Design-Silicon Timing Correlation — A Data Mining Perspective *

Li-C. Wang
Univ. of CA - Santa Barbara
licwang@ece.ucsb.edu

Pouria Bastani
Univ. of CA - Santa Barbara
bastanip@ece.ucsb.edu

Magdy S. Abadir
Freescale Semiconductor, Inc
M.Abadir@freescale.com

ABSTRACT

In the post-silicon stage, timing information can be extracted from two sources: (1) on-chip monitors and (2) delay testing. In the past, delay test data has been overlooked in the correlation study. In this paper, we take path delay testing as an example to illustrate how test data can be incorporated in the overall design-silicon correlation effort. We describe a path-based methodology that correlates measured path delays from the good chips, to the path delays predicted by timing analysis. We discuss how statistical data mining can be employed for extracting information and show experimental results to demonstrate the potential of the proposed methodology.

Categories and Subject Descriptors: B.8.2 [Hardware]: Performance and reliability

General Terms: Algorithms, Performance and Reliability

Keywords: Statistical Timing, Learning, Correlation, Timing, Test

1. INTRODUCTION

As feature sizes of device and interconnect continues to shrink, design behavior becomes more sensitive to process and environmental variations and uncertainties. Consequently, pre-silicon modeling and simulation alone may not be sufficient to answer all design-related questions. For some questions left unanswered, a common practice is to analyze the behavior of first-silicon chip samples. For example, it is difficult to predict the actual speed-limiting paths in a high-performance processor. Hence, speed-path identification is usually done by analyzing silicon samples. These paths are often different from the critical paths estimated by a timing analyzer [1].

During a design process, there can be many effects not modeled and simulated accurately. Each effect may or may not significantly impact silicon behavior. One commonly-asked question is among those effects which ones cannot be ignored. The answer to a question of such can be design dependent, design methodology dependent, and process technology dependent.

In the past, the magnitude of mismatch between simulated behavior and actual behavior is relatively small. Unless our design goal is to push for extremely high performance, the miscorrelation is often not analyzed in detail. A common practice is to classify a chip as defective if its behavior is far from the norm. Then, we utilize various diagnosis methods to pin-point the potential locations that may cause the unexpected behavior, followed by a rather tedious silicon debug process to uncover the root cause(s). Historically, unexpected chip behavior is assumed to be mostly due to manufacturing defects. Hence, diagnosis and silicon debug methods are optimized to look for defects. These methods analyze chips individually and the analysis is carried out on (suspected) failing chips only.

Diagnosis and silicon debug [2] [3] [4] [5] can be seen as the traditional ways to extract information from silicon. If our goal is to extract design-related information, a typical approach is to look for *systematic* failures. For example, a weak gate may cause a collection of chips to fail timing in a similar way. Diagnosis and debug on a few of these chips individually may uncover the location(s) of the weak gate and consequently help to improve the current design.

For helping design, analyzing failure data makes sense if (1) we have a collection of failing chips that fail in systematic ways and (2) we can afford to fix the current design. From this perspective, diagnosis and silicon debug can be used to recover significant yield loss due to systematic design errors. This is feasible if we can afford the cost associated with (1) silicon debug that is usually a tedious process involving the use of expensive equipment and (2) design respin that can be overly expensive in some cases.

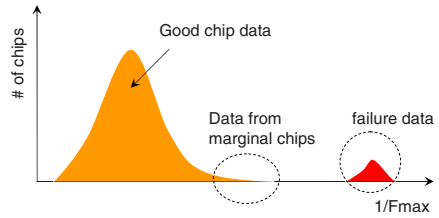


Figure 1: Three categories of chips to analyze

For studying design-silicon correlation, failure data should not be the only place to look for information. Figure 1 illustrates three categories of chips that may be analyzed. When analyzing a collection of supposedly good (and marginal) chips, the data can be inherently statistical due to process and environmental variations and uncertainties. Moreover, the number of chips to be analyzed can be large and it would be more effective to analyze their behavior collectively rather than individually. In this case, traditional diagnosis and debug methods can be ineffective.

When analyzing failing chips, the objective is clear, that is to identify the cause(s) of the failures. When analyzing good and marginal chips, the objective can vary. For example, the goal can be to validate certain assumptions in a timing model. One common approach is to place on-chip monitors in various locations of a die. For exam-

*This work is supported in part by National Science Foundation, Grant No. 0541192 and Semiconductor Research Corporation, project task 1585.001.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

ple, process monitors are for checking certain low-level parameters such as L_{eff}, V_{th}. On-chip voltage monitors can be used to check IR drop. Test structures, such as ring oscillators, have been used to monitor integrated circuit performance for many years [6, 7]. Test structures are primarily designed to provide a measure of performance, power and variability of the current design process. The data measured from test structures relates these measures to the properties of low level device parameters, in particular to MOSFETs and to parasitic delay elements [8, 9].

