Speedpath Prediction Based on Learning from a Small Set of Examples *

Pouria Bastani[1], Kip Killpack[2], Li-C. Wang[1], Eli Chiprout[2]
[1]University of California - Santa Barbara
[2]Intel Corporation, Hillsboro, OR

ABSTRACT

In high performance designs, speed-limiting logic paths (*speedpaths*) impact the power/performance trade-off that is becoming critical in our low power regimes. Timing tools attempt to model and predict the delay of all the paths on a chip, which may be in the millions. These delay predictions often have a significant error and when silicon is measured there is a large variation of path delays as compared to the prediction of the tools. This variation may be caused by process, environmental or other effects that are often unpredictable. It is therefore desirable to use early silicon data to better predict and model potential speedpaths for subsequent silicon steppings. In this paper, we present a novel machine learning-based approach that uses a small number of identified speedpaths to predict a larger set of potential speedpaths, thus significantly enhancing the traditional timing prediction flows post-silicon. We demonstrate the feasibility of this approach and summarize our findings based on the analysis of silicon speedpaths from a 65nm P4 microprocessor.

Categories and Subject Descriptors

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

General Terms

Algorithm, Performance, Reliability

Keywords

Speedpath, Timing analysis, Learning

1. INTRODUCTION

In high-performance chip design, timing analysis and optimization does not stop at first silicon. Silicon information is often analyzed carefully to drive further speed or power improvements. This process of *silicon steppings*, allows the detection and fixing of speed limiting paths (*speedpaths*). A speedpath is a path that limits the performance of a chip where the performance can be defined by observing the result of applying functional legacy tests. Because speedpaths can be observed at different cycles of a functional test sequence where different parts of the chip are exercised, a chip can have multiple speedpaths [1].

*This work supported in part by CA Micro/Intel project No. 07-079

DAC 2008, June 8–13, 2008, Anaheim, California, USA

Speedpaths are not well-predicted by the timing flows. The deficiency of these CAD flows comes about due to a multitude of process, design and environmental effects that are either unknown or too difficult to model and simulate for millions of paths. It is therefore sometimes easier to uncover speedpaths in silicon and fix them to produce a new silicon stepping. Simultaneously, based on the silicon feedback to the timing flows, other potential speedpaths can be modeled more accurately and fixed even before they are detected in the next silicon. These iterations may continue until performance is pushed to a satisfactory level, at which time design changes are frozen and the design is introduced to the market in high volume.

Each silicon stepping has significant associated manufacturing cost. Moreover, silicon speedpath detection demands a large amount of engineering effort. It is economical therefore to be able to utilize silicon data effectively for improving subsequent silicon steppings. This may be done by enhancing the CAD for timing and path prediction flows with the silicon data. However, it is not obvious how to best use the information obtained from the few detected speedpaths to accomplish this.

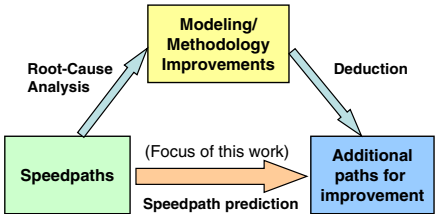


Figure 1: Two approaches to utilize speedpaths

Once a set of speedpaths are found, there are two fundamental ways by which one may enhance the delay prediction of all paths on the die. Figure 1 illustrates the two approaches. In the first approach, one tries to get to the root cause(s) of speedpath mis-prediction. A root cause may lead to improvements in delay modeling or in timing methodology or both. Based on the improved model and/or the improved methodology, more accurate design optimization can be carried out before the next stepping.

While the first approach is intuitive, it may not be the most desirable approach when time-to-market is considered. This is because finding root causes and improving modeling can be hard and time-consuming. The interesting point to observe is that, if the goal is simply to isolate potential speedpaths one need not improve the modeling and methodology first. Instead, a second approach can be employed, which can take place in parallel to the first, that predicts other potential speedpaths based only on observed *properties* of known speedpaths instead of enhanced models.

The second approach is attractive because it avoids the difficulty of finding the root causes of mispredicted timing paths. However, it may not be as effective as the first approach, if it does not con-

form to true root cause effects. To enhance the effectiveness of the second approach, it is therefore desirable to develop a methodology that can search the design and identify all paths that have *similar* characteristics to those found speedpaths. With such a methodology in place, the second approach can potentially have the same effectiveness of the first approach, without the high cost associated with finding the root causes. This may then be a complimentary approach to the standard timing flows.

