A Study of Outlier Analysis Techniques for Delay Testing *

Sean H. Wu, Dragoljub (Gagi) Drmanac, Li-C. Wang
Department of ECE, UC-Santa Barbara

Abstract

This work provides a survey study of several outlier analysis techniques and compares their effectiveness in the context of delay testing. Three different approaches are studied, an Euclidean-distance based algorithm, Random Forest, and one-class Support Vector Machine (SVM), from which more advanced methods are derived and analyzed. We conclude that one-class SVM using a polynomial kernel is most effective for detecting delay defects, while keeping overkills minimized. The best models were successfully validated and a feasible approach to delay testing using one-class SVM is proposed.

1 Introduction

Testing is intended to distinguish between good and bad chips. A golden reference for the good chips is needed in order to decided what is bad. In delay testing, this golden reference can be associated with a test clock. For example, if a design is signed-off at 800MHz, one may desire to have a test clock matching that frequency. This may involve tuning the pattern set and/or the setting of test conditions, such that known good chips are included and known bad chips are screened out with an 800MHz clock setting.

As technology advances beyond 65nm, this seemingly simple strategy may face several challenges. The first set of challenges is associated with the assumption that the sign-off frequency is the golden reference. The second set of challenges concerns the screening of "small delay defects."

For example, the authors in [1] show that timing models can be twice as pessimistic with respect to the actual timing seen on silicon. It is well known that designs follow the principle of conservatism. The sign-off frequency is supposed to be the upper bound of the operational timing, which may or may not accurately predict the actual timing on silicon. Then, as the sign-off frequency is used as the golden reference, the effectiveness of delay testing inevitably depends on the accuracy of timing models and timing analysis tools. As pointed out by the authors in [2], design-silicon timing correlation has become a challenging problem in the nanometer era. For example, it is unrealistic to expect a timing analysis tool to correctly predict actual speed-limiting paths seen on microprocessor chips [3]. If it is difficult to guarantee prediction accuracy from timing models and timing analysis, one may desire a delay test methodology that is independent of timing models and analysis.

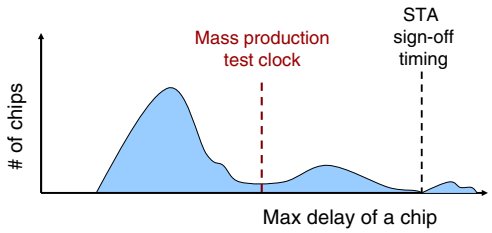


Figure 1. Selecting a test clock is not trivial

If one does not take the sign-off frequency for granted, deciding a golden reference becomes an open-ended issue. Figure 1 illustrates the situation. By checking the max-delay distribution on a collection of sample chips (without knowing which are good or bad), it may not be intuitive to decide an optimal test clock. In addition, this optimization depends on one's test-escape and over-kill requirements.

The burden of deciding a golden reference can be alleviated if one is allowed to have known-good and known-bad sample dies. Then, the problem becomes finding an optimal boundary to separate the two sets. This can be seen as a Binary Classification problem and the authors in [4] discuss algorithms for its optimization.

Suppose we do not want to depend on the timing models and analysis, nor do we have known-good and known-bad dies on hand. We ask the following question: Is Figure 1 the best we can do to decide a golden reference, i.e. does it make the best sense to determine the golden test clock using max-delay distribution? This question brings us to the second set of challenges in delay testing, i.e. capturing small delay defects.

What is a small delay defect? Intuitively, a small delay defect causes a small delay increase somewhere in the chip. If we take the perspective of Figure 1, for capturing small delay defects, there are two approaches we may follow. The first is adjusting the test clock, i.e. using a tighter test clock. The second is using a set of patterns that sensitize long-delay paths. While both approaches can be

*This work is supported in part by National Science Foundation, Grant No. 0541192 and Semiconductor Research Corporation contract No. 2007-TJ-1585

feasibly implemented in practice, their perspectives of capturing the small delay defects may become questionable if one takes statistical variations into account. For example, they do not directly answer the following questions: Should random variations be considered as small delay defects? If the magnitude of a small delay defect is smaller than the magnitude of variations, do we still care about capturing it? If we do, what method should we use?

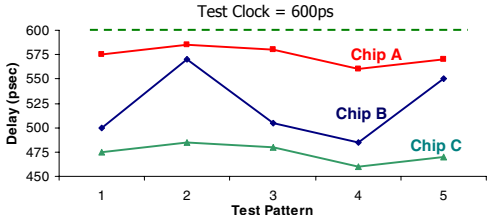


Figure 2. C behaves differently from A and B

Consider Figure 2 where the timing behavior of three chips are shown. Five patterns are applied and their delays are plotted for each chip. Suppose the test clock is set at 600ps. From the perspective of Figure 1, all three would be classified as good chips. If we compare the behavior between A and C, we observe that they follow a similar trend. It may be reasonable to assume that the delay differences seen between A and C are due to systematic process variation. On the other hand, we see that B behaves differently from A and C, i.e. the delays of pattern 2 and 5 seem to be excessively long compared to the trend observed with A and C. We may conjecture that B has "small delay defects" because its behavior does not follow the *normal trend*.

1.1 Normal trend and outliers

Given a set of sample chips, suppose we can somehow build a model to capture the "normal trend" among these chips. Then, we can use this model to screen out the *outliers* from the set. Moreover, the model can be used to differentiate between normal chips and outliers on a new set of samples. If we consider a chip that follows the normal trend as good, and a chip that does not follow the normal trend as bad, then we see that the problem discussed above for deciding a golden reference becomes the problem of deciding a boundary for the model.

We note that depending on the characteristics of the sample chips, a "normal" chip may not be good. For example, suppose a significant portion of the samples are all defective in a similar fashion. Then, what is "normal" given the samples can mean abnormal or defective. However, if this is indeed the case, then there must be something fundamentally wrong, for example, a systematic error may exits in the design. If this is the case, then such a problem needs to be fixed before applying outlier analysis.

In this work, we assume that it is safe to call a chip good if it follows the normal trend. Then, we treat delay testing as the process of identifying "outliers." This perspective has two key advantages: (1) The normality is entirely defined by the behavior of the sample chips and hence, defining what is good is totally independent of pre-silicon timing models and analysis. (2) There is no need to have known-good and known-bad dies in advance.

