Diagnosis of design-silicon timing mismatch with feature encoding and importance ranking – the methodology explained *

Pouria Bastani[1], Nick Callegari[1], Li-C. Wang[1], Magdy S. Abadir[2]
[1]Department of ECE, UC-Santa Barbara
[2]Freescale Semiconductor, Inc.

Abstract

For sub-65nm design, there can be many timing effects not explicitly and/or accurately modeled and simulated. For design-silicon timing convergence, this paper describes a novel path-based diagnosis approach that analyzes and ranks potential design related issues causing the unexpected timing effects. We explain in detail how a path can be encoded with a set of diverse "features" based on one's knowledge of the potential issues. We explain how these features can be interpreted differently in a data learning algorithm based on adjusting a so-called kernel function. Then, we explain how kernel-based data learning can be used to rank the importance of features such that a feature contributing the most to design-silicon timing mismatch is ranked the highest. We conclude the paper by showing an application result on an industrial ASIC design.

1 Introduction

As technology scales, it has become increasingly difficult to have predicted timing from modeling and simulation match actual timing observed on silicon [1]. Each generation of technology can introduce new variations and effects that are either not modeled or not modeled accurately. When design-silicon timing mismatch is observed, one desires to identify the most important effects in order to improve modeling and simulation and consequently, improve the current and future designs and their timing yields.

To diagnose, a common approach would be to hypothesize a cause and run simulations and/or physical debug to identify its effects. As the induced effect showed similar results to the observed mismatch, it would presumed to be the problem. Furthermore, any residual mismatch requires subsequent hypotheses and validations. This continues until the mismatch can be fully explained by a chain of causes. The effectiveness of this approach diminishes as there can be a large number of potentially unmodeled and mismodeled effects in the nanometer era [2]. When considering all possible such effects and their combinations, the search space can be enormous.

Recently, the authors in [3, 4] proposed an alternative diagnosis methodology for analyzing design-silicon timing mismatch. The approach collectively analyzes many potential sources of uncertainty by formulating the problem as a data learning problem. Figure 1 illustrates the methodology.

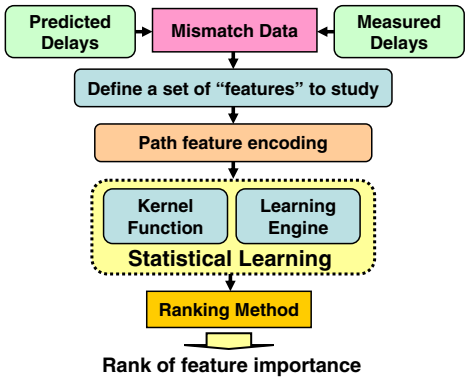


Figure 1. Overview of methodology

First, mismatch data is based on predicted path delays, e.g. from a static timing analysis tool, and measured path delays on silicon chips. To diagnose this mismatch data, a set of *features* is selected. These features define the search space for potential causes of the observed mismatch. Then, these features are used to encode paths into path vectors, i.e. each path is represented as a vector of numerical values. Together, they form a data matrix where a learning algorithm can be applied to build a model. This is a prediction model from the path vectors to the mismatch data, i.e. given a path vector, the model predicts the amount of mismatch on the path. The final step is to utilize this model to evaluate the importance of features and then rank them accordingly.

The methodology was first introduced in [4] with the emphasis on the explanation of the learning algorithm, in particular the Support Vector Machine (SVM) classifier. The features were based on cells and wires and the kernel function used to interpret them was fixed. With a limited set of features and the simple kernel function, the authors then suggested a feature ranking method that worked effectively.

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

This paper intends to generalize the scope of the methodology, by improving on three aspects: (1) One should be able to use a diverse set of features representing diverse sources of timing effects without restriction (2) The values on a subset of features can be interpreted differently and/or weighted differently from another subset, based on one's domain knowledge. (3) The ranking method should be general enough to work with any feature definition and any kernel function in use.

With these three objectives in mind, the rest of the paper follows as below. Section 2 reviews the background and related work. Section 3 uses illustrative examples to explain what the proposed methodology intends to achieve. Section 4 presents a comprehensive view on feature generation and encoding. Section 5 explains how a kernel function interprets features. Section 6 gives a brief overview of the learning engine. Section 7 discusses feature ranking and proposes a new and more general ranking method. Section 8 presents results on an industrial ASIC design. Section 9 references some work that has been done in the area of silicon repair and concludes the paper.

2 Background

Unpredictable silicon performance is a growing concern in sub-65nm process technologies. It is too expensive to consider all possible effects causing the unpredictability. Hence, one desires to have a methodology to uncover the most prevalent effects so that resource and effort can be properly allocated to address them. Such effects can be design, design methodology and/or process dependent.

Traditional diagnosis is based on the notion of "failing chips." Typically, failure data is analyzed for timing defects using fault-based approaches [7] [8]. Advancements in this area include taking multiple faults into account [9] or taking statistical timing defects into account [10]. When the causes are not location-based and/or fault-based, the effectiveness may be questionable. Moreover, when analyzing design-silicon timing mismatch, one may not have a clear notion of "failing chip" nor a clear definition of a "fault list."

This work assumes that design-silicon mismatch is due to unmodeled and/or mismodeled systematic and random timing effects. The goal is to uncover the most important systematic effects in the presence of random noise. Mismodeled effects may include those not accurately modeled or over-modeled (meaning too much added margin).

As mentioned in the Introduction, previous works [3, 4] focused the study on the learning algorithm, i.e. how and what algorithms should be used for learning and how to rank features. The work in [5] improves the SVM classifier algorithm used in [3, 4] with the ε-insensitive Support Vector Regression (ε-SVR) algorithm [27]. Although these works were important for developing the overall diagnosis methodology, they also raise questions regarding the applicability of the proposed approach in practice.

The main limitation of the methodology lies in the definition of the features. All previous works [3, 4, 5] use cells and groups of wires as features. In practice, the concerns can go beyond just cells and wires. Unmodeled and mismodeling effects can be associated with a diverse set of sources from layout, location, variations, etc. at gate, transistor, or SPICE level. To provide maximal flexibility, we need a methodology that can handle any type of feature.

