Synthesizing a Representative Critical Path for Post-Silicon Delay Prediction

Qunzeng Liu
University of Minnesota
liuxx575@umn.edu

Sachin S. Sapatnekar
University of Minnesota
sachin@umn.edu

ABSTRACT

Several approaches to post-silicon adaptation require feedback from a replica of the nominal critical path, whose variations are intended to reflect those of the entire circuit after manufacturing. For realistic circuits, where the number of critical paths can be large, the notion of using a single critical path is too simplistic. This paper overcomes this problem by introducing the idea of synthesizing a representative critical path (RCP), which captures these complexities of the variations. We first prove that the requirement on the RCP is that it should be highly correlated with the circuit delay. Next, we present two novel algorithms to automatically build the RCP. Our experimental results demonstrate that over a number of samples of manufactured circuits, the delay of the RCP captures the worst case delay of the manufactured circuit. The average prediction error of all circuits is shown to be below 2.8% for both approaches. For both our approach and the critical path replica method, it is essential to guard-band the prediction to ensure pessimism: our approach requires a guard band 30% smaller than for the critical path replica method.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUITS-Design Aids

General Terms

Algorithms, Design

Keywords

Representative Critical Path, Post-Silicon Optimization

1. INTRODUCTION

For feature sizes in the tens of nanometers, it is widely accepted that design tools must take into account parameter variations during manufacturing. These considerations are important during both circuit analysis and optimization, and are essential to ensure adequate manufacturing yield. Parameter variations can be classified into two categories: across-die variations and within-die variations. Across-die variations correspond to parameter fluctuations from one chip to another, while within-die variations are defined as the variations among different locations within a single die. Within-die variations of some parameters have been observed to be spatially correlated, i.e., the parameters of transistors or wires that are placed close to each other on a die are more likely to vary in a similar way than those of transistors or wires that are far away from each other. For example, among the process parameters for a transistor, the variations of channel length L and transistor width W are seen to have such spatial correlation structure, while parameter variations such as the dopant concentration N_A and the oxide thickness T_{ox} are generally considered not to be spatially correlated.

Process parameter variations have resulted in significant challenges to the conventional corner-based timing analysis paradigm, and statistical static timing analysis (SSTA) has been proposed as an alternative [1–6]. The idea of SSTA is that instead of computing the delay of the circuit as a specific number, a probability density function (PDF) of the circuit delay is determined. Designers may use the full distribution, or the 3σ point of the PDF, to estimate and optimize timing. Efficient statistical timing analysis tools have been developed based on parameterized block-based statistical timing analysis [1, 2], taking into consideration spatial and structural correlations of the parameter variations in the circuit to be analyzed. The computational efficiency of these methods is made practical through a preprocessing step, proposed in [1], which has shown that Gaussian-distributed correlated variations can be orthogonalized using principal component analysis (PCA).

The above-mentioned statistical timing analysis tools are useful for presilicon analysis over an entire population of die, and are intended to maximize the yield over the population. The post-silicon analysis and optimization problem is complementary to any such presilicon analysis. The diagnosis problem addresses the issue of estimating the performance of a manufactured die, or determining the critical path (or paths) on the manufactured die. While this information may be gathered using time-intensive delay testing schemes, there are many instances where a faster diagnosis is necessary, e.g., in post-silicon tuning methods.

In previous literature, the interaction between presilicon analysis and post-silicon measurements has been addressed in several ways. In [7], post-silicon measurements are used to learn a more accurate spatial correlation model to refine the SSTA framework. A path-based methodology is proposed in [8] to correlate post-silicon test data to presilicon timing analysis. In [9], a statistical gate sizing approach is presented to optimize the binning yield. The work is extended to simultaneously consider the presence of post-silicon-tunable clock tree and statistical gate sizing in [10]. Post-silicon debug methods and their interaction with circuit design are discussed in [11]. A joint design-time and post-silicon tuning procedure is described in [12].

In this paper, we focus on post-silicon tuning methods that require replicating the critical path of a circuit. Such techniques include adaptive body bias (ABB) or adaptive supply voltage (ASV) [13–15]. The approach that is used in [13–15] employs a replica of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISPD'09, March 29–April 1, 2009, San Diego, California, USA
Copyright 2009 ACM 978-1-60558-449-2/09/03 ...$5.00.

the critical path at nominal parameter values (we call this the nominal critical path), whose delay is rapidly measured and used to determine the optimal adaptation. However, this has obvious problems: first, it is likely that a large circuit will have more than a single critical path, and second, a nominal critical path may have different sensitivities to the parameters than other near-critical paths, and thus may not be representative. We quantitatively illustrate this problem in our experimental results. An alternative approach in [16] uses a number of on-chip ring oscillators to capture the parameter variations of the original circuit. However, this approach requires measurements for hundreds of ring oscillators for a circuit with reasonable size and does not provide an explicit critical path.

Another post-silicon optimization technique uses dynamic voltage scaling [17,18]. In [17], a delay synthesizer, composed of three delay elements, is used to synthesize a critical path as part of a dynamic voltage and frequency management system. However, the control signals of the synthesizer is chosen arbitrarily and therefore it is not able to adapt to a changing critical path as a result of process variations. In [18], the authors compensate this problem using a pre-characterized look up table (LUT) to store logic speed and interconnect speed inside different process bins. A logic and interconnect speed monitor is then used as an input to select through the LUT control signals to program a critical path. However, the authors use simplified circuitry for the speed monitor, consisting of only one logic dominated element and one interconnect dominated element, and assume that the results are generally applicable to all parts of the circuit. In the presence of significant within-die variations, this assumption becomes invalid. Moreover, the approach requires substantial memory components even for process bins of a very coarse resolution, and is not scalable to fine grids.

