DDRO: A Novel Performance Monitoring Methodology Based on Design-Dependent Ring Oscillators

Tuck-Boon Chan[†], Puneet Gupta[§], Andrew B. Kahng[‡], Liangzhen Lai[*]
UC San Diego ECE[†‡] and CSE[‡] Departments, La Jolla, CA 92093
UC Los Angeles EE Department[§*], Los Angeles, CA 90095
E-mails: tbchan@ucsd.edu[†], puneet@ee.ucla.edu[§], abk@cs.ucsd.edu[‡], liangzhen@ucla.edu[*]

Abstract—As CMOS technology scales, circuit performance becomes more sensitive to manufacturing and environmental variations. Hence, there is a need to measure or monitor circuit performance during manufacturing and at runtime. Since each circuit may have different sensitivities to process variations, previous works have focused on synthesis of circuit performance monitors that are specific to a given design. In this work, we study the potential benefit of having multiple design-dependent monitors. We develop a systematic approach to the synthesis of multiple design-dependent monitors, as well as a corresponding delay estimation method.

Our approach synthesizes *design-dependent ring oscillators* (DDROs) using standard library gates. This has the advantage of quick design turnaround time and reduced schedule impact, because the DDRO implementation can leverage automation in conventional implementation flows. Our delay estimation method seeks to minimize the number of parameters as well as computing resources (i.e., to limit information storage and exchange) used in delay estimation based on monitoring results. Experiments show that our delay estimation method using multiple DDROs reduces overestimation (timing margin) by up to 25% compared to use of a single DDRO.

Keywords— circuit performance monitoring, design for manufacturing, ring oscillators.

I. Introduction

Circuit performance variability continues to increase due to process variability, wide operating ranges, and other factors. Performance variability can often be compensated if accurate circuit performance estimation is available. For example, (1) circuit performance can be estimated early in the manufacturing flow for process tuning, or (2) circuits with adaptive mechanisms can optimize the tradeoff between energy and performance based on feedback from runtime circuit performance monitors. In this paper, we define circuit *performance monitoring* as a process which estimates the worst-case delay of a circuit, based on measurements obtained from on-chip monitors.

Previous works on VLSI circuit performance monitoring can be classified according to the taxonomy shown in Figure 1. Generic monitors range from simple inverter-based ring oscillators (ROs) to more sophisticated process-specific ROs (PSROs) [2] and alternative monitoring structures such as phase-locked loops (PLLs) [11]. However, such generic monitors are inadequate to capture design characteristics such as mix of device types, which cause differing responses to process variations. As a result, delay estimation using generic monitors is less accurate and hence incurs larger margins.

Design of monitoring structures that are correlated to circuit performance (design-dependent monitors) has been addressed in several ways. Liu and Sapatnekar in [13] propose a method to synthesize a single representative critical path (RCP) for post-silicon delay prediction. The RCP is designed such that it is highly correlated to all critical paths for some expected process variations. This approach uses only a single RCP to estimate the worst-case delay of multiple critical paths. Since the critical paths may have different sensitivities to process variations, using multiple RCPs can potentially improves delay estimation accuracy. The tunable replica circuit (TRC) method in [9] can synthesize different delay paths to more flexibly mimic circuit performance, but has higher design overhead compared to RO approaches. TRC also requires costly calibrations to obtain configurations that correspond to different operating conditions.

By coupling process parameters extracted from parametric monitors with a design-specific delay model, more accurate and design-dependent delay estimation can be obtained from generic test structures [4] [15] [6]. Such an approach is flexible because an arbitrary delay model can be used and calibrated post-manufacturing. Meanwhile, parametric monitors can be designed such that they are highly sensitive to the targeted process variation. However, this approach requires many calibrations and resources for storage and computation of parameters. Another class of design-dependent monitors [3] [10] [14] [16] [19] [20] estimates circuit performance by tracking delays of critical paths. Although these monitors show good estimation accuracies, having a monitor per path incurs high area overhead as well as longer design turnaround time.

In this paper, we propose a systematic methodology to synthesize multiple design-dependent ROs (DDROs) for circuit performance monitoring. A crucial and enabling observation is that critical path *delay sensitivities* form natural clusters (see Figure 4). Therefore, we can capture the design-specific delay sensitivities by synthesizing a monitor to match

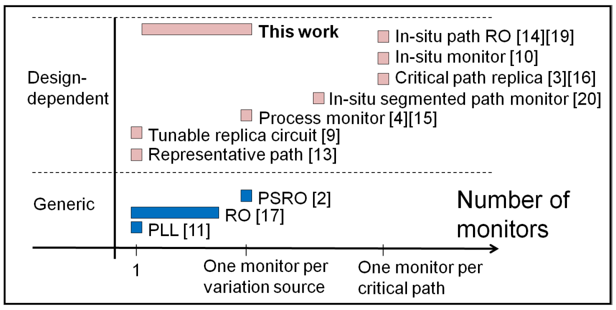


Fig. 1. Taxonomy of performance monitoring methods.

the delay sensitivities of each cluster. This approach has a lower implementation overhead compared to tracking each critical path because the number of clusters is much smaller than the number of critical paths.

Our DDRO approach offers several potential benefits compared to previous works. First, DDROs are more accurate compared to conventional ROs because they are synthesized to match the delay sensitivities of critical paths. Second, DDROs are more accurate compared to a single representative critical path because multiple DDROs are used to account for the differences between critical paths. Third, DDROs are less intrusive compared to in-situ monitoring methods. Fourth, DDROs can be used during early manufacturing stages since the DDRO routing can be limited to local metal layers. Fifth, the total number of ROs (silicon area) is greatly reduced due to the clustering of critical paths. Only a few DDROs are required to provide accurate delay estimation. Finally, DDROs can be used for both early process tuning and real-time performance monitoring. Switching the monitoring purpose is simply a matter of redefining target variation sources (manufacturing or real-time variations) with minimal design modifications.

Our experimental results below show that use of multiple DDROs can reduce delay overestimation by 15% - 25% compared to use of only one DDRO. Additional results show that the mean delay overestimation of our delay estimation method has negligible difference compared to a reference method, but the number of parameters used by our estimation method is significantly reduced compared to the reference case. Our contributions are summarized as follows.

