Path selection for monitoring unexpected systematic timing effects

Nicholas Callegari[1], Pouria Bastani[1], Li-C. Wang[1], Sreejit Chakravarty[2], Alexander Tetelbaum[2]
[1]Department of ECE, University of California, Santa Barbara
[2]LSI Corporation

Abstract— This paper presents a novel path selection methodology to select paths for monitoring unexpected systematic timing effects. The methodology consists of three components: path filtering, path encoding, and path clustering. Given a large set of critical paths, in path filtering, the goal is to filter out paths that cannot be functionally sensitized. To explore the space of unexpected timing effects, a set of features are defined to encode paths into path vectors. Each feature is a source of concern that may potentially contribute to the cause of an unexpected timing effect. Finally, a kernel-based clustering algorithm is employed to group similar path vectors into clusters from which the best representative paths are selected for post-silicon monitoring. The effectiveness of our proposed methodology is demonstrated through experiments on an industrial ASIC design.

I. Introduction

At each transition between technology nodes, as well as evolution in process and manufacturing, models and design methodology are constantly changing for accurately predicting silicon. However, even with significant effort put forth to minimize design-silicon mismatch, unexpected timing effects can still arise due to sources or combinations of sources that are not, or not accurately, modeled and analyzed. These timing effects can be the reasons for the loss of timing yield and hence, need to be carefully monitored in the post-silicon stage.

One way to monitor these effects are through on-chip test structures. Test structures, such as ring oscillators, have been used to monitor integrated circuit performance for many years [1, 2]. Test structures are primarily designed to provide a measure of performance, power and variability [3]. The data measured from test structures relates these measures to the properties of low level device parameters, in particular to MOSFETs and to parasitic delay elements [4, 5]. Using on-chip test structures requires one to hypothesize the effects and develop dedicated methodologies to measure them [6]. Because test structures are precious on-chip resources, they are often used only to monitor first-order known timing effects, and usually not sufficient to explore the entire space of unexpected timing effects.

The alternative is to monitor the effects through AC scan. In AC scan, one can use two types of tests: transition fault tests and path delay tests. Using transition fault tests may be a convenient choice because often they are already prepared for AC delay testing. However, there is no effective methodology existing today to correlate measurement results from transition fault tests back to models, design, and/or design methodology. Moreover, there is no reliable method proposed to estimate the effectiveness of a transition fault test set with respect to monitoring unexpected timing effects.

For diagnosing unexpected systematic timing effects, the authors in [7, 8] proposes a path based methodology. In the methodology, path delay tests are used to measure the delays of a set of paths on a set of silicon samples. The measured delays are correlated back to the predicted path delays from a static timing analyzer. The result of this correlation analysis is a rank of so-called *features* that point to the sources of unexpected effects. In the methodology, the set of paths to analyze is assumed to be given. Therefore, it does not answer the question of what paths need to be monitored in the first place.

In this work, we follow the path based framework employed in [7, 8]. However, instead of solving a diagnosis problem, we solve a path selection problem, and the goal is to develop a methodology to select an optimal set of paths to be measured by path delay tests. In this context, the merit of a path set is evaluated based on its size and the coverage of the space of the unexpected timing effects.

The rest of the paper is organized as the following. Section II explain the path selection problem investigated in this work. Section III presents an overview of the methodology proposed. Section IV discusses path filtering where functionally unsensitized paths are identified using a commercial ATPG tool. Section V describes in detail the application of a clustering algorithm to select representative paths. Section VI explains the experimental methodology and presents results to demonstrate the effectiveness of the clustering based path selection approach. Section VII concludes the paper.

II. Background and Problem Description

Design-silicon timing mismatch has been a known phenomenon for high-performance processor design for many years [8]. For example, timing critical paths reported from timing analysis are not the actual speed-limiting paths showing up on silicon samples [9]. However, only in recent years has the timing mismatch begun to raise concern for

ASIC designs. The main reason for the concern is the low timing yield often observed at 65nm and below. Because timing models and tools are not perfect, it is possible that an unexpected timing effect occurs in a design and causes some of the yield loss. Uncovering such an effect and fixing the design to avoid it can be the two steps for improving yield.