In this paper, we propose a new way of thinking about the problem. We automatically build an on-chip test structure that captures the effects of parameter variations on all critical paths, so that a measurement on this test structure provides us a reliable prediction of the actual delay of the circuit, with minimal error, for all manufactured die. The key idea is to synthesize a test structure whose delay can reliably predict the maximum delay of the circuit, under across-die as well as within-die variations. In doing so, we take advantage of the property of spatial correlation between parameter variations to build this structure and determine the physical locations of its elements.

The test structure that we create, which we refer to as the *representative critical path* (RCP), is typically different from the critical path at nominal values of the process parameters. In particular, a measurement on the RCP provides the worst-case delay of the whole circuit, while the nominal critical path is only valid under no parameter variations, or very small variations. Since the RCP is an on-chip test structure, it can easily be used within existing post-silicon tuning schemes, e.g., by replacing the nominal critical path in the schemes in [13–15]. While our method accurately captures any correlated variations, it suffers from one limitation that is common to any on-chip test structure: it cannot capture the effects of spatially uncorrelated variations, because by definition, there is no relationship between those parameter variations of a test structure and those in the rest of the circuit. To the best of our knowledge, this work is the first effort that synthesizes a critical path in the statistical sense. The physical size of the RCP is small enough that it is safe to assume that it can be incorporated into the circuit (using reserved space that may be left for buffer insertion, decap insertion, etc.) without significantly perturbing the layout.

The remainder of the paper is organized as follows. Section 2 introduces the background of the problem and formulates the problem mathematically. Next, Section 3 illustrates the detailed algorithms of our approach. Experimental results are provided in Section 4, and Section 5 concludes the paper.

2. PROBLEM FORMULATION

In this paper, we use the grid-based model from [1] to capture spatially correlated parameter variations. The chip is divided into a number of grids tailored for the size of the circuit. Variations of the same process parameter inside each grid are taken to be fully correlated, and the correlation is a decreasing function of distance: specifically, variations inside nearby grids show higher correlation than variations within grids that are far away. For different process parameters, it is assumed that there are no correlations.

Our overall approach can be summarized as follows. We have a circuit whose delay can be represented as a random variable, d_c. Using the method presented in this paper, we build the RCP whose delay can be represented by another random variable, d_p. After the circuit is manufactured, we measure the delay of the RCP, and find that it equals d_{pr}. In other words, d_{pr} corresponds to one sample of d_p for a particular set of parameter values. From this measured value of d_{pr}, we will infer the value, d_{cr}, of d_c for this sample, i.e., corresponding to this particular set of parameter values.

We assume that all parameter variations are Gaussian-distributed, and the delay of both the circuit and the critical path can be approximated by an affine function of those parameter variations. From previous work, e.g., [1], we know that we can get these functions by performing SSTA, and we can obtain both d_c and d_p as Gaussian-distributed PDFs.

Let $d_c \sim N(\mu_c, \sigma_c)$, $d_p \sim N(\mu_p, \sigma_p)$, and let the correlation coefficient of d_c and d_p be ρ. Then, from the basic theory of statistics, we know that the joint PDF of d_c and d_p is

$$f(d_c = d_{cr}, d_p = d_{pr}) = \frac{1}{2\pi\sigma_c\sigma_p\sqrt{1-\rho^2}} e^{C_1}$$

where
$C_1 = -\frac{1}{2(1-\rho^2)}\left(\frac{(d_{cr}-\mu_c)^2}{\sigma_c^2} + \frac{(d_{pr}-\mu_p)^2}{\sigma_p^2} - \frac{2\rho(d_{cr}-\mu_c)(d_{pr}-\mu_p)}{\sigma_c\sigma_p}\right)$.
The conditional PDF of $d_c = d_{cr}$ given the condition $d_p = d_{pr}$ is

$$f(d_c = d_{cr} | d_p = d_{pr}) = \frac{f(d_{cr}, d_{pr})}{f(d_{pr})} = \frac{1}{2\pi\sigma_c\sqrt{1-\rho^2}} e^{C_2}$$

where
$C_2 = -\frac{1}{2\sigma_c^2(1-\rho^2)}\left(d_{cr} - \left(\mu_c + \frac{\rho\sigma_c}{\sigma_p}(d_{pr}-\mu_p)\right)\right)^2$.
Therefore the conditional distribution of d_{cr} is a Gaussian with mean $\mu_c + \frac{\rho\sigma_c}{\sigma_p}(d_{pr}-\mu_p)$ and variance $\sigma_c^2(1-\rho^2)$.

The mean can be interpreted as the predicted value of the delay of the circuit, then the variance is the mean square error of infinite samples. From a least squares perspective, it is desirable to minimize the variance, so that the mean is an estimate of the circuit delay with the smallest mean square error. For the term representing the variance of the conditional distribution, σ_c is fixed because we have no control over the original circuit, and therefore, the variance of the conditional distribution is dependent only on ρ. Minimizing the variance is thus equivalent to maximizing ρ. In other words, this is a formal statement of the intuitive observation that our problem is to build a RCP whose delay has the maximum correlation coefficient with the delay of the whole circuit.

3. GENERATION OF THE CRITICAL PATH

3.1 Overview of the SSTA Framework

Block-based parameterized statistical timing analysis procedures propagate the PDF of the arrival time at the output of each gate during a topological traversal of the circuit, using a canonical form. This canonical form typically consists of a mean (i.e., the nominal value) and a set of normalized independent sources of variation (equivalent to Principal Components (PCs), which can be obtained by applying PCA to the covariance matrix of the spatially correlated process parameters), and a term for spatially uncorrelated sources of variation.

We use parameterized SSTA to obtain d_c as an affine function in the canonical form. We will show that this canonical form, in which the variables in the affine function consist of the m PCs and the independent parameter, makes the calculation of the correlation coefficient ρ defined in Section 2 much easier.

