SlackProbe: A Flexible and Efficient In Situ Timing Slack Monitoring Methodology

Liangzhen Lai Vikas Chandra Robert Aitken Puneet Gupta

***Abstract*—In situ monitoring is an accurate way to monitor circuit delay or timing slack, but usually incurs significant overhead. We observe that most existing slack monitoring methods focus exclusively on monitoring path endpoints, which is not cost efficient from power and area perspectives.**

In this paper, we first propose *SlackProbe* methodology, which inserts timing slack monitors like "probes" at a selected set of nets, including intermediate nets along critical paths. *SlackProbe* can be used to detect impending delay failures due to various reasons (process variations, ambient fluctuations, circuit aging, etc.) and can be used with various preventive actions (e.g., voltage/frequency scaling, clock stretching/time borrowing, etc.). Then we perform thorough analysis of the potential benefits and caveats of *SlackProbe* over conventional approaches in terms of number of monitors required, monitoring efficiency and observability, delay margin, and design perturbation. Experimental results on commercial processors show that with 5% extra timing margin, *SlackProbe* can reduce the number of monitors by 12-16X as compared to the number of monitors inserted at path ending pins. *SlackProbe* can also improve the monitoring efficiency by up to 1.9X and improve the monitoring observability by up to 32%, as compared to endpoint monitoring.

***Index Terms*—timing, average case design, delay testing, low-power design, network flow algorithm**

I. Introduction

With increasing amounts of manufacturing variability, ambient fluctuation and circuit wear-out (e.g., NBTI, HCI etc.), it is necessary to identify chip delay either statically (e.g., speed binning) or dynamically with both hardware and software adaptive schemes [1]. There are various classes of monitors that are targeted at measuring circuit path delay.

Canary or replica circuits [2], [3] are stand-alone circuits which are intended to mimic the timing behavior of the original circuits. The delay of the real circuit can be estimated through measuring delay of the replicas. Tunable [4] and design-dependent [3] replica circuits can reduce the mismatch of the real circuit and replica. Replica monitors are usually non-intrusive, but may fail to capture the variations that are local to real circuits such as random manufacturing variations and circuit aging.

In situ monitors measure the delay directly from the circuit paths. Fick et al. [5] use a Time-to-Digital Converter (TDC) to measure the critical path delay. Wang et al. [6] measure delay by reconstructing the critical path as Ring-Oscillators (ROs). Another approach to measure circuit path delay is to measure the timing slack. Since critical paths typically end at registers, special flip-flops can be used as slack monitors. Razor [7], [8] uses customized flip-flops to detect timing failures due to setup time violation and correct them through a pipeline flush or architectural replay. Similar approaches that reduce timing margin, but not to the point of failure, include delaying data signals [9], advancing clock signals [10] or using different flip-flop structures [11]–[15].

In situ monitors can accurately capture the real path delay, but with significant overhead, especially when large number of registers are timing critical. Some methods can be used to reduce the overhead (e.g., [16]), but with a loss in accuracy. Better monitor designs, e.g., [17], can also reduce the overhead, but are still fundamentally limited by the large number of monitors requited. We observe that existing methods focus *exclusively* on monitoring path endpoints (i.e., destination registers). In this work, we propose *SlackProbe*, a low overhead, in situ, on-line timing slack monitoring methodology. *SlackProbe* monitors in situ timing slack of selected circuit nets, including intermediate nets along circuit paths, which is more power and area efficient. This paper is an extension of [18]. The key contributions of our paper include the following:

1) We propose a novel slack monitoring methodology allowing placing monitors at intermediate nets along circuit paths
2) We develop a metric named "*opportunism window*", which allows us to flexibly select the set of critical paths to be monitored.
3) We formulate and convert the path-based monitor insertion formulation into a edge-based Linear Programming (LP) problem and solve it near its theoretical lower bound
4) We perform a thorough analysis of the potential benefits and caveats of such a monitoring methodology, including monitoring cost, monitoring efficiency and observability, timing margin and design perturbation.

