Custom On-Chip Sensors for Post-Silicon Failing Path Isolation in the Presence of Process Variations

Min Li, Azadeh Davoodi, and Lin Xie*
Department of Electrical and Computer Engineering
University of Wisconsin at Madison, WI 53706 USA
*Cadence Design Systems, San Jose, CA 95134 USA
Email: {adavoodi}@wisc.edu

***Abstract*—This work offers a framework for predicting the delays of individual design paths at the post-silicon stage which is applicable to post-silicon validation and delay characterization. The prediction challenge is mainly due to limited access for direct delay measurement on the design paths after fabrication, combined with the high degree of variability in the process and environmental factors. Our framework is based on using on-chip delay sensors to improve timing prediction. Given a placed netlist at the pre-silicon stage, an optimization procedure is described which automatically generates the sensors subject to an area budget and available whitespace on the layout, in the presence of process variations. Each sensor is then generated as a sequence of logic gates with an approximate location on the layout at the pre-silicon stage. The on-chip sensor delay is then measured to predict the delays of individual design paths with less pessimism. In our experiments, we show that custom on-chip sensors can significantly increase the rate of predicting if a specified set of paths are failing their timing requirements.**

I. Introduction

Timing prediction at the post-silicon stage is required for tasks such as circuit adaptation [8], manufacturing test [4], characterization of delay models [5], and isolation and diagnosis of failing paths during post-silicon validation [9], [11]. It is a challenging and time-consuming task due to limited access to the layout after fabrication. The increase in within-die and die-to-die process variations as well as environmental factors also make timing prediction inaccurate. This inaccuracy is higher when predicting at a finer level of granularity, i.e., delays of individual paths in the design as opposed to the circuit timing for one manufactured die.

Previous works have shown that post-silicon measurements can significantly help reduce the variance in predicting the circuit timing in the presence of process variations. For example, adding generic on-chip test structures such as ring-oscillators as well as custom (referred to as representative) paths have been proposed in [7] and [8], respectively. In [4] custom test structures are added to the scribe lines on the wafer to capture die-to-die and wafer-level variations. The main challenges that are not addressed by the above works for a comprehensive timing prediction framework are extending the timing prediction for individual paths and accounting for within-die process variations.

Timing prediction of individual design paths is an important step during post-silicon validation [1], [9], since it helps to predict/isolate the failing paths which violate the timing requirement. However, failing path isolation is a challenging task; at the post-silicon stage, the failing paths may not be among the critical paths identified based on pre-silicon delay models and timing analysis [5].

A number of previous works have focused on path-level timing prediction at the post-silicon stage in order to isolate the failing paths.

This research was supported by National Science Foundation under Grant 1053496.

978-3-9810801-8-6/DATE12/©2012 EDAA

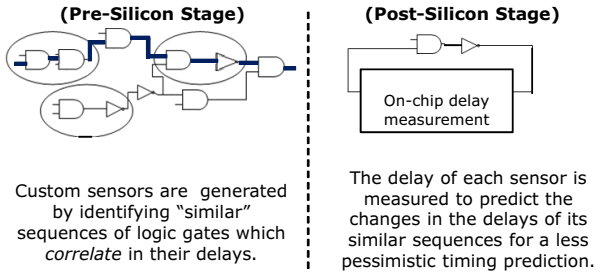

Fig. 1. Sensor-assisted timing prediction.

The approaches include statistical learning from a set of previously identified post-silicon failures [3], clock shrinkage to narrow down the location and cycle of a failure after its observation [6], and bound-based isolation using parametric static timing analysis based on post-silicon measurements [9]. Some of these techniques *rely on* observing existing timing failures in order to predict the future ones [3]. Consequently, they require identifying possibly-many failure cases to collect sufficient number of samples for learning which can become time-consuming and expensive if it has to be repeated for each design. On the other hand, the approach in [9] relies on post-silicon measurements prior to identifying the failures but it does not elaborate which components on the die should be measured. It also does not consider within-die process variations. Another previous work [11] focuses on identifying and measuring representative design paths to predict the delays of a larger set of potentially-failing paths. However, designation of potentially-failing paths may not be accurate at the design stage [5]. In addition, the approach in [11] only handles a small number of paths because and its runtime complexity is proportional to the number of designated paths.

This work proposes a framework which generates custom on-chip sensors for a design at the pre-silicon stage. It then uses post-silicon delay measurements on these sensors for timing prediction, as shown in Figure 1. Our framework captures within-die process variations while allowing timing prediction of arbitrarily-specified paths and isolating the failing ones with less pessimism. Compared to the path-based framework of [11], it operates on the entire circuit.

