Statistical Analysis and Optimization of Parametric Delay Test *

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

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

Abstract

In this work, we present using Random Forests statistical learning to analyze post-silicon delay test data. We introduce the concept of parametric delay test as a new perspective for extracting more information from delay test. First, a methodology for outlier identification is presented to aid defect characterization of initial sample chips. Second, a methodology for production test is presented, including automated pattern-set reduction analysis. Finally, a strategy for adaptive test is presented.

1 Introduction

Delay testing faces a new set of challenges as mainstream production continues to move into deep-submicron process technologies. Specifically, subtle small delay defects will increasingly escape traditional pass/fail delay test methodologies and may cause reliability problems in the field. Due to advances in automated test equipment (ATE) and on-chip clock generation, at-speed and faster-than-at-speed test can offer increased screening power. However, with these new test capabilities, comes an enormous volume of generated test data and also a larger array of choices that test engineers must face. In this work, we introduce methodologies for automatically analyzing post-silicon delay test data and optimizing delay test. These methodologies are independent of pre-silicon ATPG, operating purely in the post-silicon phase of the test engineering effort.

1.1 Evolution of Delay Test

For many years, structural delay test methodology has consisted of applying test patterns that have been generated by ATPG using a fixed test clock. Steadily, the speed of delay test has increased, in an effort to decrease the slack. Figure 1 illustrates the situation. At the top of the figure, a slow-speed test clock cycle is shown. In the past, ATE limitations forced this test-clock to be quite slow relative to the rated-speed of the circuit under test. In determining how to fail chips, delay test inherited a hard pass/fail rule from stuck-at logical tests: *if any test pattern fails at the test clock, fail the chip*. This was well justified since the tests were occurring at such a slow speed. If any test pattern failed, it implied the presence of a gross delay defect.

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

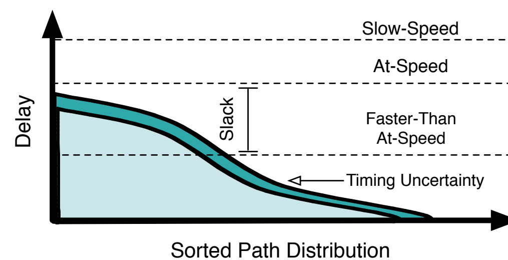


Figure 1. **Delay Distribution and Clock Cycle Time**

As advanced design-for-test techniques were employed and faster ATE became available, at-speed testing became possible. Shown as the middle clock cycle in Figure 1, at-speed test moved the test clock cycle time much closer to the critical path delay, reducing the slack interval so smaller defects could be captured. Timing-aware ATPG tools could be employed to generate tests that would attempt to propagate faults through longer paths. The test clock setting became more important now, as setting the clock too fast could induce overkill. Thus, the timing-behavior of test patterns would have to be validated via simulation. To be even more confident of the test clock settings, batches of known-good chips could be analyzed to find the optimal test clock with a percentage guard-band applied to pad the test clock.

Even at-speed testing is insufficient for capturing small defects on non-critical paths. In response, commercial ATPG tools have included faster-than-at-speed test generation. In Figure 1, a faster-than-at-speed clock cycle is shorter than the more critical paths. An example faster-than-at-speed test methodology would mask the responses from these more critical paths. The more critical paths would then have been covered with the at-speed clock. However, with increased timing uncertainty due to process variations, cross-talk and IR-drop, it is harder to trust pre-silicon timing models that ATPG and simulation tools use. Validating these test-patterns via simulation may not be sufficient, so increased burden is placed upon the test engineer in the post-silicon phase to validate patterns on known-good die and to carefully select test clocks.

1-4244-1128-9/07/$25.00 ©2007 IEEE

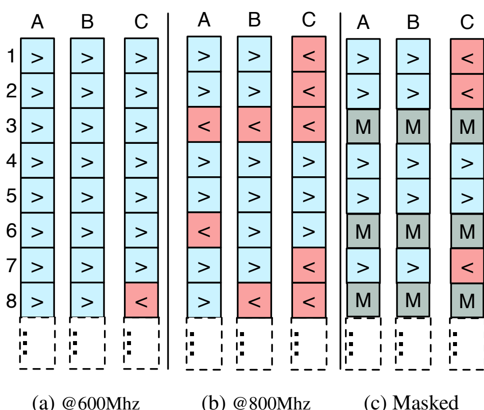
1.2 Changing the Rules of Delay Test

In the traditional pass/fail testing methodology, once the test engineer receives the test pattern set, their main concern is validating the test patterns and finding the best test clock. To avoid overkill, the test engineer may throw out problematic patterns and slow the test clock down (guardbanding). Unfortunately, both these actions result in reduced screening capability. Another way to view this is that by treating delay test in this manner, overkill is minimized with little regard to test-escapes. As the number of test-escapes due to delay defects increases, ignoring test-escapes in this manner will make less sense economically. Especially since the cost of a test-escape is usually more significant than an overkill. A different perspective is needed to better manage the trade-off between overkills and test-escapes.

1.3 Parametric Delay Test

The perspective that each delay test pattern is a pass/fail test has been inherited from stuck-at fault logical testing. The truth is, delay tests patterns are not pass/fail tests. Delay tests should really be seen as *measurements* more similar to that of a parametric test. At the lowest resolution, a pattern is applied at a single test clock - and the result is a rough measurement of whether the maximum pattern frequency is greater than the test clock frequency. For a higher resolution measurement, more clocks can be used (perhaps in a binary-search) to more accurately gauge the maximum pattern frequency. There is no fundamental reason the result of a single measurement must lead directly to passing or failing the chip. Instead, it makes sense to consider the collection of measurements, or *delay test signature*, as a whole.

