On Path-Based Learning And Its Applications In Delay Test And Diagnosis *

Li-C. Wang
Department of ECE
UC-Santa Barbara

T.M. Mak
Intel Corporation
Santa Clara, CA

Kwang-Ting Cheng
Department of ECE
UC-Santa Barbara

Magdy S. Abadir
Motorola, Inc
Austin, TX

ABSTRACT

This paper describes the implementation of a novel path-based learning methodology that can be applied for two purposes: (1) In a pre-silicon simulation environment, path-based learning can be used to produce a fast and approximate simulator for statistical timing simulation. (2) In post-silicon phase, path-based learning can be used as a vehicle to derive critical paths based on the pass/fail behavior observed from the test chips. Our path-based learning methodology consists of four major components: a delay test pattern set, a logic simulator, a set of selected paths as the basis for learning, and a machine learner. We explain the key concepts in this methodology and present experimental results to demonstrate its feasibility and applications.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance Analysis and Design Aids

General Terms

Algorithm, Performance, Reliability

Keywords

Delay test, Statistical timing simulation, Machine learning

1. INTRODUCTION

With the advance to nanometer technologies (<130nm), circuit timing reflects many important sources of effects such as process variations, power noise, crosstalk, thermal effects, etc. [1, 2, 3]. These effects are hard to predict and model deterministically. For these effects, traditional discrete-value timing models become ineffective. Statistical timing analysis and timing simulation approaches are among the many that promise to better handle these deep sub-micron (DSM) timing effects [4]-[11].

Statistical timing analysis and timing simulation are at the core of the delay test and timing validation methodologies based on statistical timing models. The objective of statistical timing analysis is to improve the accuracy of critical path identification in a design cycle. For testing, it improves the selection of critical paths for delay test generation [12]. Statistical timing simulation can more accurately quantify the delays of patterns. Hence, it can be used as a tool for selecting high-quality delay test patterns from a given test set [13]. Moreover, it can be used to predict the expected silicon timing behavior in diagnosis and debug applications [14][15].

In a pre-silicon design environment, timing models may not be correct and 100% complete. Without an accurate timing model, results from timing analysis and simulation may be misleading. This means that we often need to produce a set of test chips, for example, to validate the speed paths on the silicon or derive the actual test clock frequency based on a set of delay patterns. Moreover, even with a reasonably accurate timing model, due to the increasing complexity in the timing models for DSM effects, statistical timing simulation can be orders-of-magnitude more expensive than logic simulation.

Because of the two issues above, our objective is to develop an alternative methodology that can complement the existing statistical timing analysis and simulation approaches. The core idea of this work is to utilize machine learning techniques [16, 17] to accomplish *path-based learning* originally outlined in [18], where timing behavior can be *learned* either from an accurate but slow statistical timing simulator, or from the behavior of a collection of test chips.

In this work, we use a statistical timing simulator developed in the past [14] which was recently enhanced in terms of its modeling capability and efficiency. Given a set of paths and a set of *training* patterns, we utilize path-based learning to develop an approximate *regression simulator* for the statistical simulator. Then, given another pattern set, the regression simulator can be applied to efficiently extract patterns with similar delay characteristics as those in the training set.

In the post-silicon phase, we assume that a set of test chips are available. By testing them with a pattern set and a given test clock, we can obtain their pass/fail behavior. Then, given a set of paths, our goal is to derive the most important (statistically significant) ones that are sufficient to *explain* the pass/fail behavior. In this process, machine learning is treated as an *explanation tool*. This process is similar to solving the *feature selection* (or *feature reduction*) problem described in machine learning literature [20, 21]. Although feature selection in general is a difficult problem, we will demonstrate that for path-based learning, the problem is not as hard as the general problem.

The primary purpose of this work has twofold: to demonstrate that path-based learning is indeed feasible and to show how it can be applied in test and diagnosis applications. In section 2, we give a brief introduction about the machine learning problems and an overview of our path-based learning methodology. In Section 3, we briefly describe the statistical timing simulator. Section 4.1 explains our path-based learning methodology and how a well-known machine learning technique called Support Vector Machine (SVM) [17] is incorporated in our learning framework. In Section 4.2, we demonstrate how path-based learning can be applied to derive critical paths based on the pass/fail behavior of test chips. Section 5.1 explains the methods for measuring the effectiveness of path-based learning. Section 5.2 presents experimental results to demonstrate the effectiveness of our path-based learning approach. Section 5.3 presents results on criti-

*This work was supported in part by National Science Foundation CCR-0312701 and in part by Semiconductor Research Corporation 1173.001.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2004, June 7–11, 2004, San Diego, California, USA.
Copyright 2004 ACM 1-58113-828-8/04/0006 ...$5.00.

cal path selection. Section 5.4 discusses the characteristics of selected paths. Section 6 concludes the paper and suggests future work.