In fact, both Figure 1 and Figure 2 can be viewed as forms of outlier analysis. In Figure 1, the analysis is one-dimensional. Each chip's behavior is characterized by its max delay. The normal trend is defined by a test clock, where all chips on the left of the cut-off delay are considered normal. Chips on the tail of the distribution are considered outliers (bad). In Figure 2, the analysis is five-dimensional based on the delays of five patterns. A trend is defined by comparing the relative delays of these patterns and by comparing the five delays across all chips.

In this paper, we take the perspective of Figure 2 and consider delay testing as a multi-dimensional outlier analysis problem. There are three objectives for this work:

- Conducting a comprehensive study on state-of-the-art outlier analysis techniques.
- Based on the study, deriving the best outlier analysis algorithm for delay testing.
- Demonstrating that small delay defects can be effectively captured by an outlier analysis approach, even in the presence of statistical process variations.

With these objectives in mind, the rest of the paper is organized as follows. Section 2 discusses related works using outlier analysis tecniques in testing. Section 3 introduces the four basic components that constitutes an outlier analysis approach. Section 4 describes three outlier analysis approaches. Section 5 explains the experimental setup and Section 6 shows experimental results comparing the three baseline approaches. Section 7 suggests several improvements to the best baseline approach, the one-class Support Vector Machine (SVM) [25] and concludes its overall superiority. Section 8 presents experimental results that validate the SVM one-class model. Section 9 concludes the paper.

2 Related Work

Outlier analysis has been studied extensively in Iddq testing because of the difficulty in setting a reliable threshold to separate the intermixed defective and defect-free behavior. The authors in [5] and [6] analyze all measurements collectively as a *current signature* instead of comparing each measurement to a set threshold. In [7], the authors propose using the *current ratio* between one sample's maximum and minimum measurements to screen outliers. Utilizing multiple tests or metrics for screening outliers are presented in [5] [8] [9]. Similar methods are also applied in analog testing [20] [21]. Various works propose using the residual between the measured and estimated values to reduce the variance seen in the dataset [12] [15] [16]. Outliers

become more obvious when the variance of most data is reduced. All the above works try to solve an outlier analysis problem by finding the best set of *features* to use. With a better set of features (a better definition of axises), the sample data can be plotted in such a way that outliers are more easily identified. As we will discuss in Section 3, *feature generation* is one of the four components that constitute an outlier analysis approach. In general, one should not be restricted to using a small number of features.

In [14], clustering technique is applied on Iddq data. Different methods of determining a cluster boundary are discussed in [11][10]. These works focus on how to define a proper boundary for normality. Statistical analysis on Iddq test data has proven benefits and has been put into manufacturing test flow as described in [17][19]. The authors in [22] briefly discuss the use of Random Forests for outlier analysis in delay testing. Some encouraging initial results are shown but it remains unclear if Random Forests is the best approach to perform outlier analysis

3 Overview of Outlier Analysis

There are four major components in an outlier analysis approach. They are (1) feature generation, (2) similarity measure, (3) defining the boundary of the normality space, and (4) model validation or confidence estimation. Figure 3 illustrates these four components.

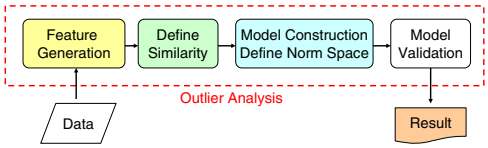
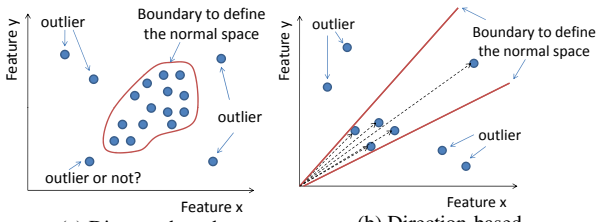


Figure 3. 4 components in an outlier analysis approach



(a) Distance-based (b) Direction-based

Figure 4. Illustration of outlier analysis

Figure 4 shows two examples to illustrate the first three components. Both examples use only two *features*, x and y. For instance, one may view these as delay measurements on two patterns. In a more complicated situation, x and y each can be a linear combination of delay measurements on a subset of patterns. The point here is that the behavior of each sample is characterized by a two-dimensional vector (x,y) that can be plotted as shown in the figures. Deciding what features to use in the outlier analysis is called *feature generation* or *feature selection*.

In Figure 4(a), a boundary of the normality space is drawn. Samples (dots) inside this boundary are considered as normal. Samples outside are outliers. The boundary becomes clearer as more samples cluster inside, while outliers spread outside. However, notice that this intuition is implicitly built on the notion of Euclidean distance. When we see that two samples are close, what we are really saying is that their Euclidean distance measure is small. If we treat such a distance measure as a *similarity* measure, we can also say that they are similar to each other.

Euclidean distance is one of the many ways to measure similarity between two samples. Figure 4(b) shows a different measure based on the angle between two vectors. In this figure, two samples are similar if the angle between the two vectors representing them is smaller (usually measured by the cosine function). Notice that the boundary of the normality space is no longer a closed space. In this case, it is a spread over an angle. More interestingly, observe that the upper-right sample is included in the normality space. If we use the Euclidean distance measure, that sample would have been treated as an outlier.

To illustrate this point further, consider the following example. Suppose we have three samples characterized by three 3-dimensional vectors $v_1 = (3,2,3)$, $v_2 = (3,0,3)$ and $v_3 = (1,0,1)$. Suppose the similarity is measured by the Euclidean distance $d()$, then we have $d(v_1,v_2) = \sqrt{(3-3)^2+(2-0)^2+(3-3)^2} = 2$, $d(v_2,v_3) = \sqrt{(3-1)^2+(0-0)^2+(3-1)^2} = \sqrt{8} = 2.8284$, and $d(v_1,v_3) = \sqrt{(3-1)^2+(2-0)^2+(3-1)^2} = \sqrt{12} = 3.4641$. We see that v_1, v_2 are closer (shorter in distance) than v_1,v_3 or v_2,v_3. A boundary can be drawn by setting the threshold at 2 such that v_1,v_2 are within the boundary and v_3 becomes an outlier.