Ring oscillators have several beneficial features. They take up minimal area and can be placed in small open spaces in a design. They can be directly measurable by a test probe to minimize test measurement error. Because ring oscillators are simple circuitry, there are aspects of design that cannot be studied by the methodology. For example, margins are often added at various stages of a timing analysis flow. The impact of these decisions on silicon timing usually cannot be measured by using only on-chip monitors.

In addition to on-chip monitors, delay testing can also be used to study silicon timing behavior. For studying silicon behavior, a separate testing methodology is often required in order to support the collection of test data that contain the required information for the analysis. This delay testing methodology is different from the one used in production testing. As illustrated in Figure 2, this methodology is to test for information. Tests are usually more comprehensive and provide higher resolution. In contrast, a production delay testing methodology is often optimized for cost and for defect-screening capability. The size of the test pattern set is an important consideration. The number of test clocks may be strictly limited.

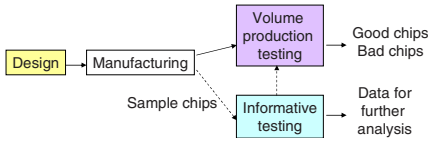


Figure 2: Informative testing is different from production testing

For example, consider production delay testing where a test clock is pre-determined. A chip is defective if its delay on any test pattern exceeds this clock. In testing for information, test clock can be a programmable value. The goal can be to estimate the failing frequency of each test pattern targeting a specific critical path.

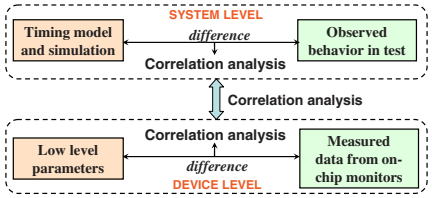


Figure 3: Three types of correlation analysis

Figure 3 gives an overall picture of design-silicon timing correlation that includes three types of correlation analysis. At the low-level, a methodology can be based on on-chip monitors. The targets of the analysis can be the within-die variations of device parameters, voltage, and temperature. These variations are usually large and considered as important factors to impact timing.

At the high-level, the methodology is based on delay testing. This type of correlation analysis has been overlooked in the past primarily due to the fact that delay testing traditionally is optimized for cost and for defect screening. Because using it for correlation analysis is rather new, the rest of the paper will focus on this type of analysis. In particular, the analysis will be based on the *differences* between predicted path delays from a timing analysis tool and measured path delays on a set of silicon samples. We focus on such a path-based analysis approach to avoid the complication of dealing with noises such as cross-coupling effects. Therefore, for a path to be included in the analysis, we require a test pattern that sensitizes only the path.

Figure 3 shows a third type of correlation analysis that tries to correlate the results between the high-level analysis and the low-level analysis. The development of this type of methodology needs to wait until the high-level and low-level methodologies are fully developed. However, in this paper when we discuss the high-level analysis, we will also show that the effectiveness of the analysis can be independent of the low-level parameter shifts.

2. AN INDUSTRIAL EXPERIMENT

This section describes an industrial experiment that studied the correlation between structural path delay testing (PDT) and a nominal static timing analyzer (STA). The STA is capable of producing a critical path report. This is a list of paths that the tool has determined having the least amount of timing slack with respect to a timing requirement. For late-mode analysis, the timing requirement is usually a setup-time constraint. From the critical path report, the individual cell delays, net delays, clock skew, setup-time and slack for the listed critical paths can be determined. Each path can be expressed as an equation:

$$STA_{delay} = \sum c_i + \sum n_j + setup = clock + skew - slack \quad (1)$$

c_i are the cell delays (including the launch flip-flop's delay) , n_j are the net delays, $clock$ is the clock period, and $skew$ is the clock skew. Structural path delay tests are generated to target paths from the STA's critical path report. The tester is programmed to search for an individual path delay test's maximum passing frequency. This measured path delay can be expressed as another equation:

$$PDT_{delay} = \sum \hat{c_i} + \sum \hat{n_j} + \hat{setup} = measured + \hat{skew} \quad (2)$$

In the above equation, the variables with hats are the actual delays that cannot be directly measured. What we measured from the tester was, $measured$ the minimum passing period (reciprocal of the maximum passing frequency). Note there is no slack variable, because at the minimum passing period, we assume the slack is zero.