An unexpected timing effect can be due to something that is not, or not accurately modeled and analyzed in the design process. Because in pre-silicon modeling and analysis, assumptions are made and approximations are taken everywhere, the space for potential unexpected timing effects can be enormous. To effectively explore this space, the authors in [8] utilize a set of *features* to encode the space. A feature represents a source of concern that may contribute to the cause of an unexpected timing effect. The work in [7] then provides a survey of where those features may be defined.

Given a set F of n features, $F = \{f_1, \ldots, f_n\}$, one can hypothesize a space of unexpected timing effects based on F. For example, one may concern all effects associated with any feature individually as well as any combination of two features. In this case, the total number of effects to consider is $n +_n C_2 = n + \frac{n(n-1)}{2}$.

Given a set of m paths, each path is encoded with the features into a path vector. For simplicity, let's assume that all features are occurrence based [7]. That is, each path is a vector v consists of n 0's and 1's. $v[i] = 1$ if feature f_i shows up on the path. Otherwise, $v[i] = 0$. Then, the goal of path selection is to select a subset of k paths such that all $n + \frac{n(n-1)}{2}$ effects are covered.

In this work, we assume that the feature set F is given. In addition, the hypothesis for the space of unexpected timing effects can also be user-defined. Typically, the number k of selected paths is determined by the available test resource, such as the number of test patterns allowed. Given k, the goal of path selection is then to select the best k paths to maximize the coverage of the hypothesized unexpected timing effect space.

The formulation of the path selection problem is different from others that have been studied in the past. For example, the authors in [10] use a statistical method, and location based correlation for critical path selection in the context of delay testing. The goal is to maximize coverage for statistical, location-based delay defects. The authors in [11] present a statistical path delay fault coverage metric, taking into account structural and spatial correlation. It is designed for timing validation against variations and not for monitoring unexpected timing effects. The authors in [17] use graph theory combined with ATPG to identify a set of the longest testable paths that cover every gate, however due to the complexity of todays designs identifying and testing every sensitizable gate may not be reasonable.

III. Overview Of The Methodology

Figure 1 illustrates the path selection methodology proposed in this work.

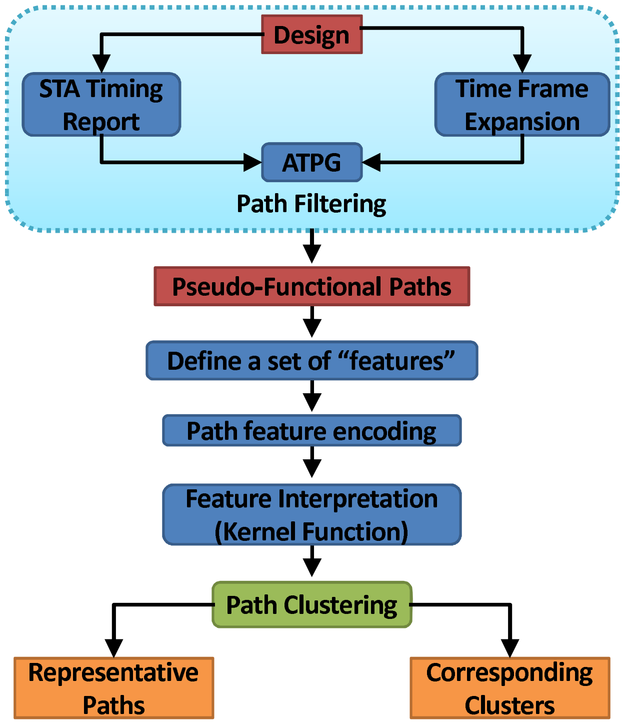


Fig. 1. **Overview of the path selection methodology**

The methodology begins with a design, netlist, in which a static timing analyzer (STA) is applied to produce a timing report. In this report, an initial set of timing critical paths are included. The netlist is modified via time frame expansion that will be explained in Section IV. The unrolled, time frame expanded, netlist is combined with an ATPG tool to analyze the sensitization of each critical path. A path that fails to be sensitized is then removed from further consideration. The resulting paths, which can be sensitized on the time frame expanded netlist, are called pseudo-functional paths [12].

