by the path examples, the decision for the potential speedpaths cannot be guaranteed to be conservative. Hence, binary classification is not a preferred way for similarity search.

4.1 One-class support vector machine

Suppose that a path P can be described as a vector $\vec{v} = (f_1, \ldots, f_n)$ of n numerical values, where each f_i is a *feature* variable. $\vec{v}$ describes the characteristics of the path. Given r paths (either speedpaths or non-speedpaths), we assume that we have r vectors $V = \{\vec{v}_1, \ldots, \vec{v}_r\}$ describing the characteristics of these paths. Our goal is to develop a method that can learn from V and produce a model M that captures the characteristics of the r paths. In this work, we use the one-class SVM algorithm [6] to build the model M.

A one-class SVM model always takes the form:

$$M(\vec{v}) = \sum_{i=1}^{r} \alpha_i k(\vec{v}_i, \vec{v}) \quad (1)$$

where $k(\cdot,\cdot)$ is a kernel function. $k(\vec{x},\vec{z})$ measures the similarity between the two vectors $\vec{x},\vec{z}$. For example, a commonly used kernel function is an exponential function $k(\vec{x},\vec{z}) = e^{-\gamma||\vec{x}-\vec{z}||^2}$ where $||\vec{x}-\vec{z}||^2$ is the square distance between the two

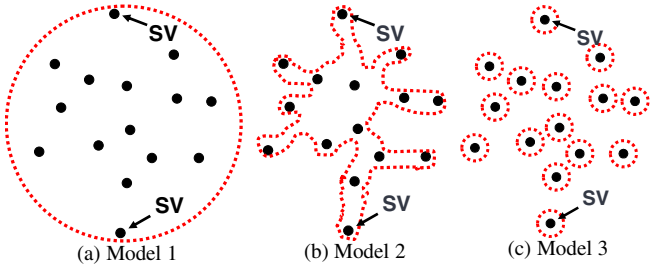
vectors and γ is a constant defining the "width" of the exponential distribution. We see that a larger distance means lower value in similarity. Also observe that a larger γ means that the kernel function is more conservative in considering the two vector to be similar.

In the model, some α_i's are zero. Only path vectors with $\alpha_i \neq 0$, are used to build the model. These vectors are called *support vectors*. Others are called non-support vectors. For a path $\vec{v}_i$, a larger α_i indicates that the path is more important for determining the value of M. We see that the model M computes a weighted average of the similarity between the path vector $\vec{v}$ and all the support vectors.

If we take a threshold value t and consider all vectors $\vec{v}$ such that $M(\vec{v}) \geq t$, essentially this defines a subspace (possibly discontinuous) in the n dimensional space. We can then call this subspace the region of potential speedpaths because any path whose description vector falls inside the region, is at least t-similar to the speedpaths.

For a support vector model, its *model complexity* can be measured by the number of support vectors [6]. The larger this number is, the more complex the model becomes. A more complex model is also more conservative in classifying a path as a potential speedpath. A less complex model is more aggressive. Note that by using the exponential kernel $k(\vec{x},\vec{z}) = e^{-\gamma||\vec{x}-\vec{z}||^2}$, the complexity of the model can be influenced by varying γ. A larger γ tends to produce a more complex model.

To illustrate the concept of model complexity, Figure 5 shows three possible models for a simplified path vector set on a two dimensional space.



(a) Model 1 (b) Model 2 (c) Model 3

Figure 5. Illustration of model complexity

Figure 5(a), shows the simplest model that is a circle region defined based on only two support vectors. The remaining dots are non-support vectors. Suppose all these vectors represent speedpaths. If we consider all other points inside the circle as being similar to the speedpaths, we observe that the area defined by this model is the largest. Figure 5(b), shows a more complex model, where the number of support vectors defining the region is increased. This model is more specific than the previous because its area is smaller, i.e. there exist paths determined as potential speedpaths by the first model but not by the second. Figure 5(c), shows the most complex model, where every training sample becomes a support vector. This model spans the minimal space needed to cover all the speedpaths.

Note that a more general model reduces the chance of false negative, i.e. the chance of rejecting a path that is actually a potential speedpath. A more specific model, on the other hand, reduces the chance of false positive, i.e. the chance of accepting a path that is actually not a speedpath. Intuitively, one would think that we want a model with a reduced chance of false positive. This is because we do not want to miss any potential speedpaths.