In this paper, we present a novel methodology that utilizes a machine learning technique called *support vector analysis* [10] to learn from the characteristics of speedpaths. The result of learning is a model of speedpaths. This model can then be used to search the design and collect additional paths whose characteristics are similar to the speedpaths.

The challenge of building such a learning methodology lies in the fact that the number of speedpaths used in the analysis is often small, in the 100's. Hence, it is difficult to treat the problem as a *binary classification* problem [2] and try to learn a model that can differentiate between speedpaths and non-speedpaths. Instead, we need to develop a method that focuses the learning only on the speedpaths. This is called *one-class learning*. In this paper, we describe how to apply one-class learning to solve the speedpath prediction problem where the number of speedpaths given is small. We will discuss how to derive path "characteristics," how to define the similarity between paths, and how to build an effective model to predict additional speedpaths.

In order to check the validity of the proposed learning methodology, we conducted detailed analysis on a set of speedpaths, i.e. the root-causing step shown in Figure 1. Our objective was to see if the result of this detailed analysis would confirm the result given by the learning approach. In particular, we were interested in observing if the paths identified by the learning would also be the paths suggested from the detailed analysis. We systematically consider three potential sources of root cause: (1) multiple input switching (MIS), (2) coupling noise and (3) localized powergrid noise. For some of the speedpaths, we discovered convincing evidences that they may be due to one of the above three reasons. Then we manually inspected the paths identified by the proposed methodology and observed that those paths shared the same characteristics as the speedpaths, which lead to the reasons that caused them to be speedpaths.

The rest of the paper is organized as the following. In Section 2 we give an overview of our speedpath data collection. Section 3 explains the theoretical foundation for how we use support vector analysis to build a model and use it to predict potential speedpaths. In Section 4 we describe what path characteristics are and how we use them in our analysis. In Section 5, we present the results from our searching method. In section 6 we validate our proposed methodology by conducting root cause analysis and in section 7 we conclude our work and point to future work.

2. SPEEDPATH DATA COLLECTION

In microprocessor silicon steppings, speedpaths are identified using functional testing. This process employs various intrusive and non-intrusive methods such as shmoo, LADA, TRE, LVP, pico-probe, FIB, On-Die-Clock-Shrink (ODCS), and programmable clock-edge skewing [5]. In addition to the traditional method, [4] proposes using logic and timing CAD to identify speedpaths. This CAD method takes tester shmoo data as input and identifies the corresponding speedpaths. Using tester shmoo data from a 65nm P4 microprocessor and the method described in [4], we analyzed 56 sightings observed in four large functional blocks. For each of the 56 sightings we identified a single speedpath, out of which 15 were topologically unique (same physical path, but on different clock cycles).

The fact that a path shows up as a speedpath, leads to the logical conclusion that there are some $special$ things happening on the path to cause it to become a speedpath (during the particular cycle and under the particular vector). This special combination of "things" is the root cause and usually it is not easy to uncover.

As mentioned in the Introduction, to eliminate a speedpath and thus improve silicon frequency, it is not necessary to know the root cause. Hence, even without knowing the root causes for the 56 speedpaths, we can still improve them so that they do not show up as speedpaths in future revisions. This is recommended, but is not the only way we can take advantage of the 56 speedpaths. For the purpose of improving design, it would be more effective if one can also identify additional paths *similar* to the 56 speedpaths, in the sense that these additional paths have the potential to be speedpaths in future revisions. By improving the delays also on these additional paths, we take away the chance of them being the speed limiting paths in future revisions of the design. This can be done in an effort to improve performance of the design over silicon steppings. In the following, we describe such a methodology.

3. SUPPORT VECTOR ANALYSIS

Suppose that a path p can be described as a vector $\vec{v} = (f_1, \ldots, f_n)$ of n numerical values, where each f_i is called a *feature*. $\vec{v}$ describes the characteristics of the path. For now, we postpone the discussion of how to generate these features. It will be discussed in section 4 later. Given the 56 speedpaths, we assume that we have 56 vectors $V = \{\vec{v_1}, \ldots, \vec{v_{56}}\}$ describing the characteristics of these paths. Our goal in this section is to develop a method that can learn from V and produce a model M which can be used in the following way.

$M(\vec{v})$ for any path p described by $\vec{v}$, gives us a value telling us how likely the path p may become a speedpath. The "likeliness" is evaluated based on how similar path p is to one or more of the 56 speedpaths. The similarity between path p to a speedpath p_i is evaluated based on a *kernel* function $k(\vec{v}, \vec{v_i})$, which takes the two vectors as inputs and produces a value that measures the degree of the similarity between them.