The idea of a delay test signature is that it will expose a spectrum of chip behaviors. It is key to design delay test such that the measured delay test signatures have a maximal amount of information.



(a) @600Mhz (b) @800Mhz (c) Masked

Figure 2. Single Clock Delay Test Signature

In Figure 2 a-b, three parts are shown with their signatures from an at-speed clock and a faster-than-at-speed clock. The signatures are a vector of comparative measurements associated with the test patterns for each part. The '>' and '<' symbols indicate passing and exceeding the test clock respectively. Intuitively, the at-speed signature in Fig. 2 a) is not as informative as the faster-than-at-speed signature in Fig. 2 b) as there is no patterns distinguishing from parts A and B. Assume that the ATPG tool cannot guarantee that patterns 3, 6 and 8 will meet the faster clock speed on all "good" chips, in a traditional methodology, these patterns would then be masked so the standard pass/fail method could be used (Fig. 2 c). However, by masking these patterns, the information is lost on how part A and B are distinguished. In our work, we will demonstrate there is an advantage in utilizing the full signature even when limited to a single test clock.

Higher resolution delay test signatures can be obtained by using multiple test clocks to search for the maximum passing frequency of each test pattern, providing a rough measurement of a test pattern's delay. In Figure 3, the power

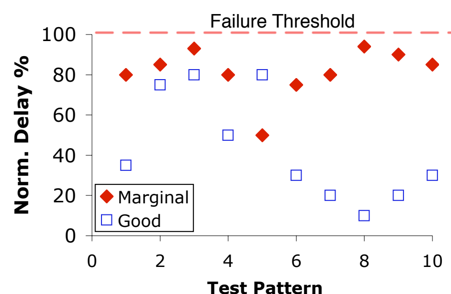


Figure 3. High Resolution Delay Test Signatures

of higher resolution delay test signatures is shown. In this example, several test patterns' measured delays for a good die and a marginal die are shown relative to a traditional at-speed test clock cycle. A single at-speed clock cannot identify the marginal die since each pattern "passes" despite many of the patterns being near the threshold. Using the simple pass/fail rule for each high-resolution measurement individually is insufficient. To properly capture the marginal chip, it is necessary to use all the measurements as a whole in a more complicated decision rule. Because this rule will necessarily be more complicated, the analysis to produce the rule must be automated.

1.4 Delay Test Signature Analysis

Automated delay test signatures analysis can be useful in two phases of the post-silicon test effort. Before production ramp-up, delay test signatures can be analyzed to identify outlier parts that may be interesting to study in detail for defect characterization. This is similar to outlier analysis for IDDQ measurements, as it is difficult to know what exactly to expect pre-silicon, making the post-silicon analysis more critical. During this phase, since the number of chips is limited, higher resolution delay test signatures with more test patterns can be used. For production testing, lower resolution delay test signatures should be gathered, requiring as few test clocks as possible, and using a minimal set of

patterns. The key objective is to create a function that maps the lower resolution test signature to the pass/fail decision. Statistical learning techniques are well suited for the analysis required in both phases.

1.5 Role of Statistical Learning

Statistical learning has been applied in many fields in which there is uncertainty in the systems at work but there is a body of empirical data that can be analyzed. Statistical learning is employed to make accurate predictions in applications such as medical diagnosis and financial markets. For delay test analysis, we accept that pre-silicon ATPG tools may not accurately model process variation, crosstalk, IR-drop effects, etc. Additionally they will also not have an accurate picture of the delay defect distribution. However, via automated testing, we can collect a wealth of data from initial sample chips. This mix of modeling uncertainty and available empirical data makes statistical learning perfect for analyzing of delay test signatures .

The rest of the paper is organized as follows. In Section 2, we will discuss related work. Section 3 discusses the experimental setup. Section 4 briefly introduces Random Forests statistical learning. Section 5 will demonstrate outlier analysis of high resolution delay test signatures. Section 6 will demonstrate supervised analysis of delay test signatures for production test. Section 7 will introduce an adaptive test methodology for production test. Section 8 concludes the paper.

2 Related Work

There have been many efforts to use statistical learning methods on post-silicon data. Researchers at Intel have published several case studies in applying data mining techniques toward optimizing the production test flow[8]. A Bayesian learning based method for learning spatial delay correlations from path delay testing was proposed in [13]. Unsupervised learning methods such as clustering have been suggested for IDDQ test data [9].

In [6], statistical techniques for identifying latent defects and outlier screening were proposed. A Support Vector Machine (SVM) learning technique was proposed for improving delay test in [14]. In this work, probability estimates and cost were not considered. The goals of using statistical learning methods stem from the objective of achieving *adaptive test*, a test methodology that will intelligently adapt based on the data that is collected. The most common ideas are that the test methodology will adaptively reorder patterns, or adaptively use different test suites for different IC's. These ideas have been explored in [4, 5].

Specific to delay testing, there has been a great deal of work addressing the concern of small delay defects. It has been well established that transition fault coverage does not necessarily correspond with delay defect coverage. It has been suggested that a "superset" of patterns is generated by the ATPG in order to ensure the inclusion of sufficient tests to detect a wide variety of defects. For example, the n-detect test set [16] and the k-longest path delay test set [15] are two notable examples of this sort. However, these test sets may contain patterns that are redundant and/or ineffective with respect to detecting the actual defects. In [12], the authors proposed using multiple test clocks to screen for small delay defects based on pre-silicon statistical pattern simulation. This method does not have a post-silicon component however and assumes accurate statistical timing models.

