Issues on Test Optimization with Known Good Dies and Known Defective Dies — A Statistical Perspective*

Benjamin N. Lee, Li-C. Wang
Department of ECE, UC-Santa Barbara
benlee, licwang@ece.ucsb.edu

Magdy S. Abadir
Freescale Semiconductor, Inc.
m.abadir@freescale.com

Abstract

As the timing behavior of the good and defective chips become statistical, the traditional notion that there exists a one-dimensional timing boundary to separate the good and defective behavior may no longer be true. This paper studies issues in test optimization for screening statistical delay defects. After the first silicon tapeout, test data learning based on silicon samples can be utilized to optimize the test set for mass production. This approach depends on the availability of known good and known defective samples. This paper focuses the discussion on silicon sample based test optimization. We relate this problem to binary classification and pattern selection to the feature selection problem in statistical learning. Experimental results are presented to explain the methodologies and the new concepts.

1 Introduction

The objective of test optimization is to minimize the test effort while maximizing the defect coverage. This paper studies the problem of test optimization in the context of applying scan patterns for screening statistical delay defects. A more general concept of test optimization is *adaptive test* [1, 2]. While this work focuses on optimization based on only one type of test set with one test delivery method, adaptive test intends to achieve optimal result across all test sets and all test application methods.

With our narrower focus, test optimization can be thought of minimizing the test set while maximizing the defect coverage. Historically, this problem has been well studied with respect to logic faults and defects. For example, in [3, 4], the authors propose a defective part level prediction model called MPG-D. The optimization flow can begin with a superset of patterns such as an n-detect stuck-at fault set S_n. The MPG-D is used as a *reference model* to select a subset of patterns from S_n. The selection objective is that the predicted defect level by MPG-D based on this subset of patterns, is kept the same as that given by original pattern set S_n, while the size of this subset is minimized.

MPG-D predicts the defect level of a test set T based on three things: two parameters τ, A and a *coverage vector* $\vec{C} = [c_1, c_2, \ldots, c_k]$ where the assumption is that there are k potential defective sites (such as k nets). Each c_i is a coverage measured on the site i by simulating T. This coverage is *observation-based*, meaning that it is based on counting how many times a particular site is observed by the pattern set T [3].

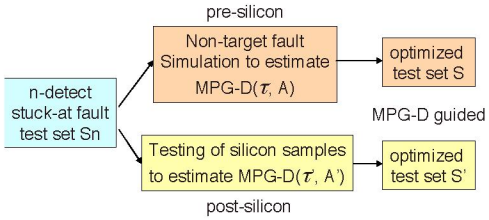


Figure 1. Test optimization based on the defective part level prediction model MPG-D [4] w.r.t. logic defects

In essence, an MPG-D model can be re-written as MPG-D$(\tau, A, \vec{C})$ where $\vec{C}$ is simulation-obtainable. The the actual values of the two parameters τ, A can be estimated either based on a non-target fault simulation (such as a bridging fault simulation) or based on applying S_n to a collection of known good and defective sample chips. The authors demonstrate that by allowing two degrees of freedom in the defective part level prediction model, MPG-D can accurately model the defect detection behavior and hence, is a good model to guide the test set optimization.

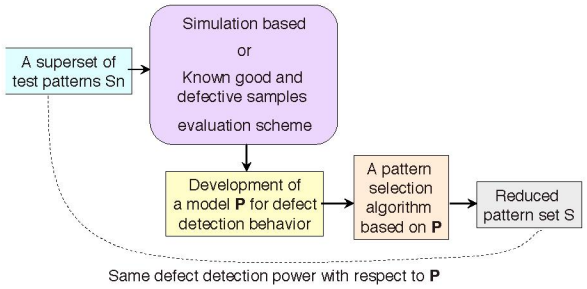


Figure 2. Test set optimization methodology

Figure 2 summarizes the test set optimization methodol-

*This work was supported in part by National Science Foundation, Grant No. 0312701and Semiconductor Research Corporation, project 2004-TJ-1173.

ogy just described. This methodology consists of four basic components: (1) a superset S_n of patterns to begin with, (2) a test set quality evaluation scheme that can be either simulation based or silicon samples based, (3) a model P for modeling the defect detection behavior by a pattern set, and (4) a pattern selection algorithm based on P. The quality of the selected set S, as predicted by the model P, should be kept the same as that given by S_n. The size of S should be minimized. Hence, the methodology defines a *constrained minimization* problem.

Inspired by the work in [3, 4], the work in [5] extends the analysis to consider statistical delay defects. Instead of using an n-detect stack-at fault test set, the optimization begins with an n-detect transition fault test set. Essentially, the authors demonstrate that given a set of good and defective sample chips, if there exists a one-dimensional separation boundary between them (Figure 3-(a)), then a defective part level prediction model that was intended for modeling logic defect detection (such as MPG-D), may still be used for guiding the test optimization. What remains unclear is what if such a one-dimensional boundary does not exist.

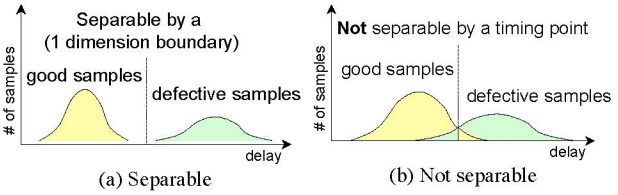


Figure 3. **Illustration of a one-dimensional timing separable boundary between good and defective samples**

Suppose that by applying a variety of different thorough testing methods (Iddq, low voltage, high voltage, functional test, etc.), we obtain a set of known good samples and a set of known defective samples. If we plot the worst-case delays (under scan with normal test environmental condition) of the good and the defective samples based on a pattern set, the two histograms may overlap, as illustrated in Figure 3-(b). In this case, we will need a *multi-dimensional boundary* in order to separate them.

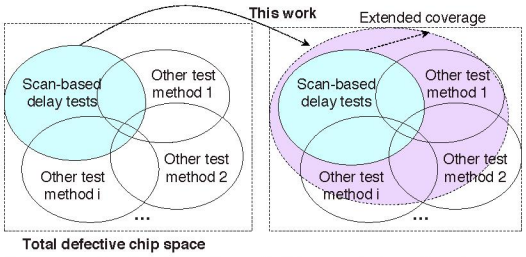


Figure 4. **Application of this work in adaptive test**

The use of a multi-dimensional boundary allows the scan-based delay tests to provide higher screening capability on the defective chips. Putting this into the context of adaptive tests, Figure 4 then illustrates a potential application of this work. In other words, by extending the coverage of scan-based delay tests, this work can facilitate the global test optimization in adaptive test.