This intuitive strategy makes sense only if the accuracy of the model is also high, meaning that it does not predict too many good paths as potential speedpaths. For example, if we identify 1000 potential speedpaths and only 10 of them are real, then the remaining 990 paths lead to a waste of engineering effort on improving them. Moreover, speeding up these paths may result in increased power consumption with little gain on the actual frequency.

As discussed before, speedpaths usually are due to special things not explicitly considered by models and simulation. As a result, we would expect their characteristic descriptions $\vec{v}_1,\ldots,\vec{v}_r$ to be quite unique, meaning that they may spread widely in the n dimensional space. If we try to build a general model shown in Figure 5(a), this model may potentially cover a large region in the space. Consequently, many paths would be considered as potential speedpaths. On the other hand, if we try to build a very complex model shown in Figure 5(c), this model covers a very small space. Thus, only paths that are identical to a observed speedpath are going to be considered as potential speedpaths. We would like to find a optimal point in between these two extreme cases.

To summarize, because the number of speedpaths used to build the model is small and because the characteristics of these paths are diverse, it is unrealistic to expect we can learn a general model that captures only potential speedpaths without including other good paths. Therefore, in our algorithm we do not try to build a general model. Instead, our strategy is to start by building the most complex model, followed by a sequence of models with gradually reduced model complexity. Then, we develop a method to decide when to stop and use the model at the stopping point as our optimal model.

Suppose that M_i and M_{i+1} are built in two consecutive steps where M_i is more complex than M_{i+1}. Suppose we use these two models to search the design and identify the top N paths that have the highest potential to be speedpaths. Let these two sets of paths be S_i and S_{i+1}, respectively. We can check to see if $S_i \approx S_{i+1}$. If the two sets almost agree with each other, then we have a higher confidence that M_i and M_{i+1} are good models to use.

Although in this paper we do not provide a theoretical reason why $S_i \approx S_{i+1}$ can be a good heuristic to use, we do experimentally validate that it does lead to more meaningful results.

5 Hypothesis Pruning and Ranking

Hypothesis Pruning and Ranking utilizes the defined hypothesis space in order to identify which hypotheses are responsible for why a path is a speedpath. A hypothesis is defined a subset of features which exists in a speedpath. Top hypotheses are those which best separates a speedpath from all non-

speedpaths, the characteristics which makes it unique. HPR operates on the assumption that the most likely cause responsible is the hypothesis which is least similar to non-speedpaths. We believe hypotheses that are the most similar to non-speedpaths are more than likely not be the cause, because there is firm evidence that its presence does not cause a speedpath.

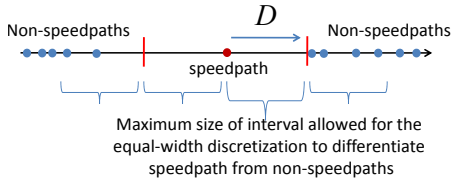


Figure 6. First-order maximum margin method

HPR determines for each hypothesis the greatest distance to the nearest non-speedpath, which we call the maximum margin, for all hypotheses. Figure 6 shows a simple illustration where given a single feature f_1, the similarity is determined by identifying the distance to the nearest non-speedpath. This example illustrates what the maximum margin separation means in a one-dimensional Euclidean space. This distance, defined by D is the maximum margin to the closest speedpath. Figure 7 illustrates this same distance calculation for a second-order subset of features, i.e. f_1, f_2. The maximum margin is determined for all possible combinations of features, hypotheses, present in a speedpath and are then ranked by greatest distance to identify the top hypotheses.

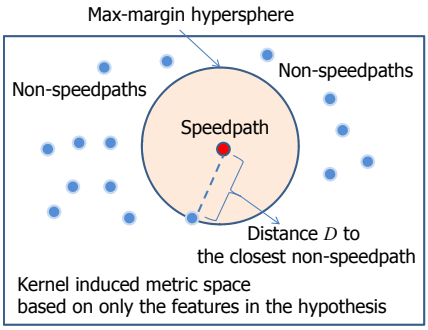


Figure 7. Find the hypothesis with maximum margin

5.1 HPR pruning

Due to the difficulty of efficiently computing the distance for all hypotheses in a large dataset, pruning is a essential. Pruning is performed in two steps; first by filtering zero-impact features, second by pruning the first order space.