For example, suppose we are interested in two types of features, the X-Y location of a path (described in section 4.3) and the drive strength of a cell characterized by transistor size. The values for the X and Y coordinates have completely different meanings from the value of transistor size. These values have to be properly interpreted by the learning algorithm. In SVM learning, feature interpretation is decided by the *kernel function* [29]. Therefore to allow using a diverse set of features, one needs to adopt a kernel that can properly reflect the meaning of those features.

As we allow feature definition to go beyond wires and cells, the feature ranking method in [3, 4, 5] needs to be generalized as well. This is because the previous ranking method depends on the fact that all feature values can be interpreted in a certain way. When we allow multiple ways to interpret subsets of features, the method may become ineffective. Hence, a more general method is desired.

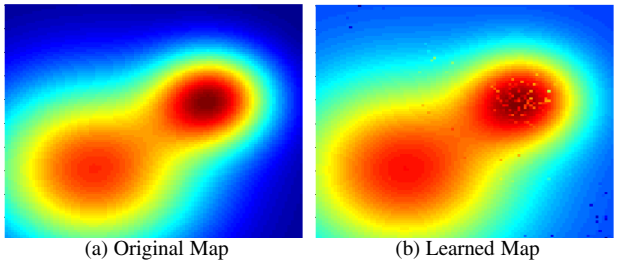
3 Explaining the essence of the proposed methodology with simple examples

In this section we use simple examples to illustrate how the impact of both intra-die and inter-die [11] variations can be analyzed with the proposed methodology.

3.1 Systematic intra-die variation

Systematic intra-die variation is often caused by layout and topological interaction with the process. Intra-die variation can be classified into two groups: process related variations and environmentally induced variations [12]. Examples of process variations are optical proximity effects, chemical-mechanical planarization effects and spatial variation effects due to lens aberration. Examples of environmentally induced effects are voltage and temperature values that vary across the die. The magnitude of these effects depend on the design, process and manufacturing precision. Our proposed methodology can analyze the impact of intra-die variation along with other systematic effects. We use the following hypothetical example to illustrate this point.

Figure 2(a), shows hypothesized mismatch map measured on paths across a die. If the mismatch is from different paths, the values shown can be thought of as the percentages of mismatch delay differences with respect to the predicted delays. If the mismatch is from copies of the same path, the values can be thought of as the actual delay differences between predicted delays and measured delays.



(a) Original Map (b) Learned Map

Figure 2. Learning an X-Y map with 22 features where the first two are X,Y coordinates and the rest are uncorrelated features

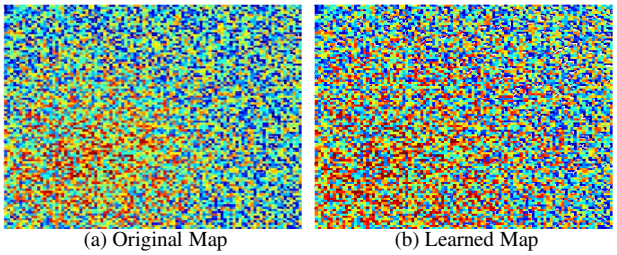
Suppose we select 22 features to study the mismatch. The first two are the X and Y coordinates of the path. The rest happen to be features not correlated to the mismatch. The point to show is that, if the methodology works, the first two features should be ranked higher than the rest. This is because we know that the map is modeled using two Gaussian functions with the X and Y locations of the grid as variables, i.e. $f(X,Y) = e^{\frac{-\left[(X-c_{11})^2+(Y-c_{12})^2\right]}{\sigma_1}} + e^{\frac{-\left[(X-c_{21})^2+(Y-c_{22})^2\right]}{\sigma_2}}$. Where c_{1*} and c_{2*} are the X and Y center of the two distributions and σ_* controls the width of the distribution. Therefore, we knew in advance that the map was independent of the 20 features.

After applying the learning algorithm ε-SVR [5] to learn the mismatch map, for every path the learned model predicts a mismatch value. If learning is effective, plotting these values should produce a map similar to the original map. Figure 2(b), shows the *predicted* mismatch map. We observe that the predicted map is similar to the original map, showing good learning accuracy.

Next we use the learned model to evaluate the importance of each feature (we will discuss the method later). We rank features based on this importance. After normalizing feature importance values into the range [0,1], the importance values of the first two features are 1 and 0.645, and the rest are between 0.03 and 0.04. This shows that the methodology correctly identifies the X and Y coordinates are the features contributing to the mismatch the most. It is noted that the methodology does not tell why X and Y coordinates are important, i.e. it does not say what original function f is. All it says is that f depends on X,Y the most.

Suppose the systematic mismatch can be the result of temperature, OPC, CMP, etc. To uncover which has the greatest impact, one needs to add three features T,O,C corresponding to the three sources of uncertainty. For example, one may have a hypothesized temperature map based on power analysis. The value of feature T on a path can then be the hypothesized temperature degree based on the location of the path on the temperature map.

After ranking, if one of the T,O,C features is ranked higher than X,Y, we know that the feature contributes to the mismatch the most. If none of them is ranked higher than X,Y, then we know that additional features may be needed to explain the map. We note that the map may be due to interaction between two of T,O,C or among all three. In this case, the two or all three should be ranked higher than X,Y. In the work [6], this is called a *higher-order* effect, where the effect depends on a combination of multiple features rather than just a feature individually.



(a) Original Map (b) Learned Map

Figure 3. Learning an X-Y map with an additional feature V

To illustrate the above discussion, suppose we have a new mismatch map using a new function $f(X,Y,V)' = 0.1 * f(X,Y) + V$, where V is a new feature variable. The point here is that now the mismatch depends more on the value of V than the X,Y coordinates. Again, we included the 20 uncorrelated features in the analysis.

Figure 3(a), shows the values of this new function $f(X,Y,V)'$. We see that in this case, the map does not show a clear dependency on the X,Y coordinates.