1.1 Use of a binary classifier

Finding a multi-dimensional boundary to separate two statistical distributions in general is the problem of *binary classification* that has been studied extensively in the statistical learning literature [6]. Therefore, it is natural to relate the delay testing problem in Figure 3-(b) to the problem of binary classification.

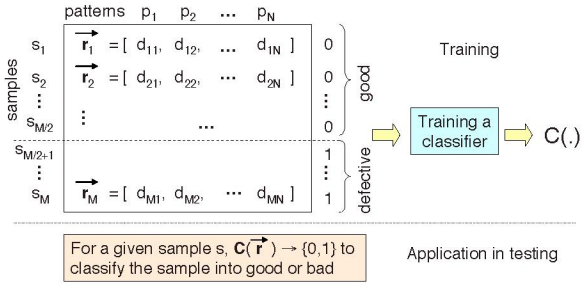


Figure 5. **Training a binary classifier for delay testing**

Figure 5 illustrates the binary classification problem in the context of learning from the test data based on known good and known defective samples. Without loss of generality, we assume that there are $\frac{M}{2}$ good samples and $\frac{M}{2}$ defective samples. We assume that there are N patterns. For each sample s_i, let $\vec{r}_i$ denote the test results $[d_{i1}, d_{i2}, \ldots, d_{iN}]$ by applying the N patterns on the sample. In the training phase, we apply a learning algorithm [6] to train a binary classifier $C()$. In the application phase, for each chip s which does not appear in the original sample set, we obtain its test result r. Then, $C(r)$ is used to classify s as good or defective, i.e. $C(r) = 0$ means good and $C(r) = 1$ means defective. The *learning error* is defined as the probability that $C(r)$ miss-classifies a given sample s. The *learning accuracy* (or *generalization accuracy*) is the probability that $C(r)$ correctly classifies s. This accuracy can also be thought as the *screening accuracy* in the context of testing.

By formulating the problem into such a binary classification problem, test set optimization involves three issues:

Test data How to define d_{ij} in order to allow more screening resolution (more classification power)? In statistical learning, this is the problem of *feature generation* [6]. For example, the simplest case can be that each d_{ij} is a value $\in \{0, 1\}$. We use a given test clock and check if pattern p_j on sample s_i result in a delay greater than this clock. If it does, $d_{ij} = 1$ and otherwise, $d_{ij} = 0$. In order to provide higher screening capability, we may apply a sequence of clocks $clk_0 < clk_1 < \cdots < clk_k$ such that d_{ij} is the first failing clock.

Pattern set reduction How can we reduce the number of patterns without affecting the screening power of the

resulting classifier? This is the problem of *feature selection* [6].

Sample size How do we know that the collection of the samples we have used in the training is enough? This is the problem of *generalization* of the learning model [11].

Refer back to Figure 2, what we have discussed so far is to utilize the binary classifier $C()$ as the model P in silicon samples based test set optimization, i.e. minimizing the test set without sacrificing the classification power. Essentially, $C()$ becomes the *reference model* as MPG-D does in Figure 1 before,

1.2 Use of a statistical timing analyzer

If the optimization is carried out in the pre-silicon stage, an intuitive approach would be to replace the non-target fault simulator in Figures 1 and 2 with a statistical delay defect simulator. However, as pointed out in [7], statistical delay defect simulation may be impractical due to its high cost. And for pattern selection, we do not necessarily need a defect simulator. We can apply a statistical timing simulator and use the concept of *timing slack* on every site to select patterns [7].

The work in [7] used a Monte Carlo statistical timing simulator for pattern selection, which is not efficient for practical use. Hence, in an earlier work [8], an efficient hazard-aware pattern-based statistical timing analyzer (PB-STA) was implemented. The PB-STA was applied to select an optimal pattern set from the 15-detect transition fault pattern set. The selection was based on a *minimum-timing-slack* principle [8]. A key issue in the PB-STA is the lack of a reliable statistical timing model to begin with.

In a recent work [9], the authors propose a Bayesian learning framework where systematic delay correlations can be learned from path delay testing and these learned correlations can be used to refine the statistical timing model for more accurate statistical static timing analysis (SSTA). The same idea can be applied to PB-STA because the PB-STA utilizes SSTA as its core engine [8].

If we combine PB-STA and the delay parameter learning methodology in [9], we obtain a simulation based test set optimization framework. An interesting question to look into is: How the pattern selection is impacted by the improvement in the accuracy of the statistical timing model, i.e. would the test sets selected before and after the Bayesian parameter learning [9] be very different? We will come back to this question in Section 7.

1.3 Overview

Figure 6 summarizes the two test optimization approaches described above. In this paper, we focus the discussion on the silicon samples based approach because

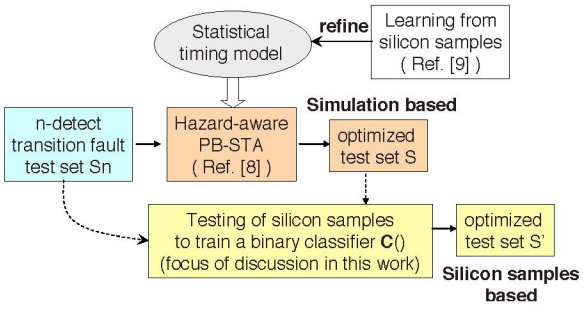


Figure 6. Simulation based and silicon samples based test set optimization in statistical timing domain

key issues in the simulation based approach have been addressed in [8, 9]. The optimization begins with a large n-detect transition fault test set. Note that this is arbitrary and can be replaced with any other test set intended for delay testing, ex. [10]. The requirement is that this initial set has to be a superset. This is because the subsequent steps only perform test set reduction, not test set enhancement.