A set of features are defined to explore the space of unexpected timing effects [7]. Each pseudo-functional path is then encoded with these features as a path vector. A *kernel* function is used to interpret the features. The interpretation is in the following sense. Given two path vector p_1, p_2, a kernel function $k(p_1, p_2)$ gives a similarity measure between the two vectors [13]. For example, let $p_1 = (1, 1, 0)$ and $p_2 = (0, 1, 1)$. A dot-product kernel calculate the similarity as $k(p_1, p_2) = 1 \times 0 + 1 \times 1 + 0 \times 1 = 1$. Note that there are many kernels widely used in diverse machine learning applications [13]. Also note that in the path selection methodology, the kernel in use depends on one's hypothesis of the unexpected timing effect space. For example, in Section V we will explain a *Spectrum* kernel that takes the consecutive ordering of the cells along a path into account.

Once a kernel function is decided, the clustering algorithm then groups paths into clusters where similar paths are put into the same cluster. A path is selected from each cluster to form the set of representative paths. As we can see, representative paths are dissimilar paths for maximizing the coverage, where the similarity is defined by the kernel function. From this perspective, the kernel function implicitly define the unexpected timing effect space to be explored.

IV. Path Filtering

With a commercial STA tool, we are able to select the top i critical paths ending at every flip flop. We begin by including as many paths as possible, limited by the available computation resource. This path set is pruned by removing functionally unsensitizable paths. Similar analysis has been done before to path-delay test in [14], by applying ATPG to identify which of the STA reported critical paths are actually sensitizable. More advanced technique has been used in [12] by applying Boolean satisfiability solver to identify pseudo-functional paths.

Our implementation utilizes a design unrolling technique where the design is unrolled based on time frame expansion (TFE) to simulate multiple clock cycles. As described in [12], Boolean SAT solver can be applied to the unrolled design in combination with a path to determine if previous time frames (clock cycles) can provide the proper inputs to the last time slice to properly sensitize the path. What is achieved by implementing TFE is pseudo-functional path identification, where a path that cannot be sensitizable by structural test, can be filtered. By using this method, paths that are identified pseudo-functional are not guaranteed to be functionally sensitizable paths. However, paths that are identified not to be pseudo-functional are guaranteed not to be functionally sensitizable paths. Therefore by using this process we are able to reduce the original set of paths, and be assured not to remove any functionally sensitizable paths. In our implementation, instead of using a Boolean SAT solver, we used a commercial ATPG tool because it is readily applicable to industrial designs.

One issue that can be seen with time frame expansion is the increase in design size. For each time frame the design is unrolled, the new unrolled design becomes $N \times D + D$, where N is the number of times the design is unrolled, and D is the size of the original design. Techniques can be incorporated such as partitioning the design by clock domain, which is common practice in large scale designs. If the increase in design size cannot be overcome, ATPG alone without TFE can still provide a significant reduction in paths and does not hinder the overall representative path selection flow. Nevertheless, although relatively design dependent, shown in Table I, in our experience, with the minimal overhead of only unrolling our design one time frame we are able to filter out the majority of non-functional paths identified by TFE.

To test our implementation we conducted time frame expansion on an industrial ASIC design consisting of 220K gates. We varied the the number of time frames from 0, 1 and 5, 0 being the original design, and 1 and 5 being the the original design plus 1 and 5 copies of the design to simulate 1 and 5 clock cycles prior. We show the number of sensitizable paths identified, as well as the reduction of paths compared to the original number of paths.

TABLE I
Path Filter Results

Total Paths	0-cycles	1-cycles	5-cycles
8332	3029	1567	1540
%Reduction	63.65%	81.19%	81.51%

Because we obtained such a significant reduction in paths in the first unrolling, and minimal improvement thereafter, we chose to select the 1,567 paths to represent our pseudo-functional paths to continue with the remainder of the flow.

V. Clustering

The Fuzzy c-means clustering algorithm was used in this work, however other clustering algorithms can be considered for the methodology. We selected Fuzzy c-means because the algorithm incorporates fuzzy logic in which each path has a degree of belonging to each cluster, rather than only belonging to one specific cluster. Defined by [15] Fuzzy c-means clusterings objective is to maximize the inter-cluster variance and minimize the intra-cluster variance. The cluster center is placed where the objective is reached given the constraints of the number of iterations and/or the procedure converging to a local minimum of the objective function. The objective function of Fuzzy c-means clustering is as follows;

$J_m(U, V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m \times dist(x_k, v_i)$, where:

•$X = \{x_1, x_2, ..., x_n\} \subset R^s$ is the data set to be studied

•$c, 2 \leq c \leq n$ is the number of clusters in X

•$m, 1 \leq m$ is a weighting factor exponent; when m is close to 1, fuzzy c-means behaves similarly to the K-means clustering algorithm.