On the other hand, suppose we use a different function, such as the cosine function $cos(v_a,v_b) = \frac{\langle v_a,v_b\rangle}{||v_a||||v_b||}$ (where $\langle v_a,v_b\rangle$ is the *dot product* of the two vectors). Then, we have $cos(v_1,v_2) = \frac{3\times3+2\times0+3\times3}{\sqrt{3^2+2^2+3^2}\sqrt{3^2+0^2+3^2}} = \frac{18}{19.8997} = 0.9045$, $cos(v_2,v_3) = \frac{3\times1+0\times0+3\times1}{\sqrt{3^2+0^2+3^2}\sqrt{1^2+0^2+1^2}} = \frac{6}{6} = 1$, and $cos(v_1,v_3) = \frac{3\times1+2\times0+3\times1}{\sqrt{3^2+2^2+3^2}\sqrt{1^2+0^2+1^2}} = \frac{6}{6.63325} = 0.9045$. In this case, we see that v_2,v_3 are more similar (larger cosine value) than v_1,v_2 or v_1,v_3. Hence, a boundary can be drawn to make v_1 an outlier.

It should be apparent now that how the similarity is measured between a pair of samples can change how the boundary is drawn. In fact, for a modern outlier analysis algorithm such as one-class SVM to work, all the information it needs is in the *similarity matrix* that records the similarity measure between every pair of samples. The interesting point here is that, the absolute values in a sample vector are not important. What is important is its relative similarity to other samples, defined by a similarity measure function. Such a function therefore defines the metric space for an outlier analysis algorithm to work on.

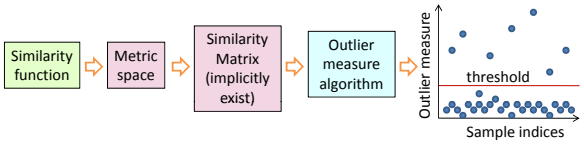


Figure 5. Typical working principles of a modern outlier analysis algorithm

Figure 5 summarizes the working principles of a modern outlier analysis algorithm. A similarity function defines a metric space for the algorithm to work on. Once such a metric space is defined, a similarity matrix can be computed. This matrix may implicitly exist in the implementation of an algorithm to save memory, i.e. when the similarity value between a pair of samples is required, it is computed on the fly. The algorithm utilizes information contained in the similarity matrix to compute an *outlier measure* for each sample. These measures are then plotted and a threshold can be drawn to decide the outliers. For example, in Figure 4(a) the outlier measure can be the distance of a sample point to the center of the normality region defined by the boundary.

3.1 Curse of dimensionality and feature selection

The *Curse of dimensionality* is a well known concept in statistical and machine learning research. Basically, it says that as the the number of dimensions in a problem increases, the complexity of the problem can grow exponentially. Since the dimension of a problem corresponded to the number of features in use, it inspired applying Principal Component Analysis (PCA) to select *important features* before utilizing a learning algorithm. PCA produces new and uncorrelated features (Principal Components) from the original set of features. Usually, by selecting a rather small number of principal components, one can reasonably approximate the statistics contained in the dataset represented by the original set of features. Because the number of dimension is reduced, it is expected that a learning algorithm can learn more effectively from the transformed dataset represented by the principal components.

In delay testing, the number of features is the number of patterns. If one uses 10K patterns, the number of dimensions is 10K. This requires an outlier analysis algorithm to work on a 10K-dimensional space. Because of this large number, the authors in [22] discuss ways to select important features using Random Forests.

While it is intuitive to conjecture that selecting important features can be a crucial step in outlier analysis for delay testing due to the large dimensionality, in Section 7 we will demonstrate that, if a proper similarity function is used with a right outlier measure algorithm, large dimensionality becomes a non-issue. This is an important finding of this work, i.e. the curse of dimensionality is not an important issue in outlier analysis for delay testing. This means that one can always use a large number of patterns to perform the analysis.

3.2 Model validation

An outlier analysis algorithm builds a model to represent the normality space and then uses that model to compute an outlier measure value for every sample. This is an *unsupervised learning* process because the algorithm learns what the normal trend is from a set of sample and then uses that knowledge to classify the samples themselves. Once the learned model is built, it is desired to estimate the confidence of the model by applying it to a new set of samples. Theoretical estimation of this confidence has been a very difficult problem. In practice, the most commonly-used approach is *k-fold cross validation*.

In k-fold validation, $\frac{k-1}{k}$ fraction of samples are used to build the model and the remaining $\frac{1}{k}$ samples are used for validation for each run. If the results from k trials are mostly consistent, then one has a high confidence in the model. Otherwise, one does not. Typically, $k = 3$ or 5.

4 Outlier Analysis Techniques

Feature selection	PCA, Random Forests (RF)
Similarity function	Euclidean, RF, kernel functions*
Outlier measure/ Model building	distance-based, multi-dim. scaling* 1-class SVM

*to be discussed later

The table above summarizes the techniques studied in this work. In this section, we first analyze three baseline approaches as explained below.

PCA+Euclidean distance The first approach uses PCA to reduce data dimensions, and Euclidean distance measure to define the metric space. Then, the outlier measure is calculated as the distance to the sample mean, i.e. for every sample its distance to the mean sample vector is calculated.

Random Forests (RF) The second approach uses RF to select important features and construct a *proximity matrix* [22]. Then, *multi-dimensional scaling* is used to calculate the outlier measure for each sample.

1-class SVM The third approach does not include feature selection. It uses a *kernel function* to measure similarity and the 1-class SVM algorithm to build a hyperspherical model in a high dimensional space. The outlier measure is calculated based on this model.

Since PCA, [23], is a well known technique, we skip further discussion of the first approach. Next we will explain the basic principles of Random Forests and 1-class SVM with kernel functions.

4.1 Random Forests

Random Forests (RF) was introduced by Breiman in 2001 based on decision tree theory [29]. RF was introduced as a *supervised learning* approach where its most popular

use was in Binary Classification [22]. A forest is a collection of decision trees where each tree is an over-fitting model for a randomly-selected subset (of equal size) of samples. RF prediction is based on voting from all trees.