The rest of the paper is organized as the following. Section 2 reviews the background and prior related work. Section 3 describes our experimental framework and present results to motivate this work. In Section 4, we discuss binary classifier for screening defective samples. We utilize *Support Vector Machines* (SVM) [11] as our learning algorithm. We relate the characteristics of the *support vectors* in SVM to the generalization issue for screening *unseen* chips in mass production. Section 5 discusses the feature selection problem for test set optimization. Section 6 summarizes the experiments for the silicon samples based approach. In section 7, we briefly review the simulation based pattern selection approach in [8] and discuss how the accuracy in the timing model may impact the selection. Section 8 concludes the paper.

2 Background and related work

Delay defects are of growing concern with deep submicron process technologies. Effective screening of delay defects often requires a combination of parametric and non-parametric testing methods [12]. Past studies have demonstrated that the problem of drawing a reliable boundary between the good and the defective chips in delay testing is inherently statistical [13].

Tests are optimized to detect defects, not faults whose primary purpose is to guide ATPG. Because true defects can only be seen from actual silicon test data, when preparing the initial test set at the pre-silicon design phase, we often want to construct a "superset" in order to ensure the inclusion of all tests required to detect a wide variety of potential defects. A typical way to construct such a superset is by employing some sort of randomness in the ATPG process. For example, the n-detect test set [14] and the k-longest path delay test set [10] are two notable examples of this sort. How-

ever, these test sets may contain patterns that are redundant and/or ineffective with respect to detecting the actual defects.

Optimization on the initial test set obtained from the ATPG can be approached by two angles: (1) Because the ATPG is timing independent, a pattern may be redundant or ineffective if we consider timing. (2) In preparing for the initial test set, we have to assume the worst for the defects. After seeing the initial test data, we may discover that some patterns are not needed because they do not provide additional defect screening power.

The idea of beginning with a superset and then optimizing based on actual test data, is a popular concept in analog testing [15]. In analog testing, we often begin with a large number of tests and measurements in order to observe the maximal screening resolution. Then, for mass production, a reduced set is selected such that the same screening resolution is maintained from a statistical sense. It is recognized that separating the good and the defective chips in analog testing may require a multi-dimensional (and non-linear) boundary. For example, the recent work in [16] develops a neural network based classifier to achieve that boundary.

As mentioned before, *adaptive test* [1, 2] is a general concept of test optimization. However, the idea of "learn-then-optimize" can also be a fundamental principle in adaptive test [2]. For example, researchers have started to investigate the potential of applying statistical learning and data mining techniques to optimize the production test flow [17].

On the design side, statistical static timing analysis (SSTA) has been a popular research topic in recent years [18]-[19]. If we consider test patterns, SSTA becomes pattern-based statistical timing analyzer (PB-STA) [8, 20]. PB-STA estimates the variations reflected in the pattern delays and reports these delays as random variables in terms of their means and standard deviations. The statistical analysis relies on a statistical timing model that captures both die-to-die and with-in-die variations at different voltage and temperature corners [8]. PB-STA can also assess the uncertainties in pattern delays so that non-robust patterns [21] can be discarded before moving into the first silicon stage [8].

3 Experimental methodologies

Our study is simulation based. The advantage of simulation is that we can have a complete control on the statistical systems to produce the good and the defective behavior. This control facilitates the study to answer more in-depth questions.

Because our research results are based on the assumptions employed in the simulation, it is crucial to devise an experimental framework that is reasonable to reflect the complexity of the problems to be studied. The simulation is Monte Carlo (MC) based where n samples, whose delay values are statistically drawn from a statistical timing model (STM), are simulated in each run. Hence, each sample is with a fixed but different delay configuration.

To allow efficient simulation, our STMs are *cell-based*. On each cell, the pin-to-pin delay is a function of four variables: input slew, output load, Vdd, and temperature. This is quite standard in the industrial static timing analysis practice. Because the models are statistical, each pin-to-pin delay is a random variable. We assume Gaussian random delay variables so that only delay means and their standard deviations are required to be recorded (rather than recording the actual probability density functions). There are three statistical timing models (STM) considered in our simulation framework, as illustrated in Figure 7:

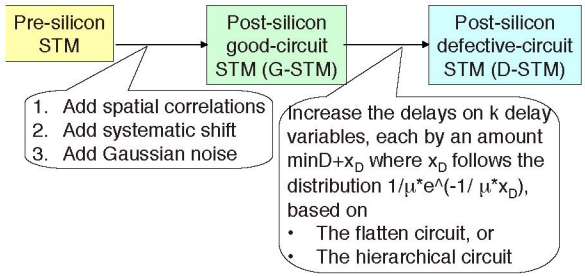


Figure 7. Statistical timing models used in our study

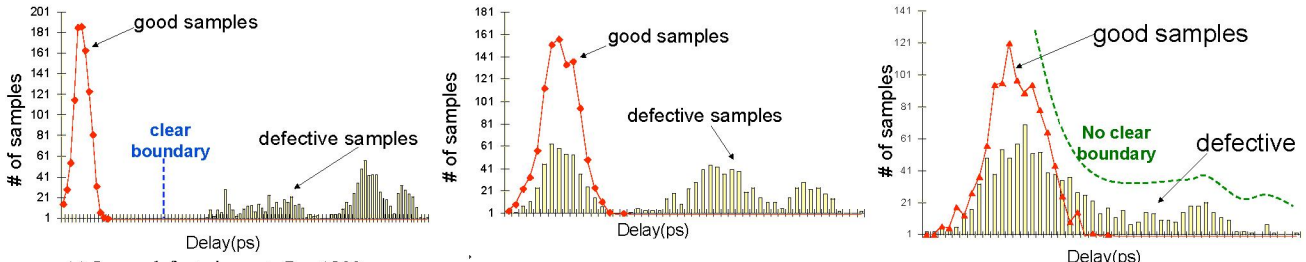
Pre-silicon STM (P-STM): This STM was characterized through Monte Carlo SPICE simulation based on a 90nm technology file [20]. This STM is what we have in the pre-silicon analysis, before seeing the test data from simulated silicon samples. The STM supports typical logic gates up to four inputs.

Post-silicon good STM (G-STM): We assume that the P-STM is inherently inaccurate [22]. This inaccuracy can be due to three common reasons:

1. *Spatial delay correlations* among pin-to-pin delays are often hard to characterize in process characterization [22]. These correlations are due to systematic process variations or layout dependent variations [23]. These correlations significantly impact the circuit worst-case performance [19]. In the G-STM, a distance-function based grid model on with-in-die spatial delay correlations is employed [9]. This model is unknown to the P-STM.