•$u_{ik} = u_i(x_k)$ is the membership of x_k in cluster u_i; usually, we have $\sum_{i=1}^{c} u_i(x_k) = 1$.

•$U = [u_{ik}]$ is a fuzzy c-partition of X

•v_i is the center of cluster i

•$dist(x_k, v_i)$ is the OG distance of any inner product norm on R^s from x_k to v_i

Mathematically defined;

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{dist(x_k, v_i)}{dist(x_k, v_j)} \right)^{\frac{2}{m-1}}}$$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik} x_k}{\sum_{k=1}^{n} u_{ik}}$$

Given a kernel function $k()$, the OG distance can be calculated as $k(x_k, x_k) - 2k(x_k, v_i) + k(v_i, v_i)$. Note that this is the kernel-based extension of the original Fuzzy c-means algorithm. In the original algorithm, the distance is nothing but the Euclidean distance calculation $|| x_k - v_i ||^2$.

Fuzzy c-means performs iterative optimization of the objective function J, updating the membership, u_{ik}, of all paths, x_k, to all clusters, and updating the cluster centers v_i. The iteration stops when the user specified termination criterion is met, which is when the improvement in u_{ik} (the sum of membership for all paths for each cluster) is less than the termination criterion. This is essentially a local minimum of the objective function J.

Once the termination criterion is met, the path with the greatest membership to each cluster is included into the representative path set. By selecting the path with the greatest membership we select the centroid path, the path with the most similarity to every other path in the cluster. We output the resulting membership values for every other path for each cluster as our resulting clusters.

Although less critical paths are being considered, our objective is to maximize coverage of the space of unexpected timing effects. Therefore if a less critical path is selected, from delay test we can determine if mismatch exists. From further analysis the unexpected timing effect causing the mismatch can be identified for more critical paths.

A. Spectrum kernel

In this work, we use a Spectrum kernel [13] to measure similarity between a pair of paths. The hypothesis is that an unexpected timing effect can be due to the presence of a single cell or a combination of j cells connected in a certain order, for example a weak cell driving a large cell. To illustrate the kernel computation, suppose we have 5 cells A, B, C, D, E. Consider a path $p_1 = ABACCEA$ and $p_2 = CBACD$. For the case of single cell, $p_1 = \{A, B, C, E\}$ and $p_2 = \{A, B, C, D\}$. Hence, they share 3 cells. For $j = 2$, $p_1 = \{AB, BA, AC, CC, CE, EA\}$, and $p_2 = \{CB, BA, AC, CD\}$. They share BA and AC. Hence, their 2-Spectrum measure is 2. Together, we have $k(p_1, p_2) = 2 + 3 = 5$.

The Spectrum kernel can be defined for j up to the maximum number of cells on a path. For demonstration purpose, in this work we only experiment with $j = 2$. Note that with a Spectrum kernel, the features are individual cells themselves.

B. Selecting The Number of Clusters

A difficulty accompanied with clustering algorithms is selecting the optimal number of clusters. If we select too few clusters we may not cover all potential unexpected effects. If we select too many clusters, we may exceed the limit on the number of test patterns.

C. Elbow Criterion

Introduced by [16], the concept of the elbow criterion can be applied to identify an optimal number of clusters based on the objective function. The elbow criterion describes how one can select the number of clusters in such a manner that adding another cluster does not add sufficient information, i.e. explain variance. More specifically, what is seen is the law of diminishing returns, in which if one graphs the variance explained by the clusters against the number of clusters. When initial clusters are added, information gained is large, where as with subsequent clusters added, the amount of information gained reduces. When applying the elbow criterion to this concept, one essentially pick the elbow, the number of clusters in which the gain by adding additional clusters starts to diminish.