To see how M might look, consider two naive examples. In the first example, M can be the following simple function: $M(\vec{v}) = 1$ if $\vec{v}$ is identical to any $\vec{v_i}$; otherwise $M(\vec{v}) = 0$. If we use this model to search the design, essentially we try to find all paths that have identical characteristics to one of the speedpaths. Intuitively, this naive model does not work well. Consider that we do find a path whose description is exactly the same as one of the speedpaths. The fact that the path is not an observed speedpath implies that our method to describe the path characteristics (feature generation method) is not complete. If all the information required to determine if a path is a speedpath is contained in the path description vector, this would not have happened. From this perspective, the naive model should not be used to find potential speedpaths. However, it might be used for a sanity check on the completeness of the feature generation method.

In the second example, one can try to generalize from the characteristic descriptions of the 56 speedpaths to establish an average speedpath characteristic. Note that each vector $\vec{v_i}$ can be seen as a point in an n dimensional space. For any two such points v, v', we can compute its Euclidean distance $d(v, v')$. Let the similarity function be $k(v, v') = \frac{1}{d(v,v')}$. Then, given the 56 points, we can find its *center mass* which is also a point $\vec{v_c}$ in the n dimensional space. A center mass is the point $\vec{v_c}$ such that the total distance $\sum_{i=1}^{56} d(\vec{v_c}, \vec{v_i})$ is the minimum.

Given any path description $\vec{v}$, then we can compute its similarity to the center mass as $k(\vec{v_c}, \vec{v})$. Then, the model M is nothing but the function $k(\vec{v_c}, \cdot)$ that measures how similar the path is to the average speedpath characteristic represented by the point of the center mass.

While in the first example, we have a model that is too specific, which does not give us any result, in the second example we may have a model that is too general, which can mislead the result. For example, the 56 points may spread widely in the n dimensional space. As a result the center mass may be far from any of the points. Then, any path vector close to the center mass may not be similar to any one of the speedpaths.

The two naive examples illustrate how one can learn from the 56 path descriptions and construct a learned model. Because neither of them works effectively, we need to develop a more sophisticated algorithm. In this work, we develop such an algorithm based on support vector analysis [10].

3.1 Support vector unsupervised learning

There are three types of support vector algorithms [11]: regression, classification and one-class. Regression and classification belong to supervised learning and one-class belongs to unsupervised learning. Typical unsupervised learning problems include clustering of data points as well as finding outliers in a given set of data points. In this work, we use the support vector one-class algorithm. In our application, a support vector model takes the form:

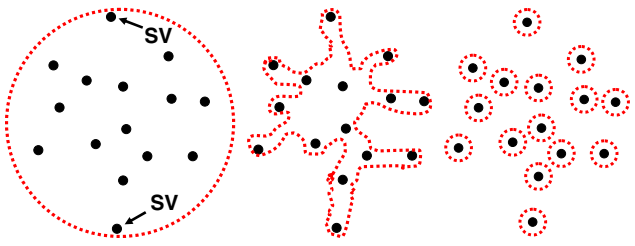
$$M(\vec{v}) = \sum_{i=1}^{56} \alpha_i k(\vec{v}_i, \vec{v}) \quad (1)$$

In this model, some α_i's are zero. Only path vectors with $\alpha_i \neq 0$, are used to build the model. These vectors are called *support vectors*. Others are called non-support vectors. For a path $\vec{v}_i$, a larger α_i indicates that the path is more important for determining the value of M. We see that the model M computes a weighted average of the similarity between the path vector $\vec{v}$ and all the support vectors.

If we take a threshold value t and consider all vectors $\vec{v}$ such that $M(\vec{v}) \geq t$, essentially this defines a subspace (possibly discontinuous) in the n dimensional space. We can then call this subspace the region of potential speedpaths because any path whose description vector falls inside the region, is at least t-similar to the speedpaths.

For a support vector model, its *model complexity* can be measured by the number of support vectors [11]. The larger this number is, the more complex the model becomes. A more complex model is also more specific. A less complex model is more general. In essence, we can think that a more complex model is more strict in predicting a path as a potential speedpath. On the contrary, a less complex model is less strict in doing so.

To illustrate the concept of model complexity, Figure 2 shows three possible models for a simplified path vector set on a two dimensional space.