The main focus in our framework is design of on-chip sensors. First, at the post-placement stage, the layout is partitioned into several regions. For a given netlist which falls within a region, the optimization procedure generates custom sensors in that region in the presence of process variations. This is subject to an area budget specified by the user and available white space in the region. Each sensor is a sequence of logic gates and is generated to highly correlate in its delay with other "similar" sequences in the netlist. *Two sequences are represented by the same sensor, if their sensitivities*

to process variations are less than a small error tolerance which define the "similarity constraints" during optimization. Two similar sequences could be made of different logic gates but the idea is that the change in the delays of the two sequences are almost identical. Therefore, measuring the delay of a sensor at the post-silicon stage allows making timing prediction for all the sequences which are represented by it. For example in Figure 1 (left), the path shown in bold contains two sequences which highly correlate in their delays with an identified sensor. Using post-silicon measurements, the uncertainty in the path delay is reduced to only those portions which are not *covered* by the sensor.

Our optimization framework forms the sequences such that the coverage of the timing graph corresponding to the netlist is maximized. Each edge in the timing graph can also be assigned an arbitrary weight, reflecting its criticality. The optimization is independently solved for each region which makes the framework scalable with the design size. Within-die process variations are accounted for: The gates of different regions have a lower correlation in their delays than the gates within a region. Furthermore, the gates within a region are not assumed to be fully correlated.

There are many options for on-chip delay measurement (of either design paths or sensors) such as [5], [10]. For a choice of measurement infrastructure, an associated measurement error can be considered in our framework during post-silicon analysis. However, design and integration of the measurement infrastructure is beyond the scope of this paper.

The contributions of this work are summarized below:

- A scalable optimization framework which generates custom on-chip sensors for a placed netlist subject to an area budget and available white space;
- Post-silicon sensor-assisted prediction of failing paths considering within-die and die-to-die process variations.

In our experiments we use Monte Carlo simulation to model post-silicon process variations of many dies in 45nm technology with up to 682 independent random variables. For the ISCAS89 circuits and a budget of 15% of the core area, we show sensor-assisted prediction increases the rate of determining if a set of paths identified to be statistically-critical at the design stage are failing post-silicon. This is compared to the case when the prediction is made without using the sensors. To show the effectiveness of our sensor generation, we also compare to an alternative procedure which generates custom sensors by matching the netlist against a set of *pre-specified* logic sequences and ignores within-die variations.

In the remainder of the presentation, Section II gives an overview of our framework. The details of our sensor generation procedure are presented in Section III. Simulation results are presented in Section IV, followed by conclusions.

II. Overview of the Framework

For a given netlist at the post-placement stage, first, rectangular regions are created on the layout. Smaller-sized and local netlists are then defined based on the regions and an optimization procedure is applied to find custom sensors for each case independently. For example, as shown in Figure 2, each region may contain a few custom sensors based on the timing characteristics of the local netlist and of process variations. Also, it is possible that some regions do not contain any sensors because no white space is available or because the area limit has been reached. Two sensors at two different regions may be made of identical gates and interconnects, however they are considered different due to different process variations in their corresponding regions.

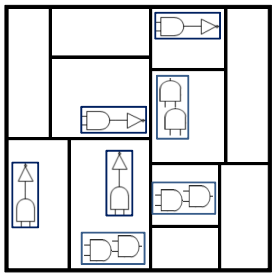

Fig. 2. Identifying sensor(s) at each region allows capturing delay sensitivity to within-die process variations.

To form the rectangular regions, we apply recursive bi-partitioning of the layout in horizontal and vertical directions. At each step, a region is partitioned into two smaller ones such that the number of cells are as close as possible among them. The bi-partitioning of a region terminates when the number of cells within a region is less than 100 and when each region contains at least 50 nets. This termination condition is defined regardless of the benchmark instance, based on our experimental observations. The goal is to form a small-sized yet meaningful optimization problem and ensure a relatively fast solution procedure while still capturing enough information about the local netlist. Furthermore, the gates inside the region may have different sensitivities to process variations and may not be fully correlated. However, the small size of the region allows modeling the correlation among these gates with fewer number of random variables which also contributes to reducing the size of the regional optimization problem as we explain later.

Boundary cases are properly handled. After the partitioning is terminated, the following procedure is applied to extract a netlist for a region: The cells in the netlist are those which fully or partially fall inside the region, as well as those that fall outside the region but connect to the interior ones directly. All the nets that connect these cells to each other are also included. This strategy ensures each cell in the original netlist maps to at least one region. It is possible that the cells which are close to the boundary of multiple regions map to all of them. *So a boundary cell may be included in the optimization problem of more than one region and thus be captured by more than one sensor in the end for a more accurate timing prediction.* The local netlist in each region is then used to solve smaller-sized problems independently and achieve a scalable optimization framework.