To apply this concept to our dataset, we ran the Fuzzy c-means clustering algorithm on the 1567 paths obtained after the path filtering. The resulting variances are obtained directly from our objective function, $J_m(U, V)$, by varying the number of clusters from 5 to 500. As shown in Figure 2, the elbow is somewhat ambiguous, where any number of of clusters between 100 to 200 could be seen as the elbow. Conversely, the elbow is specific in the sense that we can eliminate between 0 to 100, and greater than 200 clusters. For the experiment we selected 150 clusters for estimating its coverage.

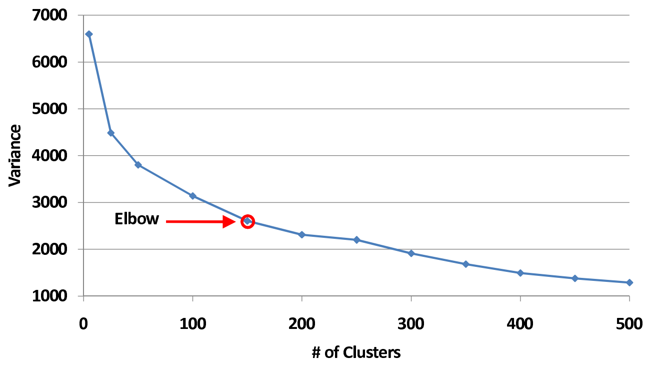


Fig. 2. **Selecting the optimal number of clusters using Elbow Criterion**

D. Feature coverage criterion

A straightforward method, in addition to the objective function, for deciding the number of clusters is to implement a feature coverage estimator. For example, given n features, there are n individual features to cover and $n(n-1)$ ordered pair of features to cover. Given a set of clusters, one can then calculate the percentage of these $n + n(n-1)$ effects being covered. Figure 3 shows such a coverage estimation, varying the number of clusters from 5 to 500.

Two interesting things can be observed in this plot. First, the selection of 150 clusters based on the elbow criterion earlier does represent a point where the return, in terms of feature coverage, of adding more clusters is diminishing. Second, by adding more clusters the coverage result never decreases. This monotonic behavior of the path selection algorithm is desirable because then it is safe to select more paths if the test resource allows.

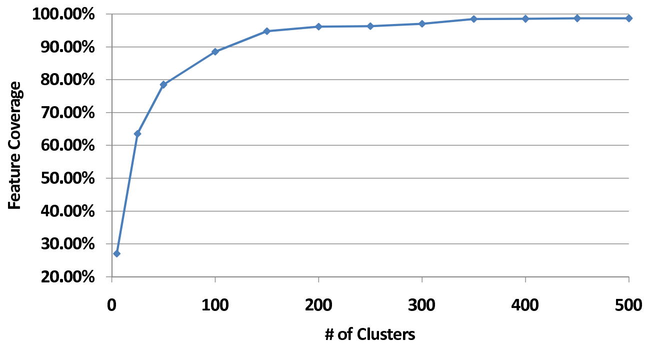


Fig. 3. **Selecting the optimal number of clusters using feature coverage criterion**

VI. Experiments

To show the effectiveness of our algorithm, we randomly inject a single timing error into the design and calculate an error coverage by a given path set. The timing errors are to simulate systematic unexpected timing effects. A timing error can be associated with a feature or an ordered pair of features.

For comparison, we also selected two other path sets, one by random selection from the 1567 paths and the other by selecting the top timing critical paths, where the criticality was decided by the STA tool. For the random selection, we repeated the experiment 100 times and took the average to avoid bias. For the clustering representative paths we extracted the most centroid point from each of the 150 clusters.

In each experiment, we randomly formed 100 timing errors and compared the coverage results based on the three path sets on these 100 errors. The results are shown in table II. These results clearly show that the clustering representative paths are significantly better than the other two sets of paths.