- We propose a systematic methodology to design *multiple* DDROs for chip frequency estimation. Experiments show that using from 3 to 7 DDROs can achieve similar performance as "perfect" replica-based monitors.
- We propose a method to estimate chip delay and minimize guardband margin by using multiple DDRO measurements, within practical limits on information exchange between design and manufacturing.

In the following, Section II gives an overview of our methodology. We present two delay estimation methods in Section III. In Section IV, we discuss implementation details of DDRO synthesis. In Section V, we present experimental data to illustrate the use of DDROs to estimate circuit timing. Finally, we summarize our conclusions and future work in Section VI. All notations in this paper are defined in Table I.

II. Overview of DDRO Approach

An overview of our monitoring strategy is shown in Figure 2. First, we extract critical paths of a design and characterize their delay sensitivities to variation sources. Delay sensitivity of path i ($\mathbf{V}_i^{path}$) is obtained using finite differences, i.e.,

$$\mathbf{V}_i^{path} = \left[\frac{d_{i1}^{path} - d_i^{nom}}{d_i^{nom}} \dots \frac{d_{iQ}^{path} - d_i^{nom}}{d_i^{nom}}\right] \quad (1)$$

where d_{ij}^{path} is the delay of path i when the j^{th} variation source is biased by +1σ from its nominal value, and d_i^{nom} is the nominal delay of path i. Second, we cluster the critical paths based on their path sensitivities, and synthesize DDROs to match delay sensitivity of the clusters. By matching DDRO and cluster delay sensitivities, we ensure that the synthesized DDROs have good correlation with the critical paths. Since we use only standard cells (gates) to synthesize the DDROs, the design and placement of DDROs can be easily integrated with conventional implementation flows. Based on DDRO frequencies, we can estimate chip delay during manufacturing or runtime.

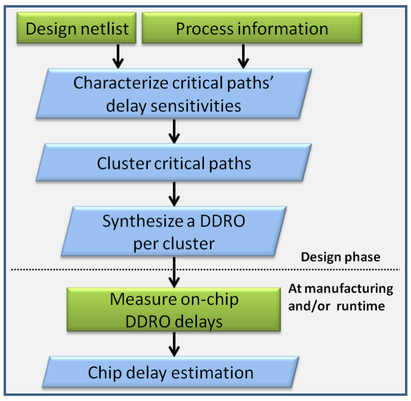
TABLE I
Glossary of Terminology

Term	Description
w_i	Probability that critical path i fails to meet circuit timing
Z	User-defined confidence, $0 \leq Z < 1$
s_h	Total number of gate modules h in a DDRO
z_h	Indicates whether gate module j is inverting
h	Index for gate instance or gate module
i	Index for path
j	Index for variation source
k	Index for DDRO
x	Index for cluster
N	Total number of critical paths
H	Total number of gate instances
Q	Total number of variation sources
M	Total number of DDROs
Y	Total number of gate module types
$d_k^{nom\_ro}$	Nominal delay of DDRO k
$d_x^{nom\_clust}$	Nominal delay of a cluster x
$d_i^{nom\_path}$	Nominal delay of path i
$d_h^{nom\_gate}$	Nominal delay of gate h
d_k^{ro}	Delay of DDRO k
d_x^{clust}	Delay of a cluster x
d_i^{path}	Delay of path i
d^{max}	Maximum delay of a chip
d'^{max}	Estimated maximum delay of a chip
d_{ij}^{path}	Delay of path i when variation source j is = +1σ
$\mathbf{V}_k^{ro}$	Delay sensitivity of DDRO k to all Q variation sources
$\mathbf{V}_x^{max}$	Delay sensitivity of cluster x to all Q variation sources
$\mathbf{V}_i^{path}$	Delay sensitivity of path i to all Q variation sources
$\mathbf{V}_x^{res\_clust}$	Residue of delay sensitivity in a cluster x
$\mathbf{V}_i^{res\_path}$	Residue of delay sensitivity of path i
$\mathbf{V}_h^{gate}$	Delay sensitivity of gate module h to all Q variation sources
b_{ik}	Constant coefficient
lb	Lower bound of b_{ik}
ub	Upper bound of b_{ik}
a_{xk}	Constant coefficient
$\mathbf{R}$	Correlation matrix for local path delay variation
$\mathbf{G}$	Global variation vector with Q variation sources
l_i^{path}	Local variation of path i
u_i	Uncertainty of delay estimation for path i
r_x	Local delay variation of cluster x
$\mathcal{N}(\cdot,\cdot)$	Gaussian random variable
$\sigma(\cdot)$	Standard deviation function
$\mu(\cdot)$	Expectation (mean) function
$erf(\cdot)$	Error function of Gaussian distribution
$P(\cdot)$	Accumulative probability function

A circuit performance monitor typically feeds back the estimated delay with some margin to reduce the probability of reporting an underestimated delay value. However, the margin should be minimized to avoid significant performance loss due to a pessimistic delay estimation. Thus, we define the goal of circuit performance monitoring as:

$$\begin{aligned} &\text{minimize: } \mu(d'^{max} - d^{max}) \\ &\text{subject to: } P(d'^{max} \geq d^{max}) > Z \end{aligned} \quad (2)$$

Here $d^{max} = \max_{i=1}^{N}\{d_i^{path}\}$, where d^{max} is actual chip delay, d'^{max} is the estimated chip delay, d_i^{path} is the delay of the i^{th} critical path, $\mu(d'^{max} - d^{max})$ is the expectation of delay



Design netlist
Process information
Characterize critical paths' delay sensitivities
Cluster critical paths
Synthesize a DDRO per cluster
Design phase
At manufacturing and/or runtime
Measure on-chip DDRO delays
Chip delay estimation

Fig. 2. Overview of DDRO design methodology.

overestimation, $P(d'^{max} \geq d^{max})$ is the probability that d'^{max} is larger than d^{max}, N is the total number of critical paths of a chip, and Z is a user-specified confidence.