3 Experimental Setup

In this work, the experiments are 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 m samples, whose delay values are statistically drawn from a statistical timing model (STM), are simulated. Hence, each sample has a different, fixed 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).

To drive the experiment, thousands of defect-free circuit instances are generated using normal process variation. A subset of circuit instances are generated with randomly sized and randomly located delay defects injected. The size of these defects are drawn from an exponential random distribution with a mean selected to be relatively small with respect to the at speed clock cycle time. For each instance, a 15-detect transition fault pattern set was applied, and the delay test responses were recorded assuming a discrete set of test clocks. We refer to this data as the *delay test signature*. Although transition faults tests are being used for this experiment, our methodology is completely agnostic towards the pre-silicon generation of delay tests; the delay tests can be any mixture of tests.

4 Random Forests

Random Forests classification was proposed by Breiman in 2001 [2]. The technique has become popular due to its best-of-class performance combined with its relative simplicity. The name of *random forests* describes the two main characteristics of the algorithm. Since a number of deci-

sion trees are grown, the group of trees is simply a *forest*. The term *random* describes the process of how each tree is grown. During the tree growing process, the randomness is applied in two different steps. First, each tree is grown based on a *random* sample of the training data, which is known as *bagging* in the machine learning community. Then, at each split node of each tree, a *random* selection of the patterns is used (Section 6.4.1 will give additional detail on how to grow a tree). The randomness that is thrown in during the forest construction ensures that each tree is similar but different. It was proved that generalization error, the error that happened when the constructed model is applied to entire data space, of random forests converges when the forest contains a large number of tree [2].

Random Forests is capable of efficiently analyzing data sets with a large number of samples as well as a data sets with high dimensionality(there are a large number of variables associated with each sample). Unlike statistical learning techniques such as Neural-Networks and Support Vector Machines which essentially produce black-box models, Random Forests can offer interpretability of its generated models. Specifically, Random Forest models can offer insights on how similar samples are to each other, which can be useful in clustering and outlier identification. It can also rank the importance of input variables, which can be useful in test-optimization. A custom implementation of Random Forests, libRF [19], was used to analyze the data.

Random Forests are typically used in *supervised learning*, in which there is a labeled training set and the goal is to predict the labels on subsequent unseen data. However, Random Forests can also be used for *unsupervised learning*, when there is no labeled training data. In the subsequent sections, we will delve into how these two forms of statistical learning can be applied to delay testing.

In the following, we present the unsupervised learning application of outlier identification first, followed by the supervised learning application of production test optimization because in a typical flow, outlier analysis occurs before production testing. In Section 6.1, we will present a detailed discussion of Random Forest supervised learning.

5 Outlier Identification

In unsupervised learning, the data set provided to the learning algorithm does not have labels. The learning algorithm is then asked to group the samples based on the similarity and difference presented in the data. In our case, it means that the data set only contains the delay response of each test pattern for every sample. Unsupervised learning can be useful for analyzing the *high resolution* delay test signatures of preliminary sample chips when a detailed test methodology has not been established yet. The term *high resolution* means that a 15-detect transition fault pattern set was applied using all test clock frequencies available. During this time before production test, failure analysis and defect characterization is crucial. Unsupervised learning can model the distribution of delay test signatures and identify non-trivial outliers.

5.1 Trivial Outlier Analysis

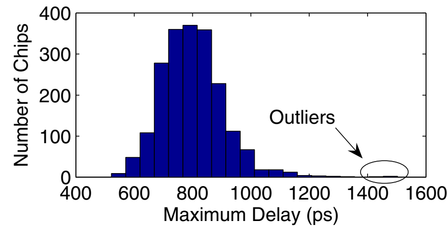


Figure 4. Ind32: Trivial outlier identification

Finding outliers in 1-dimension is a rather trivial problem. As a baseline for comparison, we looked for outliers in the maximum delay behavior of all the chips under analysis. In Figure 4, a histogram of the delay behavior of the chips is shown, with the outliers circled. As a threshold, it is convenient to use a 3σ threshold, in our case, we chose to approximate σ using *median absolute deviation* [17] a measure of the variance that is more immune to outliers. In essence, this outlier analysis works by reducing each chip to a single dimension (its maximum delay). While this method should be able to identify chips that have gross delay defects, this method will not capture more subtle defects.

5.2 Random Forest Outlier Analysis

To identify more subtle defects it is necessary to consider all the dimensions available (potentially thousands of delay test patterns). Outliers are chips that are distant from the main distribution. However, it is well known in statistical learning, that as the number of dimensions increase traditional distance measures like Euclidean distance become ineffective. This is known as *the curse of dimensionality* and is due to the exponential increase in volume as the number of dimensions increase. Because of this phenomenon, every chip will seem distant to every other chip in a high-dimensional space. Random Forests, however is able to provide a useful distance measure that works well despite the number of dimensions in the data.

Random Forests unsupervised learning is built-upon Random Forests supervised learning. To apply supervised learning, a second synthetic data set is constructed based on the original data set. Samples in the original set are labeled as one class and samples in the synthetic set are labeled as the other class. Then, random forest is constructed as a binary classifier to differentiate these two classes of samples.