(a) Model 1 (b) Model 2 (c) Model 3

Figure 2: Illustration of model complexity

Figure 2(a), shows the simplest model that is a circle region defined based on only two support vectors. The remaining dots are non-support vectors. All these vectors represent speedpaths. If we consider all other points inside the circle as being similar to the speedpaths, we observe that the area defined by this model is the largest. Figure 2(b), shows a more complex model, where the number of support vectors defining the region is increased. This model is more specific than the previous because its area is smaller, i.e. there exist paths determined as potential speedpaths by the first model but not by the second one. Figure 2(c), shows the most complex model, where every training sample becomes a support vector. This model spans the minimal space needed to cover all the speedpaths.

Note that a more general model reduces the chance of false positive, i.e. the chance of rejecting a path that is actually a potential speedpath. A more specific model, on the other hand, reduces the chance of false negative, i.e. the chance of accepting a path that is actually not a speedpath. Intuitively, one would think that we want a model with a reduced chance of false positive. This is because we do not want to miss any potential speedpath.

This intuitive strategy makes sense only if the accuracy of the model is also high, meaning that it does not predict too many good paths as potential speedpaths. For example, if we identify 1000 potential speedpaths and only 10 of them are real, then the remaining 990 paths lead to a waste of engineering effort on improving them. Moreover, speeding up these paths may result in increased power consumption with little gain on the actual frequency.

As discussed before, speedpaths usually are due to special things not explicitly considered by models and simulation. As a result, we would expect their characteristic descriptions $\vec{v_1}, \ldots, \vec{v_{56}}$ to be quite unique, meaning that they may spread widely in the n dimensional space. If we try to build a general model shown in Figure 2(a), this model may potentially cover a large region in the space. Consequently, many paths would be considered as potential speedpaths. On the other hand, if we try to build a very complex model shown in Figure 2(c), this model covers a very small space. Thus, only paths that are identical to a observed speedpath are going to be considered as potential speedpaths. We would like to find a optimal point in between these two extreme cases.

To summarize, because the number of speedpaths used to build the model is small and because the characteristics of these paths are diverse, it is unrealistic to expect we can learn a general model that captures only potential speedpaths without including other good paths. Therefore, in our algorithm we do not try to build a general model. Instead, our strategy is to start by building the most complex model, followed by a sequence of models with gradually reduced model complexity. Then, we develop a method to decide when to stop and use the model at the stopping point as our optimal model.

Suppose that M_i and M_{i+1} are built in two consecutive steps where M_i is more complex than M_{i+1}. Suppose we use these two models to search the design and identify the top k paths that have the highest potential to be speedpaths. Let these two sets of paths be S_i and S_{i+1}, respectively. We can check to see if $S_i \approx S_{i+1}$. If the two sets almost agree with each other, then we have a higher confidence that M_i and M_{i+1} are good models to use.

Although in this paper we do not provide a theoretical reason why $S_i \approx S_{i+1}$ can be a good heuristic to use, we do experimentally validate that it does lead to more meaningful results. Sections 5 and 6 explain the experimental results and the validation process. Moreover, it is intuitive to see that if S_i differs from S_{i+1} much, it cannot be the case that both of them are good. Even though one of them may be a good model to use, we would not have a clear way to judge which one. Hence, the logical choice is to continue on building M_{i+2}.

In this work, our heuristic is based on checking the five sets S_{i-2}, S_{i-1}, S_i, S_{i+1}, S_{i+2} for every model M_i (i.e. two before and two after). We determine the model M_i such that these five sets agree with each other the most. Then, we consider M_i our best model. The effectiveness of this heuristic will be discussed in section 5.

3.2 Kernel Function

In support vector analysis, the resulting model complexity can be controlled by how we define the kernel function. As explained before, a kernel defines how to measure the similarity between two path vectors. One common kernel is a radial based function:

$$k(\vec{v}_i, \vec{v}) = e^{\frac{-\|\vec{v}_i - \vec{v}\|}{\sigma}} \quad (2)$$

Suppose that we have four paths, described by $\vec{v}_{SV}, \vec{v}_1, \vec{v}_2, \vec{v}_3$, where each vector consists of two values: $\vec{v}_{SV} = (1, 0.2)$, $\vec{v}_1 = (1.02, 0.19)$, $\vec{v}_2 = (3, 0.1)$, and $\vec{v}_3 = (5, 1.2)$. Suppose that $\vec{v}_{SV}$ describes a speedpath. If we were to select potential speedpath(s) from the other three, by observing the values in the vectors, we would pick $\vec{v}_1$. Suppose that we build a model using $\vec{v}_{SV}$ as the only support vector and let its α, from eq. (1), be 1. Then the model is simply $k(\vec{v}_{SV}, \cdot)$. Table 1 shows the similarity values given by the model based on three different chosen σ's in the kernel function.