III. Proposed Method

A. Delay and Variation Model

In this work, we use the variation model in [7], whereby lot-to-lot, wafer-to-wafer, and die-to-die process variations are lumped and modeled as global variation of a chip. The global variation also includes supply voltage and temperature fluctuations. Within-die gate delay mismatch is modeled as uncorrelated Gaussian random variables. Spatial correlation is ignored in the current work as it is small for most chips [7]. When the effect of spatial correlation is significant, DDROs can be distributed within a die as in [17] to improve correlations between DDROs and critical paths. The critical path delay is represented as a linear function of the variation sources, i.e.,

$$\begin{aligned} d_i^{path} &= d_i^{nom\_path}(1 + \mathbf{V}_i^{path} \cdot \mathbf{G} + l_i^{path}) \\ \begin{bmatrix} l_1^{path} \\ \vdots \\ l_N^{path} \end{bmatrix} &= \mathbf{R} \begin{bmatrix} \mathcal{N}(0,1) \\ \vdots \\ \mathcal{N}(0,1) \end{bmatrix} \end{aligned} \tag{3}$$

where $\mathbf{G}$ is a $[Q \times 1]$ vector that represents the global variation of Q variation sources, l_i^{path} is local delay variation of the i^{th} path, $\mathbf{R}$ is the correlation matrix for local delay variation, and $\mathcal{N}(0,1)$ are independent Gaussian random variables.

To verify the accuracy of our delay model, we first simulate a critical path using HSPICE [24] with a set of random global variations (100 trials)[1]. Then, we compare the simulated path delays with the ones calculated from the linear delay model in (3). Figure 3 shows that path delays obtained from the linear model correlate very well with those from HSPICE simulation.

For DDROs, we use the same delay model as in (3). Since each RO has many identical gates, uncorrelated local variation is insignificant due to averaging of uncorrelated delay deviation. Therefore, we do not model local variation in the DDROs, i.e.,

$$d_k^{ro} = d_k^{nom\_ro}(1 + \mathbf{V}_k^{ro} \cdot \mathbf{G}) \tag{4}$$

where d^{nom_ro} is the nominal delay of the DDRO (obtained from simulation) and $\mathbf{V}_k^{ro}$ is a $[1 \times Q]$ vector that represents delay sensitivity of the k^{th} DDRO to global process variations.

[1]Variation sources are listed in Table II.

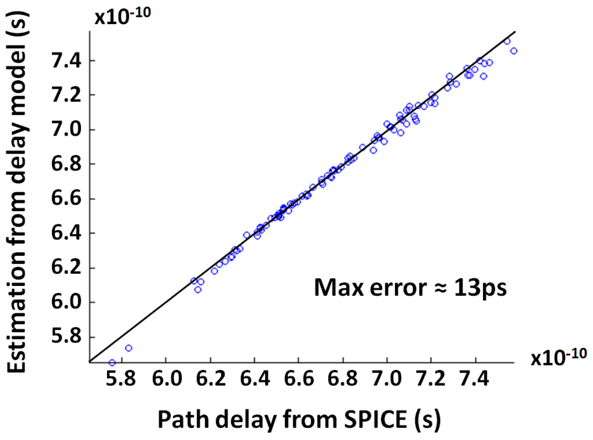


Fig. 3. Rank correlation between delays obtained from HSPICE simulation and linear delay model.

B. A Reference Approach

A straightforward delay estimation method is to extract global variation using multiple process variation-specific monitors and calculate chip delay based on the linear delay model in (3). In other words, monitoring methods in [4] [15] and [6] can be combined and extended for delay estimation. However, we use this approach only as a reference because it requires a large amount of memory to store parameters, as well as long computation time.

Given M DDROs, we can represent $\mathbf{V}_i^{path}$ as a linear combination of $\mathbf{V}_k^{ro}$ $(k = 1, ..., M)$ to utilize measurements from the DDROs:

$$\mathbf{V}_i^{path} = \sum_{k=1}^{M} b_{ik} \cdot \mathbf{V}_k^{ro} + \mathbf{V}_i^{res\_path} \tag{5}$$

where b_{ik} is a $[1 \times M]$ matrix containing constant coefficients and $\mathbf{V}_i^{res\_path}$ is a $[1 \times Q]$ matrix that represents the residue. Substituting $\mathbf{V}_i^{path}$ in (3) as a linear combination of $\mathbf{V}^{ro}$, we obtain

$$\begin{aligned} d_i^{path} &= d_i^{nom\_path}(1 + \overbrace{\sum_{k=1}^{M}(b_{ik}\mathbf{V}_k^{ro} \cdot \mathbf{G})}^{\text{measurable}} + \underbrace{u_i}_{\text{uncertainty}}) \\ \text{where } u_i &= l_i^{path} + \mathbf{V}_i^{res\_path} \cdot \mathbf{G} \end{aligned} \tag{6}$$

Equation (6) shows that d_i^{path} consists of a measurable term and an uncertainty term. While the value of the *measurable* term can be determined from the delays of DDROs, the value of the *uncertainty* term cannot be measured directly. Since a larger u_i leads to a larger $d'^{max} - d^{max}$ in (2), we should choose the value of b_{ik} to minimize u_i.

Assuming that $\mathbf{G}$ is multivariate Gaussian, we can calculate the distribution of d^{max} using the method in [18] based on (6) and (3). I.e., we repeatedly approximate the maximum of two path delays as a Gaussian distribution by matching the first two moments. After that, we can approximate the maximum delay

d_{max} as a Gaussian distribution[2].

$$\begin{aligned} d^{max} &= \max_{i=1}^{N}\{\mathcal{N}(\mu(d_i^{path}), \sigma(d_i^{path})), \mathbf{R}\} \\ &\approx \mathcal{N}(\mu(d^{max}), \sigma(d^{max})) \end{aligned} \tag{7}$$

where $\mu(d_i^{path})$ and $\sigma(d_i^{path})$ are the mean and standard deviation of the i^{th} path delay. Given $\mu(d^{max})$ and $\sigma(d^{max})$, d'^{max} can be readily obtained using the *erf* function for Gaussian distribution:

$$erf(\frac{d'^{max} - \mu(d^{max})}{\sigma(d^{max})}) > Z. \tag{8}$$

C. Clustering

The next step is to minimize delay margin and find $\mathbf{V}_k^{ro}$. Equations (7) and (8) show that the value of d'^{max} is mainly determined by the $\mathbf{V}^{res\_path}$, i.e., a larger $\mathbf{V}^{res\_path}$ will increase the magnitude of $\sigma(d_i^{path})$, which leads to a larger d'^{max}. Therefore, it is desirable to select $\mathbf{V}^{ro}$ that minimizes $\mathbf{V}^{res\_path}$. We find $\mathbf{V}_k^{ro}$ by clustering critical paths with similar $\mathbf{V}^{path}$ into the same group, then assigning the centroid of the cluster as $\mathbf{V}_k^{ro}$. To cluster the paths, we use the *kmeans++* algorithm in [1] and choose the best clustering solution in 100 random starts. The objective function of the clustering is defined as