The original data set can be represented as a matrix $A = |a_{ij}|_{i=1...n, j=1...k}$ where i is the index for chip sample and j is the index for pattern delay. The synthetic data set is another matrix $B = |b_{ij}|_{i=1...m, j=1...k}$. Each b_{ij} is randomly

sampled from the column values $\{a_{1j}, a_{2j}, \ldots, a_{nj}\}$. The sampling is done for each b_{ij} individually. Essentially, the univariate distribution of each pattern delay across all chips in the synthetic set is equivalent to the original set. However, the correlations between the pattern delays of the same chip in the original set have been entirely destroyed due to the random sampling scheme. Breiman [2] shows that, if the constructed forest for the binary classification can successfully differentiate between the original and synthetic data, then the forest can be used for unsupervised learning by constructing a *proximity matrix*.

In our application, the binary classifier forest is utilized to measure the *distances* (similarities) between pattern delay signatures of chips. Suppose the forest consists of 100 trees $T_1, \ldots, T_{100}$. The distance between two pattern delay signatures (from two chips) is measured by counting the number of trees that use the same paths to classify the chips. Suppose this number is 60. Then, the distance is recorded as 0.60. Based on this measure, the *proximity matrix* is built to record the distances between any pair of chips. This proximity matrix is then used for outlier identification.

Figure 5 shows a proposed flow in which Random Forests analysis operates directly on a collection of delay test signatures and produces a *proximity matrix*.

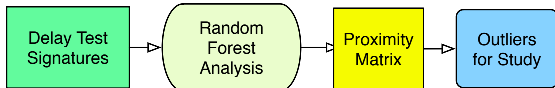


Figure 5. **Unsupervised Random Forest Analysis**

The proximity matrix depicts how similar each chip is to every other chip. For visualization of the proximity matrix, multi-dimensional scaling can project the matrix into 2-dimensions. Note that this is for visualization only, as the scaling coordinates do not represent any meaningful values.

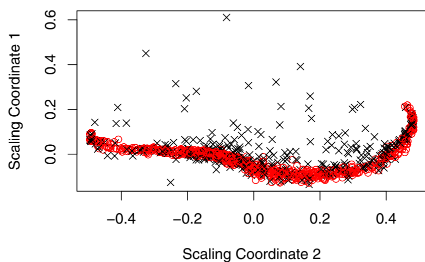


Figure 6. **Ind32: 2-D projection of proximity matrix (points of defect-injected samples are marked as $\times$)**

Figure 6 shows an example projection. Notice that good samples are clustered in a strip while many defect-injected samples locate outside the strip. Some defective samples mix with the good samples and do not appear to be outliers. This is possible as injected small-delay defects may not cause them to behave differently from the good samples.

Another way to analyze the proximity matrix is via the

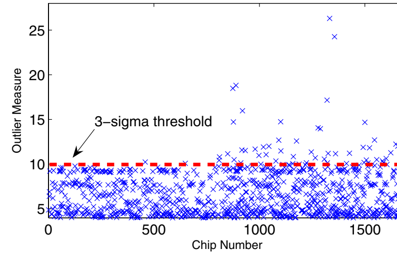


Figure 7. **Ind32: Outlier Measure plot**

outlier measure, a score derived for each sample to measure the average distance of the sample from all other samples. In Figure 7, the outlier measure of each sample is plotted. By selecting a threshold for this score, we can consider samples that exceeds the threshold as outliers.

Table 1. Outlier Analysis

Circuit	Trivial	RF Outliers
s15850	4	197
s9234	0	236
Ind32	28	137

Recall that in Figure 4 a trivial outlier identification method is shown. In Table 1, we compare how many defect-injected samples can be found via the trivial outlier method versus the outlier measure method. For each circuit, delay test signatures from 2000 samples were examined. For one thousand of the samples, small delay defects were injected. These defects are clearly hard to detect, as the trivial outlier analysis reveals few of them. However, by analyzing delay test signatures using random forests, these hard-to-find defective chips can be identified via outlier analysis.

6 Supervised Learning and Delay Testing

In this section, a robust analysis which directly targets pass/fail labeling and minimizes overkill and test escapes is discussed. For this type of analysis, supervised learning is well suited. In supervised learning there are two phases, a training phase and an application phase. In the training phase, a labeled training data set is first used to generate a learned model. In the application phase, the learned model is applied on unseen and unlabeled data, and the labels are predicted. For production delay test, the training data is the delay test signatures of known-good dies and known-bad dies. The labels we want to predict are *pass* and *fail*.

Delay testing as a supervised learning problem is shown in Figure 8, M refers to the number of samples (chips) contained in the matrix, and N is number of the variables (test patterns). The learning algorithm takes the matrix and tries to construct a classifier $C(.)$ that will be used to predict the label of unlabeled vector $\vec{r}$ of length N [14].

Since we have the advantage of having full simulation knowledge, golden labels, *pass* and *fail*, were determined by examining if injected defects had an effect on critical

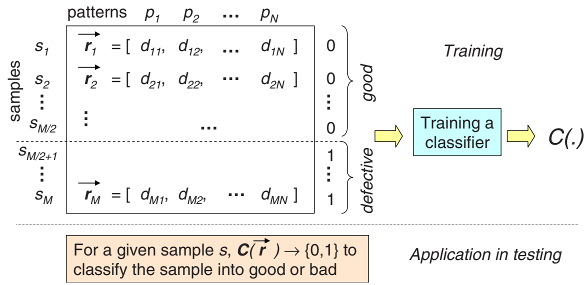


Figure 8. Delay Test as a supervised learning problem

path delays. Thus, not all of the defect-injected die are labeled as fail. We believe this will mimic the results of extensive testing for delay defects in reality.