The rest of the paper is organized as follows: Section II gives an overview of the proposed monitoring methodology. Section III describes the critical path selection and circuit graph reduction process. Section IV presents the monitor insertion problem formulation and solution. Section V discusses the monitor cost metrics and comparison between *SlackProbe* and conventional approaches. Section VI presents

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

V. Chandra and R. Aitken are with ARM Inc., San Jose, CA 95134 USA. e-mail: (Vikas.Chandra@arm.com; Rob.Aitken@arm.com).

This work is supported in part by NSF Variability Expedition grant CCF-1029030.

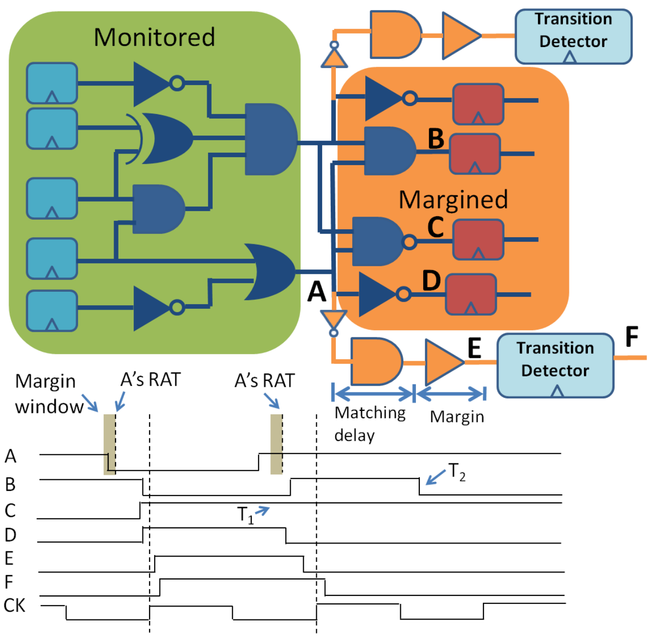


Fig. 1. *SlackProbe* working principle. As shown in the timing diagram, compared to inserting monitors at destination registers, the monitor inserted at A can monitor the path delay even when the transition does not propagate to the destination register (i.e., T_1 at C). But the monitor inserted at node A cannot capture transitions that do not pass through A (i.e., T_2 at B).

the experimental results. Section VII concludes the paper.

II. *SlackProbe* Overview

A. Monitor Working Principle

The monitor working principle is shown through an example in Fig. 1. If a monitor is inserted at an intermediate node A, a "probe", which consists of delay matching gates and a transition detector, is connected to A through a minimum size inverter. Signal transitions at node A are transferred through the delay chain to the transition detector and compared with the incoming clock edge. If the transition is close to its Required Arrival Time (RAT), i.e., within the margin window as in Fig. 1, a corresponding signal transition will arrive at node E after the clock edge. This triggers the transition detector and flags a signal indicating an impending delay failure.

The monitor inserted at node A is capable of monitor the delay of all critical paths passing through A. As shown in Fig. 1, instead of monitoring all four destination registers, *SlackProbe* can use only two monitors while achieving the same path coverage.

Different transition detector designs as in [10], [19]. can be applied here. *SlackProbe* also allows monitors to be inserted at path endpoints where monitors as in [7]–[9], [11] can be used as well. Since the additional margin makes the monitor detect an impending timing failure rather than an actual one, there is no datapath metastability issue as raised and discussed in [19]. The metastability issue of the monitor signal either results in a more pessimistic detection or is guardbanded by the monitor delay margin.

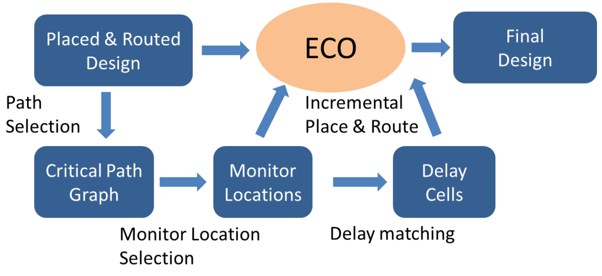


Fig. 2. Monitor insertion flow

B. Monitor Insertion Flow

With the proposed monitoring strategy, the problem now becomes *when*, *where* and *how* to insert these monitors. In this work, we propose the monitor insertion flow as in Fig. 2. Different alternatives will also be discussed and compared against conventional approaches.