The canonical expression for d_c is shown below:

$$d_c = \mu_c + \sum_{i=1}^{m} a_i p_i = \mu_c + \mathbf{a}^T \mathbf{p} + R_c, \qquad (1)$$

where d_c, μ_c are defined in Section 2, and μ_c is the mean of d_c obtained from SSTA, and represents the nominal value of d_c. The random variable R_c is the independent term defined in [19] whose variance is recorded as SSTA is performed. The random variable p_i corresponds to the i^{th} PC, and is Gaussian distributed as $N(0,1)$; note that p_i and p_j for $i \neq j$ are uncorrelated by definition, due to the property of PCA. The parameter a_i is the first order coefficient of d_c with respect to p_i. We have stacked all a_i variables together to form the vector $\mathbf{a}$, and $\mathbf{p}$ is the vector that contains all p_i.

The values of these principal components for a given manufactured part are identical for the circuit and the RCP since they both lie on the same chip. A statistical timing analysis of this path yields another delay expression in canonical form:

$$d_p = \mu_p + \sum_{i=1}^{m} b_i p_i = \mu_p + \mathbf{b}^T \mathbf{p} + R_p \qquad (2)$$

where d_p, μ_p are defined in Section 2, and $p_i, b_i, \mathbf{b}, \mathbf{p}, R_p$ are all inherited from Equation (1). The correlation coefficient of d_c and d_p is easily computed as

$$\rho = \frac{\mathbf{a}^T \mathbf{b}}{\sigma_c \sigma_p} \qquad (3)$$

where $\sigma_c = \sqrt{\mathbf{a}^T\mathbf{a} + \sigma_{R_c}^2}$ and $\sigma_p = \sqrt{\mathbf{b}^T\mathbf{b} + \sigma_{R_p}^2}$. An important point to note is that ρ only depends on the coefficients of the PCs for both the circuit and the critical path and their independent terms.

As discussed in Section 2, the mean of the conditional distribution $f\left(d_c = d_{cr} | d_p = d_{pr}\right)$, which is used to estimate of the circuit delay, is:

$$\bar{\mu} = \mu_c + \frac{\rho\sigma_c}{\sigma_p}\left(d_{pr} - \mu_p\right) = \mu_c + \frac{\mathbf{a}^T\mathbf{b}}{\sigma_p}\left(d_{pr} - \mu_p\right). \qquad (4)$$

The variance which is also the mean square error of the circuit delay estimated using the above expression, is $\sigma_c^2\left(1-\rho^2\right)$. Our goal is to build a critical path with the largest possible ρ.

Our theory assumes that the effects of systematic variations can be ignored, and we will show, at the end of Section 4, that this is a reasonable assumption. However, it is also possible to extend the theory to handle systematic variations in parameters that can be controlled through design: for a fully characterized type of systematic variation, we can compensate for it by choosing a shifted nominal value for the parameter.

3.2 Two Approaches for Generating the Critical Path

In this work, we propose two methods for generating the RCP. The first is based on sizing gates on an arbitrarily chosen nominal critical path, while the second synthesizes the RCP from scratch using cells from the standard cell library.

3.2.1 Method I: Critical Path Generation Based on Nominal Critical Path Sizing

As described in Section 1, the nominal critical path falls short of our need to capture the worst case delay of the circuit over all reasonable parameter variations. However, it is intuitively true that variations along a critical path have some relationship to the variations in the circuit. Our first approach proceeds along this intuitive direction: it begins with a critical path of the circuit, and modifies it to meet the criteria described in Section 2, in order to ensure that it closely tracks the delay of the critical path in the manufactured circuit.

For an optimized circuit, it is very likely that there are multiple nominal critical paths with similar worst case delays at nominal parameter values. To make our approach as general as possible, we pick the one nominal critical path that has the largest worst case delay at nominal values, even if its delay is only larger than a few other paths by a small margin. If there are multiple such paths, we arbitrarily pick one of them. We show in Section 4 that even with this relaxed choice, after the optimizations presented in this section, our method can produce very good results.

Algorithm 1 Variation-aware critical path generation based on sizing.

```
1:  Perform deterministic STA on the original circuit and find the
    maximum delay path as the initial RCP. If there is more than
    one such path, arbitrarily pick any one.
2:  Perform SSTA on the original circuit to find the PC coefficients
    corresponding to the vector $\mathbf{a}$ and the variance of the indepen-
    dent term.
3:  Perform SSTA on the initial RCP to find its PC coefficients and
    the variance of its independent term. Calculate the correlation
    coefficient $\rho^0$ between the delay variables of the original circuit
    and the initial RCP.
4:  k = 1
5:  while (1) do
6:    for each gate $i$ on the critical path do
7:      Bump up its size by multiplying it by a factor $F$, keeping
        all other gate sizes unchanged from iteration $k-1$
8:      Compute $\rho_i^k$ as the correlation coefficient for this modi-
        fied RCP with the original circuit
9:    end for
10:   Choose $i$ such that $\rho_i^k$ is the largest, and set $\rho^k = \rho_i^k$
11:   if $\rho^k > 0$ then
12:     Set the RCP to be the RCP from iteration $k-1$, except
        that the size of gate $i$ is bumped up by factor $F$.
13:   else
14:     break
15:   end if
16: end while
```

An outline of the procedure is illustrated in Algorithm 1. We begin with a nominal critical path of the circuit, chosen as described above, and replicate it to achieve an initial version of the RCP. This is refined by iteratively sizing the gates on the path, using a greedy algorithm, in such a way that its correlation with the circuit delay is maximized.

The first step of the approach involves performing STA on the circuit to identify a nominal critical path, which is picked as the initial version of the RCP. Next, we perform SSTA on the circuit to obtain the PDF of the circuit delay, d_c, in canonical form. In other words, this analysis provides us with the coefficients of the PCs in the circuit delay expression. We repeat this procedure for the RCP, to obtain the coefficients of the PCs in the expression for the delay, d_p, of the RCP. Based on these two canonical forms, we can compute the correlation coefficient, ρ^0, between the two delay expressions.