TABLE II
Error Injection Coverage

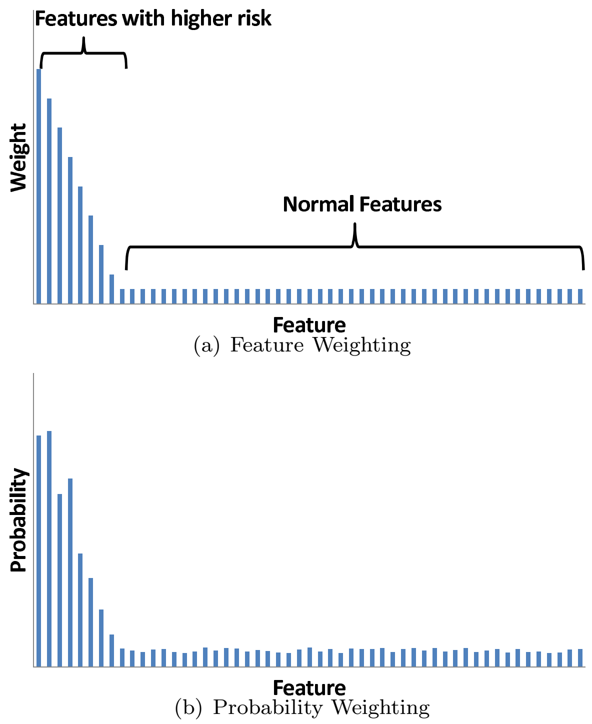
Method	Paths	Error Injected	Error Covered
Clustering	150	100	94.80%
Random	150	100	78.26%
Top Critical	150	100	32.00%

One characteristic the clustering algorithm provides, that a typical set covering algorithm cannot, is the ability for a user to weight features based on their assumed importance. The interface for the user can be as simple as selecting a subset of features which are believed to have some higher levels of risk, and order them based on that level. Features that are not selected are still considered by the clustering algorithm, but emphasis is placed on the selected ones.

Consider the Spectrum kernel example discussed in Section A again. Suppose we believe that cell A, B have a higher risk for causing unexpected effects. To force the path selection to focus more on A, B, we can assign a larger weight, said 2, to A and B while leaving the weight for others as 1. Now consider the path $p_1 = ABACCEA$ and $p_2 = CBACD$ again. For the case of single cell, $p_1 = \{A, B, C, E\}$ and $p_2 = \{A, B, C, D\}$. Hence, they share 3 cells A, B, C and because A, B are weighted by 2, the total similarity contribution by this comparison is $2+2+1 = 5$. Again, for $j = 2$, $p_1 = \{AB, BA, AC, CC, CE, EA\}$, and $p_2 = \{CB, BA, AC, CD\}$. They share BA and AC. Hence, their 2-Spectrum measure can be calculated as $2\times2+2\times1 = 5$. Together, we have $k(p_1, p_2) = 5+5 = 10$.

To set up the experiment we selected 10% of the features, and weighted them based on the weights shown in figure 4(a). Some of the features we selected were sparse features, features that do not appear very often in the design. We believe this would be realistic because features that exhibit higher risk are usually used when absolutely needed. Additionally, we selected featured that were not originally covered by the clustering algorithm in the previous experiment to verify the effectiveness of this weighting.

Again, we randomly formed 100 timing errors. This time each error is based on features that have unequal probabilities to be selected. This probability distribution is plotted in figure 4(b) and was obtained by modifying the feature weight distribution of figure 4(a) with 10% random noise. The idea is to simulate the situation that the assumed risk levels associated with the features, as reflected by the feature weights, are roughly correct but not entirely correct for capturing the reality.



(a) Feature Weighting

(b) Probability Weighting

Fig. 4. **Feature weighting in path selection and probability weight in random error injection**

In the results shown in table III we can observe a significant improvement in coverage by the clustering representative paths. With further inspection on the features that were covered, we can attribute this improvement to the fact that all features which were not covered in the original experiment were then covered when the weighting was applied. This confirms the increase in effectiveness by using this weighting scheme. The error coverage for random and top critical significantly decreased because emphasis is not placed on selecting paths which contain features that exhibit higher risk.

TABLE III
Weighted Error Injection Coverage

Method	Paths	Error Injected	Error Covered
Clustering	150	100	98.48%
Random	150	100	56.66%
Top Critical	150	100	23.00%

VII. Conclusion

In this paper, we present a path selection methodology to select paths for monitoring unexpected timing effects. In the methodology, an ATPG based approach is applied to identified functionally unsensitizable paths and remove them from further consideration. A set of features together with a kernel function to interpret them, define the hypothesized space of unexpected timing effects to be explored. In this space, paths are selected for maximizing the coverage. Our path selection algorithm is based on a kernel-based clustering algorithm. Similar paths are grouped into a cluster where the most representative path for the group is selected from most centroid point of the cluster. The effectiveness of the path selection algorithm is demonstrated based on simulation of assumed timing errors on an industrial ASIC design.