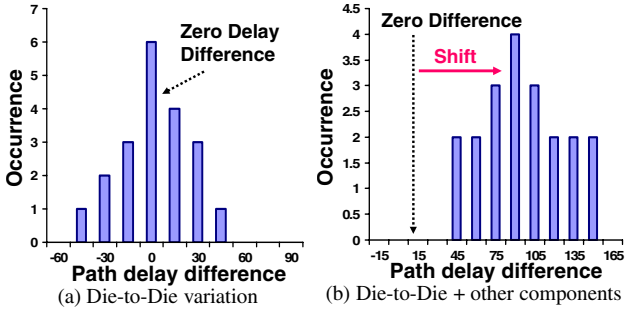
Figure 3(b), shows the predicted mismatch map. Then, with the learned model, we rank the 23 features again. After normalizing the feature importance values into [0,1], the importance of feature V, X and Y were 1, 0.14 and 0.26 respectively, showing a much higher importance for feature V. The rest had an importance between 0.03 and 0.06, showing very little impact on the mismatch data.

In this example, one may think V as the temperature T (or any other feature that may or may not be location based). In this case, the ranking result shows that temperature can explain most but not all the variation. This is because the importance values of X,Y are still significant (relative to the rest 20). Therefore, we can conclude that there are other variation effects not included in the feature list. We note that the proposed methodology does not put a strict limit on the number of features. Hence, if one is unsure about what features are relevant, the strategy is to include as many as possible and let the learning engine and ranking method identify which are most important.

3.2 Systematic inter-die variation

The above discussion explains how the proposed methodology can be used to analyze one die. What if one want to analyze multiple dies together to consider effects due to inter-die variation? Systematic inter-die variation can be due to normal manufacturing tolerances that affect

the mean value of a parameter from one die to another (die-to-die) across different reticles, wafers or lots [13]. The key concern is how much the performance varies for one path across different die. Inter-die variation is typically design independent and related to process and equipment.

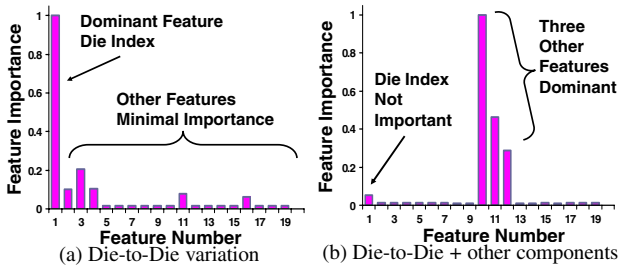


(a) Die-to-Die variation (b) Die-to-Die + other components

Figure 4. Path delay difference for a path on 20 different dies

Figure 4(a) and (b), show two hypothesized timing mismatch histograms for a path across 20 different dies. The distribution in Figure 4(a) has a mean of $0ps$ and a 3-σ of $\pm 50ps$. The distribution in Figure 4(b) has a mean of $90ps$ and a 3-σ of $\pm 40ps$. For this figure, the variation of the delay differences could be due in part to inter-die variation; however the shift of the mean from $0ps$ to $90ps$ would most likely be the result of other systematic effects.

To include inter-die variation into consideration, we add *die index* as a feature. When analyzing the same topological path, but on different dies, this feature allows us to check if the mismatch is due primarily to inter-die variation or to other systematic effects.



(a) Die-to-Die variation (b) Die-to-Die + other components

Figure 5. Ranking of features for a path on 20 different die

Figure 5(a) and (b), show the feature importance results using the datasets in Figure 4(a) and (b), respectively. Figure 5(a), shows a very strong importance for the die index. This means that almost all of the timing mismatch is due to inter-die variation. Other features had little impact. On the contrary, Figure 5(b), shows little importance for the die index, but strong importance for three other features. This means that other features contributed more to the timing mismatch than the inter-die variation.

3.3 Summary and discussion

Again, if one desires to know why (or which) inter-die variation impacts the mismatch the most, features originating from inter-die variations should be included in the analysis (as that in the intra-die example). In summary, the above examples illustrate several key points in the proposed methodology: (1) The methodology can be used to analyze a single die or a collection of dies. (2) Features can be defined at different levels of abstraction. For example, X,Y coordinates may tell the mismatch is due to intra-die variation but not which variation. A temperature feature can point to temperature variation, but it does not explain why. To understand why, one may need features corresponding to lower-level sources of temperature variation, i.e. static and dynamic power consumption. (3) The methodology suggests that diagnosis can be done in a hierarchical manner, from the most abstract view first, to a detailed view for finding the root causes. (4) The methodology is not for solving a well-defined diagnosis problem. It is more for solving an ill-defined diagnosis problem where a user has limited knowledge of what is going on. It provides the infrastructure that allows a user to apply limited domain knowledge and efficiently study and understand the mismatch data.

The benefit the methodology brings is efficiency and flexibility, allowing a user to hypothesize a variety of features and quickly narrow down to the most relevant. When one is unsure about the reasons behind the mismatch, she/he can always begin with a large number of features. One may wonder what features to begin with. In our view, this depends on the design, design methodology (and hence design company), and the process (and hence the fab). In other words, the definition of features may become proprietary information to a company because it represents the list of potential holes in the design and/or manufacturing process. With this in mind, the next section gives an overview of where features may be defined, providing a starting point for a user to develop more advanced features.

4 Feature generation and encoding

As shown in section 3, any feature that does not have a significant impact on the path delay difference will be ignored in the learning and subsequently ranked with low importance. Note that adding more features does not have a large effect on the learning algorithm in respect to accuracy nor runtime [6].

Features can be *occurrence*-based or *descriptive*-based. Occurrence-based counts how many times a feature appears in a path. For example, the number of times a 2-input NAND gate appears in a particular path is 3. Descriptive-based are real value descriptions of a feature. For example, the average functional temperature for a particular area of a die where a path is located is 74°F.

In order to find the most relevant features to analyze we

conducted a literature survey to see which effects have the highest potential to be mismodeled. We divide features into five groups: cell-based, interconnect-based, location-based, dynamic effects and margin related mismatch.

4.1 Cell-based features

In order to generate an accurate cell timing library, much effort is spent on collecting statistical information on the variation of performance based on process, voltage and temperature parameters (PVT). From this statistical data, SPICE level transistor models are generated. The problem is that due to process variations, temperature dependencies and influence of voltage variations there is no single silicon that can serve as a reference [20]. Thus, there can easily be mismodeled effects in the transistor models.