6.1 Random Forests Supervised Learning

As mentioned in Section 4, a number of trees are grown based on the training set. When a new sample is to be classified, the sample is presented to each tree in the Random Forest. Each tree makes a classification decision on the sample. These decisions can be thought of as *votes*. For each sample, the random forest can then provide *class probability estimate*, based upon the votes. The ratio of votes a class receives relative to the number of decision trees is the probability the sample being classified to the class. Intuitively, this can be thought of as a *wisdom of the crowds* effect, in which the forest as a whole performs better than any individual decision tree. The performance of random forests is quite competitive with other top of the line classifiers[3].

6.1.1 Making Cost-Sensitive Pass/Fail Decisions

Applied to delay test, a Random Forest will generate *failure probability estimates*: $Pr(fail|tests)$, the conditional probability of failure given the test results. These probability estimates can greatly improve decision making under conditions of uncertainty. For each die encountered, the decision has to be made either to pass or fail the die. Accordingly there are costs that are associated with making these decisions, specifically the cost of throwing away a good chip, $C_{overkill}$ and the cost of shipping a bad chip to the customer, C_{escape}. We can utilize the class probability estimate to present the expected costs of making a pass and fail decision (assuming that making the correct classification has zero cost).

$$E[C_{pass}] = Pr(fail) * C_{escape} \quad (1)$$

$$E[C_{fail}]] = (1 - Pr(fail)) * C_{overkill} \quad (2)$$

In other words, if a chip is passed, the expected cost of the decision is the probability that it is actually a bad chip multiplied by the cost of shipping a bad part. If a chip is failed, the expected cost of the decision is the probability that it is a good chip multiplied by the cost of throwing away a good chip. An optimal decision rule uses the failure probability estimate and makes the decision that has the least expected cost. Thus, it will be a simple threshold test based on the $Pr(fail)$. If this probability is greater than β, than the best decision is to label the sample fail, otherwise, the best decision is to label the sample pass. β can be derived by setting these two expected costs equal to each other and solving for $Pr(fail)$. Thus, the cost optimal boundary β is:

$$Pr(fail) > \beta = \frac{C_{overkill}}{(C_{escape} + C_{overkill})} \quad (3)$$

By applying this rule given a failure probability for each die, we have a theoretically optimal method for minimizing the expected cost of the decision. In contrast, traditional test flow only produces binary information and does not support this type of cost optimization.

6.2 Random Forest Learning Flow

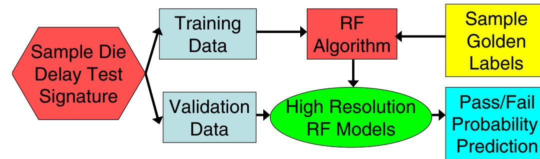


Figure 9. Random Forests Learning Flow

The basic flow of how Random Forests can be applied to delay testing is shown in Figure 9. A set of training and validation samples are made by randomly choosing from the pool of samples that has been labeled. The training samples are then given to the Random Forests (RF) algorithm to build a *high resolution* RF model using all available patterns and test clock frequencies. The validation samples are used to verify the effectiveness of the RF model.

For several circuits, we trained RF models using 2000 samples, and then tested the models on an additional 2000 samples. In Table 2, we compare the defect capture and overkill performance of a traditional at-speed single clock delay test against the RF methods where two different settings of the failure probability threshold, β, are used. Note that because the total number of defective samples is fixed, the number of test-escapes decreases relatively to the increase of the capture and overkill numbers.

Table 2. Traditional Delay Test vs RF Performance

Method	Captures	Kills	Captures	Kills	Captures	Kills
Circuit	c880		Ind32		s13027	
Traditional	9	0	28	0	4	0
RF $\beta = 50\%$	51	7	220	2	82	28
RF $\beta = 33\%$	92	38	283	16	159	90

In the traditional delay test, the test clock is set at the maximum pattern delay of all labeled good samples. Hence, it does not cause any overkill. However, it is also unable to capture many of the small delay defects. The number of defect captures increases dramatically by using RF, most notably in Ind32. This comes with the price of making a few overkills. By varying the failure probability threshold

β, the trade-off between overkills and captures can be flexibly explored. Lowering β increases overkill, which may be economically justified if the cost of test escape is significantly higher than the cost of overkill.

6.3 Identifying Marginal Chips

The probability estimates themselves may provide information about how marginal the chips are. Figure 10 is a

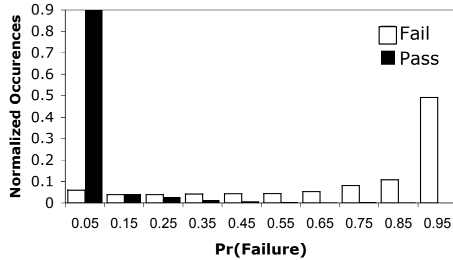


Figure 10. **Ind32: Probability estimate histograms**

histogram of how many good and bad chips fall into different failure probability estimate bins. From this histogram, we see that the first bin is dominated by passing chips which is quite reassuring. The next three bins have a small amount of good chips. We decided to examine these labeled good die that have greater than 10% probabilities of failure. Out of the 140 labeled good die with this property, 85 had latent delay defects that were injected in the simulation process but did not affect any critical paths and thus were not labeled $fail$. The size of these defects are plotted in Figure 11. We note that many of these defects are small.