To use RF for *unsupervised learning*, such as outlier analysis, the first step is to convert the problem into a binary classification problem. Given the set A of samples to be analyzed, the first step is to produce a *synthetic* set *Syn* of samples. By labeling the original samples in A as $+1$ class and the synthetic samples in *Syn* as -1 class, together they form the dataset for binary classification. RF is applied on this combined dataset to build a model M_{rf}. Figure 6 illustrates this notion. M_{rf} tries to model the boundary between

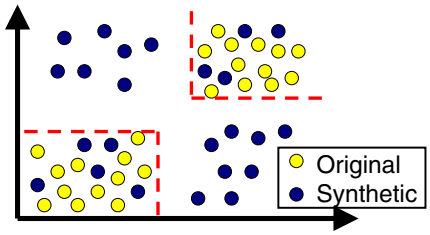


Figure 6. Building a model M_{rf} to separate the original and the synthetic samples

the space where all original sample points reside and the space where no original samples reside. The second space is represented by the synthetic samples that are produced by randomly perturbing the values of each feature across all samples in the original dataset.

Suppose the original dataset is represented as a matrix $A = |a_{ij}|_{i=1...n,j=1...k}$ where there are n chip samples and k patterns. The synthetic dataset is another matrix $B = |b_{ij}|_{i=1...n,j=1...k}$. Each column $b_{1j}, b_{2j}, \ldots, b_{nj}$ is sampled at random from the univariate distribution of $\{a_{1j}, a_{2j}, \ldots, a_{nj}\}$. Basically, the univariate distribution of each pattern feature (pattern delays) across all chips in the synthetic set is equivalent to that in the original dataset. However, the correlations, presented in the original dataset, among the pattern delays of each chip are entirely destroyed in the synthetic dataset. Breiman [29] shows that if the model M_{rf} from this synthetic binary classification dataset can successfully differentiate between the original and synthetic samples, then M_{rf} is a good model to measure a *proximity* between pairs of samples.

Refer to Figure 5 and recall that an outlier analysis algorithm starts from defining a similarity function. The model M_{rf} is for that purpose. Suppose the model consists of 100 trees $T_1, \ldots, T_{100}$. The proximity (or similarity) $prox()$ between a pair of (original) samples is measured by counting the number of trees that use the same paths to classify both samples. Suppose this number is 60, i.e. these 60 trees see the two samples as exactly the same. Then, the proximity between the two samples is 0.6.

Once the proximity matrix is constructed, a *multi-dimensional scaling* technique can be used to convert this matrix into a single dimensional outlier measure. For each sample x_i, we first calculate an average proximity, $\bar{P}$ as shown in equation (1) below, where X is the set of all samples. Let "nsample" be the number of samples in X. In equation (2), if x_i is dissimilar to most of the samples, the $\bar{P}(x_i)$ will be small, thus making $OM_{raw}(x_i)$ large. Based on all values of $OM_{raw}(x_i)$ from all samples, we find the median *Median* and also calculate their standard deviation (*STD*). The *MAD* is the *Median Absolute Deviation* defined as $0.6745 * STD$. The *outlier measure* for sample x_i, $OM(x_i)$, is then computed as equation (3).

$$\bar{P}(x_i) = \sum_{\forall k \neq i} prox^2(x_i, x_k), \forall x_k \in X \quad (1)$$

$$OM_{raw}(x_i) = nsample/\bar{P}(x_i), \forall x_i \in X \quad (2)$$

$$OM(x_i) = \frac{OM_{raw}(x_i) - Median(OM_{raw}(x))}{MAD(OM_{raw}(x))} \quad (3)$$

This process scales the high-dimensional proximity matrix into a single dimensional outlier measure. If OM(x_i) exceeds a selected threshold, x_i is classified as an outlier.

4.2 Support Vector Machine

Support Vector Machine (SVM) represents a family of algorithms that can be used with a variety of *kernel functions* [26]. Three types of SVM are available: regression, classification, and one-class. The one-class SVM is applied in this work, while the work in [4] uses the binary classification algorithm. A kernel function defines the similarity measure between pairs of samples. The most commonly-used kernels are the *Linear* (dot product) and *Gaussian* kernels.

$$Linear \quad : k(x,z) = \langle x,z \rangle = x_1 z_1 + \cdots + x_k z_k \quad (4)$$

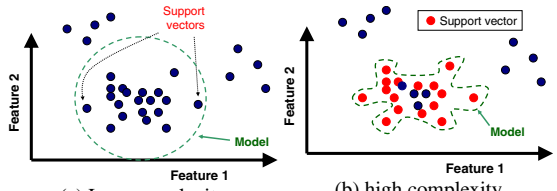
$$Gaussian \quad : k(x,z) = \exp(-\gamma ||x-z||^2) \quad (5)$$

In this work, we use the ν-SVM one class algorithm proposed in [25]. The parameter ν denotes an upper bound on the fraction of samples that should be classified as outliers. The algorithm then computes the "tightest" hypersphere that contains all sample points following a similar trend, constrained by the value ν. The hypersphere is constructed in the feature space defined by the original set of features and the kernel function. The hypersphere model is of the form:

$$M_{1svm}(x) = \sum_{\forall i} \alpha_i k(x_i, x) \quad (6)$$

where α_i characterizes the *importance* of sample x_i in building the model,and $M_{1svm}(x)$ is the outlier measure for sample x. Because $k(x_i, x)$ measures the similarity between x and a given sample x_i, we see that the model is calculating a weighted average of similarities between the sample x and all samples. These weights are decided by the importance of the samples. In the model, a sample x_i whose $\alpha_i \neq 0$, is called a *support vector* (SV). The number of support vectors

impact the shape of the model in the original feature space. For example, Figure 7 shows two example models, one with only 2 SVs and the other with many. Observe that the shape of the model in the right figure is more irregular than the shape of the model in the left. The intuition is simple: If we want to define a region that is more irregular, we need more (SV) points.



