Synthesis and Analysis of Design-Dependent Ring Oscillator (DDRO) Performance Monitors

Tuck-Boon Chan, *Student Member IEEE*, Puneet Gupta, *Member IEEE*, Andrew B. Kahng, *Fellow IEEE*, and Liangzhen Lai, *Student Member IEEE*

***Abstract*— With CMOS technology scaling, circuit performance has become 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 the synthesis of circuit performance monitors that are specific to a given design. We develop a systematic approach for the synthesis of multiple design-dependent monitors, as well as the corresponding calibration and delay estimation methods. Our approach synthesizes design-dependent ring oscillators (DDROs) using standard-cell library gates and conventional physical implementation flows. Our delay estimation method limits the memory usage overhead by clustering critical paths with similar delay sensitivities. Experimental results show that our delay estimation method using multiple DDROs reduces overestimation (timing margin) by up to 25% compared to using a single monitor. Furthermore, our silicon measurement results for monitoring an industrial microprocessor implemented in a 45-nm silicon-on-insulator process show that DDRO can reduce the mean delay estimation error by 35% compared to inverter-based ring oscillators.**

***Index Terms*— Adaptive voltage scaling, circuit performance monitoring, clustering, ring oscillators.**

I. Introduction

CIRCUIT performance variability continues to increase as a result of process variability, wide operating ranges, and other factors. Performance variability can often be compensated by accurate circuit performance estimation and subsequent adaptation. For example, circuit performance can be monitored in the manufacturing flow for process tuning, or systems with adaptive mechanisms can optimize the tradeoff between energy and performance based on feedback from runtime circuit performance monitors [12]. In this paper, we define circuit performance monitoring as a process that estimates the worst case delay of a circuit, based on measurements obtained from on-chip monitors.

Manuscript received March 31, 2013; revised August 9, 2013; accepted September 12, 2013. Date of publication October 18, 2013; date of current version September 23, 2014. This work was supported in part by SRC and in part by NSF Variability Expedition under Grant CCF-1029030.

T.-B. Chan is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: tbchan@ucsd.edu).

P. Gupta and L. Lai are with the Electrical Engineering Department, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: puneet@ee.ucla.edu; liangzhen@ucla.edu).

A. B. Kahng is with the Department of Computer Science and Engineering and Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: abk@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2013.2282742

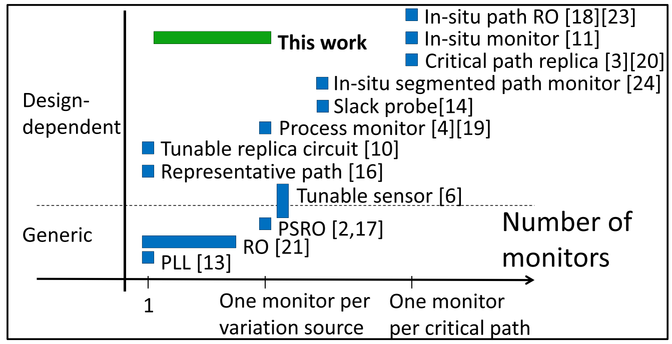

Fig. 1. Taxonomy of performance monitoring methods.

Previous works on VLSI circuit performance monitoring can be classified according to the taxonomy shown in Fig. 1. Generic monitors range from simple inverter-based ring oscillators (ROs) to more sophisticated process-sensitive ROs (PSROs) [2], [17] and alternative monitoring structures such as phase-locked loops (PLLs) [13]. However, such generic monitors are inadequate for capturing design characteristics such as mix of device types, which differ in responses to process variations. As a result, delay estimation using generic monitors is less accurate, which leads to larger timing margins.

Design of monitoring structures that are correlated to circuit performance (design-dependent monitors) has been addressed in several ways. Liu and Sapatnekar [16] 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 the 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 a single RCPs may be inaccurate. The tunable replica circuit (TRC) method in [10] can synthesize different delay paths to more flexibly mimic circuit performance, but has higher design overhead compared to RO approaches. TRC also requires costly calibration to obtain configurations that correspond to different operating conditions. Alternatively, Chan and Kahng [6] propose tunable ROs which can be used as generic or design-dependent monitors. To obtain more accurate (design-dependent) performance estimations, the tunable ROs require calibrations at skewed process corners.

By coupling process parameters extracted from parametric monitors with a design-specific delay model, more accurate delay estimation can be obtained from generic test structures [4], [7], [19]. 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 a large amount of calibration and resources for storage and computation of parameters. Another class of design-dependent monitors, called *in situ* monitors [3], [11], [14], [18], [20], [23], [24], estimates circuit performance by measuring delays of the critical paths. However, use of an *in situ* monitor for each critical path incurs a high area overhead. To reduce the number of monitors, Lai *et al.* [14] propose to selectively measure the delays of nodes in a netlist to estimate critical path delays. Although *in situ* monitors are accurate, they may increase design turnaround time because embedding *in situ* monitors interferes with the timing of actual critical paths.

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 the critical path delay sensitivities form natural clusters (see Fig. 4). Therefore, we can capture the design-specific delay sensitivities by synthesizing a monitor to match 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 the critical paths.

Our DDRO approach offers several potential benefits compared to previous works.

1) DDROs are more accurate compared to conventional ROs because they are synthesized to match the delay sensitivities of the critical paths.
2) DDROs are more accurate compared to a single RCP because multiple DDROs are used to account for the differences between the critical paths.
3) DDROs are less intrusive compared to *in situ* monitoring methods.
4) The total number of ROs (silicon area) is greatly reduced because of the clustering of critical paths. Only a few DDROs are required to provide accurate delay estimation.
5) DDROs can be used for early process tuning, post-silicon 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.

Since DDROs are replica-like monitors, they can only replicate the impact of global variation on critical paths. Thus, our monitoring approach is more suitable for long critical paths that pass through many gates. If within-die variation dominates chip performance (e.g., chip performance is limited by hold-time critical paths and within-die variation is large), an *in situ* monitor is required for accurate performance estimation. Due to this inherent limitation of replica-like monitors, we only consider setup-time critical paths in this paper.

Our contributions can be summarized as follows.

1) We propose a systematic methodology to design multiple DDROs. Our experimental results show that the use of multiple DDROs can reduce delay overestimation by 15% to 25% compared to using only one DDRO.

TABLE I

GLOSSARY OF TERMINOLOGY

Term	Description
h	Index for gate module types
i	Index for critical paths
j	Index for chips
k	Index for DDROs
x	Index for clusters
y	Index for gate instances
H	Total number of gate module types
I	Total number of critical paths
M	Total number of variation sources
K	Total number of DDROs
X	Total number of clusters
Y	Total number of gate instances
S_h	Total number of type h gate modules in a DDRO
$d_k^{nom\_ro}$	Nominal delay of the k^{th} DDRO
$d_x^{nom\_clust}$	Nominal delay of the x^{th} cluster
$d_i^{nom\_path}$	Nominal delay of the i^{th} critical path
$d_h^{nom\_gate}$	Nominal delay of the h^{th} gate
d_k^{ro}	Delay of the k^{th} DDRO
$d_k^{meas\_ro}$	Delay of the k^{th} DDRO measured from a chip
$d_{kj}^{meas\_ro}$	Delay of the k^{th} DDRO measured from the j^{th} chip
d_x^{clust}	Delay of the x^{th} cluster
d_i^{path}	Delay of the i^{th} critical path
d^{max}	Maximum delay of a chip
d^{est_max}	Estimated maximum delay of a chip
$d_j^{est\_max}$	Estimated maximum delay of the j^{th} chip
$d_j^{cal\_max}$	Estimated maximum delay of the j^{th} chip with calibration biased by one standard deviation
$\mathbf{v}_k^{ro}$	Delay sensitivities of the k^{th} DDRO to all M variation sources
$\mathbf{v}_x^{max}$	Delay sensitivities of the x^{th} cluster to all M variation sources
$\mathbf{v}_i^{path}$	Delay sensitivities of the i^{th} critical path to all M variation sources
$\mathbf{v}_x^{res\_clust}$	Residue of delay sensitivity in the x^{th} cluster
$\mathbf{v}_i^{res\_path}$	Residue of delay sensitivity of the i^{th} critical path
$\mathbf{v}_h^{gate}$	Delay sensitivity of the h^{th} gate module to all M variation sources
b_{ik}	Coefficient for the k^{th} DDRO after decomposing delay sensitivities of the i^{th} critical path according to delay sensitivities of DDROs
a_{xk}	Coefficient for the k^{th} DDRO after decomposing delay sensitivities of the x^{th} cluster according to delay sensitivities of DDROs
$\mathbf{R}$	Correlation matrix for local variation of critical path delays
$\mathbf{g}$	Global variation vector with all M variation sources
$\mathbf{g}_j$	Global variation vector of the j^{th} chip with all M variations sources
l_i^{path}	Local variation of the i^{th} critical path
e_k	A random variable that represents delay noise of the k^{th} DDRO
u_i	Delay estimation uncertainty for the i^{th} critical path
r_x	Local delay variation of the x^{th} cluster
λ	A constant coefficient
F_i	Standard normal random variable
$\sigma(\cdot)$	Standard deviation function
$\mu(\cdot)$	Expectation (mean) function
$erf(\cdot)$	Error function of Gaussian distribution
$P(\cdot)$	Cumulative probability function
Z	User-defined confidence, $0 \leq Z < 1$

2) We tape out a testchip and obtain silicon measurement results showing that DDRO can reduce the mean delay estimation error by 35% compared to a generic inverter-based RO.
3) We propose a method to estimate chip delay and minimize guardband margin by using multiple DDRO measurements. Our delay estimation method has negligible difference compared to a path-based estimation method, but the number of parameters used by our estimation method is significantly reduced.
4) We propose a calibration method to reduce delay estimation error due to a skewed process, voltage, and temperature (PVT) corner.