III. Sensor Generation Procedure

In this section we explain an optimization procedure to generate custom sensors for a netlist. *All referrals to netlist reflect the local netlist corresponding to a single region.*

A. Problem Statement

We model a given netlist as a directed acyclic timing graph $G = (\mathcal{V}, \mathcal{E})$. The gates are modeled by the nodes and their interconnections are modeled by directed edges. We define a sequence to be the nodes identified by a set of consecutive directed edges in the graph. E.g., a sequence may be portion of a combinational path. Our optimization is formulated to identify "similar" sequences which have almost-identical sensitivities to process variations and thus are highly correlated in their variation-aware delay expressions. These sequences may have different delays but their sensitivities to delay variations are similar. The objective is to maximize the coverage of the edges in the timing graph, assuming each edge is associated with a weight reflecting its criticality. Each group of similar sequences

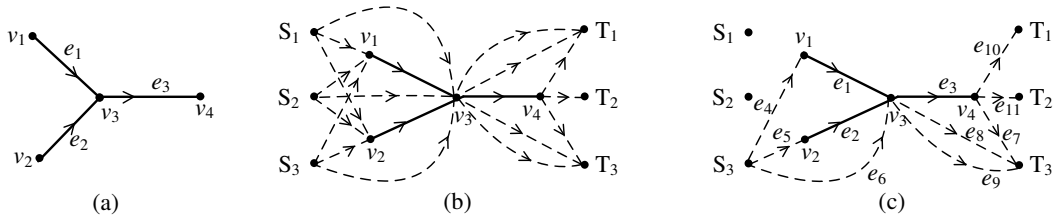

Fig. 3. (a) The graph corresponding to the original (local) netlist (b) The extended graph after modification (c) The simplified version of (b) for illustrations

is represented by one sensor which will get added to a region on the layout. The optimization further constrains the sensor area to be the minimum of a provided budget and the available white space of the corresponding region. Next, we provide an Integer Programming formulation of this optimization problem.

B. Notations and Variable Definitions

For each edge e_i we associate an area A_i^e which is the area of the gate at the starting node of the edge given by $s(e_i)$. In the presence of process variations, we assume $\mathbf{X}$ to be a column vector. Each element in $\mathbf{X}$ lies in the category of die-to-die, within-die and random variations in physical parameters. *We also assume that all elements in* $\mathbf{X}$ *are independent from each other and have mean 0 and variance 1 (after decorrelation)*. Similar to [2], we describe the variation-aware delay of edge e_i (representing delay of $s(e_i)$ and its interconnect) using the following linear expression:

$$D_{e_i} = \mu_{e_i} + \mathbf{a}_{e_i}^T \mathbf{X} \tag{1}$$

where μ_{e_i} is the nominal delay of e_i and $\mathbf{a}_{e_i}$ is its sensitivity vector corresponding to $\mathbf{X}$ after characterization. Note, the above expression already accounts for interconnect delay variations and the delay of an edge already accounts for factors such as loading. This is because the edge represents the *path* connecting the output of a gate to another.

We define a sequence s_i as a set of consecutive directed edges in G. Our goal is to *form* sequences to maximize the coverage of edges $e_i \in \mathcal{E}$, when edge e_i is associated with a weight w_{e_i} reflecting its criticality which is provided as a parameter. To simplify the problem, we assume an upper bound, denoted by $N_{\mathcal{S}}$, on the number of sequences that can be formed. (We later define $N_{\mathcal{S}}$ optimization variables which can reflect if sequence s_j is formed). While forming sequences, the optimization framework simultaneously groups "similar" sequences to be represented by one sensor. We refer to sensor k by f_k and assume the number of formed sensors is upper-bounded by parameter N_f. In our framework, we set $N_{\mathcal{S}} = 10$ and $N_f = 5$ based on empirical observations.

We define the following optimization variables (assuming $i = 1, 2, \ldots, |\mathcal{E}|; j = 1, 2, \ldots, N_{\mathcal{S}}; k = 1, 2, \ldots, N_f$):

- Let binary variable $m_{i,j}^{e:s}$ be equal to 1 if edge e_i is covered by sequence s_j, and 0 otherwise.
- Let binary variable $m_{j,k}^{s:f}$ be equal to 1 if sequence s_j is represented by sensor f_k, and 0 otherwise.
- Let binary variable y_j be equal to 1 if sequence s_j is formed and 0 otherwise.
- Let A_j^s and A_k^f denote the areas of sequence s_j, and of sensor f_k respectively. Also, let D_j^s denote the variation-aware delay expression of sequence s_j.

Using these variables, the constraints ensure that *if an edge is covered, then it is included in exactly one sequence which is ensured to be made of consecutive edges. The sequence is also ensured to be represented by exactly one sensor*.

To describe such constraints, it is helpful to consider the following modification of the original graph: we augment G by additional vertices and edges as shown in Figure 3. We first add $N_{\mathcal{S}}$ source vertices $S_1, S_2, \ldots, S_{N_{\mathcal{S}}}$ and $N_{\mathcal{S}}$ sink vertices $T_1, T_2, \ldots, T_{N_{\mathcal{S}}}$. For example, in Figure 3 we assume $N_{\mathcal{S}} = 3$. From each source node $S_i \in \mathcal{S}$, we add $|\mathcal{E}|$ edges connected to all $s(e_j)$ where $e_j \in \mathcal{E}$, where $s(e_j)$ are the starting vertices of $e_j \in \mathcal{E}$. Similarly, for each sink node $T_i \in \mathcal{T}$, we add $|\mathcal{E}|$ edges connecting from all $t(e_j)$, which are the ending vertices of edge $e_j \in \mathcal{E}$.