Monitor insertion starts with a placed and routed design, as the timing information is more accurate at this stage. Since we only care about timing-critical paths, a path selection process is applied to extract timing-critical paths and to construct the critical path graph. Then, monitor locations are picked from the graph using our proposed method. For each of the monitor locations, a delay cell path is synthesized. The final insertion flow is similar to Engineering Change Order (ECO) where the monitors are incrementally placed and routed. ECO metrics like those in [20] are applied when picking monitor locations to minimize the perturbation to the original design.

C. Possible Applications of the Monitors

Since different applications will have different requirements for the monitor, we discuss possible monitor applications here as examples before introducing the detailed implementation flow.

One possible application is to use the monitors as timing-failure event predictors and combine them with some of the existing error resilience mechanisms like in [8], [21], [22]. In this case, the monitors have to capture all signal transition events that may result in timing errors.

Another possible application is to use the monitors as speed sensors which indicate whether current operation condition is close to possible timing failure or not. This can be used by systems with adaptation capabilities like Adaptive Voltage Scaling (AVS), Adaptive Body Bias (ABB) or Dynamic Voltage and Frequency Scaling (DVFS) to account for manufacturing variations, ambient conditions as well as circuit aging effects such as Negative Bias Temperature Instability (NBTI), Positive Bias Temperature Instability (PBTI) and Hot-Carrier Injection (HCI). Since variations are either static or changing slowly, the monitor requirements can be relaxed to capture only the delay changes instead of all transition events.

III. Path Selection and Graph Reduction

A. Path Selection Criteria

As shown in Fig. 2, given a placed and routed design, the first step is to identify the part of the design that may

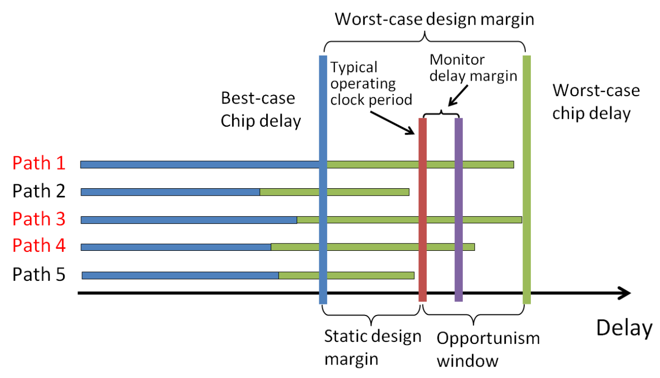


Fig. 3. Opportunism window is the margin saving comparing to worst-case design. Monitor delay margin is the delay margin of the delay matching chain which will be discussed in detail in Section IV-A.

be timing critical. Depending on the application, different criticality criteria may be applied for the selection process.

In this work, we propose a flexible path selection method by introducing user-defined *opportunism window*. As illustrated in Fig. 3, *opportunism window* and monitor margin (discussed in detail in Section IV-A) will dictate the potential monitoring benefits. Typical worst-case design sets its delay margin for the worst-case scenario, i.e., all chips will run at a frequency slow enough for the worst-case chip delay regardless of what the actual delay is. In the presence of monitors, we may reduce the design margin and decrease the default operating clock period. The paths whose worst-case delay exceeds the default operating clock period should be selected and monitored. The amount of design margin reduction is called *opportunism window*, within which the circuit will operate opportunistically at its best effort. The size of *opportunism window* determines the total number of circuit paths to be monitored, thus affecting the monitoring overhead. There is a natural trade-off between monitoring benefits and monitoring overhead when deciding the *opportunism window* and monitor delay margin. We will discuss and explore this trade-off in later sections and experiments.