2. The SPICE model is often developed in parallel to process development. Due to the complexity involved in test chips and process characterization, this model is not updated often as process becomes matured. We model this discrepancy by allowing a systematic reduction in the delay standard deviations. This is to say that as process becomes more matured, the uncertainty in delay parameters become smaller.

3. In statistical modeling, a common approach is the Response Surface Modeling (RSM) method [24]. For a delay variable, there is inherently an error associated



(a) Large defect size, $minD$ =1800ps (b) Medium defect size, $minD$ =600ps (c) Small defect size, $minD$ =300ps

Figure 8. Small defect sizes blur the separation boundary between the good and the defective samples

with the modeled delay. The RSM method is to ensure that this error is a random error (in contrast to a systematic error). In the G-STM, we therefore introduce a Gaussian noise (mean 0, standard deviation 3%) to every modeled delay random variable.

Post-silicon defect-based STM (D-STM): If we extend the G-STM to include a statistical defect model, then we obtain a defect-based STM that allows the Monte Carlo simulation to produce defective samples. Our statistical defect model contains several parameters that can be changed to alter the defect assumption: (1) We can inject k_D defects randomly on k_D pin-to-pin delays. (2) Each defect size is $minD + x_D$ where x_D is sampled from a random exponential distribution model [25]: $(1/\mu)e^{-(1/\mu)x_D}$ and $minD$ is the minimum size. The parameter μ can be thought as the average size increase from the minimum. We usually set μ equal to a third of the maximum circuit delay on 1000 Monte Carlo samples in G-STM based simulation.

We assume that each pin-to-pin delay on a flattened circuit has an equal probability to receive a delay defect and a defect always increases the delay.

4 Binary Classifier - SVM

To illustrate the binary classification problem discussed in Figure 3, Figure 8 shows three simulation results based on an industrial custom design Ind32. The simulation is based on a 15-detect transition fault pattern set consisting of 2165 patterns . For each sample, we plot its maximum delay observed during the simulation of all these patterns. The single-defect assumption is used in the D-STM, i.e. $k_D = 1$. In this experiment, the good samples are simulated samples based on G-STM and the defective samples are simulated samples based on D-STM.

In plot (a), a large minimum defect size $minD = 1800ps$ is used. In this case, we see that there exists a clear timing boundary to separate the good samples from the defective samples. This is not true for plots (b) and (c). Notice that in plot (c), the transition from the good distribution to the defective distribution is smooth and hence, there does not exist a clear way to divide the "aggregate" distribution into two parts.

The two distributions in plot (b) and plot (c) can be separated by training a binary classifier as illustrated in Figure 5 before. One popular classification algorithm in recent years is Support Vector Machine (SVM) learning [11]. SVM is popular because of its theoretical proven properties in generalization of its learning results based on the VC dimension theory.

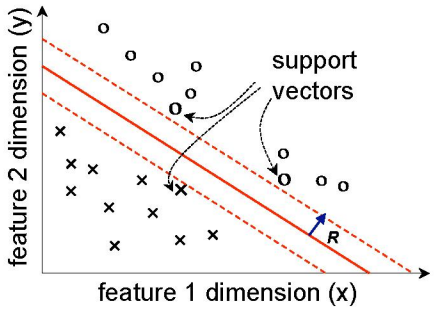


Figure 9. A 2-dimension linear classifier [11], page 97

Figure 9 depicts a simple example to help understand the basic principle of SVM. In this example, the two classes, "×" and "o", cannot be separated based on either the x dimension (1st feature) or the y dimension (2nd feature) alone. However, on the 2-dimension plane, the two classes are separable by a line. In general, given a set of 2-class samples, depending on how the "×" and "o" samples locate, they may not be separable by a line on the 2-dimension plane. However, in theory, if we go into a higher dimension (by adding more features), there may exist a smallest n such that on the n-dimension space, we can find an $n-1$ dimension *hyperplane* that separates the two classes of samples [6].

One of the simplest SVM algorithms is the *maximal margin classifier* [11]. The algorithm tries to find the separation boundary that maximizes the *separation margin* R as shown in the figure. SVM is a *kernel* method [11] which means that it utilize a pre-defined *kernel* to map the original features (pattern delays) into *kernel-induced features*. Then, in this kernel-induced feature space, a linearly separable boundary is searched.

Only the samples closest to the separation boundary are required to decide the boundary. In SVM, they become the *support vectors* (support samples) of the classifier. One can

think of the number of the support vectors as a measure of the complexity of the learned model. Another way to look at it is that, we can discard all non-support-vector samples in the training data, and the learning accuracy would still be same.

Table 1. Three-fold cross validation SVM experiments

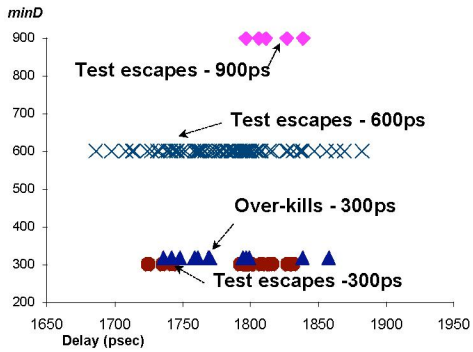
minD	avg. SVs	avg. (screening) accuracy	1-clock
300ps	394.3	94.36%	23.3%
600ps	304.7	92.53%	59.2%
900ps	186	98.26%	75.5%
1800ps	84.3	100%	100%

Table 1 presents results based on three-fold cross-validation SVM experiments [26]. In these experiments, we use pattern delays as the feature values, i.e. d_{ij} is the pattern delay in Figure 5. In each case, there are 1000 good and 1000 defective samples. We divide these samples into three sets. In each run, two sets are used for training the SVM and the other one is used for measuring the learning (screening) accuracy. In each case, the table reports the average number of support vectors and the average accuracy from the three experiments . The "**1-clock model**" is the accuracy if we simply find the best point along the timing axis (in Figure 8) to separate the good and the defective distributions, i.e. it corresponds to the traditional way of delay testing.