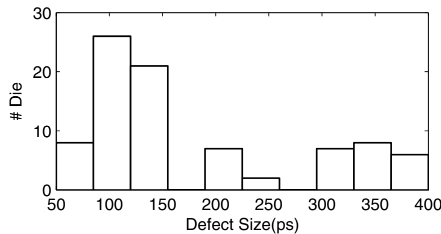


Figure 11. **Latent defect sizes found**

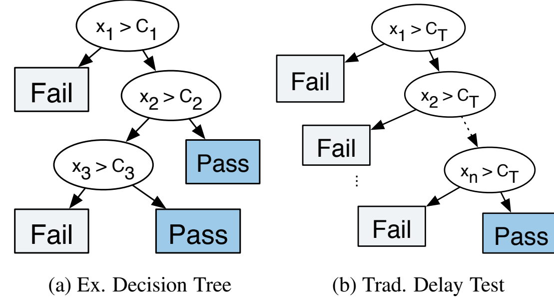
This is an important result since it shows that the probability of failure estimates can help identify marginal die that otherwise may escape traditional testing. If a die passes all traditional tests, yet has a significant probability of failure estimate from the random forest classifier, it may be worth it to do a detailed analysis to determine if there are any latent defects that are escaping the normal tests

6.4 Pattern and Test Clock Reduction

Reducing the pattern set is a critical part of post-silicon test optimization. In order for parametric delay test to be practical in production test, it is important to require a minimal number of test-patterns and a minimal number of test clock frequencies. Test engineers must be able to easily explore the trade-off between pattern set size, number of test clock frequencies, and test error cost.

6.4.1 Interpreting Random Forests Model

Fortunately, the models that are generated by Random Forests can be interpreted in a manner that can be utilized in test optimization. As previously mentioned, Random Forests models are composed of many decision trees.



(a) Ex. Decision Tree (b) Trad. Delay Test

Figure 12. **Decision Trees**

Decision trees are produced by a rather straightforward algorithm. Given the training data of test pattern responses and pass/fail labels, a decision tree learning algorithm picks which patterns to create "split" nodes with. Each "split" node is associated with a single value and will divide the data into two subsets based on whether the feature (pattern delay) is lesser or greater than a comparison value. This process is repeated recursively on each subset. The leaf nodes at the bottom of the tree will then contain the pass/fail decision. A simple example tree that may be created in this manner is shown in Figure 12 a), in which patterns $x_1...x_3$ may be compared against values $C_1...C_3$.

The comparison values in the split nodes have a special meaning with respect to delay testing. Each split node needs to know whether the pattern delay of a sample exceeds a certain comparison value. This information is obtained in delay test by applying the test pattern at a clock frequency. Thus, each split node can be viewed as a test pattern, clock period (frequency) pair.

In this manner, traditional delay test can be viewed as a special decision tree that is always constructed as shown in Fig 12 b) where one very conservative test clock period C_T is used, and a chip sample is only passed if its test pattern delays are all smaller than this clock period.

6.4.2 Calculating Test Application Cost

We can track the cost of applying the tests needed for a random forest classifier by examining each of the component decision trees of the Random Forest model. As we mentioned previously, a split node in a tree is associated with a single pattern at a specific clock frequency. This is actually a conservative view, as technically, the number of patterns and clocks can vary adaptively as different samples take different paths to reach the decision leaf. For the rest of this section we will take this conservative view that each sample

die is tested with all the pattern/clock combinations found in the random forest. Thus, for each sample die we will actually be collecting more information than is strictly necessary for the random forest to output its decision.

6.4.3 Variable (Pattern-Clock Pair) Importance

In order to automatically interpret the model, Random Forests can provide a very useful metric called *variable importance*. For each decision tree in the random forest, only a subset of the total possible test pattern/test clock combinations are used. For each of these pattern-clock pairs, the values in the data are permuted randomly, and the effect on accuracy is measured against the normal undisturbed data. This comparison is made for all pattern-clock pairs and averaged across all trees resulting in an importance score. The intuition here is that when important pattern-clock pairs are permuted, the accuracy will be significantly altered. Thus, we can obtain an importance ranking of all the pattern-clock pairs. Guided by this ranking, we can intelligently reduce the number of patterns and clocks needed by getting rid of the lowest ranking patterns and clocks.

6.4.4 Pattern/Clock Reduction Flow

The flow of the proposed pattern/clock reduction process is presented in Figure 13. Based on the high resolution RF model, the variable importance is calculated.

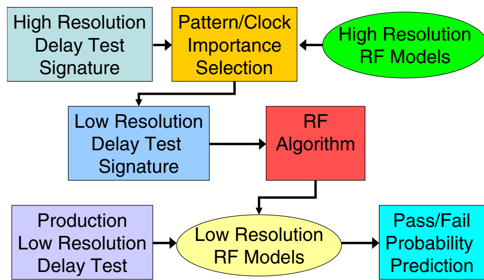


Figure 13. Pattern/Test Clock reduction flow

By selecting the top k most important test patterns with their corresponding clock frequencies, a *low resolution delay test signature* is obtained, named as such due to its reduced pattern counts and smaller number of clock frequencies. Then Random Forests learning algorithm is applied on the low resolution data, and a new low resolution RF model is generated. Since the low resolution model requires delay test signatures with fewer patterns and test clock frequencies, it is more effective for production use. Next, we present the effectiveness of the methodology.

6.4.5 Experimental Results

Using the importance scores of each pattern, we rank the patterns and generate random forest classifiers using only the top k ranking patterns. The important trade-off to examine is how reducing the number of patterns will affect the accuracy of the classification.