The iterative procedure changes the size of gates on the current RCP, using a TILOS-like criterion. In the k^{th} iteration, we process each gate i on the RCP, and alter its size by multiplying its current size by a constant factor, while leaving all other gate sizes identical to iteration $k-1$. We perform SSTA on this modified RCP to obtain the new PCs corresponding to this change, and calculate the new correlation coefficient, ρ_i^k. Over all gates on the current RCP, we greedily choose to up-size the gate j whose perturbation that provides the maximum improvement in the correlation coefficient. We then update the RCP by perturbing the size of the j, and set the value of ρ^k to ρ_j^k. We repeat until there is no improvement in the correlation coefficient is possible, or if the sizes of gates in the RCP become too large.

We can save on the computation time by exploiting the fact that the RCP is a single path, and that SSTA on this path only involves sum operations and no max operations. When the size of a gate is changed, the delays of most gates on the critical path are left unchanged. We only perform SSTA on the few gates and wires that are directly affected by the perturbation, instead of performing SSTA on the entire path. However, we still have to walk through the whole path to find the gate with the maximum improvement. If the number of stages of a nominal critical path is bounded by s, and the sizing procedure takes K iterations, then the run time of Algorithm 1 is $O(Ks)$.

The final RCP is built on the chip, and after manufacturing, its delay is measured. Using Equation (4) in Section 3.1, we may then predict the delay of the circuit.

A significant advantage of this approach is that by choosing a nominal critical path as the starting point for the RCP, and refining the RCP iteratively to improve its correlation with the circuit delay, this approach is guaranteed to do no worse than one that uses the unmodified nominal critical path, e.g., in [13–15]. For a circuit that is dominated by a single critical path, this method is guaranteed to find that dominating path.

The primary drawback of this method is also related to the fact that the starting point for the RCP is a nominal critical path. This fixes the structure of the path and the types of gates that are located on it, and this limits the flexibility of the solution. Our current solution inherits its transformations in each iteration from the TILOS algorithm, and changes the size of the circuit. However, in principle the idea could also be used to consider changes, in each iteration, not only to the sizes but also to the functionality of the gates on the RCP by choosing elements from a standard cell library, so that the delay of the modified RCP (with appropriately excited side-inputs) shows improved correlations with the circuit delay. Another possible enhancement could be to select the nominal critical path with the highest initial correlation coefficient with the circuit delay, instead of choosing this path arbitrarily. These extensions may be considered in future work, but Section 4 shows that even without them, our approach still produces good results.

3.2.2 Method II: Critical Path Generation Using Standard Cells

The second approach that we explore in this work builds the RCP from scratch, using cells from the standard cell library that is used to build the circuit. The problem of forming a path that optimally connects these cells together to ensure high correlation with d_c can be formulated as an integer nonlinear programming problem, where the number of variables corresponds to the number of library cells, and the objective function is the correlation between the statistical delay distribution, d_p, of an RCP consisting of a set of these cells, and d_c.

The integer nonlinear programming formulation is listed below:

$$\begin{aligned} \text{maximize} \quad & \rho = \frac{\mathbf{a}^T\mathbf{b}(\mathbf{N}_s)}{\sqrt{\mathbf{a}^T\mathbf{a}+\sigma_{R_c}^2}\sqrt{\mathbf{b}(\mathbf{N}_s)^T\mathbf{b}(\mathbf{N}_s)+\sigma_{R_p}^2(\mathbf{N}_s)}} \quad (5) \\ \text{s.t.} \quad & \mathbf{N}_s \in \mathbf{Z}^n \\ & \mathbf{e}^T\mathbf{N}_s \le s \\ & \mathbf{b} = \Sigma_{i=1}^n N_{si}\mathbf{b}_i \\ & \sigma_{R_p}^2 = \Sigma_{i=1}^n N_{si}\sigma_{R_{pi}}^2 \end{aligned}$$

The objective function above is the correlation coefficient, ρ, between d_p and d_c, as defined by Equation (3). The variable n represents the number of possibilities for each stage of the RCP, and the vector $\mathbf{N}_s = [N_{s1}, N_{s2}, \cdots, N_{sn}]^T$, where N_{si} is the number of occurrences of i in the RCP.

The first constraint states the obvious fact that each element of $\mathbf{N}_s$ must be one of the allowable possibilities. In the second constraint, $\mathbf{e} = [1, 1, \cdots, 1]^T$, so that the constraint performs the function of placing an upper bound on the total number of stages in the RCP. For the purposes of this computation, $\mathbf{a}$ and $\sigma_{R_c}^2$ come from the canonical form of the circuit delay, d_c, and are constant. The values of $\mathbf{b}$ and $\sigma_{R_p}^2$ are functions of $\mathbf{N}_s$, where the mapping corresponds to performing SSTA on the RCP to find the vector of PC coefficients $\mathbf{b}$ and the variance of the independent term R_p in the canonical form. The terms $\mathbf{b}_i, 1 \le i \le n$ are the PC coefficients corresponding to each stage of the RCP, and R_{pi} correspond to their independent terms, so that $\mathbf{b}$ and $\sigma_{R_p}^2$ are related to $\mathbf{N}_s$ through the last two constraints.

Since (6) does not easily map on to any tractable problem that we are aware of, we propose an incremental greedy algorithm, described in Algorithm 2, which is simpler. While this algorithm is not provably optimal, it is practical in terms of its computational cost. We begin by recalling that our problem is to make the correlation coefficient between d_c and d_p as large as possible. The algorithm begins by performing SSTA on the original circuit to determine d_c.