The above graph modification facilitates our problem definition: each sequence s_j can be described as a "single" path in the modified graph between nodes S_j and T_j which includes a subset of edges in the original graph. Next, we describe the objective expression and different types of constraints.

C. Objective Expression

We maximize an objective expression which is a weighted summation of the edges that will be covered by the sequences. This objective is given by

$$\max \qquad \sum_{j=1}^{N_{\mathcal{S}}} \sum_{i=1}^{|\mathcal{E}|} w_{e_i} m_{i,j}^{e:s} \tag{2}$$

If edge e_i is covered by sequence s_j, then $m_{ij}^{e:s} = 1$ and edge e_i will be contributing to the objective with the constant weight w_{e_i}. If $w_{e_i} = 1$ then the objective is to maximize the coverage of all edges in the graph. In this work, we also consider the case when an edge weight corresponds to its statistical criticality, reflecting the likelihood that it belongs to a path which fails the timing constraint post-silicon. We denote the statistical criticality of edge e_i by c_{e_i} as [12]:

$$c_{e_i} \triangleq Pr(AT_{s(e_i)} + D_{e_i} + RAT_{t(e_i)} > T_0), \tag{3}$$

where AT_{v_k} is arrival time (RAT_{v_k} is the required arrival time), and reflects the maximum delay from any primary input (output) to node $v_k \in \mathcal{V}$. Both AT_{v_k} and RAT_{v_k} are random variables. In [12] the authors show how these quantities and the criticality probability can be computed efficiently as a pre-processing step. In this work we study two cases of $w_{e_i} = 1$ and $w_{e_i} = c_{e_i}$.

D. Constraints

We identify and discuss the following types of constraints:

- **Sequence constraints:** each sequence should be made of consecutive edges, and each edge can be covered by at most one sequence;
- **Similarity constraints:** the sequences which are correlated in their variation-aware delay expressions should be grouped together and represented by one sensor;
- **Area constraint:** within a region, the summation of the sensor areas is bounded by the minimum of a specified area budget and the available whitespace.

1) Sequence Constraints: The sequence forming constraints are listed below:

$$\sum_{j=1}^{N_S} m_{i,j}^{e:s} \leq 1, \qquad \forall i = 1, \ldots, |\mathcal{E}| \tag{4}$$

$$y_j = \sum_{i|e_i \in \mathcal{E}', s(e_i)=S_j} m_{i,j}^{e:s}, \qquad \forall j = 1, \ldots, N_S \tag{5}$$

$$y_j = \sum_{i|e_i \in \mathcal{E}', t(e_i)=T_j} m_{i,j}^{e:s}, \qquad \forall j = 1, \ldots, N_S \tag{6}$$

$$\sum_{i_1|e_{i_1} \in \mathcal{E}', t(e_i)=v_n} m_{i_1,j}^{e:s} = \sum_{i_2|e_{i_2} \in \mathcal{E}', s(e_i)=v_n} m_{i_2,j}^{e:s} \tag{7}$$
$$\forall v_n \in \mathcal{V}, \forall j = 1, \ldots, N_S$$

$$y_j = \sum_{k=1}^{N_f} m_{j,k}^{s:f}, \qquad \forall j = 1, \ldots, N_S \tag{8}$$

We explain the above constraints using a simple example.

Example: Here, we take Figure 3(b) as an example and assume $N_S = N_f = 3$, which are equal to $|\mathcal{E}|$. For better illustration, we simplify Figure 3(b) to (c) by removing additional edges.

For edge e_3, Equation 4 guarantees that this edge can only belong to at most one sequence. Specifically, we have

$$m_{3,1}^{e:s} + m_{3,2}^{e:s} + m_{3,3}^{e:s} \leq 1. \tag{9}$$

Let us consider sequence s_3. If $y_3 = 0$, then s_3 is not formed. Otherwise, s_3 can be formed by some edges in $\mathcal{E}$. For s_3, we specify constraints in Equations 5, 6, 8 as follows:

$$y_3 = m_{4,3}^{e:s} + m_{5,3}^{e:s} + m_{6,3}^{e:s}, \tag{10}$$

$$y_3 = m_{7,3}^{e:s} + m_{8,3}^{e:s} + m_{9,3}^{e:s}, \tag{11}$$

$$y_3 = m_{3,1}^{s:f} + m_{3,2}^{s:f} + m_{3,3}^{s:f}, \tag{12}$$

To explain Equation 7, we only consider $v_4 \in \mathcal{V}$ due to lack of space but similar derivations can be obtained for the other nodes in $\mathcal{V}$. In this case, we can have