Once the transistor models are finalized, the next step is to perform timing characterization for each cell. Characterization is manually intensive, therefore it is possible for a timing model to have errors upon creation [21]. During characterization, an extracted cell netlist is simulated for different conditions by varying the input slope, output load, voltage and temperature. The extraction process itself is only an approximation [2] and may induce its own errors.

During characterization not every input slew, load, voltage and temperature is simulated. Thus, when doing delay calculations for a cell, all operating conditions that do not exactly match the simulated conditions need to be interpolated. Due to potential non-linearities in cell performance [2], this can incur some error. To add further complication, if a cell has *extreme* operating conditions outside of the scope of what was simulated, then extrapolation is necessary for delay prediction. Extrapolation can be very inaccurate, due to highly non-linear behavior seen during these conditions [20]. Examples of large timing mismatch have been shown in [14] [15] [17] [20], where a weak driving cell tries to drive a large load or a strong driving cell tries to drive a very small load. Both cases are outside of normal operating conditions and can be very unpredictable.

The authors in [20] showed that for a 180nm cell library, some complex gates (i.e. muxes) can have more than a 10% delay mismatch between Spice modeling and STA models. In addition, [22] showed that for a given interconnect load, smaller drive strengths are more susceptible to interconnect variation than larger drive strengths.

Transistor level parameters have also been shown to have mismatch. For example there can be mismatch between PFET on-current and NFET on-current [11] [14] [18]. All these inaccuracies can lead to hardware mismatch [16] and result in a large impact on the predictability of clock networks and on inverter chains.

To analyze all of these potential sources of cell uncertainty we will start by treating each cell type as a different feature. This will account for complex cells (muxes, custom blocks, etc) along with traditional cells (inv, nand, etc). Next, to analyze PFETs (pull-up) and NFETs (pull-down) separately, we will divide the cells by their output transition on the path. For example, a 2-input nand cell with a rising output transition is one feature and the same nand cell with a falling output transition is another feature. Since we are still worried about cells in "extreme" operating conditions we will further divide the cells based on the amount of capacitive load they are driving with respect to the maximum capacitive load they are characterized to drive, i.e. $\%Load = ActualLoad \,/\, MaxLoad$. If $\%Load \leq 2\%$ or $\geq 100\%$ then it will be treated as a separate feature then if $\%Load$ is between 2% and 100%. This will target the sources of uncertainty due to extreme operating regions.

In order to have some more general cell-based features, we will include active transistor types in a path as features. This number of feature depends on how many flavors (typical, high-vt, low-vt, etc) of transistors are used in a process and in a design. This is done in case there is simple systematic shift in the low-level spice parameters that is independent of cell type.

4.2 Interconnect-based features

Interconnect modeling also begins by collecting statistical information on the variation of performance based on PVT data. Therefore, interconnect modeling is prone to the same low-level mismodeling as cell modeling. RC extraction tools use these silicon models as a basis to extract as accurate parasitics as possible. However, these tools have been shown to have large computational error due to the complexity of the problem [22], in particular for very short interconnects. To control the complexity, most tools approximate further by using 2.5-dimensional extraction model [23]. Also, algorithms for calculating interconnect delay and signal edge degradation involve approximation techniques proprietary to the tool vendors [20]. This makes it difficult to estimate the exact accuracy of the tools.

Each metal and via layer presents an independent processing step in the manufacturing process, thereby insuring a high degree of miscorrelation between one layer and another [11]. The work in [17], showed a timing skew that was due to a larger than expected resistive Via3 connection.

To analyze the potential sources of interconnect uncertainties we want to have a separate feature for wires and vias. For each stage in a path we can count the number of metal layers and vias an interconnect traverses through. Then we will bin a stage based on this number. For example, if one stage goes through 2 metal layers and 2 vias to get to the next cell, then in the 2-metal layer bin and the 2-via bin we will place a 1 that corresponds to this one stage. If another stage in the path went through 2 metal layers, but had 4 vias, then the 2-metal layer bin would become 2 and the 4-via bin would receive a 1. The array for this path would look like $M = (0,2,0,0,...)$, $V = (0,1,0,1,...)$, where M is the metal layer bin and V is the via bin.

If higher resolution is required, one can use additional features associated to interconnect shapes. Many Design Rule Checks (DRC) are in place by the fab to ensure high yield on their parts. These checks *recommend* (but do not require) that certain shapes of interconnects should be avoided. One example is a $T - intersection$, where a stem endpoint can cause metal deposition to be incorrect. Another example is an $end\ of\ the\ line$ rule, where an end of a line has some metal ahead of it and some metal to the right and left. When OPC corner correction is done on this line it can become much smaller than originally designed. Anything connected to this line can be damaged, for example a via to a different metal layer. In order to analyze these optical related interconnect issues, one may have features such as how many T-intersections are on a path, how many isolated lines are surrounded by metal on a path, and the risk score from an OPC tool for the interconnects of a path.

4.3 Location-based features

Location-based features are mostly related to intra-die and inter-die variations, as mentioned in section 3. Systematic intra-die variation can be due to optical proximity effects, inter level dielectric thickness variation, chemical mechanical planarization (CMP), lens aberration, voltage variation and temperature variation [11]. To analyze these potential sources of intra-die uncertainties we can use the X,Y coordinates of a path on the die as a feature (section 3.1) as well as other features more directly linked to the sources of variations. If a path is short, then the middle X-Y coordinate of the path should suffice. If a path is very long then one may use multiple X-Y coordinates. For example, the beginning, middle and end of the path. The CMP variation is largely due to density issues on a die [2]. To analyze these effects, one can use the overall area of a path as a feature. To calculate this value we draw a bounding box around the path, then measure the diagonal distance of this box which we use as an approximation of the path area.

Systematic inter-die variation is typically considered as a shift in the mean of some parameter values equally across all devices on any one chip [13]. An example of this is the "bowl" shape variation on a wafer, where die's that fall on the same ring of a wafer exhibit similar parameter shifts. To analyze these effects we can use the die index number as a features as previously explained in section 3.2. To analyze multiple dies across different reticles, wafers or lots, an index for each of these can be used as a feature.