Although we cannot directly measure some of these delay variables, in order to explain the difference $(STA_{delay} - PDT_{delay})$ on all paths, we make some simple assumptions.

$$\alpha_c * \sum c_i = \sum \hat{c_i} \quad \alpha_n * \sum n_i = \sum \hat{n_i} \quad (3)$$
$$\alpha_s * setup = setup \qquad skew = \hat{skew}$$

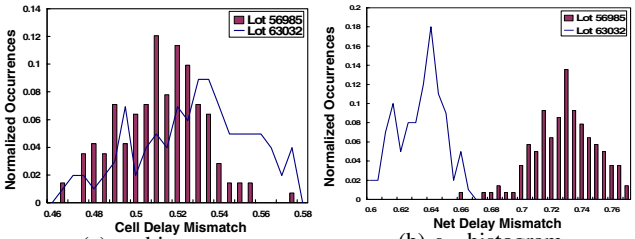
Essentially, for each chip we assume three constants $\alpha_c, \alpha_n, \alpha_s$ that can be thought as the correction factors for the lumped cell, lumped net and setup delays. These coefficients give a sense of where the mismatch between pre-silicon and post-silicon timing lies individually on each chip. In particular, α_c tracks the mismatch of cell characterization, α_n tracks the mismatch of the interconnect extraction and α_s tracks the pessimism of the setup time constraint of the flip flop. Due to the resolution of the testing, we decided not to have a correction factor on the skew.

We solve for the coefficients individually for each chip. This is an over-constrained system of equations as the number of paths is greater than the number of mismatch coefficients. This over-constrained system of equations can be solved in a least-square manner using Singular Value Decomposition to find the best fit.

2.1 Results on 240 microprocessor chips

The experiment was done based on 495 critical paths. These are latch-to-latch paths without passing through embedded memories. Results were collected on 240 packaged chips belonging to

two wafer lots manufactured several months apart. These chips are industrial high-performance microprocessors.



(a) α_c histogram (b) α_n histogram

Figure 4: Histograms of mismatch coefficients α_c, α_n

Figure 4 presents the distributions of α_c and α_n from the two lots. α_s distributions are similar to α_c distributions and hence, are not shown. The α_c and α_n results clearly show that STA are overly pessimistic, i.e. all coefficients are less than one. This may be partially because the chips were manufactured at a later point of the process, and the cell characterizations were done at an earlier point.

Unlike Figure 4-(a), the two distributions in Figure 4-(b) are separated apart. This indicates that net delays are more sensitive to the lot shift. The analysis is inconclusive because without proper tools and methodologies, we couldn't perform more detailed study. We note that equation 3 lumps all the effects into three parameters and these parameters are estimated on each chip individually. This is a very rough analysis. We would need a more sophisticated method to study the data further.

3. MODEL-BASED LEARNING

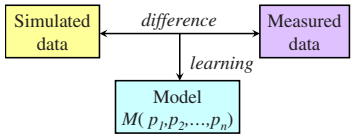


Figure 5: Model-based learning

If we have some idea on the major causes for the difference behavior, we may utilize a model-based learning approach to validate the idea. A model $M(p_1,p_2,\ldots,p_n)$ based on a set of n parameters is assumed in the learning. The goal of learning is to quantify the values of these parameters based on the difference data.

For example, in [10] the authors assume that the difference between predicted path delays and measured path delays is mainly due to un-modeled effect from within-die delay variation. A grid-based model was used and the unknown parameters to estimate became spatial delay correlations (within grid and across grids) [12]. The authors proposed a Bayesian based inference technique to quantify these parameters [13].

Because the model and the number of parameters are fixed in advance, model-based learning can be seen as a way of *parametric learning* [11]. The advantage is that we can begin by assuming a model that has a link to some physical interpretation. The limitations are twofold. First, there are aspects in the behavior difference that may not be explainable through a clearly defined model. Second, if a model is too complex, we may not have enough test data to quantify the values of all parameters with high confidence.

The limitations motivate us to take a *non-parametric learning* approach where no fixed model is assumed in advance. In this case, the goal is no longer to estimate the values of certain parameters. In the following, we will formulate a new goal.

4. IMPORTANCE RANKING

Suppose we have a timing model made up of n *delay entities* where each entity consists of a number of *delay elements*. Suppose in total there are l delay elements. To clarify, a delay entity is an abstract term that can be flexibly defined by a user. Figure 6 illustrates the difference between a delay entity and a delay element. For example, an entity can be a standard cell (i.e. Nand, Nor, etc.). In a timing model, a standard cell consists of multiple pin-to-pin delays. These delays are delay elements in the entity. An entity can also be a group of routing patterns for nets. For example, after delay calculation, the delay of each net is added into the model. Then, each net delay becomes a delay element of the entity.

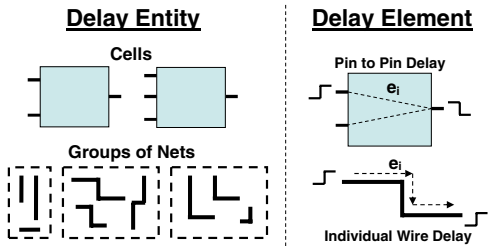


Figure 6: Delay entity Vs. delay element

Suppose we are given a set $\mathcal{Q}$ of l delay elements, $\{e_1,\ldots,e_l\}$ and a set of m critical paths $\{p_1,\ldots,p_m\}$, that are made up of those delay elements. In simulation, the delay of each path p_i is a function $T_i(\mathcal{Q})$. In our case, T_i is the estimated timing produced by a timing analysis tool. Let $\mathcal{T} = [T_i(.),\ldots,T_m(.)]$ In test, suppose that m path delays are measured on k sample chips. The result is a $k \times m$ matrix $\mathcal{D} = |\vec{D}_1,\ldots,\vec{D}_m|$. Each $\vec{D}_i$ is a column vector $[d_{1i},\ldots,d_{mi}]^T$. Each d_{ji} is the delay of path j on chip i.