2. THE PATH-BASED LEARNING SCHEME

In a typical Machine (statistical) Learning problem, we are given a collection of samples, each of the form $(\mathbf{X}, y)$ where $\mathbf{X} = [x_1, x_2, \ldots, x_n]$. "$n$" is called the *dimension*, and $(\mathbf{X}, y)$ is called a sample point (or a training sample) defined in an n-dimension *input space*. The relationship between $\mathbf{X}$ and y is through an unknown function f such that $y = f(X)$. The job of learning is to learn from a given m sample points (the training set): $(\mathbf{X_1}, y_1), (\mathbf{X_2}, y_2), \ldots, (\mathbf{X_m}, y_m)$ in order to statistically deduce an estimation f_{est} for f. This is called *Supervised Learning* [16].

After the learning, the established f_{est} serves as a predictor for f. Given another target point $\mathbf{X_{m+1}}$ not in the original training sample set, $f_{est}(\mathbf{X_{m+1}})$ can be used to predict the true value y_{m+1}.

In a *classification* problem, $f(X) \in G$ where G is a set of finite elements. The goal is to find an f_{est} to minimize the *expected probability* that $f_{est}(X) \neq f(X)$ for any given point X (not just for the samples used in the training set).

Given an n-dimensional input space for learning, each dimension defines a *feature*. Given n features for a learning problem and a learning algorithm, the *feature selection* problem is to derive an optimal subset of features for best learning performance [20, 21].

We selected SVM as our learning engine because of its capability to efficiently handle learning problems with rather large dimensions [16, 17]. In our path-based learning examples, the number of paths, which defines the dimension of input space, can be in the order of thousands. Hence, SVM is more suitable for path-based learning than other approaches.

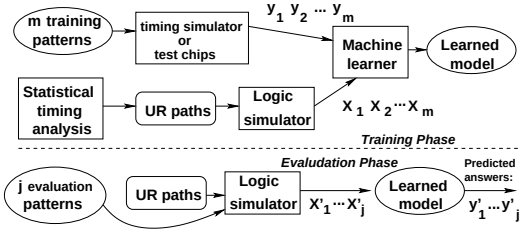


Figure 1: The path-based learning methodology

Figure 1 illustrates our path-based learning methodology. In the *training phase*, a set of m sample points are created based on a given training set of m patterns. For each pattern i, the Y_i value corresponds to the probability of failing a chip based on a given test clock when the pattern is applied. Hence, if the pattern fails p out of total q chips, then this probability is $\frac{p}{q}$. This probability can either be obtained by performing statistical timing simulation or be calculated based on a set of sample test chips.

The UR (Universal Representative) path set is derived based on our false-path-aware statistical timing analysis tool [7]. Given a cut-off clock, these are paths whose probabilities of exceeding the clock are not zero. Suppose $|UR| = n$. For a pattern j, $X_j = [x_{j1}, \ldots, x_{jn}]$ where each x_{jk}, $1 \leq k \leq n$, indicates if it is possible for the UR path k to "influence" the output delay of pattern j. This is calculated through the logic simulator and will be described in detail in Section 4.1.

After the learning, we have a regression simulator consisting of three components: the UR path set, the logic simulator, and the SVM learned model. Then, in the *evaluation phase*, the regression simulator relies on these three components to *predict* the desired answers. If the objective is to approximate the statistical timing simulator, then the evaluation pattern set can be different. If the objective is to *explain* the failing behavior of the test chips, we will use the same pattern set as the evaluation set. Then, the goal is to determine if the given UR path set is sufficient for developing a good SVM model that can accurately predict the collective behavior observed on the test chips.

3. THE STATISTICAL TIMING SIMULATOR

In our experiments, statistical timing simulation serves two purposes: (1) For the pre-silicon experiments, it is the timing simulator that the regression simulator intends to approximate. (2) For the post-silicon experiments, it simulates the production of test chips where each chip has a different (assumingly unknown) delay configuration.

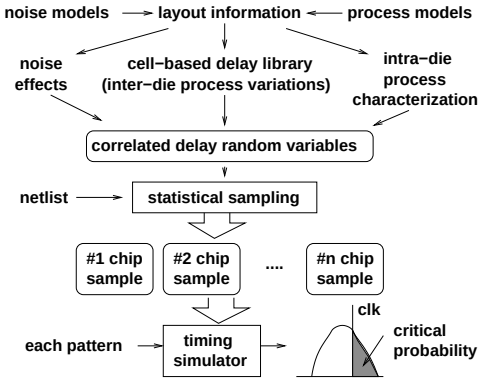


Figure 2: Illustration of the statistical timing simulator

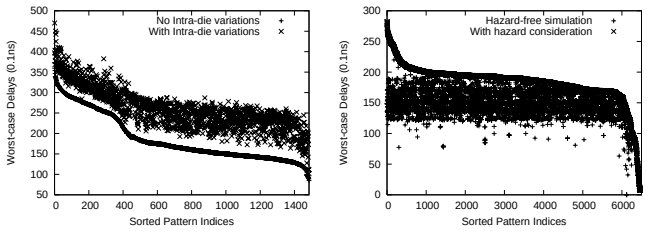
Figure 2 illustrates the flow of the statistical timing simulator design. The DSM timing effects are modeled as *correlated delay random variables*. Internally, the simulator produces n chip samples whose timings are statistically drawn from the timing model defined with the delay random variables. By simulating these n samples with each pattern from a given pattern set, every pattern's output delay distribution can be formed. Then, based on a given clock, the simulator can calculate the *critical probability* of every pattern, which estimates the probability of each pattern's delay exceeding the clock.