(a) Low complexity (b) high complexity

Figure 7. SVM outlier analysis illustration

To decide if a sample is an outlier, $M_{1svm}(x)$ is compared to a constant ρ. In Figure 7, the SVs decide the shape of the model while ρ determines its size.

5 Experimental Setup

To compare the three baseline approaches, we conducted controlled experiments with simulations. In simulation, we produced two sets of circuit samples, the good (or golden) set and the defect-injected set, based on an industrial custom block design. The good samples were generated using statistical Monte Carlo (MC) simulation [4, 22], where the delay values of each sample were statistically drawn from a statistical cell-based timing library. Hence, each sample had a fixed but unique delay configuration. Statistical process variation was modeled in the library and spatial correlation was taken into account in the MC simulation.

A good sample was converted into a defect-injected sample by including a random-sized randomly-located delay defect. The size was drawn from an exponential distribution with the mean selected to be relatively small with respect to the max delay of the circuit. One thing to note is that the maximum delay of a defect-injected sample can be smaller than the maximum delay of a good sample, because of the statistical variations considered in the simulation. Moreover, there may not exist a pattern to excite and observe the defect in a defect-injected sample. In our experiments, we ignore these issues and simply label a defect-injected sample as "defect-injected."

It is important to note that by doing so, we do not expect any method to perfectly identify all defect-injected samples. Hence, the experiments should focus on finding the most defect-injected samples, while limiting the number of over-kills. We are interested in knowing which of the three approaches can identify the largest number of defect-injected samples, while constrained to zero over-kills.

A 15-detect transition fault pattern set with 2165 patterns was used. Each pattern was evaluated against 10 test clocks, from slow (the 10th clock) to fast (the 1st clock), where the first failing clock was recorded. If no failure was observed, the recorded number is 0. Hence, the number of features was 2165 and each feature value was an integer between 0 and 10. We intentionally used the 15-detect set so that the number of dimensions in the outlier analysis problem was large enough, to test the effectiveness of each approach in avoiding the curse of dimensionality.

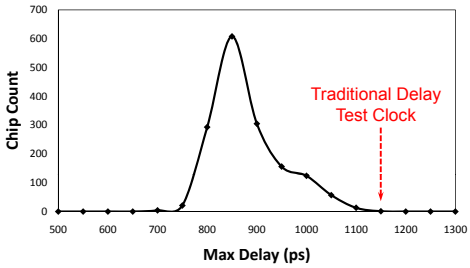


Figure 8. Max delay distribution of all samples

Figure 8 plots the max delay distribution based on all samples, good (golden) and defect-injected. We note that good samples mostly range between 650-950 ps. Defect-injected samples mostly range between 825-1200 ps. However, observe that from the distribution itself, it is hard to see that. The only obvious thing we can do is to cut off the tail of the distribution, i.e. establish a threshold as traditional delay test clock. Therefore, before running outlier analysis, obvious outliers have already been removed. After removing all samples on the (right) tail, there are 991 good samples, and 591 defect-injected samples left. These 1582 samples are the sent to the outlier analysis.

6 Baseline Experiments

We first study the performance of PCA + Euclidean distance. Based on the dataset with 2165 features, we apply PCA to select 30 and 852 principal components (PCs), which explain 75% and 99.99% of total variance in the original dataset, respectively. The two sets of PCs transform the original dataset into two new datasets. Then, the Euclidean distance based method described earlier is applied to each individually. The outlier measures for the two datasets are plotted in Figures 9 and 10. The y-axis is the scale of the outlier measure and x-axis is the circuit sample indices. Good (golden) samples are grouped to the left while defect-injected samples are grouped to the right.

In Figure 9 most samples have outlier measures ranging between 500 and 4000, with defect-injected samples having slightly higher outlier measures. The lack of vertical separation between the golden and defect-injected samples indicates that the outlier method is not effective. Even in the high outlier measure region, between 4000 and 12000, both golden and defect-injected samples are present. Hence, there is no way to draw a threshold line to screen out defect-injected samples, without involving over-kill.

Figure 10 shows a slightly improved result with more

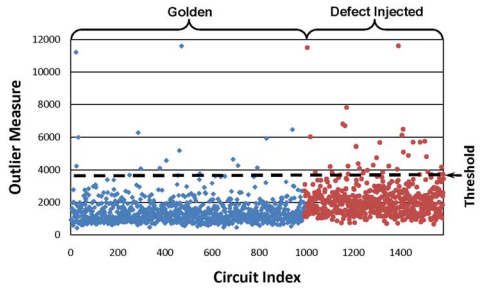


Figure 9. **Result using top 30 principal components**

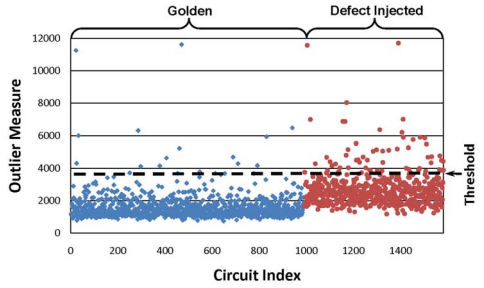


Figure 10. **Result using top 852 principal components**

PCs but the basic point stated above remains. This tells us that the Euclidean distance approach is not fundamentally feasible for outlier analysis on the dataset, regardless of the number of PCs.

6.1 Random Forests

Note that RF selects the important features during the construction of its binary classifier model based on the synthetic dataset. The RF tool we use, the *librf* package [30], calculates the optimal number of selected important features. Hence, there is no need to show different results with different numbers of features as that in the PCA experiment. Outlier measures for all samples are plotted in Figure 11.

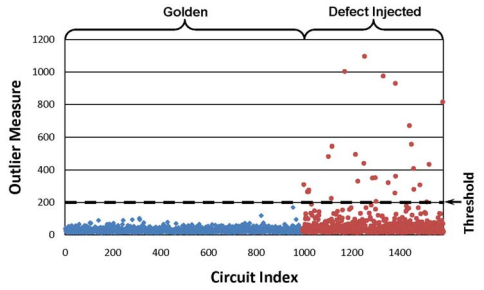


Figure 11. **Random Forests outlier measure**

Figure 11 shows better separation between golden and defect-injected samples than previous results. Most golden samples are tightly packed with outlier measures ranging between 0 and 100. Defect-injected samples spread out with outlier measures ranging from 0 to 1100. This is a desirable result because it shows that RF outlier analysis can identify some defect-injected samples as outliers, without over-killing the golden samples. In Figure 11, a horizontal threshold shown at 200 separates all golden circuits from a portion of the defect-injected ones. This threshold yields zero overkill and captures 32 defect-injected samples. While this result is more encouraging than before, it is far from satisfactory.