$$\text{minimize} \sum_{i=1}^{N}(w_i \times |\mathbf{V}_i^{path} - \mathbf{V}_k^{ro}|) \tag{9}$$

where the summation is taken over paths i in cluster k, and w_i is the probability of a critical path delay exceeding the clock period of the design. The weight factor w_i is added so that we can impose a higher penalty for having mismatched delay sensitivities on a path with higher probability to fail (less timing slack). Based on the delay model in (3) and the distribution of variation sources, we can calculate the delay distribution of path i and extract w_i. Since the upper bound for $\mathbf{V}_i^{res\_path}$ is defined by $\mathbf{V}_i^{path} - \mathbf{V}_k^{ro}$, minimizing the cost function in (9) helps reduce the upper bound of $\mathbf{V}^{res\_path}$. An example output of our clustering is shown in Figure 4.

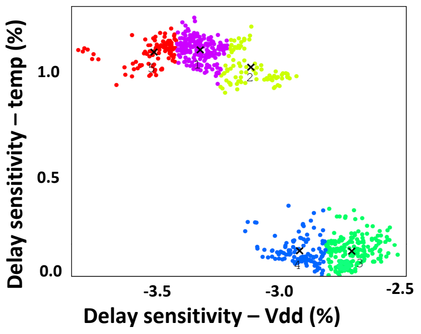


Fig. 4. Every dot in the figure represents a critical path's delay deviation for 20mV deviation in supply voltage (Vdd) and 15 degrees C deviation in temperature (temp). The critical paths are extracted from an ARM M3 processor (45nm technology) and simulated using HSPICE. We cluster the paths into 5 clusters and label them by different colors. The centroid of each cluster is marked by a black cross.

[2]In our experiments, calculating the maximum delay distribution of several hundreds of critical paths takes up to a minute of CPU time.

D. Proposed Delay Estimation Method

The reference method requires $N \times (M + H)$ parameters for runtime delay estimation. To reduce the number of parameters, we propose to design DDROs such that each of them is similar to the maximum delay distribution of each cluster. We calculate the maximum delay of paths in each cluster using the method in [18], assuming that the means of path delays correspond to their nominal values. The outcome of this step gives us the expected maximum delay. But more importantly, it also extracts the sensitivity of the maximum delay to variation sources ($\mathbf{V}^{max}$). Similar to the reference approach, we represent $\mathbf{V}^{max}$ as a function of $\mathbf{V}^{ro}$:

$$\mathbf{V}_x^{max} = \sum_{k=1}^{M}\{a_{xk}\mathbf{V}_k^{ro}\} + \mathbf{V}^{res\_clust} \tag{10}$$

where a_{xk} is a constant coefficient, and $\mathbf{V}^{res\_clust}$ is the mismatch between $\mathbf{V}_x^{max}$ and $\mathbf{V}^{ro}$. Note that when $\mathbf{V}^{ro}$ is equal to $\mathbf{V}^{max}$, $\mathbf{V}^{res\_clust} = 0$. However, the synthesized $\mathbf{V}^{ro}$ is usually slightly different from $\mathbf{V}^{max}$. Thus, having a_{xk} is useful to reduce $\mathbf{V}^{res\_clust}$. The approximate delay of cluster x is given by

$$\begin{aligned} d_x^{clust} = d_x^{nom\_clust}(1 + \sum_{k=1}^{M}\{a_{xk} \cdot \mathbf{V}_k^{ro} \cdot \mathbf{G}\} \\ + \mathbf{V}^{res\_clust} \cdot \mathbf{G} + r_x) \end{aligned} \tag{11}$$

where d_x^{clust} denotes the delay of the x^{th} cluster, $d_x^{nom\_clust}$ represents the nominal delay of the x^{th} cluster, and r_x represents the random local delay of the x^{th} cluster. After measuring DDROs, we can calculate the maximum delay distribution of a chip d^{max} as

$$\begin{aligned} \sigma(d_x^{clust}) &= \{\sigma(||\mathbf{V}^{res\_clust} \cdot \mathbf{G}||)^2 + \sigma(r_x)^2\}^{\frac{1}{2}} \\ \mu(d_x^{clust}) &= d_x^{nom\_clust}(1 + \sum_{k=1}^{M} a_{xk}\mathbf{V}_k^{ro} \cdot \mathbf{G}) \\ d^{max} &= \max_{x=1}^{M}\{\mathcal{N}(\mu(d_x^{clust}), \sigma(d_x^{clust}))\}. \end{aligned} \tag{12}$$

Then, based on the distribution of d^{max}, we can find the value of d'^{max} as in (8).

Using this approximation method, the total number of parameters reduces from $N(1 + M + H)$ to $M(2 + M)$, where $M << N << H$. Moreover, the number of operations for to calculate maximum of two delay distributions during runtime is reduced from $\log(N)$ to $log(M)$. For small M, this method can be implemented in hardware. Clearly, this estimation method is faster and more hardware resource-efficient than the reference method. As we will show later, the estimation error of this approximation approach compared to the reference method is very small.

IV. Synthesis of DDROs

A. ILP Formulation

Given a delay sensitivity target ($\mathbf{V}^{ro}$), we want to choose the number of each gate module in a DDRO, so that the delay sensitivities of the DDRO match the targeted delay sensitivities.

Since each gate module type is instantiated a discrete number of times, we formulate DDRO synthesis as an integer linear programming (ILP) problem:

$$\begin{aligned}
\text{min.:} \ & \Big| \sum_{h=1}^{Y} \{d_h^{nom\_gate} \times s_h\} \times \mathbf{V}^{ro} \\
& - \sum_{h=1}^{Y} \{d_h^{nom\_gate} \times s_h \times \mathbf{V}_h^{gate}\} \Big| \\
\text{s.t.:} \ & \sum_{h=1}^{Y} d_h^{nom\_gate} \times s_h \ \geq \ \text{minimum DDRO delay} \\
& \sum_{h=1}^{Y} s_h \qquad\qquad \leq \ \text{maximum gate count} \\
& \left.\begin{array}{ll} \sum_{h=1}^{Y} z_h - 2s_{inv} & = 1 \\ s_{inv} & \geq 0 \end{array}\right\} \text{ ensure RO oscillates}
\end{aligned} \tag{13}$$

where $d_h^{nom\_gate}$ is the nominal delay of candidate gate type h and s_h is the integer variable that indicates the number of copies of gate type h in the DDRO. Y is the total number of gates allowed, z_h is a binary variable that indicates whether gate h is inverting, and s_{inv} is a positive integer variable. In our experiments, solving the ILP with the public-domain solver [12], takes about one hour on a 3GHz single-core CPU.