This path selection method does not require any knowledge of correlation in the variations between the delay of different paths. Therefore, it can be used to select paths for applications like aging sensors, where exact delay degradations are context dependent with little pre-assumed correlation.

Another possible path selection method is based on the process corners as described in [18], where paths are selected based on their timing slacks at particular process corners. Other path selection methods such as statistical methods [23] can also be used in *SlackProbe* depending on the specific applications.

B. Circuit Graph Reduction

Although *opportunism window* is a path-based criteria, we can utilize the block-based property of Static Timing Analysis (STA) and construct a circuit graph which contains all the selected critical paths.

If a path is selected as timing critical as illustrated in Fig. 3, the path's worst-case delay falls into the *opportunism window*. Equivalently, if we run STA using the worst-case corner library, it will also imply that the path's timing slack at the worst-case corner is smaller than the size of *opportunism window*. Based on the following two key properties of STA:

1) The slacks of all pins along a path are equal or smaller than slack of the path
2) The slack of a path equals to the largest slack of all pins along that path

We can obtain a reduced circuit graph by removing the pins with slacks larger than the size of *opportunism window* and then removing the unconnected gates/nets. The critical paths in the original circuit graph are still preserved in the reduced graph because of property 1. All paths in the reduced graph are timing critical because of property 2. Therefore, we can work on the reduced graph instead of the original circuit graph when analyzing the monitor insertion. Unless otherwise specified, circuit graph discussed in the rest of the paper refers to the reduced graph.

IV. Monitor Location Selection

In this section, we describe our proposed monitor location selection method. We first discuss how monitors interact with different paths. Then we formulate the monitor location selection problem and describe our proposed solution.

A. Monitor Coverage and Delay Margin

After selecting the critical paths and obtaining the reduced circuit graph, *SlackProbe* will determine the set of candidate locations for monitor insertion. Before analyzing the location selection problem, we first discuss how paths are monitored by the inserted slack monitors.

1) Delay Margin for Paths: If the monitor is placed at some intermediate net, the path delay before the monitor can be captured by the monitor. But some extra delay margin will be required for the remaining part of the path. As shown in Fig. 4, there are three types of relations between monitor and a path:

1) The path passes through the net, for example path A in Fig. 4. Since the delay up to G4 can be captured by the monitor, the delay path should account for the delay uncertainty of G6.
2) The path branches out at some net before the monitor, for example path B in Fig. 4. Depending on the application and gate type of G4, the monitor may be treated as being inserted between G3 and G5 with G4 as part of the delay matching. If the application is speed sensing, path B can be considered as being monitored with delay uncertainty of G4, G5 and G8. If the application is event detection, only G4 with gate types that are transparent to signal transitions (e.g., inverter, buffer) are allowed.
3) None of the path instances fall in the fan-in cone of the monitor net. In this case, we consider that the path is not monitored.

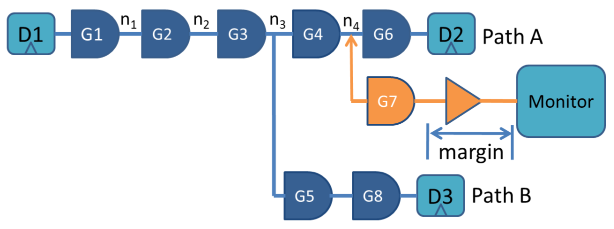


Fig. 4. Example of path-monitor pairs

2) Delay Margin for Monitors: Although different paths may require different delay margin, each monitor will have only one margin matching chain. The monitor margin should account for worst delay uncertainty after monitor insertion point and guarantee that the delay chain is always slower than margined part of monitored circuit paths.

In the example in Fig. 4, the delay margin should account for the mismatch between original path (i.e., from n_4 to D_2) and delay matching chain (i.e., from n_4 to the monitor). Theoretically, the best case delay of the delay matching chain should match the worst case delay of the original path. But this may be too pessimistic since the delay is likely to be correlated. Similar to on-chip variation modeling, in this work the delay chain is designed so that its delay at typical process corner matches the worst case delay of the original path. In Fig. 4, the equivalent delay margin in this case equals the delay of $G6$ at slow process corner (i.e., delay of the delay matching chain at typical process corner) minus the delay of $G6$ at typical process corner. This margin is considered as the delay uncertainty of $G6$.