Zero-impact filtering removes features which cannot reasonably be the cause of a speedpath. Domain knowledge from design engineers can be used to augment this filtering process. However, due to the complexity of the problem it is often the case that causes are not intuitive. Because of this, we need to reduce the risk of incorrectly removing a feature that may result as a cause. Thus, we only want to remove features that we can be certain have no impact on the analysis. These "zero-impact" features, refer to features with the numerical value of '0'. Examples of this can be if the feature is not present in the current path (occurrence-based) or there is no capacitance associated with a gate that is not present (descriptive-based). On the other hand, it is important to not to filter out, for example, a 0 description as it relates to x,y coordinates or even 0 capacitance associated with a gate that is present but may be mis-modeled.

The second pruning process, which we refer to as "pruning the first order space", is used to reduce the set of non-speedpaths that are used to calculate the potential hypotheses. Given a speedpath SP_1 and a non-speedpath NSP_1 and their associated m features, $\{f_1, \ldots, f_m\}$, where m is the number of features remaining after zero-impact filtering. We define the distance between all single order features between the two paths as $D(SP_1, NSP_1) = \{d_1, \ldots, d_m\}$.

We can reduce the set of non-speedpaths for which to calculate the distance for all 2^m hypotheses, by eliminating all non-speedpaths NSP_x where the distance, d, between every feature for the non-speedpath and the speedpaths is greater than a previously analyzed non-speedpath. In other words, if the distance of all single order features from one non-speedpath are less than that of another, $D(SP_1, NSP_1) < D(SP_1, NSP_x)$, we can eliminate calculating the distance of all 2^m hypotheses of the second non-speedpath, because the distance of all combinations of the single order features will be greater for every hypothesis.

In our heuristic we select a non-speedpath to reduce the set of non-speedpaths by first calculating the distance D between all single order features for all non-speedpaths. We take the sum $D_{sum} = d_1 + \ldots + d_m$ to identify the closest average non-speedpath. That path is used to reduce the set, and the process is continued until all non-speedpaths are either selected or eliminated. The set of selected paths is then used to calculate the distance for all hypotheses. In our experience this process can reduce a set of 1.1M non-speedpaths to an average of 50,000.

5.2 HPR similarity search

After the hypotheses are pruned and ranked, the top N ranked hypotheses are utilized to search for potential speedpaths. If an unknown path exists where its distance for any of the top N are within the maximum margin hypersphere shown in figure 7 that path is classified as a potential speedpath.

6 Experiment

To evaluate the different approaches we conduct experiments on speedpath and non-speedpath data from a high performance microprocessor design. For this design we analyze 82 speedpaths identified during silicon debug, and 1,107,862 (1.1M) non-speedpaths. The set of speedpaths and non-speedpaths were both identified by a timing analysis tool to have minimal timing slack. Each of these paths are encoded with 21 description based features described in section 3.

In machine learning, it is common to have a training set, which is used to build a model, and a validation set, which is used to verify model accuracy. In this work because we do not have silicon data from sequential silicon steppings, we use a subset of the 82 speedpaths to create a model (training set), and combine the remaining speedpaths with the set of non-speedpaths to represent the population (validation set) for the

model to search. The goal is to is to identify speedpaths in the validation set, using the model constructed by the training set. Note that none of the original speedpaths used in the training set will be contained in the validation set. The effectiveness is determined by the number of speedpaths identified in the top K ranked paths from the validation set.

In this experiment section, two key assumptions are necessary. (1)The cause of each speedpath cannot be unique. It is not necessary for all 82 speedpaths to be correlated, however a speedpath in the validation set has to have some similarity to one in the training set in order for similarity search to be applicable. (2)The feature set must contain a feature or combination of features that are related to this cause of the speedpaths. These two assumptions are required for any methodology based on a similarity search. Without them the results will be no better than a random selection algorithm. Conversely, if results yield an improvement, we validate that some speedpaths contain similar causes, and the 21 features provided by the design/debug engineers contain these causes.