Algorithm 2 Critical path generation using standard cells.

```
1: Initialize the RCP $P$ to be the initial load $INV$.
2: Perform SSTA on the original circuit to find $d_c$ in canonical
   form, and also compute the canonical form for the delay of
   each of the $p \times q$ choices for the current stage.
3: Calculate the load $L^{k-1}$ presented by the $(k-1)$-stage RCP
   computed so far.
4: With $L^{k-1}$ as the load, perform SSTA on the $p \times q$ choices for
   stage $k$.
5: Statistically add the canonical expressions for the delays of
   each of the $p \times q$ choices with the canonical form for the delay
   of the partial RCP computed so far, $P$. Calculate the correla-
   tion coefficient between the summed delays and the delay of
   the original circuit for each case.
6: Select the choice that produces the largest correlation coeffi-
   cient as stage $k$ in path $P$.
7: Go to Step 3.
```

During each iteration, the RCP is constructed stage by stage, where a *stage* is defined as a gate, together with the interconnects that it drives. If we have p types of standard gates, and q types of metal wires, then in each iteration we have $p \times q$ choices for the stage to be added. For an RCP with m stages, this corresponds to a search space of $(p \times q)^m$. Instead, our method greedily chooses one of the $p \times q$ choices at each stage that maximizes the correlation of the partial RCP constructed so far with d_c, thereby substantially reducing the computation involved.

The approach begins at the end of the critical path. We assume that it drives a measurement device such as a flip-flop, and the part of the device that the critical path drives is an inverter INV. Therefore, for the first iteration, this inverter is taken as the load, and it corresponds to a known load for the previous stage, which will be added in the next iteration.

In iteration k, we append each of the $p \times q$ choices to the partial RCP from iteration $k-1$, and perform SSTA for all of these choices to obtain the coefficients for the PCs, and the correlation with d_c, using Equation (3). The choice that produces the largest correlation coefficient is chosen to be added to the critical path. The load presented by this choice to the previous stage is then calculated, and the process is repeated. During the process of building the RCP, there may be cases where a wire on the RCP crosses the boundary between two correlation grids: if so, the current gate and the one it drives belong to two different grids, and the wire connecting them must be split into two parts to perform the SSTA.

A complimentary issue for this algorithm is related to determining the physical layout of each stage. We assume that the RCP moves monotonically: for example, the signal direction on all horizontal wires between stages must be the same, and the same is true of signal directions on all vertical wires. Because of symmetry of the PCA results, we only choose the starting points to be from the bottom grids of the die. For a given starting point, the routing would span to the right and upper part of the circuit. It should be noted that systematic variations would affect the sensitivities of the parameter values, causing PC coefficients of cells at symmetric locations not exactly symmetric. However, because systematic variations can be pre-characterized before statistical analysis by a change of nominal values at different locations, we show in Section 4 that a reasonable disturbance of the nominal values would not significantly affect the final results. The procedure continues until the number of stages in the RCP reaches a prespecified maximum, or when the monotonic path reaches the end of the layout.

If the number of stages of the RCP is bounded by s and the number of starting points we try is ω, the runtime of method II is $O(\omega pqs)$, because at each stage we have $p \times q$ choices. In comparison to Method I, if the bound of maximum number of stages for each method is comparable, then the comparison between K and $\omega \times p \times q$ determines which method has the longer asymptotic run time.

This approach has the advantage of not being tied to a specific critical path, and is likely to be particularly useful when the number of critical paths is large. However, for a circuit with one dominant critical path, this method may not be as successful as the first method, since it is not guided by that path.

4. EXPERIMENTAL RESULTS

We demonstrate the effectiveness of the approaches presented in this paper on the ISCAS89 benchmark suite. The netlists are first sized using our implementation of TILOS: this ensures that the circuits are realistic and are a reasonable number of critical paths. The circuits are placed using Capo [20] and global routing is then performed to route all of the nets in the circuits.

The variational model uses the hierarchical grid model in [21] to compute the covariance matrix for each spatially correlated parameter. Under this model, if the number of grids is G, and the number of spatially correlated parameters being considered is P, then the total number of PCs is no more than $(P \times G)$. The parameters that are considered as sources of variations include the effective channel length L, the transistor width W, the interconnect width W_{int}, the interconnect thickness T_{int} and the inter-layer dielectric H_{ILD}. The width W is the minimum width of every gate before the TILOS sizing. We use two layers of metal. Parameters on different layers of metal are considered to be independent. The parameters are Gaussian-distributed, and their mean and 3σ values are shown in Table 1. As in many previous works on variational analysis, we assume that for each parameter, half of the variational contribution is assumed to be from across-die variations and half from within-die variations. We use *MinnSSTA* [1] to perform SSTA, in order to obtain the PC coefficients for d_c. All programs are run on a Linux PC with a 2.0GHz CPU and 256MB memory.

Table 1: Parameters used in the experiments.

	L (nm)	W (nm)	W_{int} (nm)	T_{int} (nm)	H_{ILD} (nm)
μ	60.0	150.0	150.0	500.0	300.0
3σ	12.0	22.5	30.0	75.0	45.0

We first show the results of the algorithm that corresponds to Method I, described in Section 3.2.1, synthesizing the RCP by modifying a nominal critical path of the original circuit. The initial sizes of the gates are their sizes after timing optimization. We only show the results of the larger circuits, since these are more realistic, less likely to be dominated by a small number of critical paths, and are large enough to allow significant within-die variations. Of these, circuit s9234 is smaller than the others, and is divided into 16 spatial correlation grids, while all other circuits are divided into 256 grids.

In our implementation of Method I, we do not consider congestion issues. We assume both the critical path replica method and Method I can perfectly replicate the nominal critical path, including the interconnects, to give them a fair comparison. In practice, Method I can route the replicated nominal critical path in the same way as any of the prior critical path replica methods reported in previous literature.

We use a set of Monte Carlo simulations to evaluate the RCP. For each circuit being considered, we perform 10,000 Monte-Carlo simulations, where each sample corresponds to a manufactured die. For each sample, we compute the delay of the RCP, the delay of the original circuit, and the delay of the nominal critical path that may be used in a Critical Path Replica method, as in [13–15].

The delay of the RCP is then used to compute the circuit delay using Equation (4) in Section 3.1. This computed circuit delay, called the predicted delay, d_{predic}, is compared with the delay of the circuit, referred to as the true delay, d_{true}. The prediction error is defined as

$$\frac{|d_{true} - d_{predic}|}{d_{true}} \times 100\%. \tag{6}$$

For purposes of comparison, we also calculate the according prediction error for the Critical Path Replica method.