$$m_{3,3}^{e:s} = m_{10,3}^{e:s} + m_{11,3}^{e:s} + m_{7,3}^{e:s} \tag{13}$$

Illustrations: Now, we specifically explain the constraints in Equations 4-7. Equation 4 is for each edge $e_i \in \mathcal{E}$. It ensures that each edge can only belong to at most one sequence. Equation 5 is for each node $S_j \in \mathcal{E}'$ in the modified graph. For node S_j (considered to be the start point of sequence s_j), we consider all outgoing edges e_i. (See Equation 10 for s_3). If sequence s_j is formed, then $y_j = 1$ and exactly one of the binary variables $m_{i,j}^{e:s}$ corresponding to one of the outgoing edges of S_j will be equal to 1. Equation 6 is written for each node $T_j \in \mathcal{E}'$ in the modified graph. For T_j, we consider all incoming edges e_i. (See Equation 11 for s_3). If sequence s_j is formed, then $y_j = 1$ and exactly one of the binary variables $m_{i,j}^{e:s}$ corresponding to one of the incoming edges of T_j will be 1. Equation 7 guarantees that sequence s_j will form by consecutive edges e_{i1} and e_{i2} which share node $v_n \in \mathcal{V}$. (See Equation 13 for node v_4 and sequence s_3).

Equations 5-7 can guarantee that sequence s_j is formed by a set of edges initiating from a node S_j and ending at node T_j in the modified graph while covering a subset of consecutive edges in the original graph. If a sequence s_j is not formed, all $m_{i,j}^{e:s}$ variables and y_j are equal to 0.

Equation 8 helps group similar sequences and will be explained more with the constraints in the next subsection.

2) Similarity Constraints: Our goal is to group "similar" sequences to be represented by one sensor. Specifically, in the presence of process variations, two sequences are similar if they have a similar sensitivities in their variation-aware delay expression. This implies that the *changes* in the delays of the two sequences (and not their delays) are similar. So the similar sequences can be represented by one sensor and the sensor can predict the delays of the sequences.

Let us first express the delay of a sequence s_j as the summation of edge delays D_{e_i} (given in Equation 1) corresponding to edges e_i which are included in sequence s_j:

$$\begin{aligned} D_{s_j} &= \sum_{i=1}^{|\mathcal{E}|} m_{i,j}^{e:s} D_{e_i} = \sum_{i=1}^{|\mathcal{E}|} \left(m_{i,j}^{e:s} (\mu_{e_i} + \mathbf{a}_{e_i}^T \mathbf{X}) \right) \\ &= \sum_{i=1}^{|\mathcal{E}|} m_{i,j}^{e:s} \mu_{e_i} + \sum_{i=1}^{|\mathcal{E}|} m_{i,j}^{e:s} \mathbf{a}_{e_i}^T \mathbf{X} \end{aligned} \tag{14}$$

For any two sequences s_{j_1} and s_{j_2}, we can express their variation-aware delay difference as

$$\begin{aligned} D_{s_{j_1}} - D_{s_{j_2}} = & \sum_{i=1}^{|\mathcal{E}|} (m_{i,j_1}^{e:s} - m_{i,j_2}^{e:s}) \mu_{e_i} \\ & + \sum_{i=1}^{|\mathcal{E}|} (m_{i,j_1}^{e:s} - m_{i,j_2}^{e:s}) \mathbf{a}_{e_i}^T \mathbf{X} \end{aligned} \tag{15}$$

where the first term on its right-hand side (RHS) denotes the difference in nominal values and the second term represents the difference resulted from process variations. Once the sequence formation is determined (i.e., the values of $m_{i,j_1}^{e:s}$ and $m_{i,j_2}^{e:s}$ are known), we can directly compute the first term on the RHS of Equation 15. Therefore, in order to represent the sequences s_{j_1} and s_{j_2} by one sensor, we only need to pay attention to the second term on the RHS of Equation 15 which we enforce by introducing the following similarity (or sequence grouping) constraint:

$$\left\| \sum_{i=1}^{|\mathcal{E}|} (m_{i,j_1}^{e:s} - m_{i,j_2}^{e:s}) \mathbf{a}_{e_i}^T \right\|_1 \leq \epsilon \cdot m_{j_1,k}^{s:f} m_{j_2,k}^{s:f} \tag{16}$$

where ϵ is a pre-specified control parameter (as we discuss in our simulations) and $\|\mathbf{t}\|_1$ denotes L_1 norm of vector $\mathbf{t}$.

The constraint in Equation 16 is written for each combination of sequences and sensors ($\forall j_1 = 1, \ldots, N_S - 1, \forall j_2 = j_1 + 1, \ldots, N_S$, $k = 1, \ldots, N_f$). It represents that if the delay difference between $D_{s_{j_1}}$ and $D_{s_{j_2}}$ caused by process variation is negligible, we force $m_{j_1,k}^{s:e}$ and $m_{j_1,k}^{s:e}$ to be equal to 1 simultaneously. Otherwise, at least one of these two binary variables should be equal to 0. Note that the total number of constraints here is $O(N_S^2 N_f)$.