The abstract layer of delay random variables provides an interface to the detailed models in order to avoid the high cost of timing simulation directly at the layout level. The simulator utilizes the behavior from the n simulated sample chips to approximate the behavior of the real chips in production.

The statistical timing simulator assumes pin-to-pin delay random variables. The timing models are cell-based, and interconnects' delays can be included for consideration. In our experiments, the delays of cells/interconnects are modeled as correlated random variables with supposedly known probability density functions (PDFs). For experimental purpose, these PDFs were obtained using a Monte-Carlo-based SPICE simulator (ELDO) [25]. The cell-based timing model requires pre-characterization of cells, i.e., building libraries of pin-to-pin delays and output transition times (as random variables) [6]. We extracted the cells' pin-to-pin delay PDFs from a 0.25μm, 2.5V CMOS technology.

3.1 Concerns in accuracy and speed

The accuracy of statistical timing analysis and timing simulation depends on the accuracy of the timing model in use. To illustrate this point, Figure 3 shows statistical timing simulation results. These are results of simulating 1000 sample chips. In this figure, we plot the *3σ worst-case delay* achieved by each pattern (the delay distribution is constructed based on all 1000 samples). In Figure 3-(a), the lower curve is obtained by assuming no intra-die variations. Only inter-die variations are present. Above the curve are the corresponding delay values (as "$\times$") obtained by including intra-die variations. To model the intra-die variations, we adopt a simple multi-layer hierarchical model proposed in the literature [11]. The benchmark is ISCAS85 C5315. The patterns are ordered according to the lower curve.



(a) The impact of intra-die process variations, C5315

(b) The impact of hazard assumptions in simulation, s5378

Figure 3: The impact of modeling and simulation assumption

On the other hand, Figure 3-(b) compares the worst-case delays by changing the assumption of how to handle *timing hazards* in the simulation [22]. The upper curve shows the delays where hazards are considered in the simulation. Therefore, the worst-case delays represent the time points at which signals are *for sure to be stable*. Below the curve are delays where hazards are not taken into account. The benchmark used to obtain the results in this figure is ISCAS89 s5378.

Figure 3-(a) and Figure 3-(b) demonstrate that the timing model and simulation assumption can greatly affect the simulation results and consequently, affect the validity of delay test and validation methodologies that utilize these results.

All the experiments in this work were run on a P3 1GHz machine under Linux 2.4.3. In the case of C5315, the run time is about 7.32 seconds per pattern. In the case of s5378, the rum time is about 12.65 seconds per pattern. Although these run times are not unreasonable, the efficiency can be a potential concern when the number of simulated patterns is large.

4. THE PRINCIPLE OF OUR WORK

4.1 Learning delays based on paths

In our path-based learning, the exact timing model and simulation assumption employed by the statistical timing simulator is assumed to be unknown. As described in Figure 1 before, the learning is based on the output behavior of the simulator. This is a necessary assumption because one of the objectives for designing the learning scheme is that it can also be applied to learn from a set of sample test chips.

The intuition behind path-based learning is that extreme delays causing failure are the results of sensitization of some long paths. Hence, it is possible to correlate the timing of a long-delay pattern to the paths that it sensitizes. However, logic simulation cannot fully determine if a path is sensitized [24]. Timing hazards [22] can be another source of concern. Since chip delays are statistical, deterministic analysis of path sensitization for a given pattern and a given chip can be a very complex process [7, 14].

Our path-based learning scheme avoids the complication in the analysis of path sensitization in the statistical domain. The complicated relationships between paths and statistical delays are what we intend to uncover through the machine learning process. The path-based learning is accomplished in three steps:

1. Extract UR paths: The input space is defined based on a set of paths called *Universal Representative Paths* (or **UR paths**). UR paths can be extracted using the false-path-aware statistical timing analysis tool [7]. Given a cut-off clock, UR paths are those paths whose critical probabilities are not zero. The size of the UR path set is the dimension of the input space for SVM learning.

It is important to note that the timing model used in the timing analysis tool and the timing model used in the statistical timing simulator can be different. In fact, in most of our experiments, we intentionally make them different.

Given an UR set with n paths, it may be the case that only a few of them are actually statistically significant for the learning. Hence, in *path selection* (Section 4.2), we can use SVM to deduce these statistically significant paths.

2. Construct training sample points: We denote a sample point as $(\mathbf{X}, y)$. Given an UR path set with n paths, $\mathbf{X}$ is an n-dimensional vector. Let $[u_1, u_2, \ldots, u_n]$ be the UR paths. Let $[a_1, a_2, \ldots, a_n]$ be the mean delays of these paths, calculated using the statistical timing analyzer [7]. For a given pattern P_i, we define $\mathbf{X}_i = [x_1, x_2, \ldots, x_n]$ as the following.