In order to maximize yield, we must add a *guard band* for the predicted delay values to ensure that the predictions are pessimistic. Therefore in this set results we also compare the guard band needed to make 99% of the delay predictions pessimistic for both Method I and the Critical Path Replica method, respectively.

Table 2: A comparison between Method I and the Critical Path Replica (CPR) Method.

Circuit	Average error		Maximum error		Guard band (ps)	
	Method I	CPR	Method I	CPR	Method I	CPR
s9234	1.59%	2.84%	10.51%	15.20%	28.5	44.1
s13207	0.59%	1.07%	6.30%	7.33%	20.3	28.6
s15850	1.16%	2.13%	8.99%	11.52%	39.0	56.9
s35932	2.35%	5.77%	13.72%	20.83%	33.7	59.1
s38584	1.98%	3.26%	14.70%	17.66%	48.9	74.2
s38417	2.80%	5.24%	15.80%	21.32%	53.2	84.1

Table 3: Conditional standard deviation, number of stages for RCP, and CPU time of Method I.

Circuit	Avg $\frac{\sigma_{cond}}{\mu_{cond}}$	Max $\frac{\sigma_{cond}}{\mu_{cond}}$	No. stages	CPU time
s9234	2.35%	2.94%	67	24.17s
s13207	1.10%	1.47%	71	208.77s
s15850	1.43%	1.86%	96	554.33s
s35932	2.51%	3.14%	36	415.28s
s38584	2.35%	2.94%	66	158.16s
s38417	3.13%	3.90%	41	113.53s

The results of the comparisons are presented in Table 2, where the rows are listed in increasing order of the size of the benchmark circuit. For Method I as well as the Critical Path Replica (CPR) Method, we show the average error and maximum error over all samples of the Monte-Carlo simulation. All of the average errors of our approach are below 3% and both the average errors and maximum errors are significant improvements compared to the Critical Path Replica method. The guard bands needed by the two methods are listed in the last two columns. The guard band for Method I for each circuit is observed to be much smaller than the Critical Path Replica method. The advantage of Method I becomes particularly noticeable for the larger circuits.

The conditional variance derived in Section 2 defines the confidence of our estimate. Therefore we show the conditional standard deviation σ_{cond} as a percentage of the conditional mean μ_{cond} in Table 3. Because μ_{cond} is different for each sample, we list the average $\frac{\sigma_{cond}}{\mu_{cond}}$ and the maximum $\frac{\sigma_{cond}}{\mu_{cond}}$ over all samples for each circuit. In order to provide more information about the RCP we generate, we also show the number of stages for each RCP in the table. In this case, the number of stages for each RCP is the same as the nominal critical path for that circuit. The last column of the table shows the CPU time required by Method I for these benchmarks.

The run time of Method I ranges from a few seconds to around 9 minutes. The conditional standard deviation is typically below 3% of the conditional mean on average.

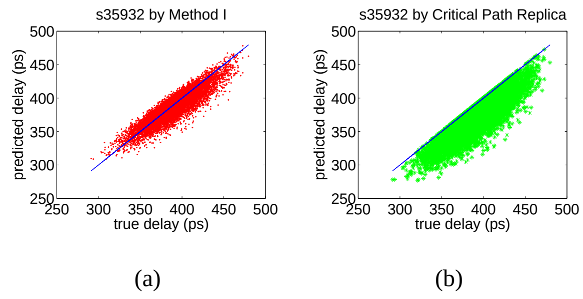


Figure 1: The scatter plot: (a) real circuit delay vs. predicted circuit delay by Method I and (b) real circuit delay vs. predicted circuit delay using the Critical Path Replica method.

To visually indicate the performance of Method I, we draw scatter plots of the results for circuit s35932 in Figure 1(a) for Method I, and in Figure 1(b) for the Critical Path Replica. The horizontal axis of both figures is the delay of the original circuit for a sample of the Monte-Carlo simulation. The vertical axis of Figure 1(a) is the delay predicted by our method, while the vertical axis of Figure 1(b) is the delay of the nominal critical path, used by the Critical Path Replica method. The ideal result is represented by the $x = y$ axis, shown using a solid line. It is easily seen that for the critical path replica method, the delay of the Critical Path Replica is either equal to the true delay (when it is indeed the critical path of the manufactured circuit) or smaller (when another path becomes more critical, under manufacturing variations). On the other hand, for Method I, all points cluster very close to the $x = y$ line, an indicator that the method produces accurate results. The delay predicted by our approach can be larger or smaller than the circuit delay, but the errors are small. Note that neither Method I nor the Critical Path Replica Method is guaranteed to be pessimistic, but such a consideration can be enforced by the addition of a guard band that corresponds to the largest error. Clearly, Method I can be seen to have the advantage of the smaller guard band in these experiments.

Our second set of experiments implement the algorithm corresponding to Method II, presented in Section 3.2.2. The maximum number of stages we allow for the critical path we create for each circuit is 50, comparable to most nominal critical paths for the circuits in our benchmark suite. We use 7 standard cells at each stage, and 2 metal layers. Therefore we have 14 choices for each stage. As in Method I, we did not consider congestion issues here and assume that the critical path replica method can perfectly replicate the nominal critical path. In practice, Method II can easily handle congestion issues by assigning a penalty to congested areas when selecting wire directions. The setup of the Monte-Carlo simulations are similar to the first set of experiments. The corresponding errors and guard bands are shown in Table 4. Since this Monte Carlo simulation is conducted separately from that in Table 2, there are minor differences in the error for the Critical Path Replica, even though both tables use the same Critical Path Replica as a basis for comparison. The average and maximum $\frac{\sigma_{cond}}{\mu_{cond}}$, the number of stages for each RCP, as well as the run times are shown in Table 5. The advantage of Method II, again, increases with the size of the circuit.