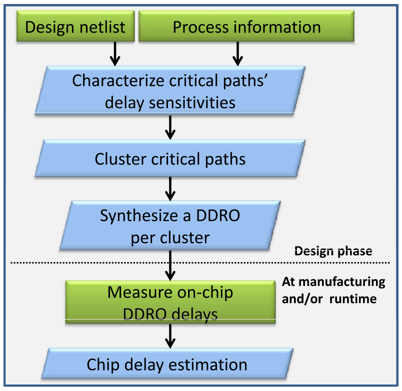

Fig. 2. Overview of DDRO design methodology.

The rest of this paper is organized as follows. Section II gives an overview of our methodology. We present two delay estimation methods in Section III. In Section IV, we discuss the 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 used in this paper are defined in Table I.

II. Overview of DDRO Approach

An overview of our monitoring strategy is shown in Fig. 2. First, given a netlist (die area for DDROs is preallocated), we extract the critical paths of a design by running static timing analysis (STA) using both fast (FF) and slow (SS) corner libraries. We consider a path to be critical if its setup timing slack differs by $\leq$10% of the clock period from the minimum (worst) timing slack over all paths at the corresponding process corner. For example, when the design has a minimum timing slack of 10 ps and clock period of 1 ns, paths with timing slack less than 110 ps are considered to be critical paths. We then characterize delay sensitivities of the critical paths to variation sources using HSPICE [30] with a typical corner process model.[1] Delay sensitivity of the ith critical path ($\mathbf{v}_i^{\text{path}}$) is obtained using finite differences

$$\mathbf{v}_i^{\text{path}} = \left[\frac{d_{i1}^{\text{path}} - d_i^{\text{nom\_path}}}{d_i^{\text{nom\_path}}}, \ldots, \frac{d_{iM}^{\text{path}} - d_i^{\text{nom\_path}}}{d_i^{\text{nom\_path}}} \right] \tag{1}$$

where d_{im}^{path} is the delay of the ith critical path when the mth variation source is biased by one standard deviation from its nominal value, and $d_i^{\text{nom\_path}}$ is the nominal delay of the ith critical path. Second, we cluster the critical paths based on their path sensitivities, and synthesize one DDRO per cluster. We formulate DDRO synthesis as an integer programming problem, in which we seek the set of gates (gate types and number of gates of each gate type) to be concatenated as a DDRO that matches cluster delay sensitivities. Since the gate delays are sensitive to the gate capacitance and slew of adjacent gates, we use gate modules (i.e., several identical gates connected in series) as basic building blocks for DDRO (see Section IV-C). To replicate the effect of interconnect, each gate module has variants with different wirelengths (e.g., INVX1 with 5 and 20 μm wirelengths). By matching DDRO and cluster delay sensitivities, we ensure that the synthesized DDROs have good correlation with the critical paths. Since we use standard cells to synthesize the DDROs, the design and placement of DDROs can be easily integrated with conventional implementation flows. By measuring on-chip DDRO delays, we can estimate chip delay during manufacturing or runtime.

[1] Improved critical-path selection algorithms have been proposed in [25] and [26]. Study of alternatives for path selection is beyond the scope of this paper.

A circuit performance monitor typically feeds back the estimated delay with some margin to ensure chip functional correctness. However, the margin should be minimized to avoid significant performance penalty due to a pessimistic delay estimation. Thus, our goal for circuit performance monitoring is

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

where d^{max} is the actual chip delay, which is defined as the maximum delay across all critical paths. Also, $d^{\text{est\_max}}$ is the estimated chip delay; $P(d^{\text{est\_max}} \geq d^{\text{max}})$ is the probability that $d^{\text{est\_max}}$ is larger than d^{max}; and $\mu(d^{\text{est\_max}} - d^{\text{max}})$ is the expectation of delay overestimation. We use Z to denote a user-specified confidence. For simplicity, we call critical paths as paths in the remainder of this paper when there is no ambiguity.

III. Delay Estimation Using DDROs

Given a set of DDROs, different chip performance estimation methods lead to different estimation errors, runtime, memory requirements, etc. We first analyze a path-based delay estimation method based on a linear model. Then, we propose a cluster-based estimation method that achieves similar accuracy but runs significantly faster and consumes less memory.

A. Delay and Variation Model

We use the variation model in [8], whereby lot-to-lot, wafer-to-wafer, and die-to-die process variations are lumped and modeled as global chip variation. The global variation also includes die-to-die supply voltage and temperature fluctuations. Within-die gate delay mismatches are modeled as random delay variations. Spatial variation is ignored in this paper as it is small for most chips [8]. When the effect of spatial variation is significant, DDROs can be distributed within a die as in [21] to improve correlations between DDROs and the critical paths. We model the critical path delay (d_i^{path}) as a linear function of the variation sources

$$\begin{aligned} d_i^{\text{path}} &= d_i^{\text{nom\_path}} \left(1 + \mathbf{v}_i^{\text{path}} \cdot \mathbf{g} + l_i^{\text{path}}\right) \\ \begin{bmatrix} l_1^{\text{path}} \\ \vdots \\ l_I^{\text{path}} \end{bmatrix} &= \mathbf{R} \cdot \begin{bmatrix} F_1 \\ \vdots \\ F_I \end{bmatrix} \\ \mathbf{R} &= \begin{bmatrix} r(1,1) & \cdots & r(I,1) \\ \vdots & \ddots & \vdots \\ r(I,1) & \cdots & r(I,Y) \end{bmatrix} \end{aligned} \tag{3}$$

TABLE II
LIST OF VARIATION SOURCES

Parameter	Descriptions[5]
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°C, $3\sigma = 30$°C.
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

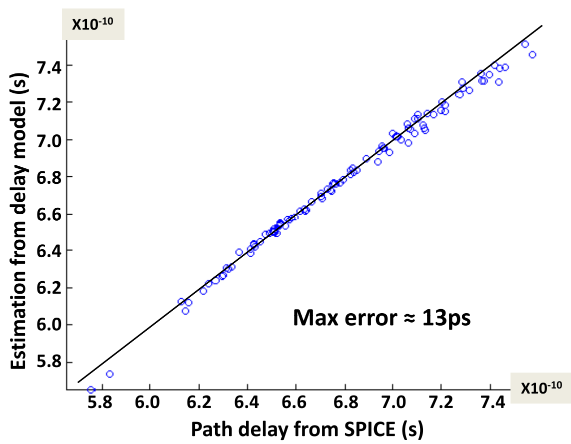

Fig. 3. Rank correlation between delays obtained from HSPICE simulation and the linear model of (3).

where $\mathbf{g}$ is an $M \times 1$ vector that represents the global variation of M variation sources. l_i^{path} is local delay variation of the ith path. $\mathbf{R}$ is a $I \times Y$ correlation matrix that represents the correlation between paths, where I is the total number of critical paths and Y is the total number of gate instances in all I critical paths. $F_1, \ldots, F_I$ are independent random variables, each of which follows a standard normal distribution. When the yth gate instance is on the ith path, the entry $r(i, y)$ in $\mathbf{R}$ is the standard deviation of the yth gate delay variation due to within-die process variation. If the yth gate instance is not on the ith path, the entry $r(i, y)$ is zero. Different gate instances with the same gate type have the same standard deviation for their gate delay variation.[2] For example, all NAND2X1 gates have the same standard deviation.

To verify the accuracy of our delay model, we first simulate a critical path using HSPICE with random global variations whose sources are as listed in Table II (100 trials). Then, we compare the simulated path delays with the delays calculated using the linear model in (3). Fig. 3 shows that path delays obtained from the linear model correlate very well with those from HSPICE simulation.

For DDROs, we also use the delay model 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., we use

$$d_k^{\text{ro}} = d_k^{\text{nom\_ro}}(1 + \mathbf{v}_k^{\text{ro}} \cdot \mathbf{g}) \quad (4)$$

where $d^{\text{nom\_ro}}$ is the nominal delay of the DDRO (obtained from simulation) and $\mathbf{v}_k^{\text{ro}}$ is a $1\times M$ vector that represents the delay sensitivity of the kth DDRO to the vector $\mathbf{g}$ of all M global process variations.