6.2 One-class ν-SVM

One fundamental issue in the previous two approaches is that a user needs to decide a threshold to screen outliers after the outlier measures are plotted. Such a decision can become arbitrary without further scientific evidence to support the decision.

The one-class SVM algorithm, ν-SVM, removes this concern by computing the threshold ρ automatically during the model construction process [25]. The software package we used is an modification of [27]. However, it requires the user to enter the value of a parameter ν that is the upper bound on the number of outliers identified by the model. In a way, the burden of choosing the threshold ρ is shifted to the burden on deciding ν.

Because ν is just an upper bound, it is much easier to decide its value. For example, if one expects a yield of 60% from a process, to be safe 0.5 can be used as ν. This means that in the worst case, we do not expect more than 50% of the samples under analysis to be classified as bad. We see that ν matches well with the estimate of yield. Since usually one has some idea about the yield, selecting a reasonable value for ν should be easier than selecting the threshold.

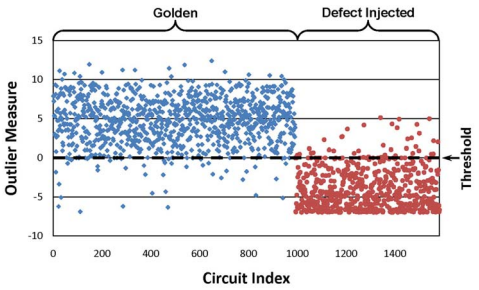


Figure 12. **SVM with Gaussian Kernal**

Figure 12 shows the result based on the Gaussian kernel. Note that the threshold line is an output of the ν-SVM and hence, is not decided after plotting the outlier measures. In particular, the algorithm scales the measures so that the threshold boundary stays at 0. Notice that this figure shows far better separation between the golden and defect-injected samples than that in the previous three plots. However, we note that because some golden samples have outlier measures similar to defect-injected samples, overkills occur.

If one desires, the threshold value ρ computed by the ν-SVM can be adjusted to achieve lower overkills or increases captures. Figure 13 presents the capture-vs-overkill tradeoff based on the outlier measures shown in Figure 12. The x-axis shows the percentage of correctly classified golden

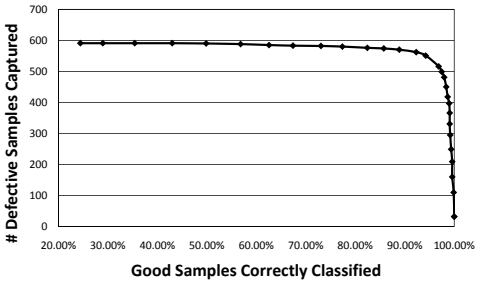


Figure 13. **Capture vs. overkill tradeoff**

samples, and the y-axis shows how many defect-injected samples are captured at that percentage. The curve shows that the Gaussian SVM model can capture most of the defect-injected samples when correctly classifying 90% of the golden samples. However when approaching a golden classification of 95%, the defect capture number plummets. At 100% golden classification, the model only screens out 37 defect-injected samples as outliers.

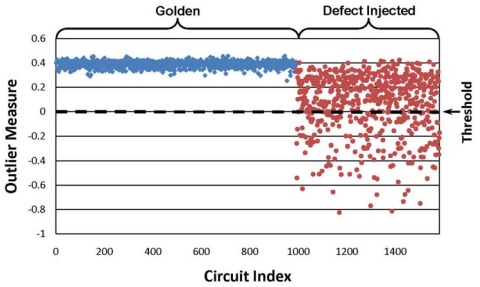


Figure 14. **SVM with Linear Kernal**

Figure 14 shows the result when using Linear kernel. A clear improvement can be seen in the spread of outlier measures for the defect-injected samples. They range between 0.4 and -0.8, while all golden measures are tightly packed around 0.4. It outperforms SVM with the Gaussian kernal, and is capable of capturing 194 defect-injected samples that fall below the 0 threshold, with no overkills.

6.3 Summary

Three baseline outlier analysis approaches are compared each with a unique definition of similarity measure. From the results presented above, one-class ν-SVM using the Linear kernal is the most effective method, which keeps the golden samples tightly packed while spreading out the defect-injected ones. Table 1 summarizes the results for the three approaches with a slightly different perspective than the outlier measure plots. For zero overkill the best model is given by SVM using the Linear kernal. However, if 5% overkill is acceptable, then using the Gaussian kernel or the Linear kernel become more comparable, both capturing more than 90% of the defect-injected samples.

7 Improvement to ν-SVM

We take the best method from the previous section, the ν-SVM with Linear kernel, and in this section we try to

	% Defective Detected	Overkill %
Euclidean Distance + PCA	0% (0/591)	0%
	37.20% (220/591)	5%
Random Forrest	5.41% (32/591)	0%
	33.50% (198/591)	5%
SVM + Gaussian	6.26% (37/591)	0%
	91.54% (541/591)	5%
SVM + Linear	32.83% (194/591)	0%
	93.40% (552/591)	5%

Table 1. **Comparison of various techniques**

improve its performance by following three strategies: (1) We replace Linear kernel with a more advanced kernel, the Polynomial kernel. (2) We reduce the dimension of the problem by applying PCA first. (3) We select important features using RF before running the SVM. We will conclude with experimental results that only the first strategy makes sense. This demonstrates that the dimensionality of the problem is unimportant for ν-SVM. Using the right kernel function is a consideration for improving its performance. The Polynomial kernel we use looks like the following:

$$k(x,z) = \left(\frac{\langle x,z\rangle}{||x||||z||} + 1\right)^d \quad (7)$$

Note that $\frac{\langle x,z\rangle}{||x||||z||}$ is the cosine function described earlier. We use 3-fold cross validation method discussed earlier to select the value for d. In the experiments below, $d = 2$. Figure 15 shows the result. Again, the threshold line was chosen by the SVM, not arbitrarily decided.