Assume we have fully characterized each entity to the best we could. Also assume that on silicon there is a systematic deviation in the delays of each entity. Our goal is to analyze $\{\mathcal{Q},\mathcal{T},\mathcal{D}\}$ and rank delay entities in terms of their deviations.

4.1 Feature ranking in binary classification

To rank delay entities, we propose a methodology consisting of the following steps: (1) We convert the dataset into a binary classification problem. (2) We apply a learning algorithm for binary classification on the dataset to build a learned model. (3) From the learned model, we obtain the importance of each *feature* as a value. Here, a feature in the learned model refers to a delay entity. (4) We use the feature importance values to rank entities.

To simplify the discussion, assume that we are using a nominal static timing analysis tool. Therefore, $\mathcal{T}$ is a vector of m estimated path delays. Let $\mathcal{D}_{ave}= [D_1,\ldots,D_m]$ be the average path delays measured on k chips based on the data matrix $\mathcal{D}$. The difference is a vector $Y=\mathcal{T}-\mathcal{D}_{ave}=[y_1,\ldots,y_m]$. Each path p_i consists of a set of q delay elements $\{e_{i_1},\ldots,e_{i_q}\}$. Recall that these elements come from n delay entities. Let $\vec{x}_i = [d_1,\ldots,d_n]$. Each d_j is the sum of all delays in $\{e_{i_1},\ldots,e_{i_q}\}$ where these delays come from the entity j. $d_j = 0$ if no delays come from the entity. In this way, each path is represented as a vector of n delays. The input dataset becomes $\mathcal{S}=\{(\vec{x}_1,y_1),\ldots,(\vec{x}_m,y_m)\}$.

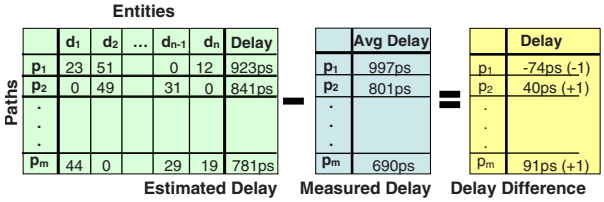


Figure 7: Converting to a binary classification problem

The first step in our methodology is to convert the dataset into a binary classification problem. Given a threshold $threshold$, we

define $\hat{y}_i = -1$ if $y_i \leq threshold$ and otherwise $\hat{y}_i = +1$. This produces a new dataset $\hat{S}$=$\{(\vec{x}_1, \hat{y}_1), \ldots, (\vec{x}_m, \hat{y}_m)\}$, as shown in Figure 7 for $threshold = 0$.

4.2 Support vector classifier

Given $\hat{S}$, we then apply a learning algorithm to build a binary classifier. Then, we analyze the structure of this classifier to quantify the *importance* of each entity. There are some classifiers whose structures allow the importance of variables to be evaluated easily and others that do not. This work examines one classifier that belongs to the former, the Support Vector Machine (SVM) classifier.

Given $\hat{S} \subset (\vec{X} \times \hat{Y})^m$, SVM implements a family of algorithms that can work with a variety of *kernel* functions [14]. We only use the *linear kernel* $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)$ which is simply the dot product of the two vectors. With a linear kernel, the SVM classifier is a hyperplane $\vec{w} \cdot \vec{x} + b$. If the two-class data are linearly separable, $\vec{w}, b$ can be obtained from solving the optimization problem:

$$\text{minimize}(\vec{w} \cdot \vec{w}) \text{subject to } \hat{y}_i((\vec{w} \cdot \vec{x}_i) + b) \geq 1, i = 1, \ldots, m \quad (4)$$

This is a *hard-margin* algorithm that finds the maximum-margin hyperplane. The margin from the resulting hyperplane $\vec{w^*} \cdot \vec{x} + b$ to its nearest point is $1/\sqrt{(\vec{w^*} \cdot \vec{w^*})}$, where $\vec{w^*}$ is the optimal solution to equation (4). The optimization problem can be solved in its dual form using Lagrange method [14]:

$$\begin{aligned} &\text{maximize } \textstyle\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \\ &\text{subject to } (\sum_{i=1}^{m} y_i \alpha_i) = 0 \text{ and } \alpha_i \geq 0 \end{aligned} \quad (5)$$

where α_i, α_j are Lagrange multipliers. Solving the problem results in optimal solution $\vec{\alpha^*}$. The solution to the primal becomes $\vec{w^*} = \sum_{i=1}^{m} y_i \alpha_i^* \vec{x}_i$. The value b is then decided based on the optimal primal solution $\vec{w^*}$. Figure 8 illustrate the relation between $\vec{\alpha^*}$ and $\vec{w^*}$. We notice that each α is associated with a path and each w is associated with a delay entity.