For each x_j, let $x_j = a_j$ if path u_j is functionally sensitized by the pattern p_i. Otherwise, $x_j = 0$. We note that, this path sensitization can be decided using logic simulation and hence, is a much faster process than statistical timing simulation. Also note that a path is functionally sensitized by a pattern if (1) there are transitions on all *on-inputs* of the path, and (2) all *side-inputs* of the path have non-controlling values from the second vector of the given pattern [24] for all gates whose corresponding on-inputs also have non-controlling values from the second vector. Functional sensitized paths are those that have a chance to affect the output delay of a given chip, depending on the delay configuration on the chip. We further note that functional sensitization is not the only way to define path sensitization [24].

For example, suppose the UR paths are $[u_1, u_2, u_3]$. Using the statistical timing analyzer, the mean delays for these paths are $[12.5,\ 10,\ 11.4]$, respectively. The training patterns are P_1, P_2. Through logic simulation we obtain that P_1 sensitizes u_1, u_3, and P_2 sensitizes u_2, u_3. Then, $X_1 = [12.5, 0, 11.4]$ and $X_2 = [0, 10, 11.4]$ for P_1 and P_2, respectively.

To obtain Y_1 and Y_2, we can simply use the critical probabilities p_1, p_2 from patterns P_1 and P_2, calculated by the statistical timing simulator or from the test chips. However, since SVM is most effective for classification, we further convert the critical probabilities into groups. We divide probability values into ten groups:

Group 9 includes all patterns whose critical probabilities ≥ 0.9.

Group i (for $0 \leq i \leq 8$) consists of all patterns whose critical probabilities fall into the range $[0.i, 0.i + 0.1)$.

For example, through statistical timing simulation, suppose that we obtain the critical probability 0.82 for P_1, and the critical probability 0.93 for P_2. Then, $Y_1 = 8$ and $Y_2 = 9$.

Treating our learning problem as a classification problem has another advantage. When measuring the learning accuracy, we can obtain the number of errors made in the classification, instead of a *mean-square* error value as that in the regression case. This simplifies the evaluation process.

3. Apply SVM: Given m patterns, either through statistical timing simulation or through testing a set of test chips, we can obtain m training sample points. Then, SVM learning can be applied to develop a learning model. In SVM, there are choices of *kernels*. Different kernels may be suitable for different problems. In our experiments, we use the *Gaussian Kernel* [17]. Our experience indicates that using other kernels is not as effective as the Gaussian kernel. The standard deviation (STD) parameter required in the Gaussian kernel could be pre-computed based on the set of sample points [23].

4.2 Feature path selection

In the post-silicon phase, we can use the path-based learning scheme as a tool for path selection. Figure 4 shows the methodology.

Similar to regression simulation that can be applied either with the statistical timing simulator or with a collection of test chips, the path selection approach can also be applied with both. However, here we assume that the objective is to search for an optimal set of paths such that based on a given set of patterns the pass/fail behavior of the test chips can be statistically *explained* through the SVM learning. Be-

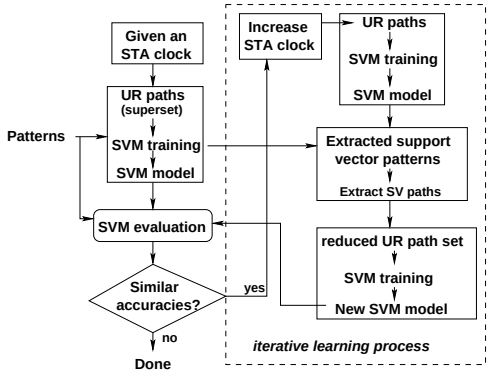


Figure 4: Path-based learning as a path selection tool

cause of this, in path selection we utilize the same pattern set as both the training pattern set and the evaluation pattern set.

As mentioned before, the path selection problem is similar to the *feature selection* problem in machine learning [20, 21]. Feature selection in general is a very difficult problem because the search space is exponential in terms of the number of features. In our case, the search space can be restricted by assuming that short paths defined in the statistical timing analysis are less likely to influence the pass/fail behavior than the long paths. The effectiveness of path reduction in Figure 4 is based on this assumption.

Starting with an STA clock, we first construct a *superset* of paths for SVM training. After the training, SVM will produce a set of *support vector* patterns (SVs). These support vectors correspond to the patterns that are critical to define the SVM model [17, 23]. Based on these SV patterns, we extract their corresponding SV paths, i.e. only paths functionally sensitized by these SV patterns. Other paths are removed from the set. The result is a *reduced UR path set*.

We train SVM with the reduced UR path set again and obtain a new SVM model. Then, we compare the two SVM models, one based on the superset and the other based on the reduced UR set. If the accuracies are similar, we proceed with *iterative learning* by increasing the STA clock setting. In each pass of the iterative learning process, we obtain a smaller UR path set that results in a smaller reduced UR set. The iteration continues until the reduced UR set cannot produce a similar accuracy as the original superset.