Table 4: A comparison between Method II and the Critical Path Replica (CPR) Method.

Circuit	Average error		Maximum error		Guard band (ps)	
	Method II	CPR	Method II	CPR	Method II	CPR
s9234	1.98%	2.84%	10.57%	15.15%	31.4	44.0
s13207	1.51%	1.06%	8.51%	7.22%	35.3	26.5
s15850	1.73%	2.14%	9.22%	10.97%	45.4	56.9
s35932	2.27%	5.80%	13.91%	21.34%	32.3	59.9
s38584	2.11%	3.29%	10.89%	17.12%	43.0	72.1
s38417	2.28%	5.27%	12.01%	22.88%	42.4	84.2

Table 5: Conditional standard deviation, number of stages for RCP, and CPU time of Method II.

Circuit	Avg $\frac{\sigma_{cond}}{\mu_{cond}}$	Max $\frac{\sigma_{cond}}{\mu_{cond}}$	No. stages	CPU time
s9234	2.18%	2.79%	49	0.1s
s13207	1.75%	2.31%	30	15.7s
s15850	1.88%	2.45%	50	15.1s
s35932	2.19%	2.81%	50	16.7s
s38584	2.14%	2.73%	50	18.6s
s38417	2.13%	2.77%	50	15.5s

It is observed that for almost all cases, the average and maximum errors for Method II are better than those for the Critical Path Replica method. The exception to this is circuit s13207, which is dominated by a small number of critical paths, even after sizing using TILOS. We illustrate this using the path delay histogram in Figure 2(a), which aggregates the delays of paths in the sized circuit into bins, and shows the number of paths that fall into each bin.

In this case, it is easily seen that the number of near-critical paths is small. In contrast, Figure 2(b) shows the same kind of histogram for circuit s9234, which is more typical over the other benchmarks: in this case it is seen that a much larger number of paths is near-critical, and likely to become critical in the manufactured circuit, due to the presence of variations.

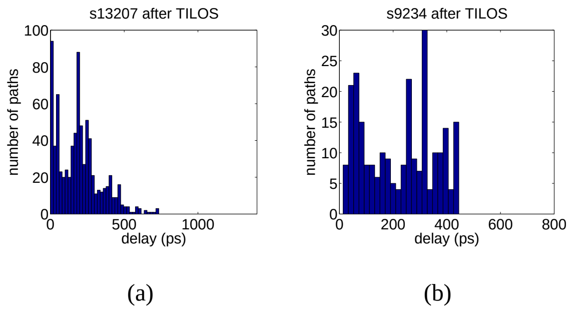


Figure 2: Histograms of path delays of (a) s13207 and (b) s9234 after TILOS optimization.

Under the scenario where the number of near-critical paths is small, it is not surprising that Method II does not perform as well as a critical path replica. First, as pointed out in Section 3.2.2, Method II does not take advantage of any information about the structure of the original circuit, and is handicapped in such a case. Moreover, the unsized circuit s13207 was strongly dominated by a single critical path before TILOS sizing; after sizing, the optimized near-critical paths are relatively insensitive to parameter variations, meaning even if one of these becomes more critical than the nominal critical path on a manufactured die, it is likely to have more or less the same delay.

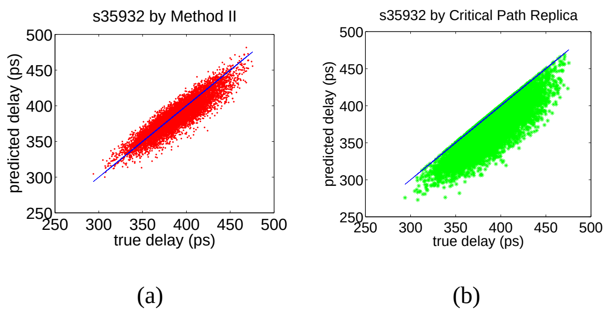


Figure 3: The scatter plot: (a) real circuit delay vs. predicted delay by Method II and (b) real circuit delay vs. predicted delay using the Critical Path Replica method.

We also show scatter plots for both our approach and critical path replica in this case, in Figure 3(a) and Figure 3(b), respectively. The figures are very similar in nature to those for the first approach, and similar conclusions can be drawn. In comparing Methods I and II by examining the numbers in Tables 2 and 4, it appears that there is no clear winner, though Method II seems to show an advantage for the largest circuits, s35932 and s38417. With our limited number of choices for each stage of the RCP, referring to discussions about run time in Section 3.2.2, it is not surprising that Method II is faster in terms of CPU time, as is shown in Table 5. The algorithm finishes within a few seconds for all of the benchmark circuits.

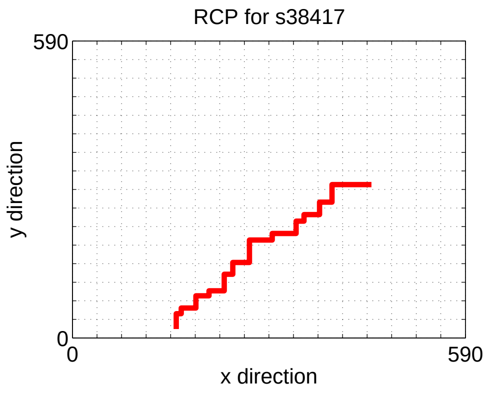


Figure 4: The RCP created by Method II for circuit s13207.

Next we show the location of the critical path we build for circuit s38417 using Method II on the chip in Figure 4. The figure shows the die for the circuit. The size of the die is determined by our placement and routing procedure, and the dashed lines indicate the spatial correlation grids. The solid bold lines are the wires of the critical path. The figure shows that the critical path grows in a monotonic direction and it starts from one of the grids at the bottom of the chip, both due to the layout heuristics discussed in Section 3.2.2.

In order to gain more insight into the trend of improvement of the correlation coefficients, Figure 5 shows the correlation coefficient of Method II after each stage is added for one starting point. The result for Method I is similar.