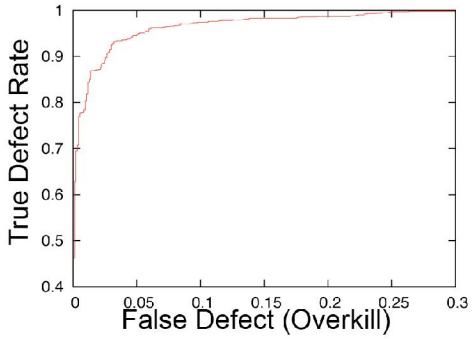
The number of support vectors increases as the defect size decreases. From Figure 8, we know that it is harder to separate the two distributions in the 300ps case than the other cases. As a result, we see that it has the largest number of support vectors, which means the SVM model is more complex.

Figure 10-(a) plots the delays of the miss-classified samples. A test escape is a defective sample classified as a good sample. An overkill is a good sample classified as a defective sample. Notice that all test escapes have delays are within the range of the good delay distribution (which is $[1650, 1933]$). Take the 600ps case as an example. From Figure 8-(b) before, we see that there are more samples in the good delay distribution range $[1650, 1933]$ than those outside. From a learning point of view, it is easier to decide those samples outside the range and harder to decide for the samples within the range. On the other hand, in the 300ps case(Figure 8-(c)), we see that many defective samples having delays similar to good sample delays. As a result, overkills happens. Note that these are the results of a classifier that is optimizing overall accuracy with no-preference for minimizing test-escapes or overkill.

In SVM, we can adjust the *decision threshold* to explore the trade-off between test-escapes and overkill with a *receiver operating characteristics* (ROC) plot [28] as shown in Fig 10-(b). From this plot, observe that if we want no test-escapes (100% true defect rate), there will be 30% overkill. Due to limited space, for the rest of the paper we will focus



(a) Distributions from miss-classified samples



(b) ROC curve for 300 ps case

Figure 10. Analysis of SVM Classifier Results

on optimizing overall learning accuracy although we note that it is possible to use a ROC plot to select an economically optimal trade-off.

4.1 Sample size Vs. the number of support vectors

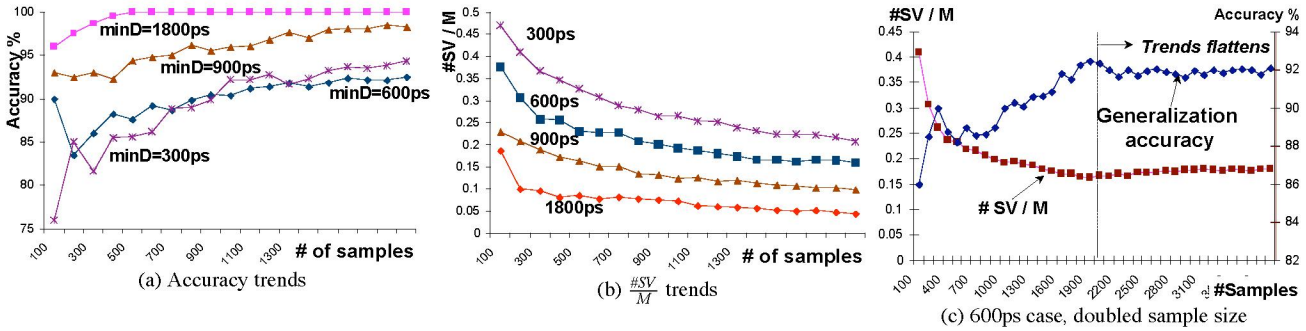
In silicon samples based test optimization, one critical step is to judge if the samples given to us are sufficient or not. A common way to decide this is to observe that adding more samples does not improve learning accuracy further. Obviously, the result depends on the power of the learning algorithm and hence, we would like to pick the best learning algorithm known to us.

A key aspect of SVM is it's strong connection to VC theory. The fewer support vectors, the greater generalization can be expected. The average generalization error $error_{avg}$ of SVM is bounded by [11]:

$$error_{avg} \leq \frac{\#SV}{M} \quad (1)$$

where $\#SV$ is the number of support vectors and M is the sample size. If the learning is effective, the number of support vectors is much less than the number of training samples and the growth on the number of support vectors is much slower than the growth of the sample size.

When the growth of $\#SV$ is much slower than the growth of M (as we continuously increase M), we see that asymptotically, $\frac{\#SV}{M} \rightarrow 0$. One thing to note is that $\frac{\#SV}{M}$ is a theoretical bound on the generalization error and is a very loose



(a) Accuracy trends (b) $\frac{\#SV}{M}$ trends (c) 600ps case, doubled sample size

Figure 11. **Accuracy trends and $\frac{\#SV}{M}$ trends (in three-fold cross-validation experiments) as sample size grows**

bound [11]. Hence, in practice, we do not expect to observe that $\frac{\#SV}{M} \rightarrow 0$.

Figure 11 shows the trends of accuracy and the quantity $\frac{\#SV}{M}$ as the sample size grow. In plot (a), for the easy cases, 1800ps and 900ps, we observe that the accuracy percentages flatten out at some points. In plot (b), we see that $\frac{\#SV}{M}$ decreases as sample size increases. In these plots, it is hard to decide that the sample size 2000 is enough. In plot (c), we double this size to 4000 and show the 600ps case. We can clearly observe that both trends flatten out at about 2000. In this case, we know that 2000 is enough – adding more samples does not help SVM to develop a better classifier.

In a typical statistical learning experiment, the accuracy trend is perhaps the only reference to decide on the sample size. For SVM, the trend of $\frac{\#SV}{M}$ can aid this decision. This is a nice feature of SVM, which other methods do not provide. Judging based on both trends gives us a higher confidence what sample size to use than judging based on the accuracy trend only.

4.2 Comparison to a multiple-clock scheme

A multiple-clock scheme is another way to enhance the screening power of a test set [27]. For example, we can apply PB-STA to a set of patterns and collect those whose delays are similar. This strategy can be used to divide the set into groups of patterns and then we can associate a test clock with each group [20]. Because the patterns in each group have similar delays, tighter clocks can be used to enhance the screening for small delay defects [20].