We begin by conducting an experiment to compare the effectiveness of the modeling approaches for the four methods. Specifically, (1)non-speedpath modeling, (2)speedpath modeling, (3)individual speedpath modeling, and (4)Hypothesis Pruning and Ranking. From the initial experiment we identify the two most effective approaches to perform cross-validation of all 82 speedpaths. We compare the results and analyze a phenomenom we identify as the *rogue path* which creates a clear distinction between the two methods.

6.1 Overall comparison

To conduct an overall comparison of all four approaches, we begin by splitting the 82 speedpaths in half, 41 used for the training set, and the other 41 combined with the 1.1M non-speedpaths used for population (validation set). Outlined in detail in each subsection, we modify this dataset slightly as each approach differs, however the overall goal is to identify the 41 speedpaths from the non-speedpaths, in the population. Each approach presents results as the number speedpaths identified in the top K ranked path selected from the validation set. We assume that it is reasonable for a designer to consider fixing up to a few hundred paths. Therefore in our comparison we focus on the critical region up to 1,000 paths.

6.1.1 Non-speedpath modeling

Non-speedpath modeling begins by building a model based on the non-speedpath set (training set) using one-class SVM with a Gaussian kernel. With this model we analyze the remaining non-speedpaths in addition with 41 speedpaths (validation set) to see which paths are included in the bounds (inliers) of the model and which paths fall outside the bounds (outliers). The objective of this experiment is to build a model on the non-speedpaths which has all the speedpaths fall outside of the bounds, while all non-speedpaths fall within the bounds. Additionally one-class SVM not only provides the boundary, but also the distance from the model, provided by the kernel, for each path in the validation set. This distance is used to rank the paths. In this case, the paths with the furthest distance are dissimilar to non-speedpaths, and therefore should be speedpaths.

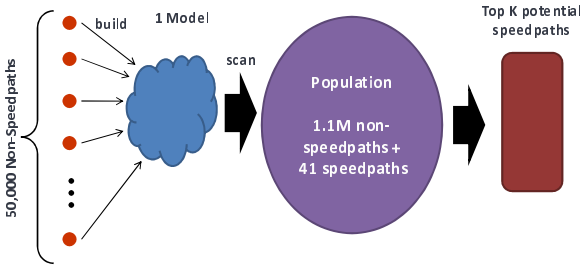


Figure 8. Non-speedpath modeling flow

Outlined in figure 8, we use a set of 50,000 non-speedpaths (training set) to build the model. Although it would be ideal to use half the non-speedpaths for the model, there is exponential growth in runtime with respect to the number of samples used to build the model. Although design dependent, given our dataset this translates into runtime in a matter of hours vs. weeks. We can see that due to the runtime issues this approach has strong limitations. The model is used to searched the population consisting of 1.1M non-speedpaths in addition to the 41 speedpaths. The result is shown in figure 9.

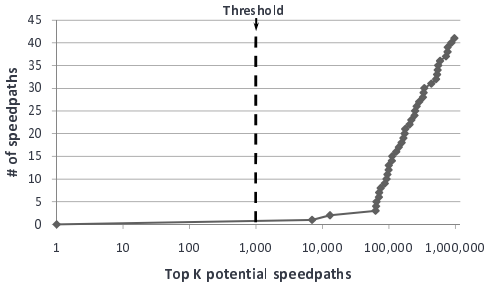


Figure 9. Non-speedpath modeling similarity search results

Illustrated in figure 9, the x-axis is the top K ranked potential speedpaths from this approach. Correspondingly the y-axis is the number of speedpaths identified in that top K. We can see at the threshold $K = 1,000$ potential speedpaths, 0 out of the 41 speedpaths were identified. In analyzing these results, two conclusions can be drawn. (1) To identify a reasonable amount of speedpaths as outliers, a significant amount of non-speedpaths are mis-classified. (2) This approach is significantly limited by the number of non-speedpaths for which to build the model.

6.1.2 Speedpath modeling

Speedpath modeling addresses the problem from the opposite perspective, where we use one-class SVM with a Gaussian kernel to build a model based on the set of speedpaths. Illustrated in figure 10, we build a model on the 41 speedpaths in the training set. The other 41 speedpaths in combination with the 1.1M represent the population. This approach also uses the distance value from the model in order to rank the paths based on similarity to speedpaths. The result is shown in figure 11.