Instead of minimizing the difference in relative delay sensitivity, the formulation in (13) minimizes the absolute delay sensitivity so that the objective function is linear in s_h. This favors a solution with a smaller DDRO nominal delay, which may be suboptimal. To compensate this inherent bias in the ILP, we add a constraint to define the minimum DDRO delay. We then sweep the value of minimum DDRO delay at 10 evenly spaced intervals along its feasible range.

B. Practical Considerations

Selecting major variation sources: To identify major variation sources that affect delay sensitivity, we simulate a seven-stage RO using HSPICE, and perturb each variation source one at a time. Based on the results in Figure 5, we can see that most of the variation sources have noticeable effect on the delay except for C_{gdl}, C_{gdo} and C_{jswg}. Therefore, we only consider 12 out of the 15 major variation sources, summarized in Table II. We do not include second-order sensitivities to the variation sources because their magnitudes are very small. This assumption is supported by the experiment data in Figure 3.

In our experimental setup, the impact of interconnect is modeled by parasitic resistance and capacitance extracted from design layout. However, we do not model interconnect as a variation source because its impact is relatively small compared to that of active devices [5]. If interconnect variations are to be included, the DDRO must be built with gate modules (see Figure 6) that are sensitive to interconnect variations. While this can be achieved by connecting the standard cells in gate modules with interconnects at higher metal layers, in such a case DDROs cannot be measured at an early manufacturing stage, making short-loop process monitoring infeasible.

[3]Where not mentioned, the σ values of variation sources are taken from a commercial 45nm process.

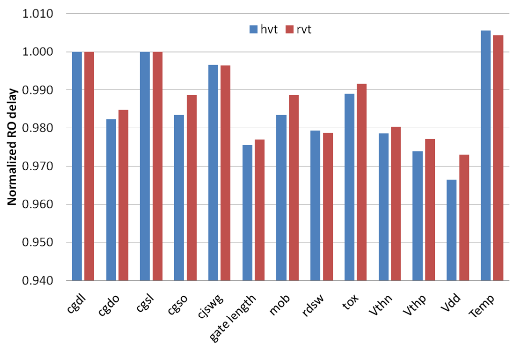


Fig. 5. Delay sensitivities of an RO to different variation sources show that most of the sources have noticeable effect except for C_{gdl}, C_{gsl} and C_{jswg}.

TABLE II
LIST OF VARIATION SOURCES

Variation source	Descriptions[3]
V_{dd}	Supply voltage. V_{dd} nominal (V_{nom}) is 0.9V, $3\sigma = 0.05 \times V_{nom} = 45mV$.
Temperature	Ambient temperature. Nominal temperature = 25^oC, $3\sigma = 30^oC$.
C_{gdo}	MOSFET gate overlap capacitance at drain junction
C_{gso}	MOSFET gate overlap capacitance at source junction
R_{dsw}	Channel series resistance per unit width
μ_0	Mobility of MOSFET
L_{gate}	MOSFET gate length
T_{ox}	Oxide thickness of MOSFET
$Rvtn$	Threshold voltage of RVT NMOS
$Rvtp$	Threshold voltage of RVT PMOS
$Hvtn$	Threshold voltage of HVT NMOS
$Hvtp$	Threshold voltage of HVT PMOS

Characterizing gate sensitivity: Our ILP formulation in (13) assumes that delay sensitivity of a gate is insensitive to other gates connected before and after it. This is a key assumption that simplifies the problem. If we model $\mathbf{V}^{gate}$ as a function of its adjacent gate type, the total number of variables and the design space become intractable.

To decouple the load and slew interaction between the gates, we introduce *gate modules* as basic building blocks for DDRO. A gate module is defined as several identical gates connected in series as illustrated in Figure 6. Simulation results in Figure 7 show that the sensitivity difference due to different input slew and output load is reduced from 0.15% to 0.03%, as the number of stages in a gate module increases from 1 to 15. In this work, we use 5-stage gate modules as a tradeoff between stability of sensitivity and total area of a gate module.

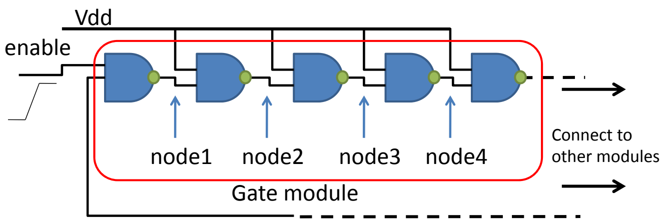


Fig. 6. Illustration of a gate module in a DDRO.

To further reduce the effect of output load, we carefully select the candidate gate types such that each of them has similar gate capacitance. Since the interconnect is also important for path sensitivity, we use two types of interconnect lengths in building

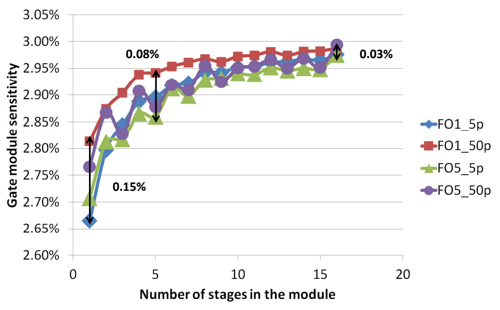


Fig. 7. Simulation results show that the sensitivities under different input slews {5ps, 50ps} and output loads {FO1, FO5} combinations converge as the number of stages in a gate module increases.

our gate modules, i.e., the interconnect between consecutive gates in a module can be either short ($5\mu m$) or long ($20\mu m$). All interconnects in a gate module have the same length and gate modules with different interconnect lengths are considered to be of different instance types even if they have the same gate type. Extra input pins of a multi-input gate are assigned to high or low to make a gate module inverting or buffering (see Figure 6).