The required delay margin for each circuit nets can be calculated based on conventional STA tool flow. We first perform timing analysis using typical process corner libraries and obtain the arrival time d_i^{tt} and timing slack s_i^{tt} for each net. Then we perform timing analysis using slow process corner libraries and obtain the arrival time d_i^{ss} and timing slack s_i^{ss} as well. The required delay margin can be derived by:

$$Margin_i = (s_{tt} - s_{ss}) - (d_{ss} - d_{ss}) \tag{1}$$

Methods as in [3] or [4] can also be applied to synthesize a replica-like path to reduce the margin for delay mismatch due to global variation.

3) Overall Margin: Because the final delay margin for the entire circuit will be dominated by the monitor with the largest margin, we define the delay margin cost as the maximum monitor delay margin constraint ϵ on each monitor. The delay margin ϵ is used to determine the set of feasible monitor candidate locations and the corresponding circuit nets that can be monitored.

For example, for a given ϵ, we can analyze the implication of inserting a monitor at a net n_i. If the margin required by n_i is smaller than ϵ, all paths passing through n_i will be monitored. Depending on the application, we may also want to consider another net n_j in the fan-in cone of n_i (like n_3 in Fig. 4). If the margin required by paths branching out at n_j is also smaller than ϵ (like path B in Fig. 4), n_j can also be monitored by the monitor at n_i. All paths passing through n_j will be monitored. Therefore, we can define a set I_{n_i} as the nets that can be included as being monitored by the monitor at n_i.

If we represent a path as the set of nets it passes through, the timing margin constraint can be stated as: with a given monitor delay margin constraint ϵ, for any critical path P_k, there exists a monitor at net n_i, such that $P_k \cap I_{n_i} \neq \emptyset$. For different candidate locations, monitor insertion may have different cost in terms of power, design perturbation etc. Here we use c_i as the cost associated with net n_i. The detailed derivation of c_i will be discussed in Section V.

B. Problem Formulation

For a given circuit, a graph can be constructed by making the nets as nodes $\mathbf{N} = [n_1 n_2 ...]^T$ and interconnect as directed edges. We denote $\mathbf{x} = [x_1 x_2 ...]^T$ as the decision vector where $x_i = 1$ when a monitor is inserted at n_i and 0 otherwise.

Since inserting a monitor at n_j implies that all nodes in I_{n_j} are monitored, we define set $O_{n_i} := \{n_j | n_i \in I_{n_j}\}$. A vector $\mathbf{y} = [y_1 y_2 ...]^T$ can be derived as:

$$y_i = \sum_{n_j \in O_{n_i}} x_j \tag{2}$$

So $y_i \geq 1$ implies that n_i is monitored because there exists some $x_j = 1$ and $n_i \in I_{n_j}$. Equation (2) can be represented as matrix representation $\mathbf{y} = \mathbf{Q}\mathbf{x}$.

A critical path matrix can be generated from the circuit graph as:

$$\mathbf{P} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots \\ p_{2,1} & p_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} where\ p_{k,i} = \begin{cases} 1 & if\ n_i \in P_k \\ 0 & otherwise \end{cases} \tag{3}$$

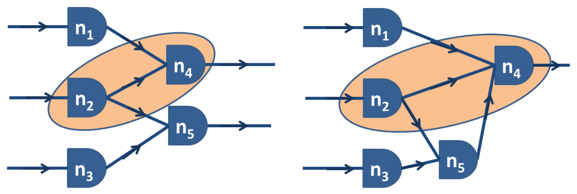
Assuming that the cost vector is $\mathbf{c} = [c_1 c_2 ...]^T$ and $\mathbf{1}$ is a row vector of 1's, the monitor insertion problem is formulated as follows:

$$\begin{aligned} \text{minimize:} \quad & \mathbf{c}^T \cdot \mathbf{x} \\ \text{subject to:} \quad & \mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{x} \geq \mathbf{1} \\ & x_i \in \{0, 1\} \end{aligned} \tag{4}$$

C. Problem Conversion

Solving (4) directly is computationally intractable for any reasonable size of circuit because of the large number of paths. We will convert the problem into a solvable edge-based problem.