Illustrated in figure 11, the x-axis is the top K ranked potential speedpaths from this approach. The y-axis is the number of speedpaths identified in that top K. We can see a def-

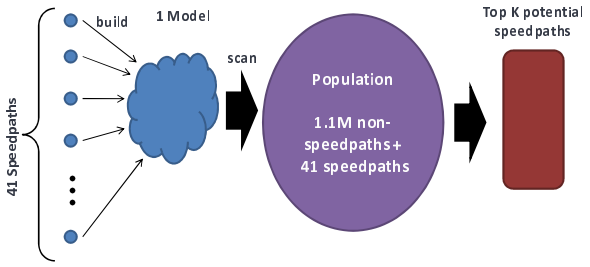


Figure 10. Speedpath modeling flow

inite improvement over non-speedpath modeling in the $K = 10,000 - 100,000$ region. However, at the threshold $K = 1,000$ potential speedpaths, 0 out of the 41 speedpaths were identified.

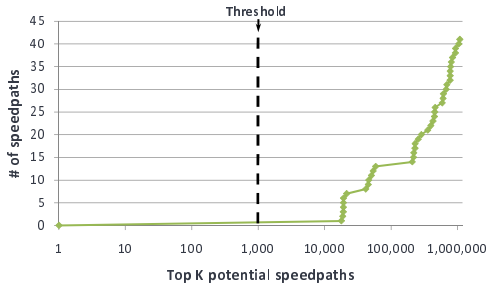


Figure 11. Speedpath modeling similarity search results

6.1.3 Individual speedpath modeling

When speedpath modeling builds a model based on a set of speedpaths, a path in the population is ranked based on the similarity to the model, and not each speedpath individually. Therefore when similarity for a path is determined, a path may be ranked higher if it is somewhat similar to multiple speedpaths in the model, than a path which is nearly identical to one speedpath but dissimilar to all others in the model. We know that each speedpath can have a unique cause as to why it is a speedpath, therefore we modify the previous approach and instead build a model for each speedpath individually as illustrated in figure 12.

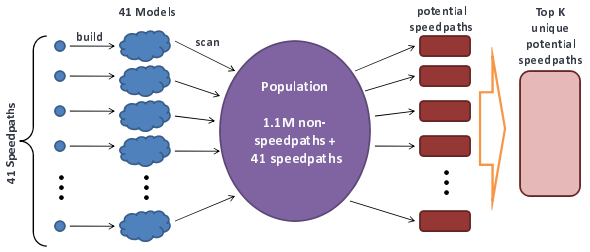


Figure 12. Individual speedpath modeling flow

We begin by building a model on each of the 41 speedpaths individually (training set), resulting in 41 models. The remaining 41 speedpaths in combination with the 1.1M non-speedpaths represent the population. For each of the 41 models, the population is searched and we obtain the top ranked potential speedpaths based on similarity to the model. We concatenate all 41 top ranked paths removing duplicates, resulting the top K unique potential speedpaths.

One shortcoming with building a model based on multiple speedpaths is the difficulty to identify the "optimal" model complexity, i.e. the tightness of the bound described in section 4.1. However, this problem is alleviated when building a model based on individual speedpaths because the model complexity is irrelevant when comparing similarity to a single speedpath.

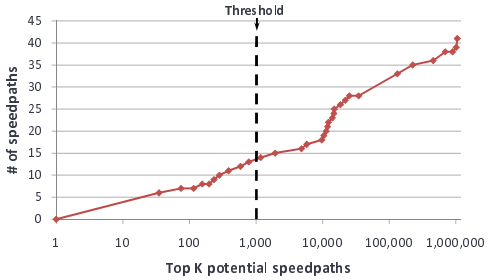


Figure 13. Individual speedpath similarity search results

The result is shown in figure 13, the x-axis is the top K ranked potential speedpaths from this approach. The y-axis is the number of speedpaths identified in that top K. The results show a significant improvement for this approach over the previous two. Specifically at the threshold $K = 1000$ potential speedpaths, 13 out of the 41 speedpaths are correctly classified.