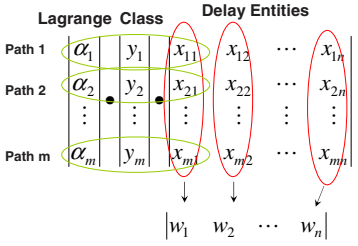


Figure 8: Illustration of $\vec{\alpha^*}$ and $\vec{w^*}$

If $\hat{S}$ is not linearly separable, in the primal formulation, we can introduce *slack variables* ξ_i, $i = 1, \ldots, m$ and the minimization objective becomes $(\vec{w} \cdot \vec{w}) + C \sum_{i=1}^{m} (\xi_i)^2$. C constrains the Lagrange multiplier, i.e. $C \geq \alpha_i \geq 0$. This becomes a *soft-margin* algorithm [14] and the approach to solve it is similar to the linearly separable case describe above.

It is interesting to note that in the optimal solution $\vec{\alpha^*} = (\alpha_1^*, \ldots, \alpha_m^*)$, some $\alpha_i^* = 0$. If $\alpha_i^* = 0$, then essentially $\vec{x}_i$ (path i) has no impact on the construction of the classifier. Hence, $\vec{w^*}$ depends only on the paths whose α values are not zero. In our methodology, we therefore use w_j^* to rank cell s_j.

4.3 Intuition behind using w_j^*

We observe that w_j^* does not directly measure the delay deviation of entity j. Note that the value of the Lagrange multiplier α_i^* measures the *importance* of the vector $\vec{x}_i$ (of path i) in constructing the classifier. A large α_i^* indicates that the vector $\vec{x}_i$ is a strong constraint for the resulting hyperplane.

x_{ij} is the amount of estimated delay contributed from cell s_j to path p_i. Hence, $x_{ij} > 0$ and $\alpha_i^* > 0$. In addition, $y_i \in \{-1, 1\}$. $y_i = -1$ means that STA under-estimates the path delay. $y_i = 1$ means that STA over-estimates. Hence, the sign of each $y_i \alpha_i^* x_{ij}$ is decided by the sign of y_i. Therefore, each $y_i \alpha_i^* x_{ij}$ is a measure of the importance of cell s_j in contributing to the over-estimation or under-estimation. w_j^* is the sum of this importance over all paths and hence, is a measure of the overall importance of cell s_i in contributing to the over-estimation or under-estimation.

5. EXPERIMENTS

In this section we discuss the experiments to validate the effectiveness of the above importance ranking methodology. Recall that our objective is to rank delay entities based on their deviations from the modeled delay values. It is important to note that the ranking methodology above does not measure these deviations directly. In this section, we will describe a linear uncertainty model for the delay deviations. Then, we will compare the importance ranking to the assumed *true ranking* based on this uncertainty model.

5.1 The linear uncertainty model

To simplify the discussion, we assume for now that a delay entity is a standard cell and delay elements are pin-to-pin delays in the cell. Later in Section 5.5, we will show that our framework can be easily extended to include analysis on net delays.

Assume that the l delay elements come from n standard cells, $\{s_1, \ldots, s_n\}$. We also assume that the actual delay of e_i, (of a standard cell s_j) on the silicon can be represented as:

$$\hat{e}_i = mean_i^0 + mean_j^{cell} + mean_i^{pin} + std_i^0 \pm std_j^{cell} \pm std_i^{pin} + \epsilon_i \quad (6)$$

where $mean_i^0$ is the estimated mean delay, std_i^0 is a Gaussian random variable with mean zero, which represents the estimated standard deviation. $mean_j^{cell}$ represents the mean delay deviation for every pin-to-pin delay of the cell. $mean_i^{pin}$ represents the additional mean delay deviation for e_i only. std_j^{cell} and std_i^{pin} are the deviations on the standard deviation, which both are Gaussian variables with mean zero. Notice that std_j^{cell} and std_i^{pin} can be used to result in reduced delay variation. ϵ_i is a zero-mean Gaussian variable which can be used to model noise such as measurement error. We assume that in the timing model, e_i is characterized only as $e_i = mean_i^0 + std_i^0$. Because std_j^{cell}, std_i^{pin}, and ϵ_i are assumed to be Gaussian with mean zero, we can also refer them as the standard deviation values. Given a cell s_j, we have two types of uncertainty, $Uncer_{mean}(s_j) = mean_j^{cell}$ and $Uncer_{std}(s_j) = std_j^{cell}$. In the experiments, we randomly select the values for $mean_j^{cell}$, $mean_i^{pin}$, std_j^{cell}, std_i^{pin}, and ϵ_i to perturb the statistical delay library. Then, we perform Monte Carlo simulation based on the perturbed library to produce the results of k samples. We use the results as if they come from measurement on k sample chips.