Table 3. Ind32: Performance vs Pattern Set Reduction

	$\beta = 50\%$		$\beta = 33\%$	
Pat. #	Overkills	Captures	Overkills	Captures
100	14	142	59	189
300	12	158	59	212
500	9	156	55	227
700	14	176	44	251
900	8	183	39	260
1100	8	195	34	272
1300	6	203	37	272
1500	4	206	29	278
1700	3	201	36	276
1900	7	206	37	281

In Table 3, the overkill and defect captures are summed up for the 2000 validation samples given random forest classifiers that use different numbers of patterns with a fixed 100 decision trees. For the sake of argument, we assume that $C_{overkill}$ is \$1 and C_{escape} is \$2. To illustrate the effect of varying the decision boundary, we show the results with both a standard decision boundary of $\beta = 50\%$ and the cost-optimal decision boundary of $\beta = 33\%$ (calculated from the equation in Section 6.1.1). It can be seen that by lowering the decision boundary, we incur more overkills to reduce test escapes. It is quite easy to change the decision boundary since the random forest classifier outputs probability estimates. The test program settings are essentially unaffected in the sense that the patterns and test clocks are still the same. This kind of flexibility is much harder to achieve in traditional delay testing where the only flexibility is in changing the test clock.

In Figure 14, the costs associated with using both boundaries are plotted against the number of pattern-clock pairs applied (test time), and we verify that the costs associated with the optimal decision boundary are lower than that of the standard boundary. Using these plots, a test engineer would be able to analyze the trade-off between the number of test patterns and the cost of making mistakes.

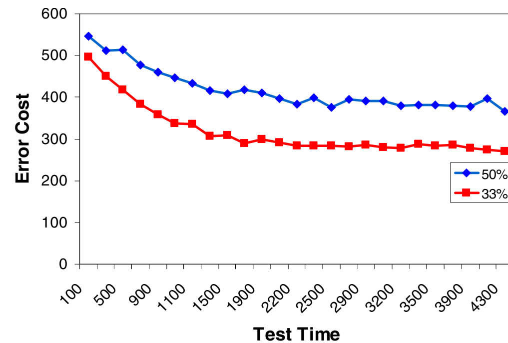


Figure 14. Decision Cost vs Test Application Cost

In Table 4, a simple comparison is made between high resolution delay signature, the 15-detection transition fault delay test pattern set applied with all 18 test clock frequencies, and the result based on low resolution delay signature, top 2200 pattern/test clock frequencies pairs based on the

variable importance metrics. The overkills from the low resolution random forest are slightly higher than the high resolution classifier on the 2000 validation samples. However, the low resolution model's test application cost, the number of pattern/test clock frequency pairs used in the model, is drastically reduced from that of the high resolution case. Only 5.7% of the high resolution model's pattern-clock pairs are needed. We choose the number 2200 for comparison because the 15-detection transition fault delay test pattern set has around 2200 patterns. In other words, the cost of this flow at production will be same as running 15-detection transition fault delay test pattern set plus the cost of test clock frequencies switching.

Table 4. High Resolution vs Low Resolution

	$\beta = 50\%$		$\beta = 33\%$		
Pat. #	Kills	Captures	Kills	Captures	Test Cost
High Res.	2	220	16	283	34650
Low Res.	6	220	35	284	2200

A histogram of the test clock frequencies used in the high resolution RF model is shown in Figure 15. We can see that majority of the split nodes uses a test clock frequency that is faster than frequency of traditional delay test.

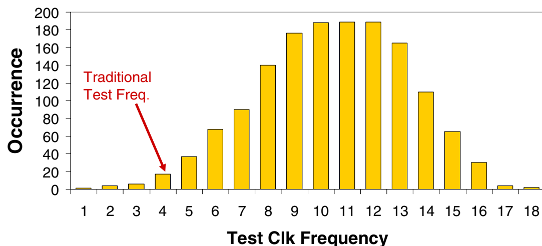


Figure 15. High-Res. Test Clock Usage Distribution

Also shown in Figure 15 is the clock frequency used in traditional delay test, which would be set at 4 to avoid overkill. For comparison, the histogram of test clock frequency usage in the reduced low resolution RF model is show in Figure 16. We see that fewer test clocks are needed in the low resolution RF model. It should also be noted that traditional delay test clock frequency, bin 4 in Figure 15, is not used in the low resolution RF model built upon the most important 2200 patter/test clock frequency pairs from the variable importance analysis.

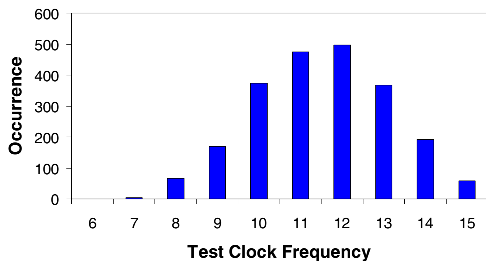


Figure 16. Low-Res. Test Clock Usage Distribution

6.5 Selecting the Best Single Test Clock

Suppose that multiple clock testing is not feasible, and only a single fixed clock can be used to generate the delay test signature. We used our reduction flow to choose the best single test clock for several circuits, and compared the results to traditional delay test.

Table 5. Single Clock Results

	Traditional		RF-based			
Circuit	Caps.	T_c	Caps.	Kills	T_c	ΔT_c %
c880	9	1425	61	7	900	-36.84%
c1355	26	1485	116	15	1000	-32.66%
Ind32	28	1087	211	33	700	-35.60%
s13207	4	2240	82	45	900	-59.82%

In Table 5, the number of small defect captures and the test clock cycle time are shown for several circuits for the traditional delay test methodology as well as the Random Forest based methodology. T_c is the test clock cycle in picoseconds. The traditional test clock cycle is set ideally with no guardband so it is actually already optimistic in the number of small defect captures. Also, the number of overkills in the RF-based flow is listed (there are no overkills in the traditional method). The best single test clock cycle from Random Forest analysis is chosen for the RF-based flow. In the last column, the percent reduction in test clock cycle time is shown. Note that for all circuits, considerably more small delay defects are captured using the RF-based flow. There is, however, a trade-off, in that a small number of overkills is incurred. It is interesting to note that the best single test clock cycle is at least 30 percent smaller than the conservative at-speed test clock cycle. This demonstrates that even if the RF-based flow is limited to using a single test clock, considerably improved results can be obtained over traditional delay test.