Since all entries in $\mathbf{P}$ and $\mathbf{Q}$ are either 0 or 1, entries in $\mathbf{P} \cdot \mathbf{Q}$ are non-negative integers and some entries may be larger than 1. This is not necessary for $\mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{x} \geq \mathbf{1}$ given that $x_i \in \{0, 1\}$. We can derive a new matrix $\mathbf{A}$ with $a_{k,i}$ equals 1 if corresponding entry in $\mathbf{P} \cdot \mathbf{Q}$ is non-zero, and 0 otherwise. The new constraint $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{1}$ is equivalent to $\mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{x} \geq \mathbf{1}$ for the optimization problem in (4). Therefore, we can replace $\mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{x} \geq \mathbf{1}$ with $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{1}$ for (4).



(a) Treat I_{n_4} as a node (b) Example of cyclic flow

Fig. 5. Example of treating the nets in $I_{n_4} := \{n_2, n_4\}$ as a node. Cyclic flow as in (b) is impossible because margin at n_2 must be larger than margin at n_5. In (b), $n_2 \in I_{n_4}$ implies $n_5 \in I_{n_4}$.

Then we relax the constraint on x_i as $x_i \geq 0$. [1] This gives us an LP problem with its dual as:

$$\begin{aligned} \text{maximize:} \quad & \mathbf{1}^T \cdot \mathbf{f} \\ \text{subject to:} \quad & \mathbf{A}^T \cdot \mathbf{f} \leq \mathbf{c} \\ & \mathbf{f} \geq 0 \end{aligned} \tag{5}$$

If we add two dummy nodes n_s and n_t as the beginning and ending nodes for all paths, (5) becomes similar to but not exactly the same as a maximum flow problem with $\mathbf{f}$ being the dedicated flow along each path.

Converting (4) into the LP problem (5) still does not reduce the problem size. To make the problem solvable, we will convert (5) into a formulation with edge-based variables and constraints. We denote $\mathbf{e} = [\ldots\ e_{i,j}\ \ldots]^T$ as the total network flow on the edges. $e_{i,j}$ equals the sum of all path flow that goes from n_i to n_j and satisfies the flow conservation constraint. Since all path flow starts from the source node n_s, the objective in (5) is equivalent to maximizing the total flow out from n_s.

If we look at the i-th row of the constraint $\mathbf{A}^T \cdot \mathbf{f} \leq \mathbf{c}$ in (5), on the left-hand side is the sum of all path flows that pass through nodes in I_{n_i}. The key enabling observation is that I_{n_i} is defined with respect to the delay margin, which is monotonic in topological order. Therefore, all nodes in I_{n_i} are connected (see example in Fig. 5(a)). If we group I_{n_i} as one single pseudo node in the graph, there will be no cyclic flow paths that exit I_{n_i} and enter again (see example in Fig. 5(b)). All paths that pass through nodes in I_{n_i} will pass through one and only one of the edges that goes into I_{n_i}. In the example in Fig. 5(a), the path flows that pass through I_{n_4} are $\{n_1, n_4\}$, $\{n_2, n_4\}$ and $\{n_2, n_5\}$, of which the sum equals all the edge flow that goes into I_{n_4}, i.e., $e_{1,4} + e_{s,2}$. Therefore, we can replace the left part of the constraint with the sum of all incoming edge flow of pseudo node I_{n_i}.