The search can be implemented as a binary search although in the figure, it is illustrated as a linear search. In our path selection approach, instead of searching for the optimal subset of paths, we search for the optimal setting of the STA clock. We note that the effectiveness of path removal can be influenced by the accuracy of the statistical timing analyzer. For example, if many short paths defined by the timing analyzer are actually long paths on the test chips, then we might not be able to remove many paths from the superset, without losing much in terms of the SVM prediction accuracy.

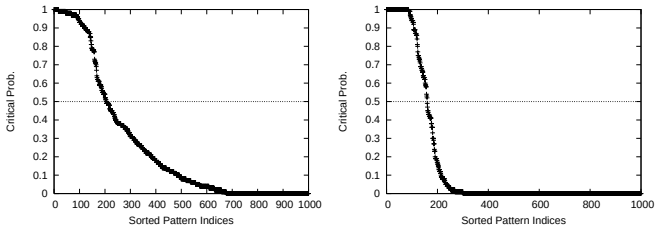
5. EXPERIMENTAL RESULTS

5.1 Measuring the effectiveness

For the purpose of training a regression simulator, we apply the statistical timing simulator to simulate 100 sample chip instances based on a training pattern set T_1. (Note that this number 100 was arbitrarily chosen so that the statistical simulation could finish within a reasonable time). Then, we apply the following two methods to evaluate the effectiveness of the learning.

We compare the results answered by the statistical timing simulator and the results answered by the regression simulator based on the same test set T_1 and simulation of a larger number of sample chip instances. Usually, this large number is set at 800. Also, we compare the results answered by the two simulators based on a different pattern set T_2 and simulation of 100 sample chip instances.

Figure 5-(a) shows the resulting critical probabilities from a set of patterns that were generated based on a critical path delay model. In this model, timing critical paths are first selected based on traditional static timing analysis using the worst-case delay model (3σ bounds). Then, non-robust tests are produced whenever it is possible by an ATPG [24]. The benchmark is the ISCAS89 C880. The clock used to calculate the critical probabilities is 19.5ns. In most of our experiments, we use this method to produce the T_1 set.



(a) T_1 set (training) (b) T_2 set (evaluation)

Figure 5: Illustration of pattern classification

In contrast, Figure 5-(b) shows the critical probabilities for another set of patterns that were generated based on a path-oriented transition fault model [19]. In this model, a test is produced through the longest propagation path from each fault site. Since the timing length of a propagation path is from each fault site to a primary output, not from a primary input to a primary output, these patterns tend to have smaller critical probabilities (shorter delays). However, we note that this is true for most of the patterns, not for all of them. In most of our experiments, we use this method to produce the T_2 set.

5.2 Results on regression simulation

Table 1: Some statistics from the experiments

	UR Path Set		run times (sec)			Pattern set
Circuit	clock	size	Learning	Reg. Sim.	Stat. Sim.	Size
C880	18.3ns	2103	58.05	3.34	1270.24	1000
C1355	22.8ns	505	220.81	5.12	4803.06	2000
C2670	33ns	2253	591.5	21.47	8283.27	1590
C7552	30.8ns	971	337.86	211.51	87666.4	9714
s1488	14ns	703	4.65	1.3	1695.26	1000
s5378	18.5ns	2102	205.98	74.01	74318.2	7345
s9234	40.5ns	290	226.31	61.55	152892	7974
s38417	39.5ns	2880	639.15	350.04	452032	7800

Table 1 shows related data from the experiments. The clocks are for the statistical timing analysis to construct the UR path sets. The sizes of the UR path sets are shown next to the clocks. The sizes depend on the selected clocks and also on the characteristics of the circuits. For example, with s9234, there are only 290 paths in the set. This is because s9234 has 290 paths whose delays are much longer than others. The sizes of the pattern sets are shown in the last column. The run times of the statistical simulator are based on 800 chip instances.

We note that the learning time excludes the statistical simulation time of the 100 samples. The regression simulation times are from the logic simulation only. The SVM machine learner usually runs in minutes. SVM run time depends less on the circuit size and more on the SVM model complexity [17].

Table 2 shows the accuracy results from various experiments. Give a pattern p, let $G_{sta}(p)$ be the group number answered by the statistical timing simulator. Let $G_{reg}(p)$ be the group number answered by the regression simulator. In the table, we compute the accuracy by counting the percentage where $G_{sta}(p) = G_{reg}(p)$.

In addition to the pattern sets T_1 and T_2 described before, we also include two multiple-detection transition fault pattern sets (TR_{15} and TR_{10} for 15-detection and 10-detection) produced by a commercial ATPG tool.