Linearization: The constraint in Equation 16 is nonlinear. Here, we discuss its linearization without approximations.

- Since $m_{j_1,k}^{s:f}$ and $m_{j_1,k}^{s:f}$ are binary variables, we can linearize the RHS of Equation 16 by replacing $m_{j_1,k}^{s:f} m_{j_2,k}^{s:f}$ using new auxiliary binary variable z and adding constraints $z \leq m_{j_1,k}^{s:f}$ and $z \leq m_{j_1,k}^{s:f}$.
- For the left hand side of Equation 16, we can introduce $|\mathbf{X}|$ auxiliary variables and $|\mathbf{X}|$ constraints to linearize it without approximations. We skip it due to the simplicity and lack of space.

In addition, Eq. 8 guarantees that if sequence s_j is formed ($y_j = 1$), then s_j is represented by exactly one sensor.

TABLE I
RESULTS FROM OUR OPTIMIZATION FRAMEWORK

Bench	#E	#Regions	#Sensors	% Area	Ave. T(min)
S1488	1277	33	54	13.4	0.26
S1494	1287	34	45	10.2	0.05
S5378	3130	123	149	10.4	0.49
S9234	2853	113	42	3.4	0.36
S13207	3433	94	84	4.6	0.53
S15850	1572	56	43	4.9	0.42
S35932	25622	1014	1457	12.0	2.79
S38417	24848	981	928	9.1	4.07
S38584	20996	548	445	5.6	0.72

3) Area Constraint: Each sensor is implemented on-chip as a sequence of gates within a specific region. The logical and layout implementation of the sensor which determines its area is identical to one of the sequences which is covered by it. Once an area is computed for each sensor, then the summation of the sensor areas should also be bounded by an area threshold A_{th} in each region. For example, we set this threshold to be minimum of a 15% budget of the core area as well as the available whitespace in the region in this work. The area budget allows controlling the whitespace for other purposes such as adding spare cells as well as the overhead for possibly adding on-chip delay measurement infrastructure.

More specifically, given constant area of edge A_i^e, we can express the area constraints as follows:

$$A_j^s = \sum_{i=1}^{|\mathcal{E}|} m_{i,j}^{e:s} A_i^e \quad \forall j = 1, \ldots, N_{\mathcal{S}} \tag{17}$$

$$A_k^f \geq m_{j,k}^{s:f} A_j^s \quad \forall k = 1, \ldots, N_f, \forall j = 1, \ldots, N_{\mathcal{S}} \tag{18}$$

$$\sum_{k=1}^{N_{\mathcal{S}}} A_k^f \leq A_{th} \tag{19}$$

Equation 17 expresses the area of sequence s_j in terms of the areas of the edges that are included in it. Equation 18 computes the area of the sensor as the maximum area of the sequences which are represented by it. Equation 19 computes the total sensor areas to be less than the given area threshold A_{th}. We observe that Equation 18 is nonlinear since non-integer variable A_j^s is multiplied by binary variable $m_{j,k}^{s:f}$. We use similar approaches illustrated in Section III-D2 to linearize it without approximations.

Overall, the IP is described by objective (Eq 2), sequence forming constraints (Eqs 4-7), similarity constraints (Eqs 8 and 16), and area constraints (Eqs 17-19).

Each instance of the described optimization formulation will be small in size and can be solved fast because of two reasons: 1) the small size of netlist as well as controlling the number of sensors and sequences inside each region; 2) small number of sensitivity parameters in Equation 16 arising from the decrease in the number of independent variations in a local neighborhood on the layout. We report a small average runtime in our simulations to solve each region.

IV. SIMULATION RESULTS

We experimented with the ISCAS89 benchmark circuits. Each instance is synthesized using a 90nm TSMC technology library and Synopsys Design Compiler for minimum area under a stringent timing constraint to ensure having many critical paths. The timing constraint is set to the case when process variations are at their nominal values. Each benchmark is placed in the timing-driven mode using Cadence.

To model post-silicon process variations, we assume variations in the Leff and Vt_0 parameters of each gate, with a Gaussian distribution and standard deviations of 10% of their mean values. To capture spatial correlations, we use the multi-level hierarchical model of [2]. The gates in the same region or in neighboring regions will share all or some of the random variables in their delay expressions and thus will be correlated to each other. This results in 42 variables for smaller benchmarks (S1488 to S9234) for a 3-level hierarchical model, and 682 variables for the larger benchmarks for a 5-level hierarchical model which is consistent with [8].

Using the above setup, we implemented our optimization framework using C++ and used CPLEX 12.0 to solve the formulation for each region. We set the parameter ϵ in Equation 16 to be 0.05 of the average gate delay in our standard cell library. We first consider the case when the edge weight $w_{e_i} = c_{e_i}$, reflecting its statistical criticality for each edge $e_i \in \mathcal{E}$. The area budget is set to 15% of the core area.