B. Path-Based Delay Estimation

A straightforward delay estimation method is to extract global variation using multiple process variation-specific monitors and calculate chip delay based on the linear model in (3). In other words, monitoring methods in [4], [7], and [19] 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 K DDROs, we can decompose the vector of delay sensitivities $\mathbf{v}_i^{\text{path}}$ as a linear combination of $\mathbf{v}_k^{\text{ro}}$ ($k = 1, \ldots, K$) to utilize measurements from the DDROs

$$\mathbf{v}_i^{\text{path}} = \sum_{k=1}^{K} b_{\text{ik}} \cdot \mathbf{v}_k^{\text{ro}} + \mathbf{v}_i^{\text{res\_path}} \quad (5)$$

where b_{ik} is a constant coefficient and $\mathbf{v}_i^{\text{res\_path}}$ is a $1 \times M$ vector that represents the residue of the delay-sensitivity decomposition.[3]

The values of b_{ik} are obtained by solving a linear program (see Section IV-D). Substituting $\mathbf{v}_i^{\text{path}}$ in (3) as a linear combination of $\mathbf{v}_k^{\text{ro}}$, we obtain

$$d_i^{\text{path}} = d_i^{\text{nom\_path}}\left(1 + \sum_{k=1}^{K} \overbrace{(b_{\text{ik}} \cdot \mathbf{v}_k^{\text{ro}} \cdot \mathbf{g})}^{\text{measurable}} + \overbrace{u_i}^{\text{uncertainty}}\right)$$

where

$$u_i = l_i^{\text{path}} + \mathbf{v}_i^{\text{res\_path}} \cdot \mathbf{g}. \quad (6)$$

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. To estimate the maximum chip delay with the uncertainty u_i, we calculate the distribution of the chip maximum frequency, $d^{\max}$, by using the method in [22]. Then, we can express $d^{\max}$ as a normal distribution using a mean $\mu(d^{\max})$ and a standard deviation $\sigma(d^{\max})$. Given $\mu(d^{\max})$ and $\sigma(d^{\max})$, $d^{\text{est\_max}}$ can be readily obtained using the erf function for Gaussian distribution

$$\text{erf}\left(\frac{d^{\text{est\_max}} - \mu(d^{\max})}{\sigma(d^{\max})}\right) > Z. \quad (7)$$

[2]The standard deviation of the gates are extracted from HSPICE simulations with a variation model that is embedded in the foundry process design kit for the 45-nm silicon-on-insulator (SOI) process.

[3]Since there will be no residue when $K = M$, it is preferred to have $K < M$. In this paper, we try $K = \{1, 3, 5, 7, 12\}$ and show that $K = 5$ is sufficient for our test cases with 12 variation sources ($M = 12$).

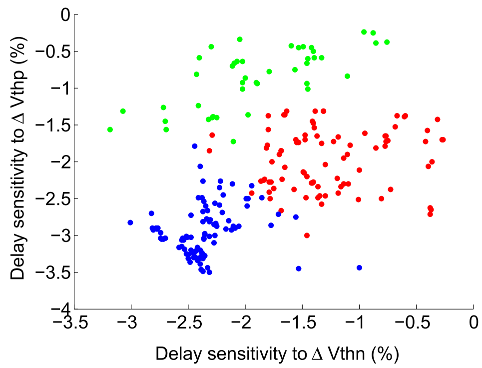

Fig. 4. Every dot in the figure represents a critical path's delay deviation for one standard deviation in nMOS threshold voltage (Vthn) and pMOS threshold voltage (Vthp). The critical paths are extracted from a benchmark circuit AES implemented using a foundry 45-nm SOI technology and simulated using HSPICE. We cluster the paths into three clusters (according to all 12 variation sources) and indicate the three-way clustering by different colors.

C. Clustering

The next step is to minimize delay margin and find $\mathbf{v}_k^{\text{ro}}$. Equations (6) and (7) show that the value of $d^{\text{est\_max}}$ is mainly determined by the $\mathbf{v}_i^{\text{res\_path}}$. A larger $\mathbf{v}_i^{\text{res\_path}}$ will increase the magnitude of $\sigma(d_i^{\text{path}})$, which leads to a larger $d^{\text{est\_max}}$. Therefore, it is desirable to select a set of $\mathbf{v}_k^{\text{ro}}$ that minimizes $\mathbf{v}_i^{\text{res\_path}}$. We find $\mathbf{v}_k^{\text{ro}}$ by clustering critical paths with similar $\mathbf{v}_i^{\text{path}}$ sensitivity vectors, and then assigning the centroid of the k^{th} cluster as $\mathbf{v}_k^{ro}$. To cluster the paths, we use the kmeans++ algorithm [1] and choose the best clustering solution among 100 random starts. The objective function of the clustering is defined as

$$\text{minimize} \sum_{i=1}^{I} \left\{P(d_i^{\text{path}} > \text{clock period}) \times ||\mathbf{v}_i^{\text{path}} - \mathbf{v}_k^{ro}||\right\}, \text{ path } i \in \text{ cluster } k. \tag{8}$$

Since the maximum chip delay is usually determined by the slowest path, we impose a higher penalty for having mismatched delay sensitivities on a path with higher probability of timing failure, i.e., $P(d_i^{\text{path}} > \text{clock period})$. For each path, the probability of timing failure is calculated based on the delay model in (3) and the distributions of variation sources, $\mathbf{g}$. Minimizing the cost function in (8) helps to reduce the upper bound of $\mathbf{v}_i^{\text{res\_path}}$ because the upper bound is defined by $\mathbf{v}_i^{\text{path}} - \mathbf{v}_k^{\text{ro}}$. An example clustering result is shown in Fig. 4.

D. Cluster-Based Delay Estimation

The path-based delay estimation method requires $O(IY)$ parameters for runtime delay estimation. To reduce the number of parameters, we represent path delays in a cluster by the delay of the cluster (d_x^{clust}). We calculate the maximum delay of paths in each cluster using the method in [22] and the path delay model (3). The outcome of this step gives us the expected maximum delay of cluster x. But more importantly, it also extracts the sensitivities of the maximum delay to variation sources ($\mathbf{v}_x^{\max}$). Similar to the path-based approach, we represent $\mathbf{v}_x^{\max}$ as a function of $\mathbf{v}_k^{\text{ro}}$

$$\mathbf{v}_x^{\max} = \sum_{k=1}^{K}\{a_{\text{xk}} \cdot \mathbf{v}_k^{\text{ro}}\} + \mathbf{v}_x^{\text{res\_clust}} \tag{9}$$

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

$$d_x^{\text{clust}} = d_x^{\text{nom\_clust}}\left(1 + \sum_{k=1}^{K}\{a_{\text{xk}} \cdot \mathbf{v}_k^{\text{ro}} \cdot \mathbf{g}\} + \mathbf{v}_x^{\text{res\_clust}} \cdot \mathbf{g} + r_x\right) \tag{10}$$

where d_x^{clust} denotes the delay of the xth cluster, $d_x^{\text{nom\_clust}}$ represents the nominal delay of the xth cluster, and r_x represents the random local delay of the xth cluster. After measuring DDROs, we can obtain the mean and standard deviation of d_x^{clust} as in (11)

$$\sigma(d_x^{\text{clust}}) = \left\{\sigma\left(||\mathbf{v}_x^{\text{res\_clust}} \cdot \mathbf{g}||\right)^2 + \sigma(r_x)^2\right\}^{\frac{1}{2}}$$
$$\mu(d_x^{\text{clust}}) = d_x^{\text{nom\_clust}}\left(1 + \sum_{k=1}^{K} a_{\text{xk}} \cdot \mathbf{v}_k^{\text{ro}} \cdot \mathbf{g}\right). \tag{11}$$

Then, we can calculate the maximum delay distribution of a chip, $d^{\max}$, using the method in [22] and find the value of $d^{\text{est\_max}}$ using (7). Although X and K need not be the same, we let $X = K$ (exactly one DDRO per cluster) for experiments in this paper. Using this cluster-based approximation method consumes less memory compared to the path-based method because the total number of parameters is reduced from $O(IY)$ to $O(K^2)$, where $K \ll I \ll Y$. Moreover, the number of operations to calculate the maximum of two delay distributions is reduced from $O(I)$ to $O(K)$. This reduces maximum-delay calculation time from 1 min (with the path-based method) to less than 1 s (with the cluster-based method).[4] The cluster-based (fast) delay estimation method enables the use of DDROs for real-time performance monitoring, which requires monitors to feed back chip performance variation (due to temperature or voltage variation) as soon as possible so that the chip can adapt to the changes accordingly. When DDROs are used for post-silicon tuning, the cluster-based delay estimation method can reduce calibration time.

IV. Synthesis of DDROs

Given a delay sensitivity target ($\mathbf{v}_k^{\text{ro}}$), we want to construct a DDRO, so that the delay sensitivities of the DDRO match the targeted delay sensitivities. This DDRO synthesis problem is difficult because there can be many combinations of gates to construct a RO. Here, we describe an integer linear programming (ILP) formulation to solve the DDRO synthesis problem. Further, we describe various aspects which must be considered during DDRO synthesis.

[4] In our experiment, calculating the maximum delay distribution of several hundreds of paths with a 3-GHz single-core CPU takes up to 1 min of CPU time.

A. ILP Formulation

Since each gate module type is instantiated a discrete number of times, we formulate DDRO synthesis as an ILP problem

$$\begin{aligned}\text{min.:} \quad & \left\| \sum_{h=1}^{H} \left\{ d_h^{\text{nom\_gate}} \times S_h \right\} \times \mathbf{v}^{ro} \right. \\ & \left. \quad - \sum_{h=1}^{H} \left\{ d_h^{\text{nom\_gate}} \times S_h \times \mathbf{v}_h^{\text{gate}} \right\} \right\| \qquad (12) \\ \text{s.t.:} \quad & \sum_{h=1}^{H} d_h^{\text{nom\_gate}} \times S_h \geq \text{ minimum DDRO delay} \\ & \sum_{h=1}^{H} S_h \leq \text{ maximum gate count}\end{aligned}$$

where $d_h^{\text{nom\_gate}}$ is the nominal delay of candidate gate module type h, and S_h is the integer variable that indicates the number of copies of gate module type h in the DDRO. H is the total number of gate module types. After solving the ILP, $|S_h|$ copies of gate module type h are used in the DDRO. If $|S_h|$ is zero, gate module type h is not used in the DDRO. In our experiments, solving the ILP with the public-domain solver [15] takes 1 h on a 3-GHz single-core CPU. Instead of minimizing the difference in relative delay sensitivity, the formulation in (12) minimizes the absolute delay sensitivity difference so that the objective function is linear in S_h. This favors a solution with a smaller DDRO nominal delay, which may be suboptimal in term of normalized delay sensitivity difference. 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. Selecting Major Variation Sources

To identify the major variation sources that affect delay sensitivity, we simulate a seven-stage RO using the foundry-supplied 45-nm SOI SPICE model. The SPICE model has 13 process-related parameters for process variation analysis. In our experiment, we perturb all of these 13 parameters (one at a time), as well as the supply voltage and temperature. Based on the results in Fig. 5, we can see that most of the variation sources have noticeable effect on the delay except for C_{gdl}, C_{gsl}, and C_{jswg}. Therefore, we only consider 12 out of the 15 major variation sources; these are 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 data in Fig. 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 components that are sensitive to interconnect variations.