The "STA" column denotes the average worst-case circuit delays calculated by the statistical timing analysis tool. In these experiments,

Table 2: Prediction accuracy from various experiments

Circuit	Train	Eval.	Clk	Accu.	Train	Eval.	Clk	Accu.	STA
C880	T_1	T_1	19	97.67 %	T_1	T_2	20	98.25%	23.28ns
C1355	T_2	T_2	19.9	96.88%	T_2	TR_{15}	19.9	96.20%	22.88ns
	T_2	TR_{15}	20.2	98.52%					
C2670	T_1	T_1	29.9	95.29%	T_1	T_2	29.5	99.14%	35.59ns
C7552	T_1	T_1	31	97.72%	T_1	T_2	30	94.15%	32.58ns
s1488	T_1	T_1	24	99.8%	T_1	T_2	24	98.4%	30.09ns
s5378	TR_{10}	TR_{10}	24	99.58%	TR_{10}	T_2	24	98%	24.94ns
	TR_{10}	T_2	23.5	97.81%	TR_{10}	T_2	20	94.18%	
	T_1	TR_{10}	20	90.1%					
s9342	T_1	T_1	37	98.42%	T_1	T_2	37	96.11%	41.08ns
s38417	T_1	T_1	41	96.85%	T_1	T_2	41	93.83%	40.27ns

we intentionally used, in the timing analysis, a timing model different from the one used in the statistical timing simulator (up to 15% difference on each pin-to-pin delay). This is why the "STA" delays could be shorter than the test clocks (the "clock" column). Otherwise, the worst-case circuit delay from statistical timing analysis should be always longer than the worst-case delays from simulating patterns.

The usefulness of regression simulation is clearly demonstrated by the results in these two tables. After training, a regression simulator can run much faster than the statistical timing simulator and produce consistent results with high probabilities.

Results in table 2 (where training set and evaluation set are different) suggest that in the pre-silicon phase, regression simulation can be used for fast construction of a pattern set with desired delay characteristics. For example, our goal may be to obtain a superset of statistically long delay patterns so that other criteria (such as logic conditions for wire coupling) can be applied afterwards in analysis to select patterns for a particular test application. Our methodology to obtain the superset can be the following. First, we produce a smaller training set and use statistical timing simulator to characterize the delays of these patterns. Then, we continue to produce more patterns and use the regression simulator to characterize the delays of those patterns. At the end, we collect all patterns falling into group 9 to form the superset. In other words, after the training, regression simulation can help to quickly identify additional patterns that have similar delay characteristics as those long-delay patterns identified by the statistical timing simulator. This avoids the high simulation cost in the process.

5.3 Results on critical path selection

Results in table 2 (where training set and evaluation set are the same) also indicate that our proposed path selection method can be feasible. Given m patterns, we use the statistical timing simulator to produce 200 test chips. Then, based on these test chips, we obtain their pass/fail behavior. In most cases, some patterns fail many more chips than others. Hence, we focus on selecting paths to explain the behavior based on those *worst-case* patterns. In other words, in our experiments we focus on explaining the *tail* of the pass/fail behavior, not the average.

Based on the critical probabilities of the original m patterns, we first select M patterns that have largest probability values (In Figure 5, these would be the leftmost M patterns). Then, we select a test clock and a cut-off probability boundary such that we can divide these M patterns into two groups with roughly equal sizes. The group 0 has those patterns with higher probabilities of failing a chip, and group 1 has the remaining patterns. We note that by using M patterns instead of the total m patterns, the learning problem becomes more specific and hence, harder (w.r.t. getting close to 100% result) because the new problem usually has much fewer sample points ($M \ll m$).

The SVM is then used as a *binary classifier*. Here, instead of giving the SVM a 10-group problem to solve as that in the regression simulation experiments, we focus on solving the binary cases. From another perspective, we utilize SVM to answer the following question: What are the critical paths that can better explain why the worst-case $M/2$ patterns fail more chips than other patterns?

Figure 6 illustrates the path selection results based on $M = 100$ and

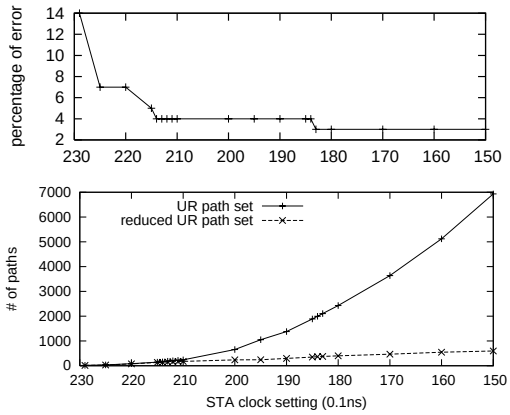
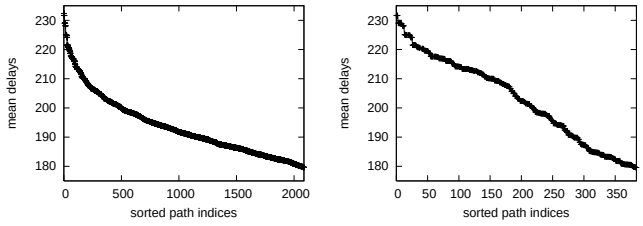


Figure 6: Illustration of path selection (C880)

the T_1 pattern set for circuit C880. It is interesting to observe that the number of UR paths grow exponentially as the STA clock decreases. However, the number of reduced UR paths grows very slowly. With the initial UR path set, the learning results in 3% of errors (3 errors because $M = 100$). This corresponds to STA clock 150 with the UR set size 6932. With STA clock 183 and a reduced UR path set size 377, we can maintain this accuracy. Hence, these 377 paths (out of the 6932 paths) are identified to be the critical ones to explain why the worst-case 50 patterns fail more chips than the other patterns.