Extracting b_{ik} and a_{xk}: Section III, represented $\mathbf{V}^{path}$ and $\mathbf{V}^{max}$ as linear combinations of $\mathbf{V}_k^{ro}$, using b_{ik} and a_{xk}, respectively. The challenge in this step is to find the values of b_{ik} (resp. a_{xk}) such that the resulting residue, $\mathbf{V}^{res\_path}$ (resp. $\mathbf{V}^{res\_clust}$), is minimized. It is important to note that critical path and DDRO delays have nonlinear dependence on parameters in Table II, and that they are subjected to process and environment variations. Thus, solving (5) and (10) using simple least-squares fitting can lead to large b_{ik} (resp. a_{xk}) value, which may magnify the noise from DDRO. For example, Figure 8(a) shows that solving (5) using a linear least-squares method (without constraints on b_{ik}) leads to little delay overestimation when we consider global variation only. However, this is not true when we repeat the experiment with global and local variations, as well as other variations that are absent in our delay model.

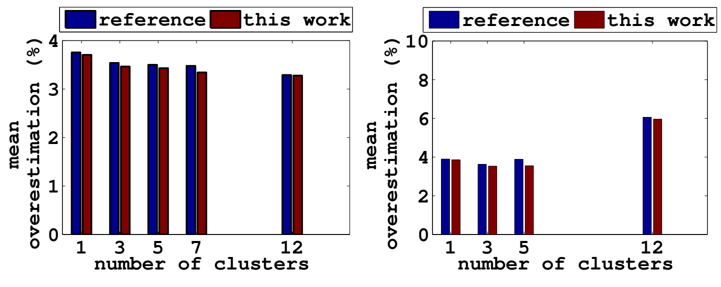
To reduce the impact of large b_{ik} (resp. a_{xk}) values, we solve the extraction problem using linear programming and apply constraints to bound b_{ik} (resp. a_{xk}). The optimization formulation that we use is

$$\text{min.:} \quad \left\| \begin{bmatrix} \mathbf{V}_1^{path} \\ \vdots \\ \mathbf{V}_N^{path} \end{bmatrix} - \begin{bmatrix} \sum_{k=1}^{M} b_{1k}\mathbf{V}_k^{ro} \\ \vdots \\ \sum_{k=1}^{M} b_{Nk}\mathbf{V}_k^{ro} \end{bmatrix} \right\|_F \tag{14}$$
$$\text{s.t.:} \quad lb \le b_{ik} \le ub, \forall\, i, k$$

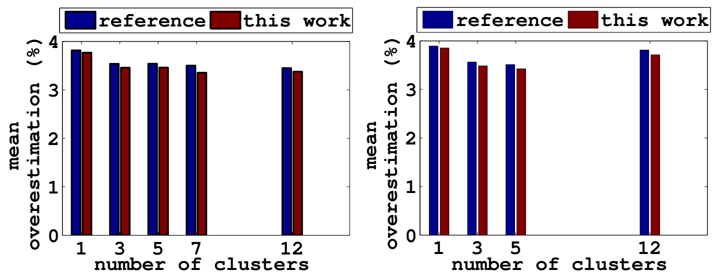
Since delay sensitivity mismatch between the critical paths and $\sum_{k=1}^{M} b_{ik}\mathbf{V}_k^{ro}$ is represented as a $[N \times Q]$ matrix, we use *Frobenius* norm function in (14) to account for each entry in the matrix. We solve the problem using the optimizer in [25], setting ub as 1 and lb as -0.5. Results in Figure 8(b) show that with the constraints in (14), the delay estimation becomes less sensitive to circuit nonlinearity and other variations.

V. Experimental Results

To validate our performance monitoring methodology, we synthesize, place and route three benchmark circuits using a



(a) Linear model results (left) vs. HSPICE results (right) without constraints on b_{ik} for MIPS testcase.



(b) Linear model results (left) vs. HSPICE results (right) with constraints on b_{ik} for MIPS testcase.

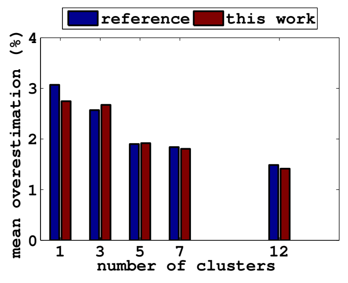
Fig. 8. In case (a), b_{ik} is extracted using linear programming without constraints. Linear model simulation results show that mean delay estimation reduces as the number of clusters increases. However, the HSPICE simulation results show a drastic increase in overestimation due to variations and nonlinearity unmodeled by the linear model. In case (b), simulation results based on b_{ik} extracted with constraints (14) shows that linear model and HSPICE results are similar. This suggests that b_{ik} must be extracted with constraints so that the mean delay overestimation is less sensitive to variations and nonlinearity.

TABLE III
PHYSICAL IMPLEMENTATION RESULTS OF BENCHMARK CIRCUITS. A PATH IS CONSIDERED TO BE CRITICAL WHEN ITS TIMING SLACK IS LESS THAN 10% OF THE SS CORNER CLOCK PERIOD. CRITICAL PATHS ARE EXTRACTED AT BOTH FF AND SS CORNERS TO CAPTURE THE SLOWEST PATHS AT DIFFERENT PROCESS CONDITIONS.

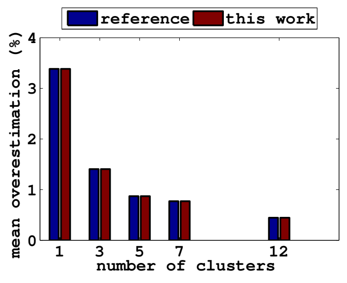
Benchmark circuits	Total number of cells	Clock period at SS corner	Number of critical paths
M0	8169	500ps	218
MIPS	8283	450ps	107
AES	10634	600ps	420

commercial $45nm$ technology. Details of the implemented benchmark designs are listed in Table III. The benchmark circuits are obtained from *ARM* [26] and *Opencores* [27]. Then, following the proposed DDRO design flow in Figure 2, we extract delay sensitivity of each critical path to each of the variation sources in Table II using HSPICE. Note that HSPICE-based sensitivity characterization is not mandatory in our design flow, and that it can be replaced by other methods (e.g., the statistical method in [20]).