Table 2. Comparison to a 10-clock scheme

	10 clocks			from Table 1		
minD	multi-clock	acc	#SVs	acc	#SVs	1-clock model
300ps	77.47%	90.05%	595	94.54%	407	23.3%
600ps	89.37%	91.86%	466	92.07%	315	59.2%
900ps	96.49%	97%	321	98.4%	208	75.5%
1800ps	100%	100%	121	100%	88	100%

Table 2 shows the results by applying a 10-clock scheme using the strategy above and the PB-STA statistical analyzer [8]. In binary classification, each data value d_{ij} of a pattern j on a sample i (in Figure 5) now becomes an integer k where $k \in \{1,2,\ldots,10\}$, i.e. the pattern delay is greater than the $(k-1)$th clock period but less than or equal to the kth clock period. The last three columns are copied from Table 1 where the SVM training was based on the data matrix that consists of the actual pattern delays.

It is interesting to observe that the multiple-clock scheme does show significant improvement from the 1-clock model. However, its screening capability is still less than that using a binary classifier. The difference becomes more noticeable as the classification problem becomes harder. It is also interesting to observe that by discretizing pattern delays into 10 ranges in the training data matrix, the learning accuracies degrade and the numbers of support vectors increase. This is understandable because the new data matrix provides less resolution on the delays.

5 Feature selection and test optimization

In our binary classification, a feature is a pattern's delay. It is intuitive to recognize that not all patterns are necessary for training a classifier. Some provide more separation power on the samples than others. If we are restricted to use only a fixed number of patterns, we would like to select the patterns that together, provide the greatest separation power. This problem is called *feature selection* in machine learning [28]. And a common approach to begin the feature selection is to calculate the *Fisher discriminant ratio* (FDR) for each feature and use these FDR scores to rank the importance of features [28]. Therefore, we begin our discussion with the FDR calculation.

5.1 Fisher discriminant ratio (FDR)

Suppose that we are given M good samples $G = \{s_1,\ldots,s_M\}$ and L defective samples $D = \{s'_1,\ldots,s'_L\}$. For a pattern p, suppose that applying the pattern on these samples obtains the delay data $[g_1,\ldots,g_M]$ for the good samples and $[d_1,\ldots,d_L]$ for the defective samples. Let $\mu_g = (\sum_{i=1}^{M} g_i)/M$, $\mu_d = (\sum_{i=1}^{L} d_i)/L$, and $\mu = ((\sum_{i=1}^{M} g_i) + (\sum_{i=1}^{L} d_i))/(M+L)$. We have:

$$FDR(p) = \frac{(\mu_g - \mu)^2 + (\mu_d - \mu)^2}{(\sigma_g)^2 + (\sigma_d)^2} = \frac{\text{between-group variance}}{\text{within-group variance}} \quad (2)$$

FDR is a popular statistic to measure the separation power of a feature [28]. The numerator indicates how well the pattern separates the good samples and the defective samples as two whole groups, i.e. the *between-group variance*. The denominator measures how the pattern differentiates samples within each group, i.e. the *within-group variance*. If $FDR(p)$ is larger, it is more likely that this pattern has greater separation power.

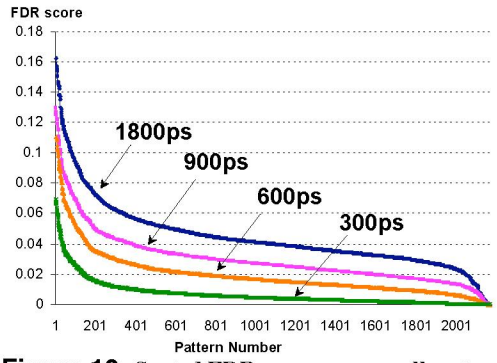


Figure 12. Sorted FDR scores across all patterns

Figure 12 shows the resulting FDR scores on the 2165 15-detect transition fault patterns. For each case, the FDR scores are sorted. We are interested in observing the trends. There are two noticeable trends in this figure. First, the FDR scores on a harder case (300ps) are consistently smaller than those on an easier case (1800ps). This is understandable because smaller FDR scores mean less separation power as discussed above. Secondly, there are about 200 patterns whose FDR scores are much larger than the rest. From 200 to 2000, the FDR scores do not change much. After 2000, the FDR scores approach to zero quickly.

5.2 FDR can be misleading

In a typical feature selection process, if we wanted to select a subset of features (patterns) based on the FDR ranking, we would probably begin by including the first 200 patterns. However, FDR scores can be misleading due to "deterministic factors". Consider the situation if there exists a pattern q such that q produces a large delay only on a single defective sample s. For all other samples, q's delays are similar and we have $\mu_g \approx \mu_d \approx \mu$ in the FDR calculation. In this case, $FDR(q)$ is very small. On the other hand, for all other patterns, the delays on s are close to their average delays on the good samples. Then, we know that q has the greatest power to separate s from the good samples, and it is likely that without q, s cannot be separated. While it is clear we want to include q if we rely on $FDR(q)$, we would have discarded pattern q due to its small FDR score. The above discussion explains that in our feature selection, there are some "deterministic" factors involved. Hence, the feature selection problem is not entirely statistical. The FDR statistic treats the set of good samples and the set of defective samples as two distributions. Hence, FDR score does not reflect the characteristics of individual samples. This motivates us to develop a new statistic that can reflect the characteristics of individual samples.

5.3 Statistics based on individual samples

Given a pattern p, let μ_p be the mean delay value of p averaging across all good samples. Note that the delay value on each sample can be a discretized value as described in the 10-clock scheme experiment before. For a given defective sample t, let d_p be the delay value of p on t. We define the *deviation distance* of pattern p on sample t as

$$Dev(p,t) = |d_p - \mu_p| \quad (3)$$

Then, for sample t, we select the pattern whose deviation distance is the largest among all patterns. If we collect all such patterns from all defective samples, we obtain the set of *baseline patterns*. For the 1000 defective samples in the experiments, the following shows the number of baseline patterns.

Table 3. Results of selected baseline patterns

minD	900ps	600ps	300ps
# of selected patterns	240	248	281
Accuracy using baseline patterns	93.25%	90.42%	85.78%
Accuracy using FDR ranking**	89.9%**	84.08%**	82.28%**
Accuracy- all patterns, Table 2	97%	91.86%	90.05%

**if the same number of patterns are selected following the FDR ranking.

In Table 3 we ran binary classification experiments based on these baseline patterns and first compared them to similar number of patterns selected by FDR rankings. The accuracies from FDR are consistently lower than those from the corresponding baseline pattern sets. The we compare the accuracy results to those listed in Table 2 where all patterns are used. The differences in accuracy can be clearly observed. This indicates that the baseline patterns are not sufficient.