If we allow 4% of errors, the iterative learning process (Figure 4) can further reduce the path set to 137 critical paths. In other words, these are 137 statistically significant paths sufficient to explain the pass/fail behavior to the desired accuracy of 96%. It is obvious that if we allow more errors, the number of paths required becomes smaller. However, notice in the figure that as we keep increasing the STA clock, at some point we may encounter a sudden increase in the number of errors. Note that this is true in all cases that we have experimented.



(a) original UR set (b) reduced UR set

Figure 7: Mean delay comparison (STA clock $= 183$)

It is interesting to find out what the delay distribution of the reduced UR path set looks like. Figure 7 compare the delay distributions between the 2103 UR paths (STA clock 183) and the 377 reduced UR paths, based on their mean delays calculated by the statistical timing analyzer. It is interesting to observe that the 377 reduced UR paths are not necessary just the long paths. These results imply that simply using long paths defined by the statistical timing analyzer to uncover the reason why the 50 patterns fail more chips can be misleading.

Table 3: Results on path selection

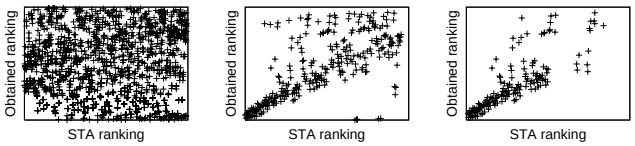
Circuit	M	test	Superset		Reduced UR sets		
	number	clock	size	accu.	size/same accu.	size	accu.
C880	307	21	1379	99.7%	436	224	99.05%
C1355	96	19.9	3389	100%	452	319	97.92%
C1355	212	19.9	3389	100%	1174	533	97.16%
C2670	516	29.9	4696	95.16%	833	568	92.45%
C7552	215	32	4323	94.89%	2339	650	87.05%
s1488	79	14	703	99.82%	16	16	99.82%
s5378	237	17	4138	100%	444	384	99.65%
s5378	260	17	4138	100%	917	447	98.18%

Table 3 summarizes the path selection results for other experiments.

The pattern sets used in these experiments are those *training sets* used in the experiments in Table 2. The "size/same accu." column are the sizes of the reduced UR path sets that can deliver the same accuracy results as the initial UR path supersets. The last two columns show further path reductions by allowing small decreases in accuracy.

5.4 The characteristics of selected paths

To further understand the characteristics of selected paths, we conduct the following experiments. We utilize a simple post-silicon path ranking heuristic to derive path ranking based on the behavior of sample test chips. First, we assume that the average delay of a pattern on the sample chips can be determined by applying it with a sequence of test clocks. Given a path P, suppose it is functionally sensitized by patterns $p_1, p_2, \ldots, p_i$ whose average delays are determined as $d_1, d_2, \ldots, d_i$, respectively. Then, we associate delay $\frac{(d_1+\cdots+d_i)}{i}$ to path P. We use these associated delays to rank paths in a given UR path set and obtain a post-silicon path ranking. On the other hand statistical timing analysis, using its average delays, can also derive a path ranking. If our sample chips behave close to that predicted by the timing analyzer, then the two rankings should correlate well.



(a) 1272 paths (b) 268 paths (c) 155 SV paths

Figure 8: STA path ranking Vs. post-silicon ranking, C880

Based on an initial UR path set with 1272 paths, Figure 8-(a) compares the path ranking (x-axis) computed using average delays from statistical timing analysis to the post-silicon path ranking (y-axis) computed based on the behavior of the sample chips. For example, a path with the two rankings as 100 and 200 would appear as a point with the coordination (100,200). As it can seen, there is almost no correlation between the two rankings in Figure 8-(a).

Then, we apply our iterative learning method to determine that, out of these 1272 paths, 268 paths are sufficient to obtain the same learning accuracy. These 268 paths are obtained by increasing the STA clock without considering SV-pattern path reduction yet (see Figure 4). Figure 8-(b) shows the correlation results based on the 268 paths. Figure 8-(c) shows results by removing those paths not sensitized by the SV patterns from Figure 8-(b). Hence, these 155 paths are the final reduced UR paths from our path selection method.

From Figure 8-(a) to (b), we observe that our iterative learning approach actually select paths where they can be better (and more easily) correlated with the results from the statistical timing analysis. Also observe from figure (b) to (c), that focusing only on SV paths can further improve the correlation by removing many noisy points. These results suggest that the proposed iterative learning methodology can serve as a path filtering tool so that another path-based methodology (like path ranking) can be applied more easily on the resulting path set for validation against the pre-silicon statistical timing analyzer.

6. CONCLUSION AND FUTURE RESEARCH

This paper describes the implementation of a path-based learning methodology and explains its potential applications. We demonstrate that path-based learning is feasible, and can be an attractive approach to provide effective solutions to complement the existing statistical timing tools and methodologies.