6.1.4 Hypothesis Pruning and Ranking

Hypothesis Pruning and Ranking begins by identifying for each speedpath the closest non-speedpath for each hypothesis, shown in figure 7. The hypotheses are sorted by maximum distance, which translates to the features which differentiate why a speedpath is a speedpath.

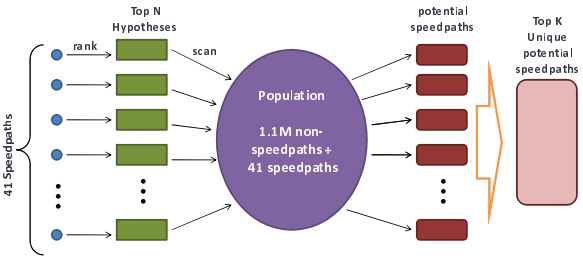


Figure 14. HPR flow

Similar to individual speedpath modeling, we apply this approach on each of the 41 speedpaths individually (training set) illustrated in figure 14. The remaining 41 speedpaths combined with the 1.1M non-speedpaths represent the population. We search the population and obtain the closest X paths for the top N ranked hypotheses. Ideally the hypotheses are examined by designers to identify the most reasonable hypothesis prior to performing the similarity search. Because the top hypothesis for each speedpath was not validated, we selected the top $N = 3$ hypotheses, as to not exclude any hypothesis that may be important. In varying X we obtain the result of the top K unique potential speedpaths.

The result is shown in figure 11, the x-axis is the top K ranked potential speedpaths from this approach. Correspondingly the y-axis is the number of speedpaths identified in that top K. We can see that the results are similar to individual speedpathing modeling. At the threshold $K = 1000$ potential speedpaths, 10 out of the 41 speedpaths are correctly classified.

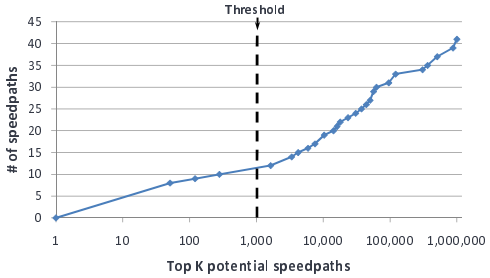


Figure 15. HPR similarity search results

6.1.5 Comparison results

To fully compare the four approaches, we must examine what is considered reasonable for a designer to fix in a silicon stepping. Illustrated in figure 16 is the result of all four approaches. We increased the resolution in the lower figure and focused on the range of between $K = 0 - 500$ potential speedpaths. Two thresholds are outlined Threshold A at $K = 200$ and Threshold B at $K = 400$. It is our belief that it would be reasonable for a designer to fix a number of paths at either of these thresholds.

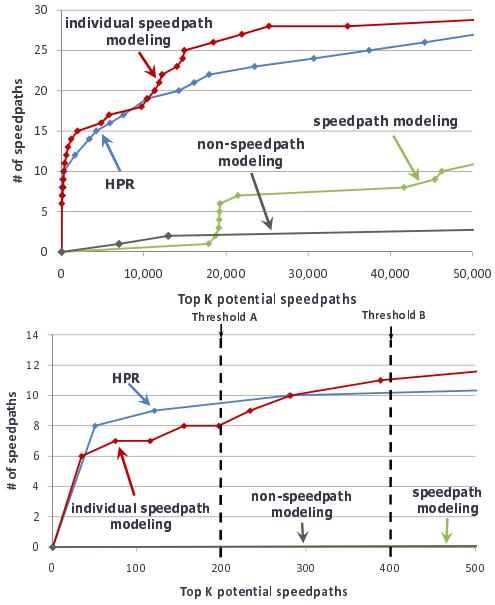


Figure 16. Comparison of similarity search results

At Threshold A, HPR is the preferred approach, identifying 9 speedpaths in the top $K = 200$. Whereas at Threshold B, individual speedpath modeling is the preferred, identifying 11 speedpaths in the top $K = 400$. Both methods show a superiority over the speedpath and non-speedpath modeling, as they do not identify a single speedpath in this range. Note that finding 9 out of 43 paths or 21% in the top 100 potential speedpaths from a total population 1.1M or 0.009% can be regarded as a very successful similarity search. Especially when considering that all paths in the population were identified critical by timing analysis. We inspected the remaining non-speedpaths for both approaches, and found a high similarity to speedpaths, and could potentially be speedpaths in future silicon steppings.