Table 1: Three different similarity calculations

	$\sigma = 1/40$	$\sigma = 1$	$\sigma = 40$
$k(\vec{v}_{SV}, \vec{v}_1)$	0.3011	0.9704	0.9992
$k(\vec{v}_{SV}, \vec{v}_2)$	≈ 0	0.1224	0.9488
$k(\vec{v}_{SV}, \vec{v}_3)$	≈ 0	0.0067	0.9048

It is interesting to observe that for a very small σ, it is easy to decide that $\vec{v}_1$ is similar to $\vec{v}_{SV}$, and $\vec{v}_2$, $\vec{v}_3$ are not similar to $\vec{v}_{SV}$. For a large σ, we see that the similarity values for all three vectors are large (above 0.9). Note that the maximum value possibly reported by this kernel function is 1. For $\sigma = 1$, the value for $\vec{v}_1$ is significantly larger than the other two. If we believe that only $\vec{v}_1$ should be called a potential speedpath, then we see that using a small σ helps to make this decision more easily. However, if we believe that all three vectors should be potential speedpaths, then using a large σ helps make this decision.

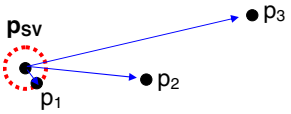


Figure 3: Radial-based kernel similarity example

It is interesting to note that a small σ leads to a more specific model (a more complex model), i.e. a path is considered to be similar to the speedpath if their description vectors only differ by very little. This is illustrated in Figure 3 where P_i represents the paths in the radial space. On the other hand, if a large σ is used, we are building a more general model.

4. FEATURE GENERATION

In this section we explain how a path can be described as a vector. To do so, we first discuss the potential sources of modeling mismatch. Based on these sources we describe how the actual features used in the experiment were selected.

For speedpath prediction it is necessary to analyze situations with potentially *unaccountable* effects. Unaccountable effects refer to special situations where silicon behaves unexpectedly. Typically these effects are not accounted for due to their rarity of occurrence along with the large additional cost of modeling them. We group them into five categories of effects:

- **Topological effects** are modeling issues related to location, orientation, ordering and configuration. This can include, XY location, layout orientation, neighboring elements, metal layer connectivity, and/or number of via connections.
- **Dynamic effects** are modeling issues based on input patterns. These effects show up during regular operation of a design. Examples are: Multiple input switching, cross coupling noise, dI/dt voltage droop, RC voltage droop, temperature variation.
- **Static effects** are modeling issues that are independent of input patterns. These effects will appear everywhere the static element is placed. An example of this is if a specific cell is mis-modeled. Everywhere the cell appears it will be mis-modeled independent of the patterns used to activate it. Other examples are during RC extraction, specific shapes or layers can be mis-modeled due to the extraction methodology.
- **Statistical effects** are modeling issues dealing with systematic variations in the physical characteristics of devices and interconnects related to the process. These effects include intra-die and inter-die variations due to lithography, dopant fluctuations and/or chemical mechanical planarization (CMP).
- **Random effects** include any effects that are not based on a statistical system. Examples of these are alpha particles present during manufacturing or seismic activity during CMP. These effects cannot be predicted or modeled.

From the set of effects described above, an engineer can develop a set of features that can be used to differentiate between good paths and speedpaths. This step is called $feature\ generation$. For example the authors in in [2] [3] use "delay entity" to describe paths to facilitate the learning of sources of timing uncertainty. In this work we use "features" to describe paths to facilitate the measurement of similarity among them. In addition we want this similarity measurement to be in such a way that if a path is very similar to a speedpath, then the unaccountable effects associated with the speedpath are also likely to show up on the path. Keep in mind that the feature definitions are entirely up to the user and can depend on the application scenario. For example, if we have prior knowledge that all the speedpaths are due to a power related issue then when applying the proposed methodology we may intentionally exclude other sources of effects in the feature definitions.

In this work, based on our domain knowledge and application scenario we choose the following features:

1. Active device type in a path due to logic sensitization. These are based on four different flavors of N and P MOS transistors.
2. The predicted arc or stage delay for a particular cell in the path. This is a function of input transition time, output load and net resistance and capacitance.
3. Percent of total path delay due to the nets as opposed to cells. $\%Net = NetDelay/(TotalPathDelay)$
4. Dynamic switching activity in the region.
5. Cross coupling aggressor impact.
6. Temperature variation, using Infrared Emission Microscopy to obtain a temperature map during functional operation [7].