4.4 Dynamic effects

Dynamic effects are timing uncertainties that are due to input patterns on a specific clock cycle. These effects include dI/dt voltage droop and cross-coupling noise. Voltage droop occurs when there is a sudden current draw (dI/dt) on a localized region of the die. If the change in current is large enough, the power delivery system cannot supply the required current fast enough for the gates and will slow down the propagation speed of a path [18] [25]. Cross-coupling noise occurs when the capacitance between two neighboring wires causes a logic event on one wire to induce noise onto the other wire which can slow-down or speed-up a particular transition [18] [24].

Voltage droop can be analyzed by using a feature that estimates the current draw (in amps) for a given size grid for the cycle of interest. Similarly, the authors in [26] showed how to approximate current draw by analyzing switching activity for a particular region surrounding a path for the cycle of interest. The switching activity number would then be used as a path feature to analyze the effect of voltage droop. Cross-coupling noise can be analyzed by approximating the amount of delay push-out (in ps) that would occur assuming worst case aggressor slope and alignment for each stage in a path as a feature [26].

4.5 Margin related mismatch

To account for all the different uncertainty on silicon, the simplest and sometime most effective approach is to design extra margin into the models. However, it is not economical to continue adding more margin to account for all the new sources of variation [11]. Adding too much margin can also have a negative impact on design-silicon convergence. The authors in [3], showed that for a particular high-performance microprocessor, some path delays were predicted $2X$ overly pessimistic by STA tools when compared to silicon measurements. By adding in this much excessive margin to a design, there will be a lot of performance left on the table. In order to analyze the impact of margin on timing mismatch, one can use the estimated margins themselves as features. If the mismatch data is highly correlated to these margins (as a result, they are ranked high), then excessive margin is the problem.

4.6 Summary

In summary, we have given a detailed description of where features may come from. We compose our list based on effects that have been known and published in literature. Figure 6 shows what an example dataset may look like before running the learning tool. Each row in the dataset is a different path and each column is a different feature. We note that feature values may need to be interpreted differently. For example, 74°F and 3 times mean different things. How to make the learning engine be aware of the difference is an interesting question. After all, when the engine is processing the data, it only sees the numbers 74 and 3. In the next section, we will explain how a kernel function can be designed to reflect one's intuition on the interpretation of feature values.

5 Kernel function

For a modern machine learning algorithm like SVM to work, all the information it needs is contained in a so-called

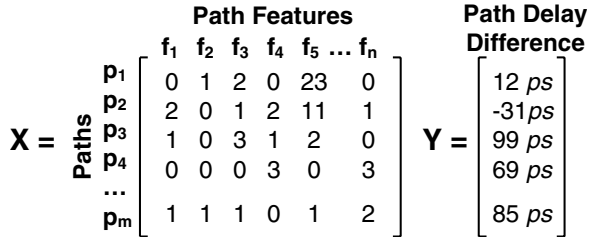


Figure 6. Example dataset

similarity matrix. This matrix measures the similarity between every pair of samples, in our case every pair of paths. The interesting point here is that the feature values on a single path by themselves have no meaning for learning. Only when we compare two vectors of feature values from two paths can meaningful information be learned.

The similarity between two vectors of feature values is measured in a feature space defined by a so-called *kernel function*. For example, suppose we have three path vectors, $S_1 = (3,3,3)$, $S_2 = (3,0,3)$ and $S_3 = (1,0,1)$. Suppose one's intuition for the feature encoding scheme is that S_1-S_2 should have a higher similarity than S_2-S_3. To capture this perspective, one can use the *dot product* "$\langle\cdot,\cdot\rangle$" as the kernel function. Then, we have $\langle S_1,S_2\rangle = 3\times3+3\times0+3\times3 = 18$. We have $\langle S_2,S_3\rangle = 3\times1+0\times0+3\times1 = 6$.

Suppose, in another case, one's intuition desires the contrary that S_2-S_3 should have a higher similarity than S_1-S_2. Then, one can use a different function, such as the cosine function $cos(x,y) = \frac{\langle x,y\rangle}{||x||||y||}$ Then, we have $cos(S_1,S_2) = \frac{18}{\sqrt{3^2+3^2+3^2}\sqrt{3^2+0^2+3^2}} = \frac{18}{22.0454} = 0.8165$. We have $cos(S_2,S_3) = \frac{6}{\sqrt{3^2+0^2+3^2}\sqrt{1^2+0^2+1^2}} = \frac{6}{6} = 1$. We see that by changing the definition of the kernel function, we can change how the feature vectors are interpreted and consequently, how the learning algorithm behaves.

What if we want one way to interpret a subset of features and another way to interpret another subset of features (the two sets may or may not be mutually exclusive)? Consider 3 paths encoded with three features: The count of the occurrences of a 2-input NAND cell in a path, the X coordinate and Y coordinate of the path on a die. For example, they are $P_1 = (2,100,200)$, $P_2 = (1,150,300)$ and $P_3 = (5,10,400)$. The range of the first feature is between 0 and the maximum number of the cell occurrences on a path. On the other hand the X,Y coordinates are limited by the size of the die. Notice that a value of 0 for NAND count means something entirely different from the 0,0 for the X,Y coordinates.

The solution is to combine two kernel functions into a single one. The first kernel is applied to the occurrence and the second is applied to the coordinates. Suppose we apply n kernel functions, $k_1,\ldots,k_n$, to interpret n subsets of features. Kernel function theory [29] states that any linear combination of kernels, $K = w_1k_1 + w_2k_2 + \cdots + w_nk_n$, is also a valid kernel function. Here $w_1 \ldots w_n$ are constants to weight the result from each kernel differently. This allows the design of a kernel function to match ones intuition on interpreting features, where some features have different meaning and importance than others.

For example, for all occurrence-based features, we use the polynomial kernel [29], $Poly(v_1,v_2) = \frac{1}{2^n}(\frac{\langle v_1,v_2\rangle}{||v_1||*||v_2||})^n$, to measure the similarity. The authors in [6], showed the superior capability of using the polynomial kernel to handle high-order timing effects due to combinations of occurrence-based features. For X,Y coordinates we can use the Gaussian kernel, $Gau(v_1,v_2) = e^{\frac{-||v_1-v_2||^2}{\sigma}}$, which showed excellent learning results in section 3.1.