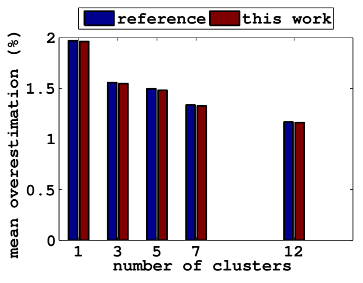
To evaluate the quality of our DDRO synthesis and delay estimation methodologies, we run Monte Carlo experiments on the critical paths and DDROs. Since each critical path is defined for a specific input and simulated independently, we cannot capture the correlation of local variation due to gate sharing. As an alternative, we run another set of Monte Carlo experiments using the linear model in (3). In both simulations, we use the path and DDRO delay sensitivities extracted from HSPICE



(a) AES

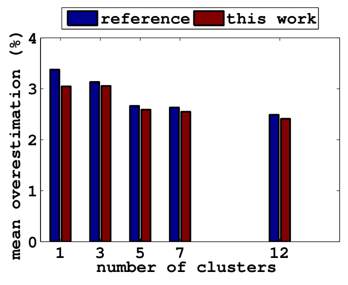


(b) M0

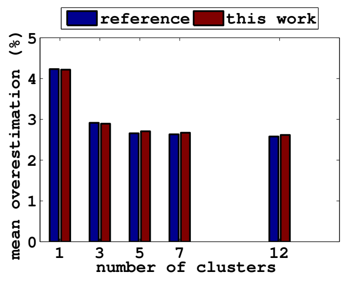


(c) MIPS

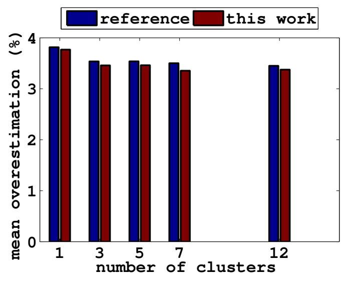
Fig. 9. Linear model simulation results with global variations only and constrained b_{ik} ($ub = 1$, $lb = -0.5$).



(a) AES

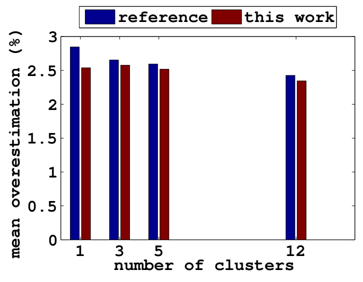


(b) M0

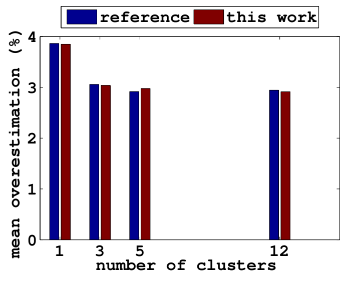


(c) MIPS

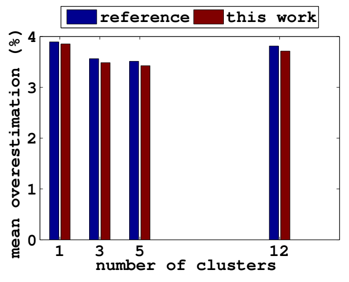
Fig. 10. Linear model simulation results with global and local variations, and constrained b_{ik} ($ub = 1$,$lb = -0.5$).



(a) AES



(b) M0



(c) MIPS

Fig. 11. HSPICE results with constrained b_{ik} for global and local variations.

simulation results to minimize the discrepancy between them. In the linear model experiment, we sample the values of variation sources by using the Gaussian random number generator in *Matlab* [23]. In HSPICE simulation, we use the built-in Monte Carlo setup in the $45nm$ commercial device model. The number of trials in the Monte Carlo experiment is 1000 and 100 for the linear model and for HSPICE simulation, respectively. In all experiments, we set the user-specified confidence Z = 99%.

A. Simulation Results

Experiments using linear model: The simulation results in Figure 9 and 10 show that our approximate delay estimation method achieves similar results compared to the reference method. The results also show that mean delay overestimations of all benchmark circuits decrease noticeably as the number of clusters increases from 1 to 12. This confirms our hypothesis that having multiple DDROs that correlate well with the critical paths can reduce chip delay overestimation. The results also show that delay overestimation is nonzero even when the number of DDROs = 12. This is because $\mathbf{V}^{res\_path}$ and $\mathbf{V}^{clust}$ are nonzero when we apply constraints in the b_{ik} and a_{xk} extractions.

We further observe that the benefit of using multiple DDROs is more significant when the local variation is relatively less compared to the global variation. This is because replica-like monitors (e.g., PSRO, DDRO, PLL) can only replicate the impact of global variation on critical paths. If local variation dominates, more intrusive monitoring is required to measure the impact of local variation.

Based on the simulation results with global and local variations, minimum delay overestimations for the AES, M0 and MIPS testcases are 2.4%, 2.6% and 3.3%, respectively. Note that the values of minimum delay overestimation correlate with the clock period of the benchmark circuits (see Table III), which is related to the magnitude of local variations. This suggests that the achievable minimum delay overestimation is limited by the local variation of a design. Therefore our performance monitoring method may be more suited for low-speed designs with longer critical paths that are less susceptible to local delay variations.

HSPICE Simulations: The HSPICE results in Figure 11 are mostly similar to the linear model results. Several sources of inaccuracies contribute to the discrepancies between HSPICE and linear model results. First, our delay estimation does not account for nonlinearity in circuit delay. Although we have shown that the impact of nonlinearity is small (Figure 3), small errors from nonlinearity could be magnified by b_{ik} or a_{xk}. In other words, due to circuit nonlinearity, the delay estimation is sensitive to the extraction of b_{ik} and a_{xk}. For example, the

MIPS benchmark circuit has a higher overestimation when the number of DDROs is 12. This is an artifact of the constraints in (14), whereby a tighter constraint can reduce delay estimation quality (see Figure 12). We leave understanding of the tradeoff between estimation quality and robustness for future work.

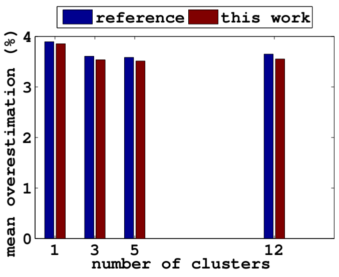


Fig. 12. After applying tighter constraints $ub = 1$ and $lb = -0.1$ for MIPS, the HSPICE results with global and local variations become less sensitive to variations and unmodeled nonlinearity.