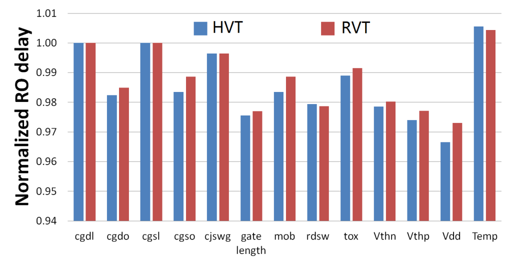

Fig. 5. Delay sensitivities of an RO to different variation sources show that most of the sources have noticeable effect except for C_{gdl}, C_{gsl}, and C_{jswg}. Delays (y-axis) are normalized with respect to the nominal delay of the RO with no variation.

C. Characterizing Gate Sensitivities

Our ILP formulation in (12) assumes that delay sensitivity of a gate (standard cell) is not sensitive to other gates connected before and after it. This is a key assumption that simplifies the problem. If we model $\mathbf{v}_h^{\text{gate}}$ 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 Fig. 6. Simulation results in Fig. 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 paper, we use five-stage gate modules as a tradeoff between stability of sensitivity and total area of a gate module.

For a gate with multiple input pins, gate delays through different input pins will have different delay sensitivities. Thus, each gate module type is defined with respect to a specific input pin. For example, gate module types NAND2X1_A and NAND2X1_B use the same gate type (NANDX1) but the gate modules toggle different input pins (A versus B). Extra input pins of a multiinput gate are assigned to high or low to make a gate module inverting or buffering (see Fig. 6). To obtain a list of candidate gate module types for DDRO synthesis, we use logic standard cells (e.g., AND, OR, OXR, INV gates) to build gate modules. For multiinput gates, we generate a gate module type for each input pin. Since there are many gate module types, we select those that have similar gate capacitance. This is because gate modules with similar gate capacitance have less impact on the delay sensitivities of adjacent gate modules when they are concatenated to form a DDRO.

Since the interconnect also affects path delay sensitivity, we use different wirelengths in building our gate modules. Gate modules with different wirelengths are considered as different instance types even if they have the same gate type. Note that the gate module wirelengths need to be defined based on both the technology and the critical paths that are to be monitored. In our experiment, the wirelengths of critical paths are typically less than 20 μm (see Fig. 8). Thus, we use two types of interconnect lengths in our gate modules, i.e., the wirelength between consecutive gates in a module can be

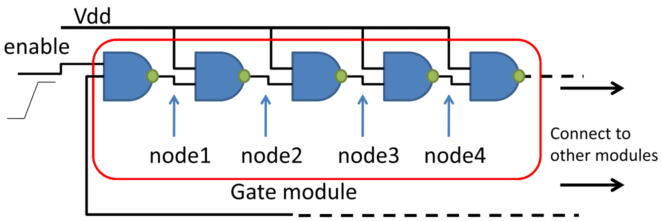

Fig. 6. Illustration of a gate module in a DDRO.

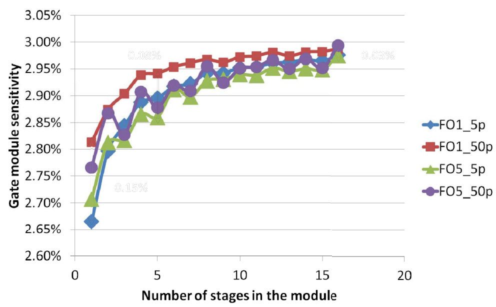

Fig. 7. Simulation results showing that the sensitivities under different input slew {5 ps, 50 ps} and output load {FO1, FO5} combinations converge as the number of stages in a gate module increases.

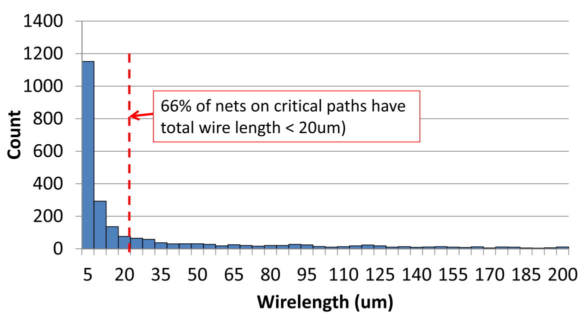

Fig. 8. Wirelength distribution of each net on critical paths. The critical paths are extracted from an ARM M3 processor implemented in 45-nm SOI process.

either short (5 μm) or long (20 μm). As depicted in Fig. 9, we create custom interconnect cells with "snaking" routes to match the desired interconnect wirelengths as well as reduce the total area of DDROs. During physical implementation, we synthesize each DDRO using gate modules which consist of standard cells and custom interconnect cells. The gates modules in each DDRO are placed in two rows to form a loop. The standard cells and the custom interconnect cells in each gate module are placed in series.

D. Extraction of b_{ik} and a_{xk}

As mentioned in Section III, we represent $\mathbf{v}_i^{\text{path}}$ and $\mathbf{v}_x^{\max}$ as linear combinations of $\mathbf{v}_k^{\text{ro}}$, using b_{ik} and a_{xk}, respectively. The b_{ik} (resp. a_{xk}) extraction is achieved by solving (5) [resp. (9)] using simple least-squares fitting to minimize the resulting residue $\mathbf{v}_i^{\text{res\_path}}$ (resp. $\mathbf{v}_x^{\text{res\_clust}}$). However, the simple fitting approach can lead to overfitting when $K \approx M$, which results in large b_{ik} (resp. a_{xk}) values and increases delay estimation error. For example, Fig. 10 (left) shows that solving (5) using a linear least-squares method without constraints on b_{ik} leads to

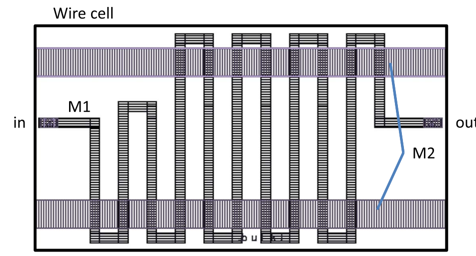

Fig. 9. Custom interconnect cell with a snaking route to reduce total area of long interconnect.

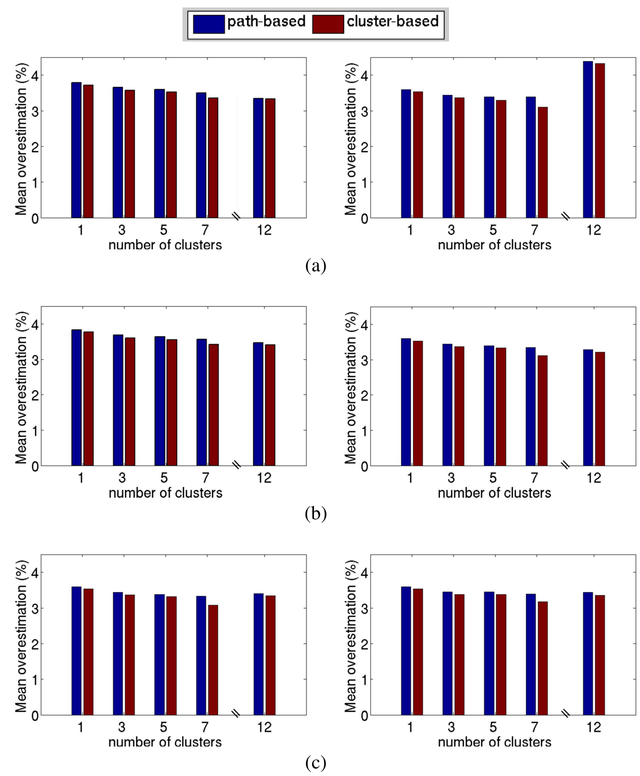

Fig. 10. Estimation error of a test case (MIPS) with different setups. (a) Linear model results (left) versus HSPICE results (right) using linear least-squares method on b_{ik} for the MIPS test case. Linear least-squares method works for linear model but becomes unstable with SPICE results. (b) Linear model results (left) versus HSPICE results (right) using our method for the MIPS test case with $\lambda = 0.02$. With our method, the results are consistent for both linear model and HSPICE results. (c) HSPICE model results with (left) $\lambda = 0.01$ and (right) $\lambda = 0.1$. Our method is robust and insensitive to the value of λ.

little delay overestimation when we consider global variation only. However, Fig. 10 (right) shows that 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. This is because the large b_{ik} (resp. a_{xk}) values magnify delay noise, i.e., the differences between the actual delays and the delays calculated using the linear delay model in (3). The delay noise is mainly due to the fact that critical path and DDRO delays have nonlinear dependence on the parameters in Table II, when subjected to PVT variations.