From the perturbed library, we can obtain the assumed *true ranking* based on the actual deviation values used to perturb the library. In the following we discuss results based on $mean_j^{cell}$. Results on std_j^{cell} are omitted because they show similar trends.

5.2 Experimental setup

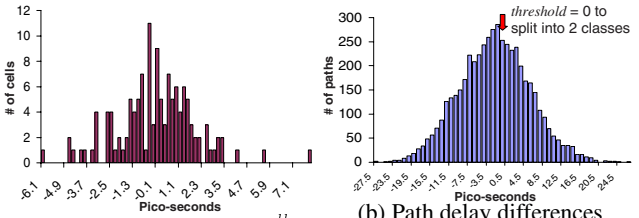
We took a cell library of 130 cells characterized based on a 90nm technology. The characterization gives each pin-to-pin delay e_i, a mean delay value $mean_i^0$ and a standard deviation std_i^0. In this baseline study, we select $m(= 5000)$ random paths. Each path consists of 20 to 25 delay elements. These paths are analyzed through a statistical static timing analysis (SSTA) tool [15] to obtain a mean and standard deviation for each path delay. The cell library is then

perturbed using the linear uncertainty model. Then, we perform Monte-Carlo simulation to produce $k = 1000$ samples.

If our objective is to rank cells based on $mean_j^{cell}$, then the initial dataset is converted into measured mean path delays. If the objective is to rank cells based on std_j^{cell}, standard deviation of each path delay is calculated based on k samples. In both cases, we use the method described before to produce the difference dataset $\mathcal{S}=\{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$. Then, a value $threshold$ is selected to convert $\mathcal{S}$ into a binary classification dataset $\hat{S}$. The dataset $\hat{S}$ is analyzed using SVM and the weight vector $\vec{w^*}$ is calculated. From the values of w_j^*, we obtain a ranking. This rankings is then compared to the true ranking that is based on either $mean_j^{cell}$ (or $|mean_j^{cell}|$) or std_j^{cell} (or $|std_j^{cell}|$).

5.3 Correlation between w^* and $mean_j^{cell}$

$mean_j^{cell}$ is sampled from a Gaussian distribution $N(\mu, \sigma^2)$ where $\mu = 0$ and $\sigma = \frac{0.2}{3} \times a_j$ where a_j is the average of all mean delays in the cell. Hence, we want $mean_j^{cell}$ to be roughly between $\pm 20\%$ of a_j. In a similar way, individually we add $mean_i^{pin}$ to the pin-to-pin delay i, which is roughly between $\pm 10\%$ of $mean_i^0$. The std_j^{cell}, std_i^{pin}, ϵ_i have no impact because we focus ranking on $mean_j^{cell}$. Therefore, their numbers are arbitrarily assigned. For example, we let std_j^{cell} be a random variable whose $\pm 3\sigma$ is $\pm 20\%$ of a_j, std_i^{pin} be a random variable whose $\pm 3\sigma$ is $\pm 20\%$ of $mean_i^{pin}$, and ϵ_i be a random variable whose $\pm 3\sigma$ is $\pm 5\%$ of a_j.



(a) Histogram of $mean_j^{cell}$ (b) Path delay differences

Figure 9: $mean_j^{cell}$ and path delay differences

Figure 9-(a) shows the histogram of $mean_j^{cell}$, $j = 1 \ldots 130$, in terms of their actual values in picoseconds. Figure 9-(b) shows the histogram of path delay differences, i.e. $y_1, \ldots, y_{5000}$. We first select $threshold = 0$ to split the distribution in the middle so that $\hat{y}_i = +1$ if $y_i \geq 0$ and otherwise $\hat{y}_i = -1$. The new dataset is then given to SVM to calculate $\vec{w^*}$.

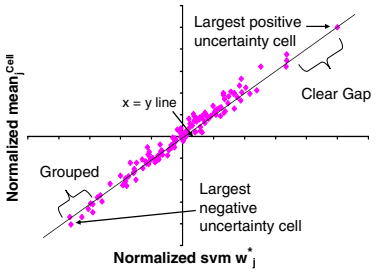


Figure 10: Correlation on normalized w^* and $mean^{cell}$

Next, we normalize the values of $mean_j^{cell}$ and w_j^* into the same range $[0, 1]$. Then, we do an X-Y scatter plot for every point using the normalized $mean_j^{cell}$ as the Y value and normalized w_j^* as the X value. Figure 10 shows the scatter plot. It is interesting to notice in Figure 10 that on the top right there is one outlier cell and then a gap followed by three cells clustering. When we compare this figure to figure 9-(a), we can see that on the right (positive) side of the histogram there is one high $mean_j^{cell}$ and then there is a gap followed by three cells with very similar $mean_j^{cell}$. If we compare the (bottom) lefts of both figures, the cells are grouped very closely together in both figures and there is no cell as a clear outlier.