5.4 Selecting additional patterns

To select more patterns, we have two intuitive options: (1) We can include additional patterns by going back to the FDR ranking (The result above does not necessarily imply that this is a bad strategy for selecting additional patterns). (2) We can expand the baseline pattern set by select the top K patterns from each sample based on the deviation distances. Figure 13 shows the results using option (1) and Table 4 shows the results using option (2).

In Figure 13, the x-axis is the percentage of the remaining patterns (excluding the baseline patterns) to be added into the baseline pattern set. For the 300ps case, we see that the accuracy would not peak until 70% of the remaining patterns are added.

If we project the accuracy results from Table 4 onto the accuracy results in Figure 13, we obtain the results shown in Figure 14 (on next page). This clearly shows that option

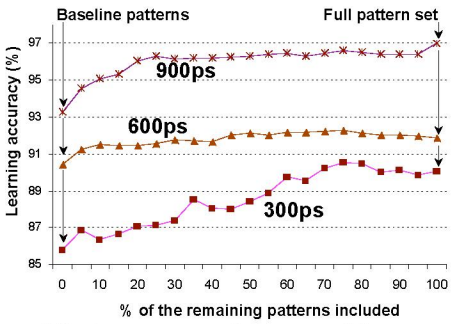


Figure 13. Accuracy results as we add patterns to the baseline pattern sets, by following the FDR rankings in 300ps, 600ps, and 900ps cases

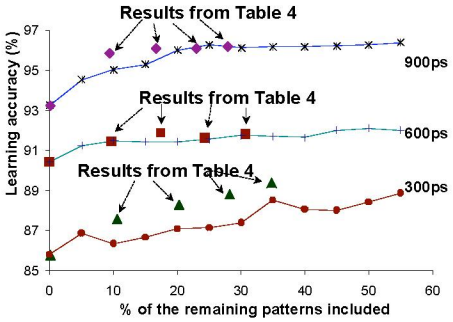


Figure 14. Projection of the results from Table 4 onto the plot in Figure 13 to compare the effectiveness of the two options of selecting additional patterns

Table 4. Expanding the set of baseline patterns

patterns per sample	$K=1$	$K=2$	$K=3$	$K=4$
300ps # of selected patterns	281	481	668	820
300ps learning accuracy	85.78%	87.58%	88.31%	88.82%
300ps #SVs	540	521	511	510
600ps # of selected patterns	248	434	582	715
600ps learning accuracy	90.42%	91.45%	91.86%	91.6%
600ps #SVs	413	378	384	395
900ps # of selected patterns	240	419	559	678
900ps learning accuracy	93.25%	95.89%	96.14%	96.14%
900ps #SVs	306	246	247	255

(2) above is a better strategy than option (1), i.e. even after the selection of the baseline pattern set, following the FDR ranking can still be misleading.

Table 5 presents the results of Table 4 from a different perspective. Here we are interested in observing the trend on the number of additional patterns included from $K-1$ to K, as K increases. Notice that this number decreases as K increases. Moreover, these numbers are smaller if the classification problem is easier.

Table 5. Trend of pattern increase as K increases

# of patterns selected per sample	$K=1$	$K=2$	$K=3$	$K=4$
300ps: # of additional patterns	281	200	187	152
600ps: # of additional patterns	248	186	148	133
900ps: # of additional patterns	240	179	140	119

In Table 4, we see that the number of support vectors does not change much as more patterns are selected. This indicates that the generalization of the learning accuracy results is rather independent of the pattern selection, ex. the confidence levels are similar between generalizing the 85.78% result to the unseen samples in the 300ps case from $K=1$, and generalizing the 88.82% result from $K=4$. This is a desirable property to have.

6 Additional experimental results

Table 6 summarizes the results from additional benchmark circuits, both using all patterns and the optimized deviation-distance based pattern set with $K=1$. All the SVM results show marked improvement over using a single clock or 10 clocks in the multiple-clock scheme [20]. Overall, the optimized set has lower complexity and comparable or improved accuracy. An exception is c1355, in which the accuracy seems to decline significantly. This seems to be because the number of patterns in the $K=1$ optimized set is rather small. The test escape and overkill percentage is also presented. Once again, we note that it is possible to explore the trade-off between these numbers by adjusting the decision thresholds in SVM, as illustrated in Figure 10-(b) before.

7 Note on simulation based pattern selection

Table 7. Minimal Slack patterns: Ind32

	1-slack	2-slack
Pattern Set	#Patterns	#Patterns
Unadjusted P-STM	323	428
Adjusted P-STM	383	509
Patterns shared by both	186	256

In Section 1.3, we use Figure 6 to illustrate two approaches for test optimization. For the simulation-based approach, one interesting question raised in Section 1.2 is how the accuracy of a statistical timing model would impact the pattern selection based on PB-STA [8]. We conducted an experiment to answer that question. First, we used the P-STM in PB-STA to select a set of patterns. Then, we used G-STM to produce a set of samples and applied the Bayesian parameter learning in [9] on these samples to learn the spatial delay correlations assumed in the G-STM. These correlations are quantified by a correlation coefficient ρ. The learned value of ρ are applied to the P-STM and we ran PB-STA to select a new set of patterns. We are interested in comparing these two sets of patterns in terms of their overlap.

Table 7 shows the number of patterns that are selected using the unadjusted and adjusted P-STMs. It is interesting to note that the patterns chosen are significantly different as the intersection between the patterns has only about half the number of patterns chosen. However, when we applied the in SVM learning experiments, we found that the SVM accuracy of the pattern sets to be very similar. With different circuits and defect models this result may not hold. Further

Table 6. Summary of experiments

		15-detect				Deviation-distance-based optimized set				Non-SVM Acc.	
Circuit	*minD*	#Patterns	Accuracy	#SVs	Overkills	#Patterns	Accuracy	#SVs	Overkills	1-clock.	10-clock.
Ind32	600ps	2165	91.86%	466	1.3%	248	91.04%	392	0.1%	59.2%	79.17%
c3540	650ps	5593	71.95%	860	11.35%	440	85.8%	607	1.05%	30.2%	65%
c2670	500ps	4296	72.45%	970	16.1%	282	88.05%	524	2.55%	12.3%	55.6%
c1355	400ps	3540	82.95%	912	5.85%	102	77.75%	763	2.2%	58.5%	72.8%
c880	300ps	1694	83.6%	626	6.2%	281	90.2%	487	0.6%	10.4%	53.3%

work needs to be done to examine how different aspects of statistical timing model accuracy affects the classification capability of the selected pattern sets.