To reduce the impact of large b_{ik} (resp. a_{xk}) values, [27] formulates the extraction problem as a linear program with upper and lower bounds on b_{ik} (resp. a_{xk}). Although the method of [27] avoids large estimation error, the upper and lower bounds are determined by trial and error to minimize delay estimation error.

In this paper, we consider both RO delay sensitivity decomposition residue and delay noise as errors and formulate the b_{ik} (resp. a_{xk}) extraction problem as a linear program

$$\text{min.:}\quad \mathbf{v}_i^{\text{res\_path}} \cdot \mathbf{g} + [b_{i1} \dots b_{iK}] \cdot \begin{bmatrix} e_1 \\ \vdots \\ e_K \end{bmatrix} \tag{13}$$

where e_k is a random variable that represents the delay noise of DDRO k introduced by the linear delay approximation in (4). Note that the e_k also includes higher order delay sensitivities, any unmodeled variation, as well as the local variation in DDRO due to process variations.

The value of e_k can be estimated by calculating the difference of the delay obtained from HSPICE Monte Carlo simulation and that from (4). Alternatively, we can define λ as the ratio between $\mathbf{g}$ and e_k and simplify the linear program (13) as

$$\text{min.:}\quad \|\mathbf{v}_i^{\text{res\_path}}\|_2 + \lambda \cdot \left\| \begin{bmatrix} b_{i1} \\ \vdots \\ b_{\text{iK}} \end{bmatrix} \right\|_2. \tag{14}$$

Based on our empirical results, we set $\lambda = 0.02$ in this paper. Results in Fig. 10 show that, by using a_{xk} extracted by solving (14), the delay estimations are not sensitive to delay noise caused by circuit nonlinearity and other variations. Moreover, Fig. 10 shows that the delay estimation errors are not sensitive to λ. Thus, the formulation in (14) is more robust than that in [27].

E. Synthesis Results

Fig. 11 shows examples of synthesized DDROs for test case M0 with $K = 3$. As shown in the figure, the synthesized DDROs have three sets of linearly independent delay sensitivities. This is an important property because we will use linear combinations of the delay sensitivities to match the delay sensitivities of critical paths or path clusters (DDROs with linearly dependent delay sensitivities are redundant). Fig. 12 shows that, by using linear combinations of delay sensitivities of DDROs (i.e., $a_{\text{xk}} \cdot \mathbf{v}_k^{\text{ro}}$), we can achieve smaller delay sensitivity errors with respect to a critical path compared to using DDROs directly or simple inverter-based ROs. The standard cells in the DDROs are described in Table III.

F. Delay Estimation With Skewed PVT Corner

The estimation methods in Sections III-B and III-D assume that the nominal RO delays ($d_k^{\text{nom\_ro}}$) are obtained from HSPICE simulation at the nominal PVT corner, i.e., the measurable term in (6) is defined as

$$\mathbf{v}_k^{\text{ro}} \cdot \mathbf{g}_j = \frac{d_{\text{kj}}^{\text{meas\_ro}}}{d_k^{\text{nom\_ro}}} - 1 \tag{15}$$

where $d_{\text{kj}}^{\text{meas\_ro}}$ is the delay of the kth DDRO measured from the jth chip and $\mathbf{g}_j$ is the global process variation

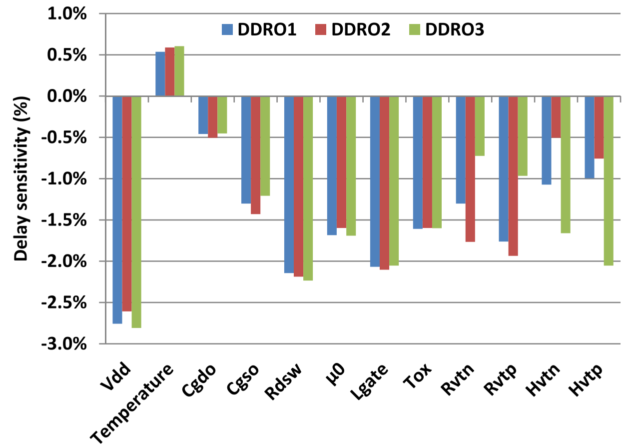

Fig. 11. Delay sensitivities of synthesize DDROs of test case M0. Cluster number = 3. The delay sensitivities (y-axis) is normalized to DDRO delay with no variation.

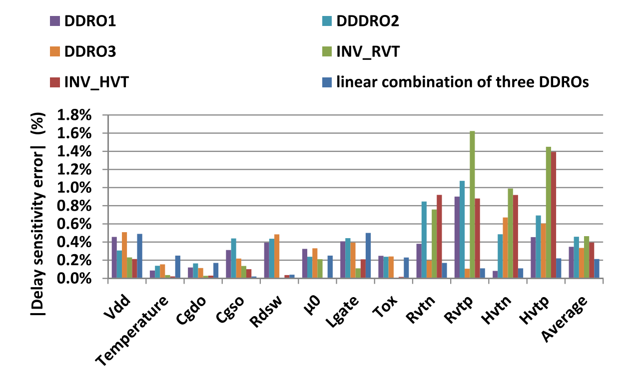

Fig. 12. Delay sensitivity errors of different ROs with respect to the delay sensitivities of a critical path in test case M0. By using linear combination of DDROs, the total delay sensitivity error is reduced compared to simple inverter ROs or DDROs without applying linear combination.

of the jth chip. If the actual operating PVT corner of the chips is significantly skewed compared to the nominal corner, $d_i^{\text{nom\_path}}$ and $d_k^{\text{nom\_ro}}$ obtained from HSPICE simulation will be inaccurate. This is especially important for low-volume production runs. Therefore, we propose a method to calibrate $d_i^{\text{nom\_path}}$ and $d_k^{\text{nom\_ro}}$ when chip samples are available. Given a set of chip samples, we can obtain the mean RO delay across all samples ($\mu(d_k^{\text{meas\_ro}})$). By replacing the $d_k^{\text{nom\_ro}}$ in (15) with $\mu(d_k^{\text{meas\_ro}})$, we compensate for the error caused by a skewed process and/or mismatch between HSPICE model and silicon data

$$\mathbf{v}_k^{\text{ro}} \cdot \mathbf{g}_j = \frac{d_{\text{kj}}^{\text{meas\_ro}}}{\mu(d_k^{\text{meas\_ro}})} - 1. \tag{16}$$

After applying the calibration in (16), we can estimate the delay of the jth chip ($d_j^{\text{est\_max}}$) using (11). Similarly, the chip delay is also susceptible to the skewed process as well as mismatch between HSPICE model and silicon data. Moreover, chip delay can be skewed differently with respect to the DDRO. To minimize delay estimation error resulting from the systematic mismatch between chip and DDRO delays, we propose to apply an additional calibration procedure during chip delay estimation. First, we obtain the expectation of

TABLE III
STANDARD CELLS IN DDROs

	Copies	wirelength	Cell type	Size	Vt
DDRO1	5	w20	NAND2	X1.4	RVT
	5	w20	AOI222	X1.4	HVT
	20	w20	AOI31	X1.4	RVT
	10	w20	INV	X1.2	RVT
	5	w20	OAI31	X2	HVT
	5	w20	OAI31	X3	HVT
	5	w20	OAI31	X3	HVT
	5	w20	XOR2	X1.4	RVT
	5	w5	OAI2XB1	X1.4	RVT
	5	w5	OAI31	X3	HVT
DDRO2	5	w20	NAND2	X1.4	RVT
	10	w20	AOI31	X1.4	RVT
	5	w20	XNOR2	X0.5	HVT
	20	w5	AOI31	X1.4	RVT
	5	w5	OAI221	X1.4	HVT
	15	w5	OAI31	X2	RVT
	10	w5	OAI31	X3	RVT
DDRO3	5	w20	NAND2	X1.4	RVT
	15	w20	AOI221	X1.4	RVT
	5	w20	OA21A1OI2	X1.4	HVT
	5	w20	OAI211	X1.4	HVT
	10	w20	OAI222	X1.4	HVT
	5	w20	OAI222	X1.4	RVT
	5	w20	OAI31	X2	HVT
	5	w20	OAI31	X2	RVT
	5	w20	XNOR2	X0.7	HVT
	15	w20	XOR2	X0.5	RVT
	25	w20	XOR3	X0.5	HVT

actual chip delay ($\mu(d^{\max})$) by calculating the average of sample chip delays. Second, we calculate the expectation of chip delay estimation ($\mu(d^{\text{est\_max}})$) by averaging chip delay estimations ($d_j^{\text{est\_max}}$) across all chip samples. In other words, ($\mu(d^{\text{est\_max}})$) is defined as the average of the expectation of estimated chip delay

$$\mu(d^{\text{est\_max}}) = \frac{1}{\text{total samples}} \sum_j \left(d_j^{\text{est\_max}}|_{Z=50\%} \right). \quad (17)$$

The calibrated maximum-delay estimate for chip j ($d_j^{\text{cal\_max}}$) is given by

$$d_j^{\text{cal\_max}} = \frac{\mu(d^{\max})}{\mu(d^{\text{est\_max}})} d_j^{\text{est\_max}}. \quad (18)$$

V. EXPERIMENTAL RESULTS

To validate our performance monitoring methodology, we synthesized, placed, and routed three benchmark circuits using a commercial 45-nm SOI technology. Details of the implemented benchmark designs are listed in Table IV. The benchmark circuits are obtained from ARM [32] and Opencores [34]. Then, we follow the DDRO design flow in Fig. 2. We first run STA using both FF and SS corner libraries. As mentioned in Section II, we consider a path to be critical if its setup timing slack at either FF or SS corner differs from the worst timing slack at the corresponding process corner by no more than 10% of the clock period. We extract delay sensitivity of each critical path to each of the variation sources in Table II using HSPICE with a typical process corner model.