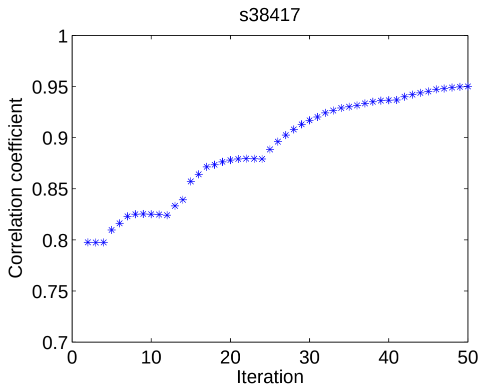
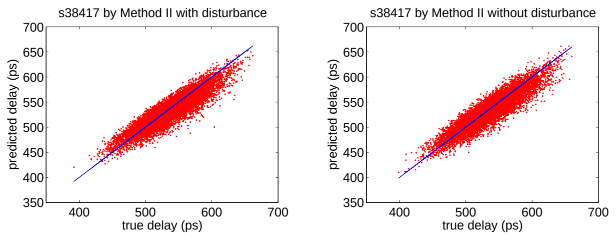


Figure 5: Trend of correlation coefficient after each iteration.

Finally, we experimentally demonstrate that our assumption of neglecting systematic variations is reasonable. We demonstrate this on Method II, and show that a reasonable change in the nominal parameter values of the RCP cells due to systematic variations would not affect the final results by much. This justifies our heuristic to only choose the starting point of the RCP at the bottom of the die.

The experiment proceeds as follows: after the RCP is built, we disturb the nominal values of all parameters associated with the RCP by 20%, while leaving those of the original circuit unperturbed. This models the effect of systematic variations, where the RCP parameters differ from those of the original circuit. We show the final results of the scatter plots for circuit s38417, with and without disturbance, in Figures 6(a) and 6(b), respectively. It is shown that the plots are almost identical, and the average error is 2.26% with disturbance as compared to 2.28% for the normal case.

The intuition for this can be understood as follows. The corre-



(a) (b)

Figure 6: Scatter plots of s38417 with and without nominal value disturbance for the RCP, to model systematic variations.

lation between the original circuit and the RCP depends on the coefficients of the PCs in the canonical expression. The coefficients depend on the sensitivities of the delay to variations, and not on their nominal values. Although the delay is perturbed by 20%, the corresponding change in the delay sensitivity is much lower, and this leads to the small change in the accuracy of the results.

5. CONCLUSION

In this paper, we have presented two novel techniques to automatically generate a critical path for the circuit to capture all of the parameter variations. Experimental results have shown that our methods produce good results.

6. REFERENCES

[1] H. Chang and S. S. Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations using a Single PERT-Like Traversal," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pp. 621–625, Nov. 2003.

[2] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-Order Incremental Block-Based Statistical Timing Analysis," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 331–336, June 2004.

[3] Y. Zhan, A. J. Strojwas, X. Li, L. Pileggi, D. Newmark, and M. Sharma, "Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 77–82, June 2005.

[4] J. Le, X. Li, and L. Pileggi, "STAC: Statistical Timing Analysis with Correlation," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 343–348, June 2004.

[5] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula, "Statistical Timing Analysis Using Bounds and Selective Enumeration," in *Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp. 29–36, Dec. 2002.

[6] A. Devgan and C. Kashyap, "Block-based Static Timing Analysis with Uncertainty," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pp. 607–614, Nov. 2003.

[7] B. Lee, L. Wang, and M. S. Abadir, "Refined Statistical Static Timing Analysis Through Learning Spatial Delay Correlations," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 149–154, July 2006.

[8] L. Wang, P. Bastani, and M. S. Abadir, "Design-Silicon Timing Correlation–A Data Mining Perspective," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 385–389, June 2007.

[9] A. Davoodi and A. Srivastava, "Variability Driven Gate Sizing for Binning Yield Optimization," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 956–964, July 2006.

[10] V. Khandelwal and A. Srivastava, "Variability-Driven Formulation for Simultaneous Gate Sizing and Postsilicon Tunability Allocation," in *Proceedings of the International Symposium on Physical Design*, pp. 11–18, Mar. 2007.

[11] M. Abranmovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A Reconfigurable Design-for-Debug Infrastructure for SoCs," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 7–12, July 2006.

[12] M. Mani, A. Singh, and M. Orshansky, "Joint Design-Time and Post-Silicon Minimization of Parametric Yield Loss using Adjustable Robust Optimization," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pp. 19–26, Nov. 2006.

[13] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1396–1402, Nov. 2002.

[14] J. W. Tschanz, S. Narendra, R. Nair, and V. De, "Effectiveness of Adaptive Supply Voltage and Body Bias for Reducing the Impact of Parameter Variations in Low Power and High Performance Microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 826–829, May 2003.

[15] J. W. Tschanz, S. Narendra, A. Keshavarzi, and V. De, "Adaptive Circuit Techniques to Minimize Variation Impacts on Microprocessor Performance and Power," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 23–26, May 2005.

[16] Q. Liu and S. S. Sapatnekar, "Confidence Scalable Post-Silicon Statistical Delay Prediction under Process Variations," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 492–502, June 2007.

[17] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura, "Dynamic Voltage and Frequency Management for a Low-Power Embedded Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 28–35, Nov. 2005.

[18] M. Elgebaly and M. Sachdev, "Variation-Aware Adaptvie Voltage Scaling System," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, pp. 560–571, Nov. 2007.

[19] H. Chang and S. S. Sapatnekar, "Statistical Timing Analysis Under Spatial Correlations," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 1467–1482, Sept. 2005.

[20] A. Caldwell, A. B. Kahng, and I. Markov, "Capo: a large-scale fixed-die placer," *available at http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement*.

[21] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda, "Path-Based Statistical Timing Analysis Considering Inter- and Intra-die Correlations," in *Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp. 16–21, Dec. 2002.