6 SVM learning engine

The learning engine part of the methodology has been studied carefully before in [3, 4, 5]. As mentioned before, the conclusion is that the SVM algorithm ε-SVR performs the best [5]. This section gives a quick overview of the SVR algorithm. More interested readers can refer to [27] [28].

Suppose we are given n features $\{f_1,\ldots,f_n\}$ and m paths $\{p_1,\ldots,p_m\}$. Let each path p_i be described as $x_i = (x_{i1},\ldots,x_{ik})$ where x_{ij} is the value of feature f_j. Suppose the path delay difference on path p_i is y_i. From Figure 6 we see that matrix X (= $[x_1,\ldots,x_m]^T$) describes the path characteristics and vector Y (= $[y_1,\ldots,y_m]^T$) records the observed path delay differences.

The goal of learning is to build a model F based on matrices (X,Y), so that $F(X)$ gives the result $Y' = [y'_1,\ldots,y'_m]^T$ such that $err = \sum_{i=1}^{m} ||y_i - y'_i||^2$ is small. In SVR, instead of minimizing *err*, the learning tries to minimize both *err* and the *model complexity*, i.e. it tries to find the *simpliest* model to best explain the dataset [28]. At the core of every SVM algorithm, it solves a convex quadratic optimization problem. Using the Lagrange method, the solution is obtained as the following form:

$$F(x) = \sum_{i=1}^{m} \alpha_i k(x_i,x) + b \qquad (1)$$

where $k()$ is the *kernel function*, b is a constant, and α_i is the Lagrange multiplier for path vector $\vec{x}_i$. Each α_i characterizes the importance of the path in building the model F. A larger α_i in magnitude means that the path vector is more important. Some α_i can be zero. In this case, the path has no effect in the model. Those path vectors whose Lagrange multipliers are non-zero, are called the *support vectors*. In support vector analysis, the model complexity is proportional to the number of support vectors.

7 Ranking method

Once the model is built, we need to convert the path importance characterized by the Lagrange multipliers (α_i) into feature importance so that features can be ranked. The method proposed in [3, 4] is the following.

$$w_j = \sum_{i=1}^{m} \alpha_i * x_{ij} \quad (2)$$

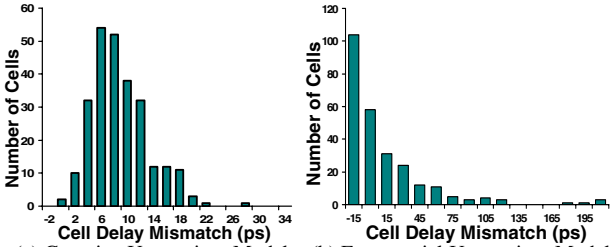
Where x_{ij} is the value of feature j for path i and w_j is the calculated weight for feature j. A higher weight means that the particular feature is more important in deciding the difference in the dataset. From equation 2, one can see that non-support vectors, i.e. unimportant paths, are discarded in the feature weight calculation. This weight calculation showed excellent results for the occurrence-based feature set used in [4] [5]. However, it is unclear if it is generalized enough to work on any type of features.

In this work we take a more general approach to calculate feature importance. This approach works with any feature list and any learned model. Once a model is learned, we take the original feature matrix and randomly perturb one feature at a time across all paths. The way we do the perturbation for each feature is by randomly shuffling that features values over all the paths. This way we do not generate any feature values out of the expected range. A similar approach was effectively used in the tree-based statistical learning algorithm *RandomForrest* [30].

After each feature value perturbation, we apply the learned model on each path using the perturbed matrix and try to predict the timing mismatch for the path. We analyze the prediction accuracy in terms of the mean square error ($MSE = \frac{1}{m}\sum_{i=1}^{m}(y_{predict} - y_{actual})^2$). Note that a separate MSE is calculated based on each feature perturbation. If the MSE is higher, then it means that the feature is more important. This is because by making the feature value random across all paths, the model prediction degrades and hence, the model depends more on the feature. If the MSE does not change, then the feature had no impact on the learned model and is considered unimportant.

We conduct three controlled experiments to compare the effectiveness of our new proposed ranking approach using the MSE to the previous ranking approach using equation 2. For these experiment we will refer to these approaches as the "New" and "Old" methods, respectively. In the first two experiments we use only cell based features, similar to the work in [5], and compare the ranking results between the two approaches using a gaussian uncertainty model and an exponential uncertainty model. For the third experiment we compare the two approaches using some of the new features generated in this work, in particular X-Y coordinates.

In this experiment, we take 130 cells and make 260 features by separating output rising and falling transitions as two features. Each feature is occurrence-based. We randomly generate 5000 paths, each consisting of 10-25 cells. Every occurrence of each feature is associated with two unmodeled timing deviations, one random and one systematic. The random and systematic gaussian deviation are generated assuming a normal distribution with 3σ equal to 20% of the average cell delay. The systematic exponential deviation is generated using an exponential distribution with mean equal to 5% of the average cell delay. Figure 7(a) and (b), show the amounts of the systematic deviations associated with the 260 features for the gaussian and exponential uncertainty models, respectively. Based on these uncertainty models we create a learned model using the SVR algorithm. Then, we used both feature ranking methods to rank the features based on importance.



(a) Gaussian Uncertainty Model (b) Exponential Uncertainty Model

Figure 7. Gaussian and Exponential Uncertainty Models for Cell Delays

Table 1. Ranking comparison between Old and New method for a gaussian uncertainty model

rank	1	2	3	4	5	6	7	8	9	10	sum
Ideal list	26.2	20.9	18.5	18.5	18.4	17.8	17.8	17.6	17.4	17.1	190.1
Old Method	26.2	18.5	20.9	17.8	18.5	17.1	16.8	18.4	17.8	17.6	189.5
New Method	26.2	18.5	20.9	18.5	16.8	17.8	17.8	17.6	18.4	17.1	189.5

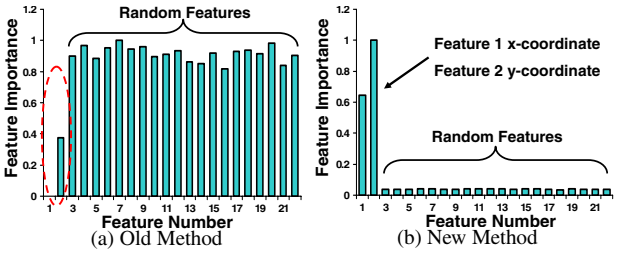
Table 1 shows the comparison of the ranking results for the top 10 features for the gaussian uncertainty model. The "ideal list" is the true ranking based on the actual systematic deviations. The sum of these top 10 deviations is $190.1ps$. Notice that the Old and New ranking methods are very comparable in accuracy (both with $189.5ps$) and both found 9 out of 10 from the ideal list. The only difference between the two rankings is a slight re-ordering of the 10 features.