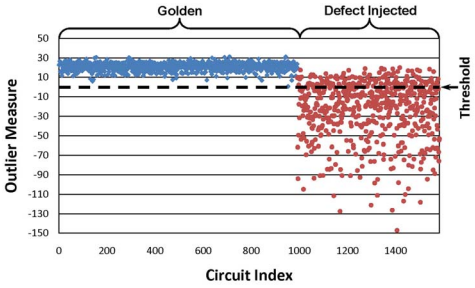


Figure 15. **SVM with polynomial kernal**

The model successfully identifies 441 defect-injected samples with zero overkill. This significantly improves the result from using the Linear kernel where only 194 defect-injected samples were screened as outliers.

7.1 PCA and RF dimensionality reduction

A common approach in support vector analysis is to use PCA to transform a given dataset into a lower-dimensional dataset before applying SVM [28]. Figure 16 shows the result of applying one-class SVM on the reduced data set using 30 PCs. Immediately it is clear that PCA has destroyed all separation between golden and defect-injected samples. It is impossible to differentiate the two classes of samples.

RF analysis was shown to be effective in selecting the important features in delay test applications [22]. By apply-

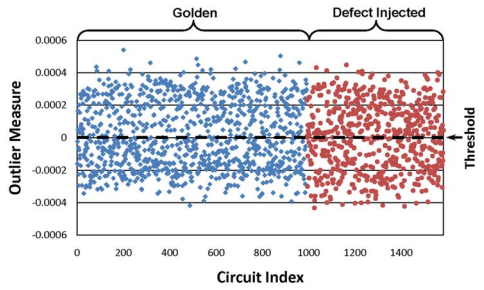


Figure 16. PCA + SVM + Linear kernel

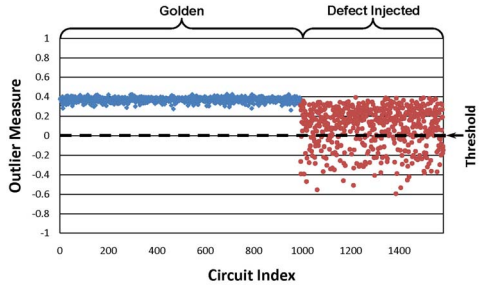


Figure 17. RF + SVM

ing RF on the synthetic dataset, it selected 286 important features in building the 100-tree forest model. Based on these 286 features, we constructed a new dataset by removing all other features. Then SVM is applied on the reduced dataset. Figure 17 shows the result. Based on the given threshold at 0, 160 defect-injected samples are identified as outliers with zero overkill. Comparing this number to the original number 194, we see that selecting importance features actually hurts the performance of the one-class SVM.

Results in this section clearly demonstrate that, with the ν-SVM outlier analysis approach, using a proper kernel function is more important than reducing the dimensionality of the problem. In fact, both PCA and RF, dimension reduction hurts SVM's performance. This conclusion is opposite to that presented in [28]. We conjecture that this is due to the unique nature of our problem, i.e. in delay testing, the behavior of an individual pattern on a chip can mean a lot (in statistical analysis, individual behavior is usually not important). Hence, when individual features are replaced with PCs (which are linear combinations of the original features), behavior of an individual feature can be masked. We note that, a similar situation was discussed in [4] before, where a statistical feature selection approach is shown to be ineffective for pattern selection in the context of delay test set optimization, due to the masking of individual behavior on important patterns.

8 Polynomial Kernel Model Validation

From a delay testing standpoint, the best outlier method to use is the one-class SVM model generated with the polynomial kernel. To make sure this model is feasible for

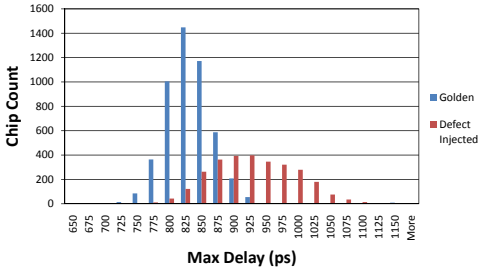


Figure 18. Max delay histogram of validation dataset

large scale classification a validation dataset was analyzed. The validation set contains 4961 golden samples, and 2852 defect-injected samples. The max delay distributions of the validation set can be seen in Figure 18, showing a clear overlap between golden and defect-injected samples.

In order to show the flexibility of one-class SVM, the original dataset was used to produce three new datasets, each with a different golden-to-defective sample ratio. These three datasets are used to train one-class SVM to model. The intent is to demonstrate that one-class SVM and the polynomial kernel train generalized models independent of the exact yield.

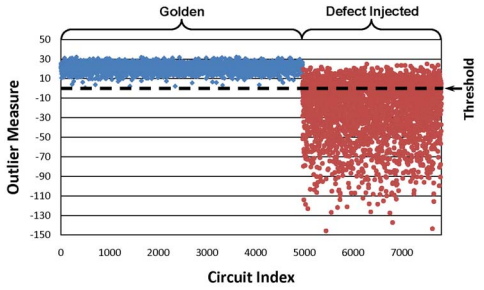


Figure 19. Validation result with the large dataset using the SVM model built on the dataset containing 90% golden and 10% defect-injected samples

Figure 19 shows the validation result from the model trained with 90% golden and 10% defect-injected samples. A zero threshold successfully separates golden samples from most of the defect-injected ones.

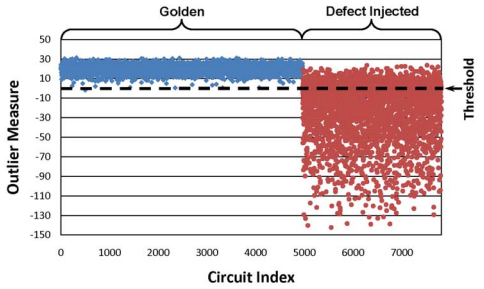


Figure 20. Similar validation result (training with 63% golden, 37% defect-injected)

Figure 20 shows a similar result based on training with 63% defect-injected and 37% golden samples. Figure 21 shows the validation result using the model trained with 50% golden and 50% defective. Table 2 summarizes the validation results from all three models. It is interesting to note that as more good samples are used to build the model, the model boundary for good is tighter. As a result, when using the model to screen the large dataset, it is less likely that some good samples are classified as bad. In general, the three models perform similarly and the performance of SVM outlier analysis is mostly independent of the yield in the original training dataset.