TABLE IV
PHYSICAL IMPLEMENTATION RESULTS OF BENCHMARK CIRCUITS

Benchmark circuit	Total number of cells	Clock period	Number of critical paths
M0	8169	1000ps	218
MIPS	8283	900ps	107
AES	10634	800ps	420

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 [24]).

To evaluate the quality of our DDRO synthesis and delay estimation methodologies, we run Monte Carlo experiments with global and local variations on the critical paths and DDROs. For HSPICE simulation, we use the built-in Monte Carlo setup in the 45-nm commercial device model. 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 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 [29]. The number of trials in the Monte Carlo experiment is 1000 and 100 for the linear model and for HSPICE simulation, respectively. Unless otherwise specified, we set the user-specified confidence $Z = 99\%$.[5]

A. Simulation Results

Experiments using linear model. The simulation results in Figs. 13 and 14 show that our approximate delay estimation method achieves similar results compared to the path-based method.[6] The results also show that mean delay overestimation of all benchmark circuits decreases noticeably as the number of clusters increases from 1 to 12.[7] 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 (i.e., $K = M = 12$). This is because $\mathbf{v}_i^{\text{res\_path}}$ and $\mathbf{v}_x^{\text{clust}}$ are nonzero.

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 the 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 (Fig. 14), minimum delay overestimation values for

[5]When number of trials is small, our delay estimation is more sensitive to the instances of the trials, especially for a high confidence $Z = 99\%$.

[6]The results in this paper are slightly different from those in [27] because we fixed an error in characterization of DDRO delay sensitivities in [27].

[7]When the number of clusters (K) = 1, our DDRO method is similar to the representative critical path replica method [16].

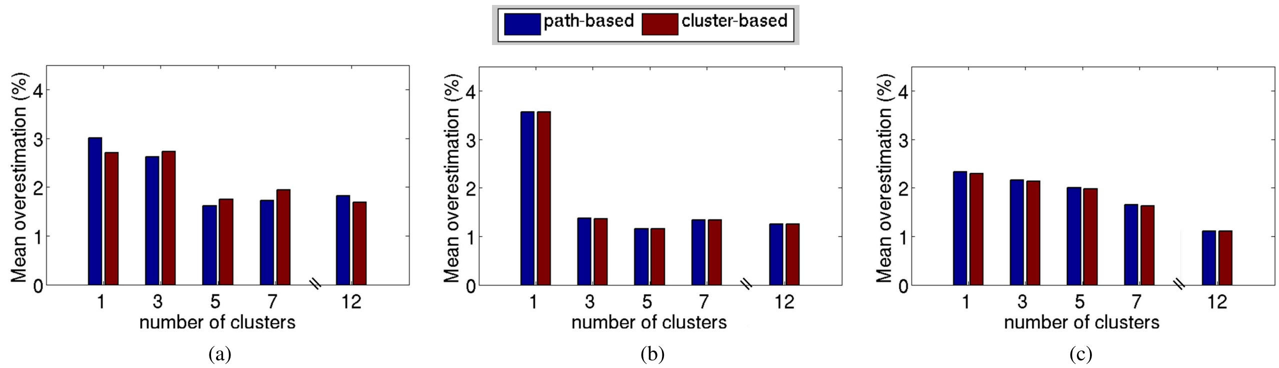

Fig. 13. Linear model simulation results with global variations only. (a) AES. (b) M0. (c) MIPS.

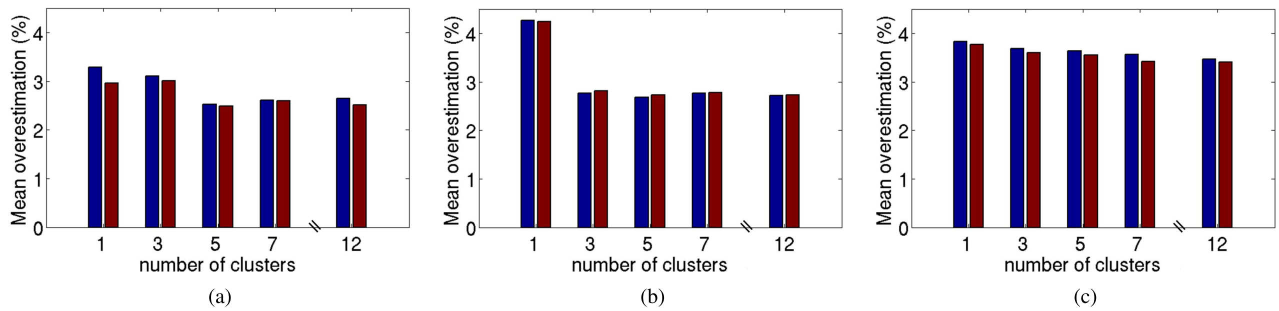

Fig. 14. Linear model simulation results with global and local variations. (a) AES. (b) M0. (c) MIPS.

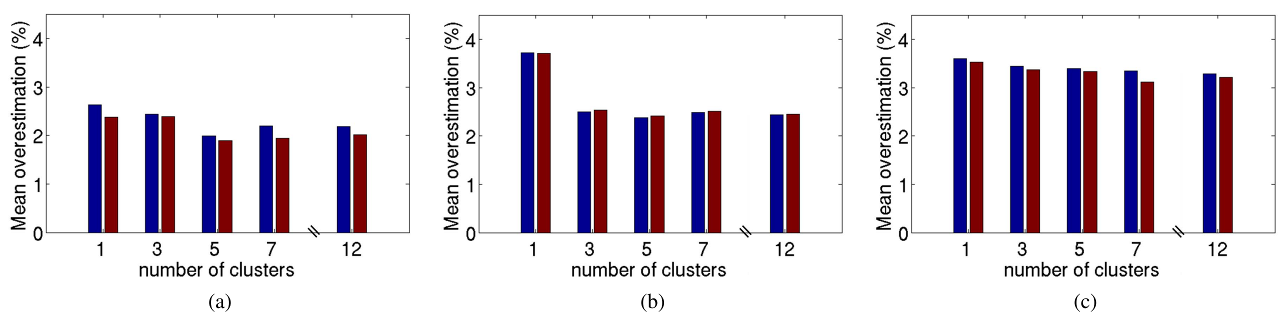

Fig. 15. HSPICE results for global and local variations. (a) AES. (b) M0. (c) MIPS.

the AES, M0, and MIPS test cases are 2.5%, 2.7%, and 3.4%, respectively. The results for $K = 12$ in Figs. 13 and 14 show 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. HSPICE results in Fig. 15 are similar to the linear model results. Discrepancies between HSPICE and linear model results are mainly due to the fact that our delay estimation does not account for nonlinearity in circuit delay. Despite a user-specified confidence of 99%, the results in Table V show that we underestimate the delays of 1.96% and 5.9% instances in the linear model and HSPICE experiments, respectively (average across three benchmarks for cluster-based estimation). 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 [22]. The HSPICE results have more underestimated instances because local variation is not modeled correctly, i.e., HSPICE simulates the 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.

B. Delay Estimation With Calibration

We set up two experiments to evaluate our calibration method in Section IV-F. First, we shift both chip and DDROs supply voltages from nominal supply voltage (0.9 V) to 0.8 V. This experiment setup represents the typical scenario where

TABLE V

AVERAGE UNDERESTIMATED INSTANCES ACROSS $K = \{1, 3, 5, 7, 12\}$

Benchmark	Linear model Global and local variations	HSPICE
AES	25.9/1000 (2.59%)	10.8/100 (10.8%)
M0	12.8/1000 (1.28%)	5.1/100 (5.1%)
MIPS	20.2/1000 (2.02%)	1.7/100 (1.7%)
Total	59.9/3000 (1.96%)	17.6/300 (5.9%)

TABLE VI

AVERAGE OF MEAN DELAY ESTIMATION ERROR NORMALIZED TO MEAN CHIP DELAY MIPS WITH 100 HSPICE MONTE CARLO TRIALS

number of samples	Chip voltage = 0.8V	Chip voltage = 0.9V		
	DDROs voltage	DDROs voltage		
	0.8V	0.8V	0.9V	1.0V
1	6.82%	6.73%	5.78%	3.65%
2	5.21%	6.34%	5.42%	2.71%
5	3.06%	7.17%	2.51%	2.16%
10	3.21%	2.61%	2.00%	2.18%
15	2.56%	2.37%	1.83%	1.70%
20	2.05%	2.22%	1.46%	1.75%
25	1.99%	2.59%	1.41%	1.88%
30	2.37%	1.99%	1.45%	1.72%
35	1.88%	1.96%	1.40%	1.85%
50	1.77%	1.84%	1.39%	1.76%
100	1.74%	1.64%	1.22%	1.52%
no calibration	1.69%	20.68%	1.25%	12.34%

the nominal PVT corner is shifted. Second, we keep the chip supply voltage at 0.9 V but shift all DDROs voltages to {0.8, 0.9, and 1.0 V}. This experiment setup represents the scenario when there is systematic within-die variation between the chip's critical paths and DDROs (e.g., voltage drop in chip's power delivery network).