Despite a user-specified confidence of 99%, the results show 2.5% and 5.3% of instances (chips) being underestimated in the linear model and HSPICE experiments, respectively. Since the results of the linear model experiment are free from nonlinearity error, the underestimation error is mainly due to the approximation in the statistical maximum function given by [18]. The HSPICE results have more underestimated instances because local variation is not modeled correctly, i.e., HSPICE simulates critical paths with uncorrelated local random variation but our delay estimation accounts for correlation between local variations. As a result, our delay estimates are slightly smaller than the path delays obtained from HSPICE simulation.

VI. Conclusion

In this paper, we have proposed methods to systematically design multiple DDROs, and to estimate circuit performance (chip delay) based on the measurements from the multiple DDROs. Our study shows that by using multiple DDROs we can reduce up to 25% (from 4% to 3%) of the mean delay overestimation of a design. We also show that our delay estimation method can achieve similar results as the reference method with significantly less parameters. Therefore, our method is more amenable to hardware implemention.

We also observe that the benefit of using replica-like monitors (such as DDROs) is more significant when the local variation is relatively less compared to the global variation. If local variation dominates, then in-situ monitoring, though expensive, will fare better. With shrinking feature dimensions, increasing wafer sizes and changing device structures (e.g. fully depleted SOI, FinFETs), it is difficult to project which of the two components of variation is going to dominate in future technologies.

To verify the performance of DDRO and the proposed delay estimation approach, we have taped out a testchip using $45nm$ technology together with an ARM CORTEX M3 CPU. Ongoing work also addresses (1) the tradeoff between estimation quality and robustness during b_{ik} and a_{xk} extraction; and (2) silicon measurements from our testchip.

Acknowledgments

This work is supported in part by SRC, the UC Discovery program IMPACT center, and NSF Variability Expedition grant CCF-1029030.

References

[1] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding", *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027-1035.

[2] M. Bhushan, A. Gattiker, M. Ketchen and K. K. Das, "Ring Oscillators for Cmos Process Tuning and Variability Control", *IEEE Transactions on Semiconductor Manufacturing* 19(1) (2006), pp. 10-18.

[3] T. Black, "A Critical Path Based Parametric Ring Oscillator", *Master's Thesis*, Texas Tech University, 2000.

[4] L. M. Burns, L. Dauphinee, R. A. Gomez and J. Y. C. Chang, "Process Monitor for Monitoring and Compensating Circuit Performance", *U.S. Patent* No. US7375540B2, May 2008.

[5] T.-B. Chan, R. S. Ghaida and P. Gupta, "Electrical Modeling of Lithographic Imperfections", *Proc. IEEE/ACM International Conference on VLSI Design*, 2010, pp. 423-428.

[6] T.-B. Chan, A. Pant, L. Cheng and P. Gupta, "Design Dependent Process Monitoring for Back-End Manufacturing Cost Reduction", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2010, pp. 116-122.

[7] L. Cheng, P. Gupta, K. Qian, C. Spanos and L. He, "Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across Wafer Variability", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30(3) (2011), pp. 388-401.

[8] B. Das, B. Amrutur, H. Jamadagni, N. Arvind and V. Visvanathan, "Within-Die Gate Delay Variability Measurement Using Reconfigurable Ring Oscillator", *IEEE Transactions on Semiconductor Manufacturing* 22(2) (2009), pp. 256-267.

[9] A. Drake, R. Senger, H. Singh, G. Carpenter and N. James, "Dynamic Measurement of Critical-Path Timing", *Proc. IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, 2008, pp. 249-252.

[10] D. Fick, N. Liu, Z. Foo, M. Fojtik, J.-S. Seo, D. Sylvester and D. Blaauw, "In Situ Delay-Slack Monitor for High-Performance Processors Using An All-Digital Self-Calibrating $5ps$ Resolution Time-to-Digital Converter", *Proc. IEEE International Solid-State Circuits Conference*, 2010, pp. 188-189.

[11] K. Kang, S. P. Park, K. Kim and K. Roy, "On-Chip Variability Sensor Using Phase-Locked Loop for Detecting and Correcting Parametric Timing Failures", *IEEE Transactions on Very Large Scale Integration Systems* 18 (2010), pp. 270-280.

[12] *lp_solve reference guide*. http://lpsolve.sourceforge.net/5.5/ .

[13] Q. Liu and S. S. Sapatnekar, "Capturing Post-Silicon Variations Using a Representative Critical Path", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29(2) (2010), pp. 211-222.

[14] H. C. Ngo, G. D. Carpenter, A. J. Drake and J. B. Kuang, "Circuit Timing Monitor Having a Selectable-Path Ring Oscillator", *U.S. Patent* No. US7810000B2, October 2010.

[15] D. J. Philling and C. Talledo, "In-Situ Monitor of Process and Device Parameters in Integrated Circuits", *U.S. Patent* No. US7583087B2, September 2009.

[16] K. Shaik, "Implementation of a Critical Path Based Parametric Ring Oscillator", *BSEE Thesis*, Texas Tech University, 2011.

[17] A. Tetelbaum and S. Chakravarty, "Electronic Design Automation Tool and Method for Optimizing the Placement of Process Monitors in an Integrated Circuit", *U.S. Patent Application* No. US20090282381, November 2009.

[18] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece et al., "First-Order Incremental Block-Based Statistical Timing Analysis", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(10) (2006), pp. 2170-2180.

[19] X. X. Wang, M. Tehranipoor and R. Datta, "Path-RO: a Novel On-Chip Critical Path Delay Measurement Under Process Variations", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 640-646.

[20] L. Xie and A. Davoodi, "Representative Path Selection for Post-Silicon Prediction Under Variability", *Proc. ACM/IEEE Design Automation Conference*, 2010, pp. 593–599.

[21] J. Xiong, V. Zolotov, N. Venkateswaran and C. Visweswariah, "Criticality Computation in Parameterized Statistical Timing", *Proc. ACM/IEEE Design Automation Conference*, 2006, pp. 63-68.

[22] Synopsys PrimeTime User's Manual. http://www/synopsys.com/ .

[23] Mathworks Matlab documentation. http://www/mathworks.com/help/techdoc/ .

[24] Synopsys HSPICE User's Manual. http://www/synopsys.com/ .

[25] CVX: Matlab Software for Disciplined Convex Programming. http://cvxr.com/cvx/ .

[26] ARM Cortex-M0 processor. http://www.arm.com/products/processors/cortex-m/cortex-m0.php .

[27] http://opencores.org