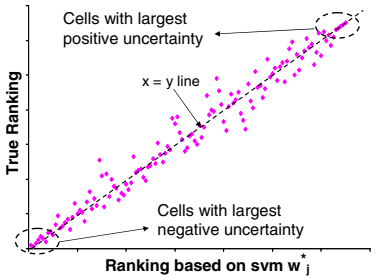


Figure 11: SVM w_j^* ranking vs. true ranking

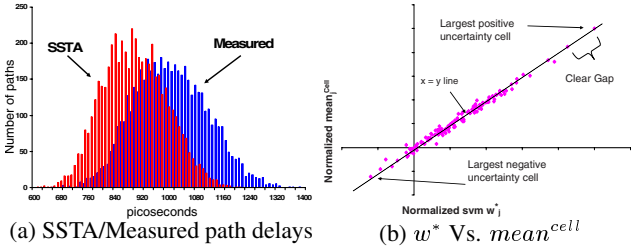
For each cell s_j, we obtain its SVM ranking based on w_j^* and true ranking based on $mean_j^{cell}$. Let them be SVM_j and $true_j$. We then do an X-Y scatter plot for every point $(SVM_j, true_j)$. Figure 11-(a) shows the ranking correlation result. We observe good correlation between the two rankings, especially on those cells with the largest uncertainties. Notice that there are two highly correlated ends. The cells with smaller SVM_j and $true_j$ (bottom left) have large negative uncertainties and the cells with larger SVM_j and $true_j$ (top right) have large positive uncertainties.

5.4 Impact of systematic L_{eff} shift

In this section, our goal is to show that a systematic process shift on a low-level parameter such as L_{eff} would not degrade the effectiveness of the proposed ranking method. This is a desired feature because our methodology analyze design-silicon correlation at the high level of Figure 3. As explained before, one can also utilize on-chip monitors to study the correlation at the low level, i.e. SPICE parameter level. Therefore, we desire our methodology to be applicable independently of a low-level correlation methodology.

We simulate the effect of a 10% systematic shift in L_{eff} and observe its impact on the ranking accuracy. Recall that we use a standard cell library that was characterized with 90nm technology. We re-characterized the library with 99nm technology and then injected the same amount of the deviations as that in the baseline study. This new and perturbed library is then used in Monte Carlo simulation to produce the measured path delays. Note that the predicted delays are still based on the original 90nm statistical timing library.

Figure 12-(a) shows the path delay distributions from the predicted SSTA results based on the 90nm library and the measured path delay results based on the perturbed 99nm library A clear shift is visible. Figure 12-(b) shows the correlation between $mean_j^{cell}$ and w_j^*. Because of the shift, we see that the y-axis in Figure 12-(b) shifts left. Compared the result to that shown in Figure 10, we see that except for the shift of the axis, the low-level parameter does not degrade the effectiveness of the method.



(a) SSTA/Measured path delays (b) w^* Vs. $mean^{cell}$

Figure 12: Impact of L_{eff} on the proposed method

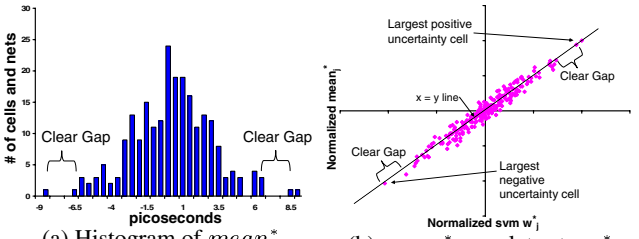
5.5 Including net delays in the ranking

As described in Section 4, the definition of a delay entity is artificial. Once this definition is given, the proposed methodology ranks

entities according to their importance. In the above experiments, we simply let an entity be a cell in a library. We can easily extend the definition of entity to include net delays.

As shown in Figure 6, a net entity should include a set of nets whose routing patterns can be deemed as similar. For example, a group of patterns can be defined as similar if their lithographic effects are similar. As far as our methodology concerns, the definition of this similarity is given by the user. In the experiment described below, we take the liberty to group nets into 100 entities. The assumption is that there is a systematic timing uncertainty on every net that belong to the same entity. Then, we would like to rank both cell and net entities together.

For this experiment we need modify equation 6. In addition to $mean^{cell}$ and $mean^{pin}$ for a cell entity, we include $mean^{sys}$ and $mean^{ind}$, where sys stands for a systematic shift on the net delays within the net entity and ind stands for individual shift on each net delay. Note that 130 cell entities and 100 net entities together give us 230 entities to rank. Again, we use $\pm 20\%$ on the systematic shifts and $\pm 10\%$ on the individual shifts as before. In the following, we use $mean^*$ to indicate $mean^{cell}$ and $mean^{sys}$ together.