6.2 Individual speedpath model/HPR X-validation

In order to conduct a thorough comparison between the two superior approaches, (1)Individual speedpath modeling, and (2)Hypothesis Pruning and Ranking, we conduct cross-validation, fully utilizing our complete set of speedpaths. This is accomplished by building a model on each of the 82 speedpaths individually (training set), with the remaining 81 speedpaths combined with the 1.1M non-speedpaths combined to represent the population. Once the model is built on one speedpath, the population is searched and we obtain, for both approaches, the top ranked potential speedpaths. We iterate this approach to each of the 82 speedpaths and remove duplicates between the rankings, resulting in the top K unique potential speedpaths.

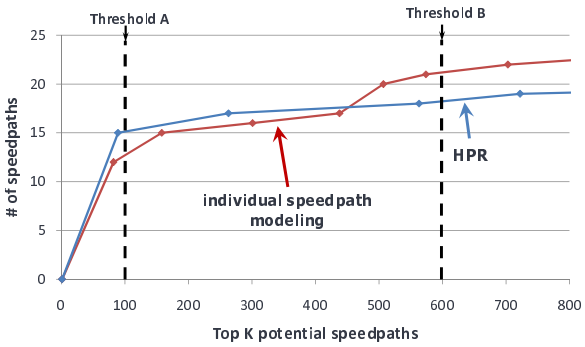


Figure 17. X-validation comparison results

When comparing the results of individual speedpath modeling and HPR, there are two findings: (1) Assume that the maximum number of potential speedpaths a designer is willing to fix is 100, HPR would have identified 15 speedpaths to fix, where individual speedpath modeling would only have identified 12. Thus, for the same effort, HPR can find more speedpaths. (2) Given unlimited effort, individual speedpath modeling can find more speedpaths. However this finding only reflects the results of our worst case HPR analysis, where no validation is conducted to select the most likely hypotheses for which to search.

6.3 X-Validation analysis and Rogue paths

In analyzing both individual speedpath modeling and HPR, we examined the set of speedpaths that one method identified in the top K potential speedpaths, but not the other. In all cases where individual speedpath modeling was able to identify a speedpath that HPR did not, it was determined to be due to lack of designers insight to validate the top hypotheses. Assuming the top hypothesis that correlated to a speedpath in population was identified by designers, we verified that (1) the hypothesis existed in the top 3 ranked hypotheses and (2) given the elimination of the other 2 hypotheses, HPR would be superior in identifying all 82 speedpaths.

Additionally we examined a reoccurring phenomenon where a HPR identified a speedpath, that individual speedpath modeling did not. Specifically a speedpath was identified by HPR in the the top $K = 100$, which was ranked in the top $K = 634,228$ by individual speedpath modeling. We refer to this as a *rogue*

path. We determined that while HPR found it to be similar to a specific hypothesis, its overall similarity with respect to all features was dissimilar.

To illustrate this event we show in table 1 two speedpaths, SP1 and SP2 and their corresponding relevant features. The values have been normalized to mask all proprietary information. SP1 was the speedpath used to build the model. SP2 was the speedpath in the population. In HPR, it was the top ranked hypothesis from SP1 that identified SP2. The hypothesis, was a combination of 4 features. We calculated the distance for the hypothesis, resulting $D(f_1, f_6, f_14, f_{17}) = 0.107$. For one-class SVM employed in individual speedpath modeling, we know that the similarity of two paths takes into account the distance of all features. The similarity measure equal to the distance of all features is $D(f_0, ...f_{20}) = 0.172$. The similarity with respect to all features is greater than that of the specific hypothesis. We therefore call this a rogue path. In practice the difference of these two methods can equate to not selecting a path as a potential speedpath because as a whole the path contains dissimilar features, however the path may contain a certain subset of similar features which could potentially be the cause of the speedpath which the HPR method can identify.