7 Adaptive Test Clocking

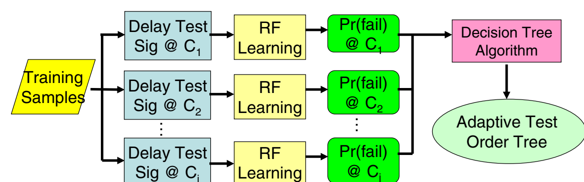


Figure 17. Adaptive Test Clocking Flow

However, if multiple test clocks are used in production test, we must consider that switching test clocks takes a non-negligible amount of time in testing. Practically, it makes sense to group patterns by test clock frequency. If test is organized this way, it is natural to make it adaptive. That is, depending on the result of the previous test clock, we may choose to apply different subsequent test clocks. The end result being that different chips will have a different battery of test clocks applied.

We propose a two-step flow toward adaptive test clocking, and it is illustrated in Figure 17. First, delay test signatures are separated and grouped by test clock frequency. Individual Random Forest models are then trained for each specific clock frequency. The outputs of each Random Forest model, the probability estimates, are then used as inputs to decision tree algorithm as shown. This results in a single decision tree that can be used to adaptively order tests.

Figure 18 shows the result after applying the above method onto a Ind32, a combinational block from an industrial design. In this example, the decision tree indicates that all chips should have test clock C_2 applied. Depending on this result, test clocks C_1 and C_3 may be applied. Note that at the bottom of the tree, a class *marginal* can be given if the leaf node is not decisive (for example, in the training phase the node contains almost equal amounts of defective chip samples and good chip samples).

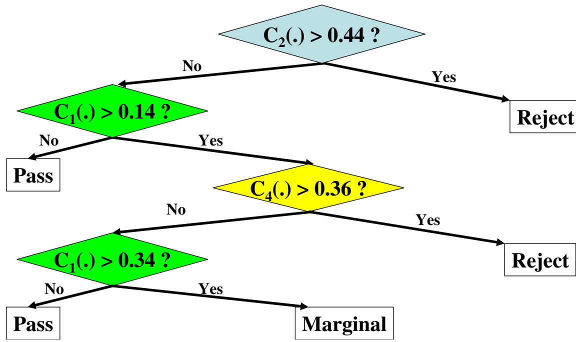


Figure 18. **Adaptive Ordering Tree for Ind32**

In Table 6, we compare the adaptive ordering result against a result using a non-adaptive ten fixed test clocks. Although the non-adaptive results in fewer overkills, the adaptive test result can actually result in more defect captures while requiring on average only 2 test clocks with reducing total tester time. By using adaptive test ordering, we can reduce the average number of test clocks needed to test each chip while still maintaining screening capability. The reduction on number of test clocks required directly translates into saving on average test time that each chip needs.

Table 6. Adaptive Ordering Performance Comparison

	# Clocks	Test Time	Kills	Captures
Adap. Test (avg)	2.05	4443	19	232
Fixed Test	10	21650	2	221

8 Conclusion and Future Work

In this work, we introduced the concept of parametric delay test and measuring delay test signatures. We demonstrated the utility of Random Forests statistical analysis of delay test signatures in several applications. For defect characterization of initial sample chips, unsupervised outlier analysis can identify hard-to-find latent defects. For production test, Random Forests supervised learning can learn to distinguish good chip delay test signatures against bad chip delay test signatures. The model built by Random Forests can even give a probabilistic output. Thus, the pass/fail threshold can be flexibly decided by economic means. To further optimize production test, the Random Forest model can be interpreted in detail. Test patterns and test clocks can be reduced while maintaining screening capability. Significant improvements in small delay defect screening can be achieved even with only a single test clock. We also demonstrated an adaptive test clocking flow that would allow for a variable number of test clocks to be applied to each chip. Finally, we note that these methods are delay test agnostic, in the sense that the methods can be applied to analyze transition-fault tests, path-delay tests or BIST patterns, even though this work assumes transition-fault tests in the experiments.

Besides multiple test clock frequencies, different voltages and temperatures can also be analyzed for adaptive test ordering. Random Forests statistical analysis could be applied with other parametric tests such as IDDQ. Finally, these methods could also be used to correlate delay test signatures to functional speed for speed-binning applications.

References

[1] Margineantu, Dragos D., Class Probability Estimation and Cost-Sensitive Classification Decisions, *European Conference on Machine Learning* 2002

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

[3] R. Caruana, A. Niculescu-Mizil An Empirical Comparison of Supervised Learning Algorithms. *ICML* 2006

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

[5] 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.

[6] 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.

[7] 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*, Jan-Feb 2001, pp. 31-41.

[8] 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

[9] S. Jandhyala, et al. Clustering Based Techniques for IDDQ Testing. *ITC*, 1999.

[10] 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*, 2003.

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

[12] 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.

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

[14] 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

[15] 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.

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

[17] S. S. Sabade, D. M. Walker, Evaluation of Effectiveness of Median of Absolute Deviations Outlier Rejection-based IddQ Testing for Burn-in Reduction. *VTS*, 2002.

[18] N. Ahmed, M. Tehranipoor and V. Jayaram, A Novel Framework for Faster-than-at-Speed Delay Test Considering IR-Drop Effects. *ICCAD*, 2006.

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