8 Conclusion

In this work, we examined various issues in test set optimization for screening small delay defects. The primary objective is to enhance the screening capability of scan delay test with limited test patterns. We developed a framework for using known-good and know-defective silicon samples to build SVM classifiers, including metrics to evaluate if we have sufficient samples. We introduced a deviation-distance based pattern selection method for reducing the number of patterns needed. We discovered that in our pattern selection, the common statistics proposed for feature selection are not effective [28]. One potential application of this work is to facilitate adaptive test optimization, as illustrated in Figure 4. By using SVM learning techniques on known good/bad silicon samples we hope to extend coverage from scan-based delay tests out to cover other test methods. Although this study was limited to simulation-based delay tests and delay defects, the potential of the proposed methodology can be clearly observed in Table 6. We plan to extend our study to include industrial silicon data in the future.

References

[1] K.M. Butler, J. Saxena. An empirical study on the effects of test type ordering on overall test efficiency. *ITC* , 2000, pp. 408 - 416.

[2] R. Madge, B. Benware, R. Turakhia, R. Daasch, C. Schuermyer, J. Ruffler. In search of the optimum test set - adaptive test methods for maximum defect coverage and lowest test cost. *ITC*, 2004, pp. 203 - 212.

[3] M. R. Grimalia, *et al.* REDO — Random Excitation and Deterministic Observation: First Commercial Experiment. *VTS*, pp. 268-274, 1999.

[4] J. Dworak, J.D. Wicker, S. Lee, M.R. Grimaila, M.R. Mercer, K.M. Butler, B. Stewart, and Li-C. Wang. Defect-Oriented Testing and Defective-Part-Level Prediction. *IEEE Design & Test*, pp. 31-41, Jan-Feb 2001.

[5] Li-C. Wang, A. Krstic, L. Lee, K-T. Cheng, R. Mercer, T.W. Williams, M. Abadir. Using Logic Models To Predict The Detection Behavior Of Statistical Timing Defects. *ITC*, Oct. 2003, pp. 1041-1050

[6] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning - Date Mining, Inference, and Prediction. *Springer Series in Statistics*, 2001

[7] M. C-T. Chao, Li-C. Wang, K-T. Cheng. Pattern selection for testing of deep sub-micron timing defects. *DATE*, 2004, pp. 1060-1065.

[8] B. Lee, H. Li, Li-C. Wang, and M. Abadir. Hazard-aware statistical timing simulation and its applications in screening frequency-dependent defects. *ITC*, 2005.

[9] B. Lee, Li-C. Wang, and M. Abadir. Refined Statistical Static Timing Analysis Through Learning of Spatial Delay Correlations. *DAC* 2006.

[10] W. Qiu and D. M. H. Walker. An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit. *ITC*, 2003, pp. 592-601.

[11] N. Cristianini, J. Shawe-Taylor. An Introduction to Support Vector Machine and other kernel-induced-based learning methods. *Cambridge University Press*, 2002

[12] R. Madge, et. al. Obtaining High Defect Coverage for Frequency Dependent Defects on Complex ASICs *Design & Test of Computers*, Special issue on speed test, September 2003.

[13] B. R. Benware, R. Madge, C. Lu, R. Daasch, R. Effectiveness comparisons of outlier screening methods for frequency dependent defects on complex ASICs. *VTS*, May 2003, Page 39 - 46.

[14] S. Ma, P. France, and E. McCluskey. An Experimental Chip to Evaluate Test Techniques: Experiment Results. *ITC*, pp. 663-672, 1995.

[15] G. Devarayanadurg, et al. Test Set Selection for Structural Faults in Analog ICs. IEEE *Tran. on CAD*, Vol 18, No 7, July 1999, page 1026-1039.

[16] H-G. D. Stratigopoulos, Y. Makris. Non-Linear Decision Boundaries for Testing Analog Circuits. IEEE *Tran. on CAD*, Vol 24, No 11, 2005, page 1760-1773.

[17] R. Goodwin, et al. Advancements and Applications of Statistical Learning/Data Mining in Semiconductor Manufacturing Intel Technology Journal, Volume 8, Issue 4, November 17, 2004, pp. 325-336

[18] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. *ICCAD*, 2003, pp. 621-625.

[19] C. Amin, N. Menezes, K. Kilpatrick, F. Dartu, U. Choudhury, N. Hakim, and Y. Ismail. Statistical static timing analysis: How simple can we get. *DAC*, 2005.

[20] B. Lee, Li-C. Wang, M. Abadir. Reducing Pattern Delay Variations For Screening Frequency-Dependent Defects *VTS*, 2005

[21] B. Kruseman et al. On Hazard-free Patterns for Fine-Delay Fault Testing *ITC*, 2004, pp. 213-222.

[22] D. Boning and S. Nassif. Models of Process Variations in Device and Interconnect. in W. Bowhill A. Chandrakasan and F. Fox, editors, *Design of High Performance Microprocessor Circuits*, chapter 6. IEEE Press, 2000.

[23] V. Mehrotra, et al. A methodology for modeling the effects of systematic within-die interconnect and device variation on circuit performance. *DAC*, 2000, pp. 172-175.

[24] S. G. Duvall. Statistical circuit modeling and optimization. *5th International Workshop on Statistical Metrology*, 2000, pp. 56-63.

[25] N. N. Tendolkar. Analysis of Timing Failures Due to Random AC defects in VLSI modules. *DAC*,1985, pp 709-714.

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

[27] Jing-Jia Liou, et al. Enhancing test efficiency for delay fault testing using multiple-clocked schemes *DAC*, 2002, pp 371-374.

[28] S. Theodoridis, K. Koutroumbas. Pattern Recognition. Chapter 5: Feature Selection, Academic Press, 2 edition, February 1, 2003.