Table 2. Ranking comparison between Old and New method for a exponential uncertainty model

rank	1	2	3	4	5	6	7	8	9	10	sum
Ideal list	208	202	202	190	173	118	117	106	102	102	1521
Old Method	202	208	190	202	173	117	118	102	102	96	1512
New Method	208	202	190	202	173	88	118	117	102	102	1504

Table 2 shows the comparison of the ranking results for the top 10 features for the exponential uncertainty model. Again, the results show that both the Old and New ranking methods can accurately rank 9 out of 10 from the ideal list.

For the third experiment we compare the ranking results between the old and new method using some of the new features from this work. This experiment is the same as the illustrative example from section 3.1, where a path delay mismatch map is generated based on X-Y coordinates. In addition to using the X-Y coordinates as features, we use 20 random variables that have no impact on the delay difference. Once a model is generated using the SVR algorithm, we rank the features based on importance.



(a) Old Method (b) New Method

Figure 8. Ranking of X-Y coordinates plus 20 random features using Old and New ranking methods. Where feature #1 and #2 are X and Y respectively.

Figure 8(a) and (b), show the ranking results using the Old and New method, respectively. In both figures, feature number 1 and 2 correspond to the X and Y coordinates. When using the Old ranking method,Figure 8(a), the X and Y coordinates are ranked as the two least important features. This result is obviously wrong, because the delay difference was generated using only these two features. Notice that the new ranking method,Figure 8(b), ranks the X and Y features correctly as the two most important features.

To summarize, both ranking methods have comparable accuracy for occurrence-based features sets, however the Old method does not work when using descriptive-based features. This is because the Old method [5] uses a weighted sum of the feature values to calculate importance. When higher feature values do not translate into more uncertainty (i.e. X-Y location, temperature, etc), then this importance metric does not make sense. The New ranking method is generalized enough to work with any feature set.

8 Industrial Experiment

In this section we will show the effectiveness of our diagnosis approach on an industrial ASIC design. The design has 220k cells based on a 90nm technology node. The die area for the design is 1170.28*um* x 1170.28*um*. To extract all the features that we listed in section 4, we needed access to the following: 1)Design netlist, 2)Lef/Def files, 3)Standard cell timing library. Using the design netlist and Lef/Def files, parasitic extraction was conducted to obtain the interconnect RC's (SPEF file) used by an STA tool for path delay prediction. Using a typical corner timing library the STA tool analyzed 4108 of the most critical paths, where each path was composed of 15 to 30 cells. This design uses a standard cell timing library consisting of 134 cells.

All cell-based features were extracted from the timing report and timing library. These features included: Cell type, cell output transition, cell *%Load* capacitance and active transistor type. The interconnect-based features and location-based features were extracted from the timing report and the Def file. These features included: Number of metal layers and vias that each stage of a path traverses through, the path area and X-Y coordinates for each path. As previously mentioned this work does not analyze any dynamic effects, so no dynamic features were extracted. Finally, we approximated each paths margin as 10% of the total delay from the timing report. This values was used as a feature to analyze the impact of margin on delay mismatch.

In order to analyze the effectiveness of our approach we conducted controlled experiments. Similar to section 7, we associate a systematic and a random timing deviation for each feature. Monte-Carlo simulations are performed on the 4108 paths using the original timing library along with the systematic and random deviations to create silicon measurements. For some paths multiple samples were created to simulate having the same paths measured on multiple die. To differentiate them, each die is arbitrarily given a die index value. To summarize, there is a potential total of 577 extracted features for this design (134 cells * 2 high/low transitions * 2 normal/extreme load + 2 transistor types + 2 X-Y coordinate + 1 path area + 10 metal layer bins + 25 via bins + 1 die index = 577). We say potential number of features because not all cells, cell transitions and cell loads appear in the 4108 paths being analyzed.

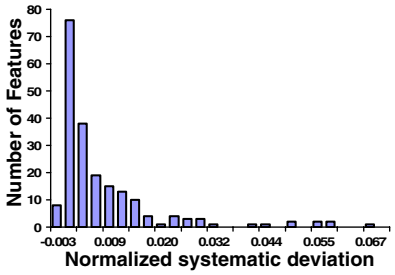


Figure 9. Systematic delay deviation

Figure 9, shows the systematic delay deviation for all features in the design. In this section all delay values are normalized to the maximum predicted path delay value. Notice that the systematic delay deviation is sampled on a exponential curve. Based on our research this seems to be the most realistic timing mismatch model, where most features have very small timing deviation and only a few have a large deviation on silicon.

Table 3. The number of the top *i* true ranking features our SVM-ranking method found

Top i true rank, $i=$	1	2	3	4	5	6	7	8	9	10
SVM rank j of the top i, $j=$	0	2	2	2	3	4	5	6	7	7

Table 3, shows the number of the top *i* true largest deviating features (consisting of both occurrence- and descriptive-based) that our proposed diagnosis methodology found, where $i = 1, 2, ..., 10$. For example, the column $i = 1$, shows whether our method correctly ranked the true largest deviating feature. Because the value is 0, it shows that our method incorrectly ranked a different feature as the most important. Column $i = 2$, shows how many of the top 2 true largest deviating features where ranked by our method, and so on. Notice that 7 of the top 10 true largest deviation

features where identified correctly (column $i = 10$). To understand why three features are missed, we analyzed their occurrences and discovered they only appear $1, 25$ and 40 times in the dataset. To give a perspective most other features appeared hundreds to thousands of times. Hence, one can hypothesize that because there are fewer paths utilizing these features, we do not have enough information in the path dataset to quantify the importance of the features and as a result they are ranked lower.