With these feature definitions every path can now be described as a vector. In the next section we describe the results of applying our proposed methodology to identify potential speedpaths.

5. RESULTS

A large design can have millions or even billions of different paths and we may not be interested in searching all of them. In this work, we select 19,000 paths that were sensitized during functional testing. Our goal is therefore to select potential speedpaths from this list.

Using a radial-based kernel and the heuristic described in section 3.1, we learned a sequence of models and determined the optimal model complexity. In the heuristic each path set S_i is the 50 most potential speedpaths resulting from model M_i. Figure 4, shows the results from the most complex model consisting of 56 support vectors to the least complex model consisting of 8 support vectors. The optimal model occurs at 24. Where 48 out of 50 paths are shared by the five models M_{22}, M_{23}, M_{24}, M_{25}, M_{26}.

It is interesting to note that with more than 26 support vectors (higher model complexity) there is little agreement between learned models. In other words, above 26 the model is unstable in the sense that a small change in complexity results in a large change in how the model determines the potential speedpaths. As complexity falls to between 23 and 19 support vectors there is no agreement between models, showing that the model becomes unstable again. Although model complexity below 19 support vectors tends to produce stability, those models are not chosen because: 1) the number of shared paths is much smaller than the 48 given by the optimal model, and 2) as mentioned before we prefer a more specific (higher complexity) model than a general one (lower complexity). It is interesting to note that if we only compare three models during our complexity heuristic, M_{i-1}, M_i, M_{i+1}, the results are very similar to Figure 4, with 24 support vectors again being optimal.

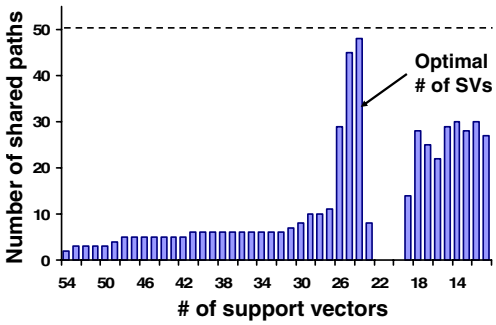


Figure 4: Sweeping model complexity

Using the optimal learned model M_{24}, we search all the 19,000 paths. For each path vector $\vec{v}$ we compute its similarity $M_{24}(\vec{v})$ and we sort all the similarity values and the result is shown in Figure 5. To select potential speedpaths for further improvement it is very easy to define a similarity threshold based on the knee of the curve. For example in Figure 5 we show such a threshold. From the figure we can see that there are 6 paths that clearly stand out from the rest. Furthermore, when zooming in on the right of Figure 5, we observe 7 additional paths that stay above the threshold. Using our searching methodology, we identified 13 paths that we deemed to be potential speedpaths. In the next section we will describe how we did root cause analysis to verify our results.

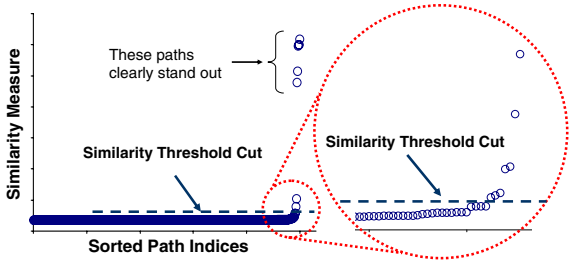


Figure 5: Result of support vector similarity ranking

6. ROOT CAUSE ANALYSIS

After identifying speedpaths and seeing the large discrepancy between predicted and measured results, we analyze them for potential root causes. We initially suspected three dynamic effects: multiple input switching, cross coupling noise and localized voltage droop.

6.1 Multiple input switching pushout

Multiple input switching occurs when multiple inputs of a gate switch in temporal proximity, in which the max (min) delay is more (less) than the pin-to-pin delays [6]. To analyze this effect, we simulate the entire logic cone for each speedpath using the test vectors from the cycle on which the speedpath is observed in silicon. We use a SPICE-accurate Monte Carlo statistical simulation, mentioned in [4], to determine which gates have an impact on cone arrival time. If two or more paths influence the cone arrival time, then we can conclude that there are MIS effects within the logic cone.

Many of the cones do not have MIS situations such as: 1) the path was only comprised of single input switching gates, or 2) the path had gates with more than one input switching, but the secondary signal switched to a non-controlling value and had no impact on the gate functionality. On the other hand, some cones had gates with multiple inputs switching but the pushout impact was negligible due to the low probability of alignment of the switching signals. In fact, only 4 cones had MIS inputs aligned close enough to impact cone arrival time. But even on those 4 cones, the alignment was such that the MIS impact was small. This analysis implies that MIS is not the reason that these 56 paths show up slower than predicted in silicon.