Although the current work is based on a simple Spectrum kernel and straightforward cell features, as described in [7], features and their associated kernels can be flexibly defined. We emphasize the point that the kernel function used in the clustering algorithm implicitly define the unexpected timing effect space to be explored. Therefore, the selected paths always depends on the kernel in use. Putting a weight scheme on the features based on their assumed risk, can be viewed as simply changing the kernel computation, as illustrated in the previous section.

With the ability to flexibly define the set of features and the kernel, we see that the proposed path selection methodology does not intend to provide a golden answer to the path selection problem. Instead, it provides a rather generic framework that allows a user to conveniently incorporate his/her domain knowledge into the path selection process and select the optimal paths based on his/her desired perspective.

It is important to note that selecting paths to monitor unexpected timing effects and selecting paths to maximize the diagnosis resolution for the unexpected timing effects, said using the diagnosis framework proposed in [8], are two different problems. This work attempts to solve the first problem that concerns maximizing the coverage of the effects. This objective can contradict to the objective of maximizing diagnosis resolution. Our conjecture is that for maximizing the coverage, we should select representative paths that are dissimilar to each other. For maximizing diagnosis resolution we should utilize paths that are similar. Based on this conjecture, a representative path from a cluster is used to monitor unexpected effects and when such an effect is observed on the path, all paths in the cluster can be used in diagnosing the cause. We plan to investigate this proposal in future work.

References

[1] L. Milor, L. Yu, B. Liu "Logic Product Speed Evaluation and Forecasting During the Early Phases of Process Technology Development Using Ring Oscillator data," Proc. International Workshop on Statistical Metrology, 1997, pp. 20-23.

[2] D. Boning, S. Nassif, A. Gattiker, F. Lui, et al. "Test Structures for Delay Variability" Proc. of the 85th ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, 2002, p.109.

[3] C. Cho, et al, "Statistical Framework for Technology-Model-Product Co-Design and Convergence," Proc. DAC 07

[4] M. Bhushan, et al, "Ring oscillator for CMOS process tuning and variability control," IEEE Trans. on Semiconductor Manufacturing, Vol. 19, pp. 10-18, 2006

[5] M. Ketchen, et al, "High speed test structures for in-line process monitoring and model calibration," Proc. IEEE Inter. Conf.Microelectronic Test Structures, 2005.

[6] R. Franch et al. On-chip Timing Uncertainty Measurements on IBM Microprocessors *Proc. ITC*, 2007.

[7] P. Bastani, et al. Diagnosis of design-silicon timing mismatch with feature encoding and importance ranking- the methodology explained, *ITC*, in press, 2008.

[8] P. Bastani, et al. Statistical Diagnosis of Unmodeled Systematic Timing Effects, *DAC*, in press, 2008.

[9] K. Killpack, C. Kashyap, E. Chiprout, "Silicon Speedpath Measurement and Feedback into EDA flows," *Proc. DAC*, 2007.

[10] J. Liou, L. Wang, A. Krstic, K. Cheng, "Experience in Critical Path Selection For Deep Sub-Micron Delay Test and Timing Validation," *ASPDAC*, pp. 751-756, 2003.

[11] W. Qiu, X. Lu, J. Wang, Z. Li, D. Walker, W. Shi, "A Statistical Falt Coverage Metric for Realistic Path Delay Faults," *VTS*, pp. 37-42, 2003.

[12] Y. Lin, F. Lu, K. Yang, K. Cheng, "Constraint Extraction for Pseudo-Functional Scan-based Delay Testing," *ASPDAC*, pp. 166-171, 2005.

[13] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press 2004.

[14] J. Zeng, M. Abadir, J. Abraham, "False Timing Path Identification Using ATPG Techniques and Delay-Based Information," *DAC*, p. 562, 2002.

[15] J. C. Bezdek, C. Coray, R. Gunderson, J. Watson, "Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy c-Lines," *SIAM Journal on Applied Mathematics*, vol. 40, no. 2, pp. 339-357, 1981.

[16] M. S. Aldenderfer, R. K. Blashfield, *Cluster Analysis,* Sage Publications, 1984.

[17] M. Sharma, J. H. Patel, "Finding a Small Set of Longest Testable Paths that Cover Every Gate," *ITC*, pp.974, 2002.