For each test case, we simulate the critical paths (obtained from MIPS) and DDRO delays using HSPICE Monte Carlo with 100 trials. Based on the simulation results, we estimate chip delay using the cluster-based method in Section III-D with five DDROs ($Z = 50\%$) and compare it with the simulated chip delay. Among the 100 trials, we randomly choose a subset of the chip samples and apply the calibration procedure described in Section IV-F. Since the delay estimation is affected by the selection of chip samples, we repeat this experiment 50 times and report the average values of mean delay estimation error.

Results in Table VI show that, when both chip and DDROs voltages are at the nominal corner (0.9 V), the mean delay estimation error is only 1.25% without applying any calibration. Even when both chip and DDRO voltages are shifted to 0.8 V, the estimation error is only 1.70%. However, if chip voltage remains at 0.9 V but DDROs voltage is shifted to 0.8 or 1.0 V, the estimation error increases significantly (12% to 21%). The estimation error can be reduced significantly when we apply our calibration method (Section IV-F). As the number of samples increases, the average mean delay estimation error reduces rapidly. For instance, the maximum of the average mean delay estimation error is less than 2.5% with 30 samples.

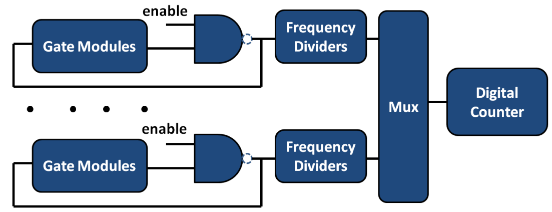

Fig. 16. RO block schematic. In this test chip, we use a 12-stage frequency divider.

TABLE VII

DESIGN INFORMATION OF THE TEST CHIP

Component	Cell count	Cell type
Cortex-M3	50196	mixed VT
DDRO1	13+100	mixed VT
DDRO2	13+85	mixed VT
DDRO3	13+100	mixed VT
DDRO4	13+90	mixed VT
DDRO5	13+85	mixed VT
inverter RO1	13+21	RVT
inverter RO2	13+21	HVT
inverter RO3	13+61	RVT
inverter RO4	13+61	HVT
inverter RO5	13+61	mixed VT

C. Proof-of-Concept Silicon Results

We have taped out a test chip with DDRO-based performance monitoring using a 45-nm IBM SOI technology with dual-V_{th} libraries. The test chip has an ARM Cortex-M3 microprocessor [33] with DDROs. To synthesize the DDROs, we extract the critical paths from the microprocessor and cluster their sensitivities into five clusters by using the kmeans++ algorithm. The results of the path sensitivities clustering is shown in Fig. 4.[8] Then, for each cluster, we synthesize a DDRO which has delay sensitivities similar to the mean delay sensitivities of paths in the cluster. The synthesis method is the same as that in Section IV.

To control DDRO oscillation, a NAND (or AND) gate is added in each RO as shown in the schematic in Fig. 16. An on-chip digital counter is used to obtain the RO frequencies, i.e., the counter will count the number of cycles of a RO within a measurement window. We repeat RO measurements with 40- and 100-ms measurement windows and measure the ROs in different sequences to make sure that the results are consistent and systematic measurement error is minimized. For comparison, we also implemented inverter-based ROs. The design information of the Cortex-M3 and ROs are listed in Table VII. The RO cell count includes the additional NAND (or AND) gate and a 12-stage frequency divider (total 13 cells). The test chip layout and die photo are shown in Fig. 17. We measured the processor maximum operating frequency and RO frequency using the test bed shown in Fig. 18. There are two microcontroller units (MCUs) on the test bed. One of the MCUs is used to control the digital counter of the RO block and to measure the frequency of the ROs. The other MCU is used to control the processor and the on-chip PLL. We measure chip frequency by running a test

[8]At the time of our test chip tapeout, the clustering method of (8) had not yet been developed.

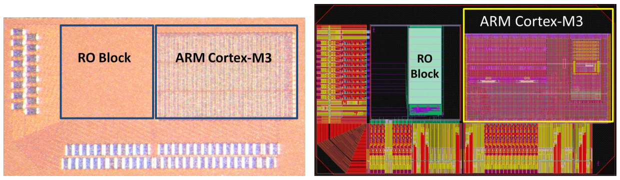

Fig. 17. Test chip die photo and layout illustration.

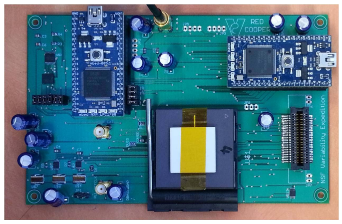

Fig. 18. Test bed for RO frequency measurement and processor frequency measurement. Two microcontroller units are designed to control the processor and RO blocks, respectively.

program (fast Fourier transform) and increasing the processor's clock frequency (through PLL) until the processor generates incorrect results compared to the precalculated golden results. For each chip, we supply both RO and processor with the same supply voltage.

The measured mean chip and RO delays (14 test chips) are about 2 times the corresponding simulation results. This suggests that the chips are operating at a very skewed PVT corner compared to the HSPICE simulation. Therefore, we use the calibration method described in Section IV-F to estimate chip delays. To minimize the estimation error, we use all 14 chips for the calibration. For each inverter-based RO, we treat it as one DDRO designed for the all critical paths, i.e., $x = k = 1$. Then we apply the same calibration as in Section IV-F and estimation method as in Section III-D for the inverter-based ROs (with $a_{xk} = 1$). The results of the mean delay estimation error are shown in Fig. 19 ($Z = 0.5$). The measurement results show that, by using five DDROs, we can reduce the mean delay estimation error by 35% (from 2.3% to 1.5%) compared to generic inverter-based ROs. To ensure that our results are not sensitive to measurement errors, we repeat the analysis by injecting random noise (standard normal distribution with $\sigma = 1\%$, 3%, and 5% with respect to RO frequency) into all RO measurements. Results in Table VIII show the average mean delay estimation error of DDRO and inverter-based ROs across 30 random trials. The improvement of DDRO over inverter-based ROs is approximately 25%–30%, which is consistent with our observation drawn from Fig. 19.

We also deploy ROs with different numbers of stages to estimate the effect of local variation. The results in Fig. 19 show that the errors of 61-stage inverter ROs are similar to those of their 21-stage counterparts. This suggests that random local variation in ROs has little impact on the estimation error in our experiment. In Fig. 20, we plot the statistics of the delay estimations. The results show that the minimum and maximum delay estimation errors using DDROs are smaller to those of inverter-based ROs. Note that our results are based on measurements on 14 test chips from a single wafer.

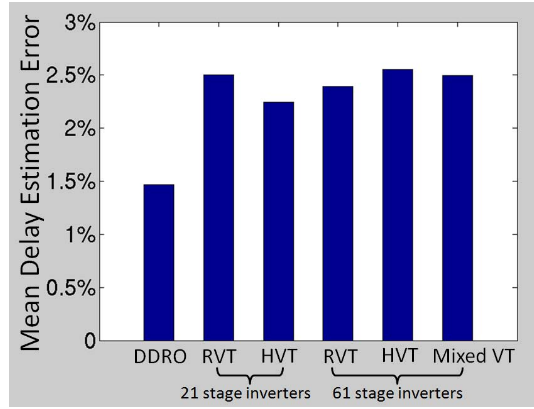

Fig. 19. Mean delay estimation error obtained from DDROs and inverter-based ROs. Estimation errors are calculated by taking the absolute difference between normalized estimation and normalized chip delay.

TABLE VIII
MEASUREMENT ERROR SENSITIVITY ANALYSIS

	mean delay estimation error					
	σ noise = 1%		σ noise = 3%		σ noise = 5%	
	Avg (%)	Improvement (%)	Avg (%)	Improvement (%)	Avg (%)	Improvement (%)
DDRO	1.60	NA	2.40	NA	3.30	NA
21 stage RVT inverter RO	2.60	38	3.20	25	4.40	25
21 stage HVT inverter RO	2.30	30	3.20	25	4.50	27
61 stage RVT inverter RO	2.50	36	3.20	25	4.40	25
61 stage HVT inverter RO	2.70	41	3.60	33	4.70	30
61 stage mixed VT inverter RO	2.60	38	3.40	29	4.40	25

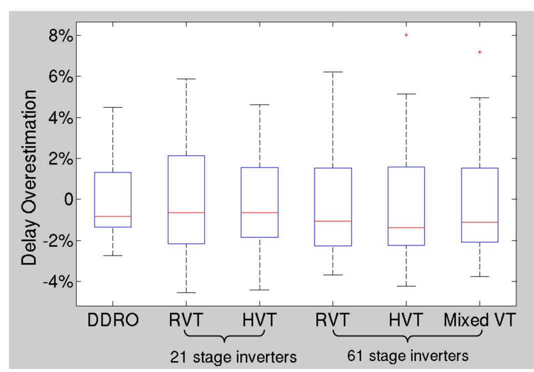

Fig. 20. Maximum and minimum delay overestimation obtained from DDROs and inverter-based ROs. The edges of the boxes are the corresponding 25th and 75th percentiles of the data.

With multiple wafers from different lots, we expect that the improvements may be different (improvement is likely to be higher since the magnitude of global variation will increase compared to local variation).