6.2 Cross coupling noise

Cross coupling noise is when the capacitance between two neighboring wires cause a logic event on one wire to induce noise onto the other wire [8]. The cross coupling delay pushout for a stage is a function of the victim and aggressor slopes, victim and aggressor drive strengths, coupling capacitance, signal alignments, and direction of the transitions. For an accurate measure of the impact of cross coupling noise all these parameters must be known. Due to the size of the potential aggressor space this can be costly to calculate. If we are only interested in bounding *potential* cross coupling effects, we can greatly simplify this calculation by making a few assumptions. Since, we are only interested in coupling slowdown, we narrow down the aggressor space to only aggressors with opposite transitions than the victim wires during the cycle on which the speedpath is sensitized. For this subset of total aggressors, we assume a worst-case aggressor slope and alignment and use a noise-based coupling model to compute an upper bound on coupling pushout.

Using this simplified model we compute the coupling pushout as a function of the number of aggressors assumed to align on each stage of the speedpath. The results of the delay pushout, as a percentage of the total path delay, are shown in Figure 6.

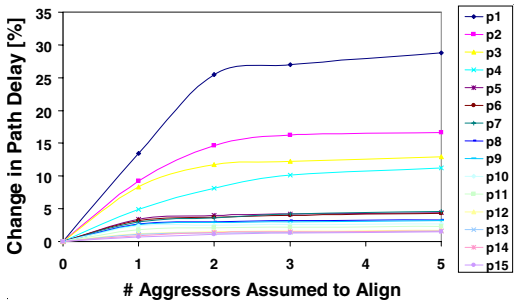


Figure 6: Aggressor Impact

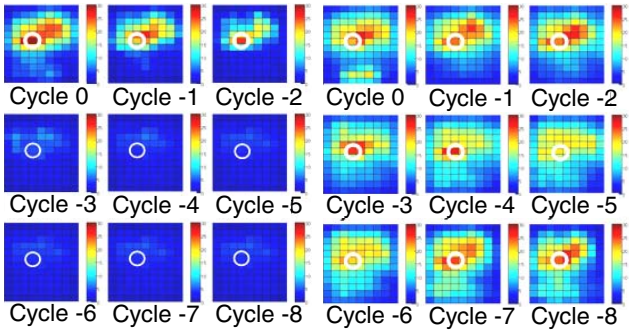
Each line $p_1, ..., p_{15}$ represents the delay pushout for a single speedpath. The x-axis shows the number of aggressors assumed to align per stage (from 0 to 5). The aggressors are sorted based on their impact. Even under our worst-case alignment and slope assumptions, along with our worst-case 5 aggressor alignment per stage, 11 of the 15 unique speedpaths experience a delay pushout of less than 5%. For these paths, coupling noise can be ruled out as the reason the paths show up slow in silicon. The other 4 paths show a much higher delay pushout due to coupling noise, thus coupling noise can be a potential reason why these paths became speedpaths.

6.3 Localized voltage droop

Localized voltage droop occurs when there is a sudden current draw (dI/dt) on a localized region on the die. If the change in the current is large enough, the power delivery system cannot supply the required current fast enough to the gates. This results in a droop in the expected voltage for those gates. The authors in [9], show that in order for dI/dt voltage droop to occur, there must be a large current $ramp$ that hits either a local or global resonant frequency of the power-grid and package. Without a large ramp, dI/dt droop is not possible.

To approximate this effect, we analyze the switching activity of a functional block for each cycle when speedpaths are observed. From the functional test patterns we know which cells are transitioning for a given cycle. Using logic transitions, capacitive load values and XY layout location for each cell, we can approximate the current draw in terms of switching capacitance. A large ramp in local switching capacitance implies a large ramp in current draw. Without this ramp we can assume that there is no change in current draw.

Figure 7 shows the results of this analysis for one of the speedpaths located in the white circle. We divided a $1100um \times 1500um$ functional block into a grid, where each individual grid is a $125um \times 125um$ square. For each grid we analyzed the switching activity over a range of 9 cycles, starting at cycle 0 and back 8 previous cycles. Cycle 0 in Figure 7(a), shows the speedpath when it appeared as a speed-limiting path on silicon. The key thing to notice is the change in switching activity at the location of the speedpath between cycle -3 and cycle 0 when the speedpath was sensitized. During cycle -3 there is almost no switching activity, shown by the dark blue color. Then by cycle 0 there is a large amount of switching activity, shown by the red color. Within 4 clock cycles this particular functional block went from no activity to large activity. This clearly shows a case where there would be a large ramp in current draw.