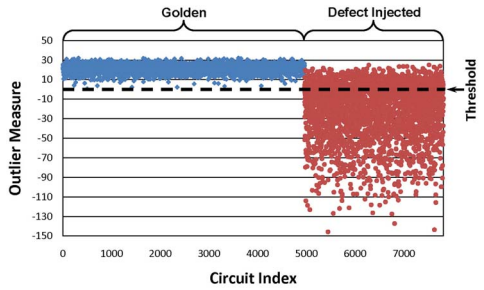


Figure 21. Similar validation result (training with 50% golden and 50% defect-injected)

Model Training Set (% Golden / % Defect-injected)	# Defect-injected samples Detected	# overkills
90/10	2292	4
63/37	2182	1
50/50	2043	0

Table 2. Comparison of using the three SVM models

9 Conclusion

In this work, we provide an overview of various outlier analysis techniques and compare their performance in the delay test application. As the difficulty of establishing a golden reference in delay testing increases due to modeling uncertainty and the presence of small delay defects, the outlier analysis approach becomes an attractive alternative. Utilizing outlier analysis avoids the dependency on the accuracy of pre-silicon timing models and analysis as well as the dependency on having known-good and known bad dies. After studying a variety of techniques, we conclude that one-class SVM with a properly chosen kernel function is the best approach, which is able to screen out a majority of small-defect-injected chips without incurring any overkill. One-Class SVM's ability to minimize overkills while detecting the majority of defective samples under varying training conditions makes it an ideal candidate for applications in delay testing.

References

[1] P. Bastani et, al., "'Analyzing the risk of timing modeling based on path delay tests"' *ITC*, 2007

[2] Li-C. Wang, Pouria Bastani, Magdy S. Abadir, "Design-silicon timing correlation — a data mining perspective," In *DAC* 2007.

[3] 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.

[4] B. Lee, Li-C. Wang, and M. Abadir, "'Issues on Test Optimization with Known Good Dies and Known Defective Dies - A Statistical Perspective"', *ITC*, 2006

[5] A. Gattiker, W. Maly, "'Current Signature: Application"', *ITC*, 1997, pp. 156-165.

[6] P. Nigh and A. Gattiker, "'Random and Systematic Defect Analysis Using Iddq Signature Analysis for Understand Fails and Guiding Test Decisions"', *ITC*, 2004

[7] P. Maxwell et al., "'Current Ratios: A Self-scaling Technique for Production Iddq Testing,"' *ITC*, 1999, pp.738-746

[8] S. Sabade and D. Walker, "'Use of Multiple Iddq Test Metrics for Outlier Identification"', "' VTS"', 2003

[9] J. Roehr, "'Very-Low Voltage (VLV) and VLV Ratio (VLVR) Testing for Quality, Reliability, and Outlier Detection"', *ITC*, 2006

[10] S.Sabade and D. Walker, "'Evaluation of Effectiveness of Median of Absolute Deviations Outlier Rejection-based Iddq Testing for Burn-in Reduction"', *VTS*, 2002

[11] P. Buxton and P. Tabor, "'Outlier Detection for DPPM Reduction"', "'ITC"', 2003

[12] C. Schuermyer et. al, "'Screening VDSM Outlier using Nominal and Subthreshold Supply Voltage Iddq"', *ITC*, 2003

[13] Huisman et al. "'Data Mining Integrated Circuit Fails with Fail Commonalities"', *ITC* 2004

[14] S. Jandhyala, et al. "Clustering Based Techniques for IDDQ Testing," (*ITC*, 1999, pp. 730-737.

[15] R. Daasch and R. Madge, "'Variance Reduction and Outliers: Statistical Analysis of Semiconductor Test Data"', *ITC*, 2005

[16] R. Daasch and R. Madge, "'Data-Driven Models for Statistical Testing: Measurements, Estimates and Residuals"', *ITC*, 2005

[17] R. Madge et al. "'The Value of Statistical Testing for Quality, Yield and Test Cost Improvement"', *ITC*, 2005

[18] A. Nahar et al. "'Burn-in Reduction using Principal Component Analysis"', *ITC*, 2005

[19] K. Butler et. al., "'Successful Development and Implementation of Statistical Outlier Techniques on 90nm and 65nm Process Driver Device"', *IRPS*, 2006

[20] L. Fang, M Lemnawar, and Y. Xing, "'Cost Effective Outliers Screening with Moving Limits and Correlation Testing for Analogue ICs"' *ITC*, 2006

[21] Jeffrey Roehr, "'Measurement Ratio Testing for Improved Quality and Outlier Detection"' *ITC*, 2007

[22] S. Wu et, al. "'Statistical Analysis and Optimization of Parametric Delay Test"', *ITC*, 2007

[23] I.T. Jolliffe, Principal Component Analysis. Springer, 1986

[24] Vladimir N. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer 1999.

[25] B Scholkoph et al., "'Estimating the support of a high-dimensional distribution"', *Neural Computation* 13, 2001, pp. 1443-1471.

[26] Nello Cristianini and John Shawe-Taylor. An Introduction to Support Vector Machine. Cambridge University Press, 2002.

[27] C. Chang and C. Lin. LIBSVM: a library for support vector machines. 2001, Software available at http://www.csie.ntu.edu.tw/čjlin/libsvm.

[28] J. Jin, X. Wang, and B. Wang "'Classification of Direction perception EEG Based on PCA-SVM"' *ICNC*, 2007

[29] Breiman, Leo, "'Random Forests"'. *Machine Learning (45) 1*, pp. 5-32, 2001.

[30] Benjamin N Lee, libRF: a library for Random Forests, 2007. Software available at http://mtv.ece.ucsb.edu/benlee/librf.html