	f_0	f_1	f_4	f_5	f_6	f_{14}	f_{15}	f_{17}	f_{18}	f_{20}
SP1	0.36	0.25	0.05	0.48	0.64	1.00	0.08	0.03	0.10	0.06
SP2	0.44	0.34	0.08	0.54	0.70	1.00	0.00	0.03	0.13	0.09
$D(f_0)$	0.005	0.008	0.001	0.004	0.004	0.000	0.006	0.000	0.001	0.001

$D(f_1,f_6,f_{14},f_{17})=0.107$ $D(f_0,...,f_{20})=0.172$

Table 1. Speedpath similarity search comparison

To further illustrate the advantage of HPR, in figure 18, we show 3 paths in the hypothesis space in both the single and two feature dimension. The X dimension represents a relevant feature, and the Y dimension represents a irrelevant feature. This illustration shows what we observed with the rogue path. When only relevant features are considered by the hypothesis from HPR (single feature dimension), the rogue path is more similar to a speedpath then when both relevant and irrelevant features are considered (two-feature dimension) which occurs in speedpath modeling.

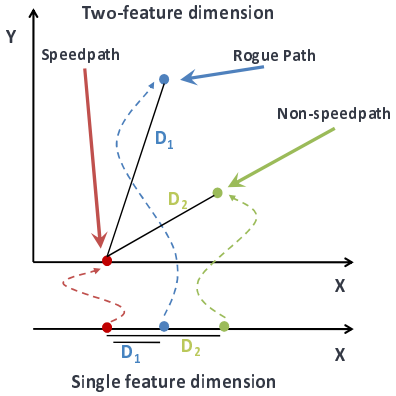


Figure 18. First order vs. second order

7 Conclusion

This paper presents a feature based similarity search approach and discusses three potential methods to implement this approach, building a model for the good, building a model for the bad, and establishing the hypotheses to explain individually why each sample is special. We apply similarity search to the speedpath analysis problem where special samples are speedpaths and the goal is to identify more paths in the design with similar characteristics to the speedpaths. The effectivenesses of the three methods were analyzed through experiments based on speedpath data collected from a high-performance microprocessor.

We demonstrate three key findings: (1) In general, it would be difficult to use the "modeling for the good" method for similarity search if the search is for bad examples, even though the number of good samples is large. (2) The "modeling for the bad" method is much more efficient to run than the method of building hypotheses while results of the two method can overlap significantly. (3) The hypothesis building method, although takes more time to run, can find special samples that are unlikely to be found by other methods. Hence, if run time allows, the hypothesis pruning and ranking method presents the most effective implementation for similarity search.

8 Acknowledgments

We acknowledge the following people at Intel for their contributions to our speedpath identification and root-cause analysis in the form of guidance and discussions: Eli Chiprout, Kip Killpack, Chandramouli Kashyap, Chriayu Amin. We also acknowledge Moshe Bensal from Intel IDC for his work in defining, identifying and collecting features and feature data for the speedpaths and non-speedpaths that was analyzed in this work.

References

[1] K. Killpack, et al, "Analysis of Causation of Speed Failures in a Microprocessor: A Case Study," To appear in IEEE *Design & Test* Special Issue on Silicon Debug and Diagnosis 2008.

[2] P. Bastani, N. Callegari, L. Wang, and M. Abadir. "Statistical diagnosis of unmodeled systematic timing effects," *DAC*, 2008.

[3] P. Bastani, et al. "Speedpath prediction based on learning from a small set of examples," *Proc. DAC*, 2008.

[4] D.Barton, P. Tangyunyong, J. Soden, A. Liang, and e. a. F. Low. "Infrared light emission from semiconductor devices," *ISTFA Proc.*, 1996.

[5] N. Callegari, P. Bastani, L. Wang, "Speedpath analysis based on hypothesis pruning and ranking," *Submitted to DAC* 2009.

[6] Bernhard Schlkopf, and Alexander J. Smola. "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond," The MIT Press, 2001.

[7] K. S. Kim, S. Mitra, and P. G. Ryan, "Delay defect characteristics and testing strategies," *IEEE D & T*, 2003.

[8] A. Gattiker and W. Maly, "Current Signatures," *VLSI Test Symp.*, 1996.

[9] P. Bastani, B. Lee, L. Wang, S. Sundareswaran, M. Abadir, "Analyzing the risk of timing modeling based on path delay tests," *ITC* 2007.