(a) Histogram of $mean_j^*$ (b) $mean_j^*$ correlates to w_j^*

Figure 13: Correlation between $mean_j^*$ and w_j^*

In Figure 13-(a), we see two clear gaps at both ends on the $mean_j^*$ histogram. It is interesting to observe that in Figure 13-(b), we also observe the same two gaps at the two ends. This shows again that in the SVM ranking, the most uncertain entities stand out as outliers. We also see that the impact of going from 130 entities to 230 on ranking accuracy is relatively small.

6. SUMMARY AND DISCUSSION

Design-silicon timing correlation can mean different things in different applications. When the focus of analysis is on failing samples, diagnosis and silicon debug aim to uncover the root causes for the failures. When the focus of analysis is on good and marginal chips, on-chip monitors aim to measure the effects on low-level parameters from process, voltage, and temperature variations. When these two approaches are not applicable, the third option is to analyze delay test data. While the first two approaches have been employed in the industry for years, this paper discusses a path-based methodology for correlating test data to timing analysis.

Delay testing has traditionally been optimized for cost and for capturing defects. Changes need to be made to the tools, methodologies and ATEs if we want to utilize delay testing for extracting design-related information. On the ATPG side, a more flexible model is required for guiding test pattern generation. This model should be easily adjustable by a user to target specific design aspects of interest. On the methodology side, a separate methodology specific for design information gathering needs to be developed. This methodology may need to integrate with the existing design tools. On the ATE side, a tester should allow related data to be gathered more easily. In the past few years, we have observed changes to the tools, methodologies and ATEs to the directions described above, although they are not necessary for the purpose of extracting design-related information as discussed in this paper.

Even with delay testing, it is impossible to test for all design concerns left unaddressed in the pre-silicon design process. Take the path-based methodology as an example. There are limited number of paths we can test at the post-silicon stage. However, there are enormous number of cells and wires in a design. It is virtually impossible to utilize the proposed method to evaluate the timing deviations for all of them. This raises an important question for the proposed path-based methodology. That is, how to select paths? Without proper path selection, analyzing path delay data may not help to address the key concerns.

In summary, an effective design-silicon correlation framework needs to address three important aspects of the problem: (1) information content, (2) information decoding, and (3) application of the information. Silicon data should contain the required information for analyzing the design aspects of interest. Containment can be thought as if given unlimited computational resource, the required information can be decoded from the data. Then, efficient methods should be developed to decode the information as much as possible. This information should be in the form that can be easily applicable. This paper only discusses the second aspect and hence, the path-based methodology is only the first step for developing a more comprehensive correlation framework. This framework should also be integrated with the on-chip monitor based correlation analysis as described in Figure 3.

7. REFERENCES

[1] L. Lee et al. "On Silicon-Based Speed Path Identification," In *IEEE VTS*, 2004, pp. 35-41.

[2] M. Abramovici, M. Breuer. "Fault Diagnosis Based on Effect-Cause Analysis: An Introduction," In *DAC* 1980, pp. 69-76.

[3] Z. Wang, K.H. Tsai, M. Marek-Sadowska, J. Rajski. "An Efficient and Effective Methodology on the Multiple Fault Diagnosis," IN *ITC* 2003, pp. 329-338.

[4] A. Kristic, K.T. Cheng *Delay Fault Testing for VLSI Circuits.* Boston: Kluwer Academic Publishers, 1998.

[5] A. Kristic, Li-C. Wang, K.T. Cheng, J.J. Lou, M. Abadir. "Delay Defect Diagnosis Based Upon Statistical Timing Models - The First Step," In *DATE*03.

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

[7] D. Boning, S. Nassif, A. Gattiker, F. Lui, et al. "Test Structures for Delay Variability" Proc. of the 85h ACM/IEEE *TAU* 2002, p.109.

[8] M. Bhushan, A. Gattiker, M. Ketchen, K. Das, "Ring oscillator for CMOS process tuning and variability control," IEEE Transactions on Semiconductor Manufacturing, Vol. 19, pp. 10-18, 2006

[9] M. Ketchen, M. Bhushan, D. Pearson, "High speed test structures for in-line process monitoring and model calibration," Proc. IEEE Int. Conf. on Microelectronic Test Structures, 2005. pp. 33-38.

[10] Benjamin Lee, Li-C. Wang, Magdy S. Abadir. "Refined statistical static timing analysis through learning spatial delay correlations," In Proc. *DAC* 2006, pp. 149-154.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. *Springer Series in Statistics*, 2001

[12] A. Agarwal, D. Blauuw, and V. Zolotov. "Statistical timing analysis for intra-die process variations with spatial correlations," In Proc. *ICCAD*, 2003.

[13] E. F. Schisterma, et al. "Estimation of the correlation coefficient using the Bayesian Approach and its applications for epidemiologic research," *BMC Medical Research Methodology*, Vol 3, 2003.

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

[15] C. Visweswariah, et al. "First-order incremental block-based statistical timing analysis" *DAC*, 2004, pp. 331-336.