As one of its early works along this line of research, this paper raises several interesting questions for future research: (1) It is obvious that the effectiveness of learning depends on the initial UR path set. Given a circuit, how to derive an optimal UR path set for learning remains an interesting question. (2) For regression simulation, the quality of the training pattern set is crucial. How to derive a good-quality and minimal training pattern set can be another interesting research topic as well. (3) Our path selection methodology is preliminary, which treats SVM as a black-box tool. How to revise SVM algorithms for better feature path selection requires further research in machine learning. (4) Finally, to provide a complete post-silicon validation and diagnosis methodology, we need to develop more sophisticated methods that can further analyze the selected paths as explained in Section 5.4.

We note that our iterative learning path selection approach is not a complete solution for post-silicon validation and diagnosis. However, it can be the first step to filter out statistically unimportant paths based on the observed pass/fail behavior from a set of sample test chips. This reduces the search space dramatically so that other deterministic path-based analysis can be applied more effectively. Moreover, our path selection approach utilizes test chips as the basis for learning and hence, avoids the potential problem of using an inaccurate timing model in the pre-silicon simulation environment.

7. REFERENCES

[1] M. A. Breuer, C. Gleason, and S. Gupta. New Validation and Test Problems for High Performance Deep Sub-Micron VLSI Circuits. *Tutorial Notes, IEEE VLSI Test Symposium*, April 1997.

[2] Anne Gattiker, et. al. Timing Yield Estimation from Static Timing Analysis. *Proc. IEEE ISQED* 2001, pp. 437-442

[3] Sani R. Nassif. Modeling and Analysis of Manufacturing Variations. *Proc. IEEE Custom Integrated Circuits Conference*, 2001, pp. 223-228

[4] D. R. Tryon, F. M. Armstrong, and M. R. Reiter. Statistical Failure Analysis of System Timing. *IBM Journal of Research and Development*, 28(4):340–355, July 1984.

[5] H.-F. Jyu, S. Malik, S. Devadas, and K. Keutzer. Statistical Timing Analysis of Combinational Logic Circuits. *IEEE Transactions on VLSI Systems*, 1(2):126–137, June 1993.

[6] J.-J. Liou, A. Krstić, K.-T. Cheng, D. Mukherjee, and S. Kundu. Performance Sensitivity Analysis Using Statistical Methods and Its Applications to Delay Testing. *Proc ASP-DAC* 2000, pp. 587-592

[7] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng. False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation. *ACM/IEEE Design Automation Conference*, June 2002.

[8] Michael Orshansky and Kurt Keutzer. A general probabilistic framework for worst case timing analysis. in *Proc. DAC*, 2002, pp. 556-561.

[9] Anirudh Devgan and Chandramouli Kashyap. Block-based Static Timing Analysis with Uncertainty. in *Proc. Digest of papers, ICCAD 2004*, pp 607-614.

[10] Chandu Visweswariah, Kaushik Ravindran, and Kerim Kalafala. First-Order Parameterized Block-Based Statistical Timing Analysis. in *ACM/IEEE TAU workshop*, 2004, pp. 17-24.

[11] A. Agarwal, D. Blaauw, V. Zolotiv. Statistical timing analysis for intra-die process variations with spatial correlations. in *Proc. ICCAD*, 2003, pp. 900-907.

[12] J.-J. Liou, L.-C. Wang, and K.-T. Cheng. On Theoretical and Practical Considerations of Path Selection For Delay Fault Testing. *Proc. International Conference on Computer-Aided Design*, Nov. 2002.

[13] Mango C-T. Chao, Li-C. Wang, Kwang-Ting Cheng. Pattern Selection for Testing of DSM Timing Defects. in *Proc. DATE*, March 2004, pp. 1060-1065

[14] Angela Krstic, Li-C. Wang, Kwang-Ting Cheng, Jing-Jia Liou, Magdy S. Abadir. Delay Defect Diagnosis Based Upon Statistical Timing Models – The First Step. in *Proc. DATE*, 2003, pp. 328-323

[15] Angela Krstic, Li-C. Wang, Kwang-Ting Cheng, T. M. Mak. Diagnosis-Based Post-Silicon Timing Validation Using Statistical Tools and Methodologies in *Proc. ITC*, 2003

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning - Date Mining, Inference, and Prediction. *Springer Series in Statistics*, 2001

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

[18] Li-C. Wang. Regression Simulation: applying path-based learning in delay test and post-silicon validation in *Proc. DATE*, March 2004, pp. 692-693.

[19] Kai Yang, et. al. TranGen: A SAT-Based ATPG for Path-Oriented Transition Faults. in *Proc. ASP-DAC*, Jan 2004.

[20] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 10:245V271, 1997

[21] R. Kohavi. Wrappers for feature subset selection. *Artificial Intelligence*, special issue on relevance, 97:273V324, 1995.

[22] Haluk Konuk. On Invalidation Mechanisms for Non-Robust Delay Tests. in *Proc. International Test Conference* 2000, pp. 393-399

[23] http://www.torch.ch

[24] A. Krstić and K.-T. Cheng. *Delay Fault Testing for VLSI Circuits*. Kluwer Academic Publishers, Boston, MA, 1998.

[25] Anacad. Eldo v4.4.x User's Manual. 1996.