(a) Droop (b) No Droop

Figure 7: Switching activity for a speedpath

Figure 7(b) shows the switching capacitance when the same speedpath was sensitized on a different cycle, however this time it was not a speed-limiting path. Again the speedpath is sensitized on cycle 0. The key thing to notice in this picture is that even though there is a large amount of switching activity in cycle 0, there is no change in activity between each cycle. This means that the current draw would have been constant, and there would have been no dI/dt voltage droop. We did this analysis for all identified speedpaths and 18 out of 56 showed potential dI/dt droop. Each of these 18 sightings came from the same functional block and each time the path was speed-limiting there was also a large ramp in switching activity.

6.4 Summary

In the previous sections we presented three different dynamic effects that we suspected to contribute to the measured silicon mismatch. MIS analysis showed very little potential impact on silicon and can be ruled out. Cross coupling analysis showed that a few speedpaths are sensitive to coupling noise and it could be problematic on silicon. Similarly, localized voltage droop analysis showed that the paths in a particular functional block became speed-limiting when there was a large ramp in switching activity.

7. CONCLUSION

After learning the results from our potential root cause analysis, we inspected the characteristics of the 13 paths identified by our support vector methodology. Not only did the paths share similar characteristics to one or more speedpaths, but also if we were to select potential speedpaths manually, after the root cause analysis these 13 paths would have been selected.

In particular, all the paths had either identical or very similar combination of active device types as one or more of the speedpaths. In addition to the active devices: 1) Six paths showed high delay pushout due to coupling noise. The amount of coupling noise pushout is equal to or greater than the calculated pushout for 21 of the 56 speedpaths. 2) Four paths had a large percent of total path delay due to the net delay as opposed to cell delay. The net delay was greater than 9.5% of the total path delay which is similar to 9 of the 56 speedpaths. 3) Two paths showed high amounts of switching activity. The level of the switching activity for these paths is similar to 15 of the 56 speedpaths. Note that 2 of the 13 paths selected had both a large pushout and a large net delay.

Of the thirteen paths that were selected, three of them did not show any potential root causes. However, each of these paths has similar active device types, net delays, coupling noise pushout and switching activity to at least one speedpath. It is important to note that for these speedpaths no potential root causes were also found. This further emphasizes our earlier point that even without knowing the root cause information, our methodology can search the design and select potential speedpaths. Although our proposed heuristic to select the optimal model works well in the experiment, the theoretical reason behind its effectiveness is not explained in this work. We plan to provide such theoretical analysis in future work.

8. ACKNOWLEDGMENTS

We acknowledge the following people at Intel for their contributions to our speedpath identification and root cause analysis in the form of guidance, discussions and implementation: Chandramouli Kashyap, Suriyaprakash Natarajan, Arun Krishnamachary and Praveen Parvathala.

9. REFERENCES

[1] L.Lee, L. Wang, P. Parvathala, TM Mak, "On Silicon-Based Speed Path Identification," *Proc. VTS* 2005.

[2] P. Bastani, B. Lee, L. Wang, M. Abadir, "Analyzing the risk of timing modeling based on path delay test," *Proc. ITC*, 2007.

[3] Li-C. Wang, P. Bastani, M. Abadir, "Design-silicon timing correlation – a data mining perspective," *Proc. DAC*, 2007.

[4] K. Killpack, C. Kashyap, E. Chiprout, "Silicon Speedpath Measurement and Feedback into EDA flows," *Proc. DAC*, 2007.

[5] B. Gottlieb, et al, "Silicon Debug: What Do You Do When Your ASIC Does Not Work as Fast as Expected?" *Proc. DAC*, 2004.

[6] A. Agarwal, D. Blaauw, and F. Dartu, "Statistical Gate Delay Model Considering Multiple Input Switching," *Proc. DAC*, 2004.

[7] D.Barton, P. Tangyunyong, J. Soden, A. Liang, F. Low, et al, "Infrared Light Emission From Semiconductor Devices," *ISTFA Proc.* 1996.

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

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

[10] N. Cristianini, and J. Shawe-Taylor, "An Introduction to Support Vector Machine," *Cambridge University Press*, 2002.

[11] Vladamir Vapnik, "The nature of Statistical Learning Theory," 2nd editions, *Springer*, 1999.