Table I shows the results in this case. For each benchmark we report the number of edges (#E), number of formed regions (#Regions), total number of sensors (#Sensors), sensor area usage (% Area) which could be completely within the available whitespace, and average runtime to solve a region. The runtime numbers are generated on a machine with a 2.8GHz Intel CPU and 12GB of memory. The average runtimes are found by dividing the total runtime by the number of partitions. This could be an indication of the wall time in the presence of more processing elements. The average number of sensors per region ranges between 0.09 and 2.02 among the benchmarks. In some regions no sensors could be inserted due to lack of space. In other regions the number of sensors was higher than this reported average.

Using these generated sensors, we then verify the post-silicon failure of a set of statistically-critical paths which are extracted following the procedure in [12]. The number of extracted paths is given in Table II (#Paths). We define a "golden model" capturing post-silicon behavior by assuming the instance of parameter variation is known and experiment by analyzing many such instances generated using Monte Carlo (MC) simulation with 50K samples reflecting various post-silicon cases. For one post-silicon case, we assume delay of a sensor is measured by plugging in the MC sample in its variation-aware delay expression, given by Eq. 1.

For each considered path p_i, we replace the delays of all the correlated sequences represented by one (or more) sensor(s) on p_i with the post-silicon sensor delays. For the remainder of the path, we compute worst-case (WC) and best-case (BC) delays when the process variables are at $\mu + 3\sigma$ and $\mu - 3\sigma$ respectively, in the variation-aware delay expression of path p_i. Error in measurement infrastructure can also be added at this point to compute the WC and BC delays.

These WC and BC delay values of the path are then used to predict if it is failing the timing constraint (T_0). For path p_i we also consider its actual post-silicon delay by plugging in the MC sample in the full variation-aware delay expression of the path. Using the predicted and actual delays of a path p_i on chip k, we predict that it fails the timing constraint, if its BC delay is higher than T_0 (denoted by $d_{p_i}^{(BC)}(k) > T_0$). Similarly, we predict that the path does not fail the timing constraint, if its WC delay is at most T_0 (denoted by $d_{p_i}^{(WC)}(k) \leq T_0$). We evaluate the rate of correctly predicting the failing and non-failing paths by reporting these metrics:

$$mr_f = \frac{\sum_{k=1}^{50K} \sum_{i=1}^{\#P} I_{d_{p_i}^{(BC)}(k) > T_0}}{\sum_{k=1}^{50K} \sum_{i=1}^{\#P} I_{d_{p_i}(k) > T_0}} \quad mr_{nf} = \frac{\sum_{k=1}^{50K} \sum_{i=1}^{\#P} I_{d_{p_i}^{(WC)}(k) \leq T_0}}{\sum_{k=1}^{50K} \sum_{i=1}^{\#P} I_{d_{p_i}(k) \leq T_0}}$$

TABLE II
COMPARISON OF FAILING AND NON-FAILING MATCH RATIOS FOR FOUR DIFFERENT CASES

Bench	#Paths	1) Without Sensors		2) Custom (w_{e_i}=c_{e_i})			3) Custom (w_{e_i}=1)			4) Pre-Specified Sequence Matching		
		mr_{nf}	mr_f	% E_{cover}	mr_{nf}	mr_f	% E_{cover}	mr_{nf}	mr_f	% E_{cover}	mr_{nf}	mr_f
S1488	22	0.64	0.00	0.06	0.97	0.62	0.14	0.82	0.00	0.13	0.94	0.05
S1494	20	0.69	0.00	0.05	0.99	0.70	0.15	0.79	0.32	0.08	0.91	0.36
S5378	1203	0.91	0.00	0.07	1.00	0.52	0.19	0.91	0.00	0.14	0.89	0.10
S9234	1305	0.58	0.00	0.02	0.98	0.96	0.16	0.86	0.86	0.03	0.63	0.08
S13207	1123	0.67	0.00	0.04	1.00	0.89	0.11	0.90	0.00	0.10	0.88	0.00
S15850	1007	0.83	0.00	0.05	0.98	0.87	0.21	0.92	0.16	0.21	0.92	0.07
S35932	1278	0.56	0.00	0.08	1.00	0.76	0.28	0.81	0.34	0.13	0.67	0.11
S38417	1105	0.51	0.00	0.05	0.92	0.92	0.17	0.71	0.48	0.09	0.80	0.05
S38584	1371	0.81	0.00	0.03	0.96	0.65	0.17	0.81	0.46	0.05	0.81	0.03

where I_E is an indicator function. If the event E holds, then I_E is equal to 1. Otherwise, it is equal to 0. The failing match ratio (mr_f) reflects the number of times that the paths are *predicted* as failing (numerator) divided by the number of times that they are *actually* failing (denominator), over the range of 50K MC samples and the extracted paths. Similarly, the non-failing match ratio (mr_{nf}) reflects the ratio of correct predictions when the paths are non-failing.