D. Comparison With Other Monitoring Methods

Table IX summarizes the differences among different replica-like design-dependent monitoring methods. The method proposed in [16] has small implementation overheads

TABLE IX

COMPARISON OF DIFFERENT REPLICA-LIKE DESIGN-DEPENDENT MONITORING METHODS

	Implementation overheads	Calibration effort	Accuracy
[16]	Small	No calibration	Low
[6]	Small	Low	Low
[10]	Medium	High	High
This work	Small	Low	Medium

because it uses a single representative critical path to estimate chip delay. Although this method does not require any calibration, it is relatively less accurate because it relies on a single representative critical path to estimate a set of critical paths.[9] The method of [6] also has small implementation overheads because it requires only a set of simple ROs. However, one-time calibrations at skewed process corners are required to make the ROs to be design-specific. Even with calibration, the method in [6] is not necessarily accurate because it calibrates the configurations of ROs to guardband for the worst possible delay. Tunable replica circuits in [10] are more accurate but require more complex circuits and calibration steps. By contrast, this paper has proposed a method that also has small implementation overheads because the monitor consists of only a few DDROs. Our method requires a calibration step to compensate for any difference between the simulation model and actual silicon as described in Section V-B. We expect that our method is more accurate than the method of [16] because we use multiple DDROs to track the delays of critical paths. Our method is also more accurate than that in [6] because we estimate the critical path delays instead of the worst-possible delay. Although our method may be less accurate than the tunable replica circuit, our method does not require calibration for every chip and also has less implementation overhead.

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. We showed that our delay estimation method can achieve similar results as the path-based method with significantly less bookkeeping overhead. We also showed that by using multiple DDROs we can reduce the mean delay overestimation by up to 25% (from 4% to 3%). The reduction is mainly limited by local variation, which cannot be captured by replica-like monitors. Further delay overestimation reduction will require *in situ*-type monitors, which have much higher area and design implementation overheads. 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, although 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 DDROs and our delay estimation approach, we taped out a test chip using 45-nm SOI technology together with an ARM CORTEX M3 CPU. Our silicon results have shown that DDRO can reduce the mean delay estimation error by 35% (from 2.3% to 1.5%) compared to generic inverter-based ROs.

[9]This approach is similar to our DDRO method with $K = 1$.

ACKNOWLEDGMENT

The authors would like to thank Prof. D. Sylvester, Dr. D. Fick, M. Fojtik, and Dr. D. Kim, University of Michigan, for their generous support in taping out the test chip.

REFERENCES

[1] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. ACM-SIAM Symp. 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 Trans. Semicond. Manuf.*, vol. 19, no. 1, pp. 10–18, Feb. 2006.

[3] T. Black, "A critical path based parametric ring oscillator," M.S. thesis, Dept. Electr. Eng., Texas Tech Univ., Lubbock, TX, USA, Dec. 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 7 375 540, May 20, 2008.

[5] T.-B. Chan, R. S. Ghaida, and P. Gupta, "Electrical modeling of lithographic imperfections," in *Proc. IEEE/ACM Int. Conf. VLSI Design*, Jan. 2010, pp. 423–428.

[6] T.-B. Chan and A. B. Kahng, "Tunable sensors for process-aware voltage scaling," in *Proc. IEEE/ACM ICCAD*, Nov. 2012, pp. 7–14.

[7] T.-B. Chan, A. Pant, L. Cheng, and P. Gupta, "Design dependent process monitoring for back-end manufacturing cost reduction," in *Proc. IEEE/ACM ICCAD*, Nov. 2010, pp. 116–122.

[8] 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 Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 3, pp. 388–401, Mar. 2011.

[9] B. Das, B. Amrutur, H. Jamadagni, N. Arvind, and V. Visvanathan, "Within-die gate delay variability measurement using reconfigurable ring oscillator," *IEEE Trans. Semicond. Manuf.*, vol. 22, no. 2, pp. 256–267, May 2009.

[10] A. Drake, R. Senger, H. Singh, G. Carpenter, and N. James, "Dynamic measurement of critical-path timing," in *Proc. IEEE ICICDT*, Jun. 2008, pp. 249–252.

[11] 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," in *Proc. IEEE ISSCC*, Feb. 2010, pp. 188–189.

[12] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. S. Rosing, M. B. Srivastava, S. Swanson, and D. Sylvester, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 8–23, Jan. 2013.

[13] 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 Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 2, pp. 270–280, Feb. 2010.

[14] L. Lai, V. Chandra, R. Aitken, and P. Gupta, "SlackProbe: A low overhead in situ on-line timing slack monitoring methodology," in *Proc. IEEE DATE*, Mar. 2013, pp. 282–287.

[15] (2010). *Lp_Solve Reference Guide* [Online]. Available: http://lpsolve.sourceforge.net/5.5/

[16] Q. Liu and S. S. Sapatnekar, "Capturing post-silicon variations using a representative critical path," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 2, pp. 211–222, Feb. 2010.

[17] I. A. K. M. Mahfuzul, A. Tsuchiya, K. Kobayashi, and H. Onodera, "Variation-sensitive monitor circuits for estimation of global process parameter variation," *IEEE Trans. Semicond. Manuf.*, vol. 25, no. 4, pp. 571–580, Nov. 2012.

[18] 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 7 810 000, Oct. 5, 2010.

[19] D. J. Philling and C. Talledo, "In-situ monitor of process and device parameters in integrated circuits," U.S. Patent 7 583 087, Sep. 22, 2009.

[20] K. Shaik, "Implementation of a critical path based parametric ring oscillator," M.S. thesis, Dept. Electr. Comput. Eng., Texas Tech Univ., Lubbock, TX, USA, 2011.

[21] 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 0 282 381, Nov. 12, 2009.

[22] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece, J. Piaget, N. Venkateswaran, and J. G. Hemmett, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2170–2180, Oct. 2006.

[23] X. X. Wang, M. Tehranipoor, S. George, D. Tran, R. Datta, and L. Winemberg, "Design and analysis of a delay sensor applicable to process/environmental variations and aging measurements," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1405–1418, Aug. 2012.

[24] L. Xie and A. Davoodi, "Representative path selection for post-silicon prediction under variability," in *Proc. ACM/IEEE 47th Design Autom. Conf.*, Jun. 2010, pp. 593–599.

[25] L. Xie and A. Davoodi, "Bound-based statistically-critical path extraction under process variations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 1, pp. 59–71, Jan. 2011.

[26] V. Zolotov, J. Xiong, H. Fatemi, and C. Visweswariah, "Statistical path selection for at-speed test," in *Proc. IEEE ICCAD*, Sep. 2008, pp. 624–631.

[27] T.-B. Chan, P. Gupta, A. Kahng, and L. Lai, "DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators," in *Proc. IEEE ISQED*, Mar. 2012, pp. 633–640.

[28] (2011). *Synopsys PrimeTime User's Manual* [Online]. Available: http://www/synopsys.com/

[29] (2011). *Mathworks Matlab Documentation* [Online]. Available: http://www/mathworks.com/

[30] (2010). *Synopsys HSPICE User's Manual* [Online]. Available: http://www/synopsys.com/

[31] M. Grant and S. Boyd. (2012). *CVX: Matlab Software for Disciplined Convex Programming* [Online]. Available: http://cvxr.com/cvx/

[32] (2013, Jun.). *ARM Cortex-M0 Processor* [Online]. Available: http://www.arm.com/products/processors/cortex-m/cortex-m0.php

[33] (2013, Jun. 11). *ARM Cortex-M3 Processor* [Online]. Available: http://www.arm.com/products/processors/cortex-m/cortex-m3.php

[34] *OpenCores* [Online]. Available: http://opencores.org

Tuck-Boon Chan (S'09) received the M.S. degree from National Taiwan University, Taipei, Taiwan, in 2007. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of California, San Diego, CA, USA.

He is involved in research under the supervision of Prof. A. B. Kahng. His current research interests include mitigating VLSI circuit variability and improving manufacturing yield through design/manufacturing co-optimization.

Puneet Gupta (M'07) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 2000 and the Ph.D. degree from the University of California, San Diego, CA, USA, in 2007.

He is currently a Faculty Member with the Electrical Engineering Department, University of California, Los Angeles, CA, USA. He co-founded Blaze DFM, Inc., (acquired by Tela Inc.), Sunnyvale, CA, USA, in 2004, and served as a Product Architect in 2007. He has authored over 100 papers, 15 U.S. patents, a book, and a book chapter.

Dr. Gupta is a recipient of the National Science Foundation CAREER Award, the ACM/SIGDA Outstanding New Faculty Award, the European Design Automation Association Outstanding Dissertation Award, and the IBM Faculty Award. He has given several tutorial talks, served on several technical program committees, and was a Program Chair of the IEEE DFM&Y Workshop in 2009, 2010, and 2011. He is the Director of the Multiuniversity IMPACT+ Integrated Modeling Process and Computation for Technology Center.

Andrew B. Kahng (F'10) received the Ph.D. degree in computer science from the University of California at San Diego (UCSD), La Jolla, CA, USA, in 1989.

He was with the Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA, from 1989 to 2000. Since 2001, he has been with the Department of Computer Science and Engineering and the Department of Electrical and Computer Engineering, UCSD, where he holds the endowed chair in high-performance computing. He has authored or co-authored more than 400 journal and conference papers, and three books. He holds 24 issued U.S. patents. His current research interests include IC physical design, the design-manufacturing interface, combinatorial algorithms and optimization, and the roadmapping of systems and technology.

Liangzhen Lai (S'12) received the B.S. degree in electronic engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2010 and the M.S. degree in electrical engineering from the University of California, Los Angeles, CA, USA, in 2012, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering.

His current research interests include hardware variability monitors and cross-disciplinary techniques to leverage variability.