In summary, the ranking result depends on the distribution of the timing deviation associated with the features. When a few features have a large deviation it is easier for the methodology to identify them. In practice, these features are a guide to fixes of the timing model or design methodology. It is likely that one can only afford to fix a few of these due to the time and engineering cost. Hence, from the perspective of fixing, the top 10 features do not have to be perfect. For example, if one can identify 7 out of 10 features with large timing deviation, fixing these 7 can still greatly improve design-silicon timing correlation. In the next section we will reference some work done in the area of silicon repair and conclude this work.

9 Conclusion

One may wonder how to improve a design once problematic features are identified. Fixing design involves an entirely different set of techniques that are out of the scope of this paper. In the literature much work has been done proposing ways to improve design [19, 31, 32, 33, 34, 35, 36, 37, 38]. Those techniques can be more effectively utilized if the most critical issues are identified first. While many potential issues have been brought into attention as technology advances, it is usually difficult to assess, among all these potential issues and for a particular design and/or design methodology, which demand the most attention. The proposed methodology provides a solution to this problem and hence, facilitate effective allocation of design resources to go after those most relevant issues.

References

[1] K. Killpack, et al," "Analysis of Causation of Speed Failures in a Microprocessor: A Case Study," To appear in IEEE *Design & Test* Special Issue on Silicon Debug and Diagnosis 2008.

[2] C. Bittlestone, A. Hill, V. Singhal, Arvind N.V, "Architecting ASIC Libraries and Flows in the Nanometer Era," In *DAC* 2003.

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

[4] P. Bastani, B. Lee, L. Wang, S. Sundareswaran, M. Abadir, "Analyzing the risk of timing modeling based on path delay tests," *ITC*2007.

[5] P. Bastani, et al,"An Improved Feature Ranking Method for Diagnosis of Systematic Timing Uncertainty," *VLSI-DAT* 2008.

[6] P. Bastani, N. Callegari, L. Wang, M. Abadir, "Statistical Diagnosis of Unmodeled Timing Effect," *Submitted to DAC* 2008.

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

[8] Y.-C. Hsu and S.K. Gupta. "A New Path-Oriented Effect-Cause Methodology to Diagnose Delay Failures," ITC, 1998, pp. 758-767.

[9] Z. Wang, et al, "An Efficient and Effective Methodology on the Multiple Fault Diagnosis," In *ITC* 2003, pp. 329-338.

[10] A. Kristic, et al, "Delay Defect Diagnosis Based Upon Statistical Timing Models - The First Step," In *DATE* 2003.

[11] P. Zuchowski, P. Habitz, J. Hayes, J. Oppold, "Process and Environmental Variation Impacts on ASIC Timing," In *ICCAD* 2004.

[12] S. Nassif, "Design for Variability in DSM Technologies," In *IEEE International Symposium on Quality Electronic Design* 2000.

[13] S. Nassif, "Modeling and Forecasting of Manufacturing Variations," In *International Workshop on Statistical Metrology*, 2000.

[14] D. Josephson, et al, "Debug Methodology for the McKinley Processor," In *ITC* 2001. pp 451-460.

[15] D. Josephson, "The Manic Depression of Microprocessor Debug," In *ITC* 2002.

[16] W. Huott, et al, "The Attack of the 'Holey Shmoos': A Case Study of Advanced DFD and Picosecond Imaging Analysis,"In *ITC* 1999.

[17] C. Pryon, et al, "Silicon Symptoms to Solutions: Applying Design-for-Debug Techniques," In *ITC* 2002.

[18] M. Bass, et al, "Design Methodologies for the PA 71000LC Microprocessor," In *Hewlett-Packard Journal*, 46(2):23-25 April 1995.

[19] D. Josephson, "The Good, the Bad, and the Ugly of Silicon Debug," In *DAC* 2006.

[20] T. Thiel, "Have I Really Met Timing? - Validating PrimeTime Timing Reports with Spice," In *DATE* 2004.

[21] O. Sinanoglu, P. Schremmer, "Diagnosis, Modeling and Tolerance of Scan Chain Hold-Time Violations," In *DATE* 2007.

[22] N. NS, T. Bonifield, et al, "BEOL Variability and Impact on RC Extraction," In *DAC* 2005.

[23] U. Narasimha, A. Hill, N. NS, "SmartExtract: Accurate Capacitance Extraction for SOC Designs," In *IEEE VLSI Design* 2006.

[24] P. Larsson and C. Svensson, "Noise in digital dynamic CMOS circuits," In IEEE J. Solid-State Circuits June 1994, pp. 655-663.

[25] Sanjay Pant, Eli Chiprout, "Power Grid Physics and Implications for CAD," In *Proc. DAC* 2006.

[26] P. Bastani, K. Killpack, L. Wang, E. Chiprout, "Speedpath prediction based on learning from small set of exmaples," *Submitted to DAC*'08.

[27] V. Vapnik The Nature of Statistical Learning Theory. Springer 1999.

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

[29] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press 2004.

[30] Leo Breiman, "Random Forests," In *Machine Learning Journal (45)* pp. 5-32, 2001.

[31] S. Kulkarni, D. Sylvester, D. Blaauw, "A Statistical Framework for Post-Silicon Tuning through Body Bias Clustering," In *ICCAD* 2006.

[32] M. Najibi, et al, "Dynamic Voltage and Frequency Management Based on Variable Update Intervals for Freq. Setting," *ICCAD* 2006.

[33] B. Quinton, et al, "Asynchronous IC Interconnect Network Design and Implementation Using a Standard ASIC Flow," *ICCAD* 2005.

[34] R. Livengood, "Design for Debug for Silicon Microsurgery and Probing of Flip-Chip Packaged Integrated Circuits," In *ITC* 1999.

[35] Lakshmanan, et al, "Incremental Dummy Metal Insertion," *US Patent No.* 7,260,803 B2. August 2007

[36] L. Tien, "Method for reducing layer revision in an engineering change order," *US Patent No.* 7,137,094. November 2006.

[37] K. Chang, I. Markov, V. Bertacco, "Automating Post-Silicon Debugging and Repair," *ICCAD* 2007.

[38] L. Yuan, S. Leventhal, G. Qu, "Temperature-Aware Leakage Minimization Technique for Real-Time Systesms," *ICCAD* 2006.