We report these match ratios for four cases in Table II: 1) when no sensors are used in which the WC and BC predicted delays of the paths are the most pessimistic and therefore the match ratios are the lowest; 2) when generating custom sensors using our framework by setting $w_{e_i} = c_{e_i}$ which is the case studied in the previous experiment; 3) same as case 2 except we use $w_{e_i} = 1$ in order to select the sensors to maximize the coverage of the edges with equal criticality; 4) custom sensors using an alternative approach by matching the entire netlist against a set of pre-specified sequences.

In case 4, we consider all the possible sequences of two logic gates in the library as potential sensors. We then traverse the netlist and apply sequence matching at each gate. We rank these potential sensors based on the number of times that they are instantiated as a sequence in the netlist and select as many top-ranked sensors, until the 15% area budget is used. Note, we do not apply layout partitioning and ignore within-die variations. The assumption of considering two gates is consistent with the average length of a sensor observed from our optimization framework in case 2.

In Table II, in addition to the match ratios, we also report the percentage of edges in the timing graph which are covered by a sensor in cases 2, 3 and 4. The failing match ratio mr_f is 0 in case 1 for all the benchmarks, indicating that the none of the failing paths can be isolated without the aid of the sensors. However, some of the non-failing paths could be accurately identified as given by the mr_{nf} column. We note that isolating the failing paths is of higher significance during post-silicon validation [1], [9].

From the table we observe case 2 offers the highest mr_f and mr_{nf} match ratios compared to the other cases. It outperforms case 3, indicating that maximizing the coverage of the graph edges with equal weights may not be the best strategy to isolate the failing and non-failing paths. The higher coverage of the graph edges in case 3 can be seen by comparing columns 5 and 8. Case 2 also consistently outperforms case 4 by offering higher match ratios, demonstrating the strength of our optimization procedure compared to using pre-specified sequences for custom sensor generation. Furthermore, the percentage of covered edges in case 3 is also always higher than case 4 (given in columns 8 and 11), reflecting the strength of our optimization framework when the goal is covering the graph edges with equal weights.

V. CONCLUSIONS AND FUTURE WORKS

We proposed a scalable optimization framework which generates custom on-chip sensors for a given netlist. It simultaneously identifies and groups sequences of logic gates in the netlist which illustrate a high correlation in their delays in the presence of within-die process variations. The on-chip sensor delays, measured at the post-silicon stage, help to predict if a set of specified paths fail the timing constraint with less pessimism. Our future plans include focusing on the physical design issues of the framework.

REFERENCES

[1] Pouria Bastani, Kip Killpack, Li-C. Wang, Eli Chiprout, Speedpath prediction based on learning from a small set of examples. In *DAC*, 2008.

[2] David Blaauw, Kaviraj Chopra, Ashish Srivastava, Louis Scheffer, Statistical timing analysis: From basic principles to state of the art. *IEEE TCAD*, 27(4), 2008.

[3] Nicholas Callegari, Dragoljub Gagi Drmanac, Li-C. Wang, Magdy S. Abadir, Classification rule learning using subgroup discovery of cross-domain attributes responsible for design-silicon mismatch. In *DAC*, 2010.

[4] Tuck-Boon Chan, Aashish Pant, Lerong Cheng, Puneet Gupta, Design dependent process monitoring for back-end manufacturing cost reduction. In *ICCAD*, 2010.

[5] Kip Killpack, Chandramouli V. Kashyap, Eli Chiprout, Silicon speedpath measurement and feedback into EDA flows. In *DAC*, 2007.

[6] Kip Killpack, Suriyaprakash Natarajan, Arun Krishnamachary, Pouria Bastani, Case study on speed failure causes in a microprocessor. *IEEE DAT*, 25(3):224–230, 2008.

[7] Qunzeng Liu, Sachin S. Sapatnekar, A framework for scalable postsilicon statistical delay prediction under process variations. *IEEE TCAD*, 28(8):1201–1212, 2009.

[8] Qunzeng Liu, Sachin S. Sapatnekar, Capturing post-silicon variations using a representative critical path. *IEEE TCAD*, 29(2):211–222, 2010.

[9] Sari Onaissi, Khaled R. Heloue, Farid N. Najm, PSTA-based branch and bound approach to the silicon speedpath isolation problem. In *ICCAD*, 2009.

[10] Xiaoxiao Wang, Mohammad Tehranipoor, Ramyanshu Datta, Path-RO: a novel on-chip critical path delay measurement under process variations. In *ICCAD*, 2008.

[11] Lin Xie, Azadeh Davoodi, Representative path selection for post-silicon timing prediction under variability. In *DAC*, 2010.

[12] Lin Xie, Azadeh Davoodi, Bound-based statistically-critical path extraction under process variations. *IEEE TCAD*, 30(1):59–71, 2011.