The converted edge-based formulation becomes:

$$\begin{aligned} \text{maximize:} \quad & \sum_{\forall i} e_{s,i} \\ \text{subject to:} \quad & \forall n_i, \sum_{\forall j} e_{j,i} = \sum_{\forall k} e_{i,k} \\ & \forall n_i, \sum_{n_j \notin I_{n_i}, n_k \in I_{n_i}} e_{j,k} \leq c_{n_i} \\ & \mathbf{e} \geq 0 \end{aligned} \tag{6}$$

[1]constraint $x_i \leq 1$ is not necessary in this case because monitor cost $\mathbf{c} > 0$ and $a_{k,i} \in \{0, 1\}$

In this edge-based LP formulation, the number of variables equals m_E and the number of constraints equals $m_E + 2m_N$, where m_E is the total number of edges and m_N is the total number of nodes. For all our benchmarks, solving (6) takes less than a minute.

D. Problem Solution

Because of the relaxation on x_i, the result of (6) will be a lower bound of that in (4).

To derive an exact solution of the monitor locations, we take the solution of (6) and extract out the set of all nets n_i with the edge flow into I_{n_i} equals c_i. This set will be a valid solution which satisfies constraints in (4). Then we identify the highest cost net that is unnecessary to maintain the constraint and delete it. This process is repeated until no more nets can be deleted. Our experiments show that in most of the cases, we can get the result that is close to its lower bound.

V. Monitor Cost Metrics and Comparison Analysis

In this section, we discuss monitor cost metrics and perform a thorough comparison between *SlackProbe* and conventional approaches of monitoring at path endpoints only.

A. Monitor Power Cost

Since the monitors are inserted as ECO, there is no direct area overhead added to the original design, but the monitors will introduce additional power overhead. Different monitoring locations have different power overheads because circuit nets have different signal switching probabilities. The length of the matching delay chain will also affect the power consumption. Therefore, power overhead can be modeled as:

$$p_o + (\lambda_i p + l) d_i \tag{7}$$

where λ_i is the signal switching probability of net n_i, p is the estimated dynamic power overhead per unit matched delay, l is the estimated leakage power overhead per unit matched delay, d_i is the delay of the path that is going to be synthesized for the monitor at n_i, and p_o is the static power overhead of the transition detector, which includes additional clock power and leakage power. The derivation of λ_i will be discussed later in Section V-D.

B. Layout Interference Cost

The monitor insertion is considered as ECO. ECO cost such as layout disturbance and timing disturbance should be considered. Layout disturbance can be evaluated by taking local layout congestion. To minimize the timing disturbance, the monitor uses a minimum size inverter to "probe" at the monitoring nets. The estimated timing slack after the inverter insertion can be used for timing disturbance evaluation. Similar to [20], we use a linear model to evaluate the design perturbation cost of inserting a monitor at net n_i as:

$$a_t \cdot exp(-1 \times s_{n_i}) + a_r \cdot r_{n_i} \tag{8}$$

where s_{n_i} is the estimated timing slack after inserting the inverter, r_{n_i} is the layout congestion around n_i, a_t and a_r are weighting parameters for different cost considerations.

C. Implementation Flow Comparison

The proposed implementation flow of *SlackProbe* was described in Section II-B and Fig. 2, of which the starting point is a placed and routed design and the monitors are inserted as additional circuit through ECO, while conventional endpoint monitoring approaches, (e.g., [10], [22]), decide the monitor locations at an earlier design phase and insert monitors by replacing the endpoint flip-flop cells.

Typically, conventional approaches determine the timing-critical flip-flops after synthesis and replace them with the monitor flip-flops. During later place and route, different sets of timing constraints are applied separately to regular flip-flops and the monitor flip-flops. The timing constraints are set to make sure that only paths ending at the monitor flip-flops can be timing critical under variations. This approach has relatively simpler physical implementation, but may unnecessarily speed up unmonitored paths and result in a sub-optimal design. While the approach as in Fig. 2 preserves the optimized design, but incurs more complicated physical implementation and design perturbation. Although both approaches can be applied to *SlackProbe*, the analysis and solution in this work is based on the the approach as in Fig. 2.

D. Monitoring Efficiency and Observability

Other than the implementation flow, the monitoring efficiency and observability is another important difference between *SlackProbe* and conventional approaches. In situ monitor relies on timing-critical signal transitions along critical paths to monitor delay changes. Depending on the circuit structure and logic function, monitors inserted at different locations will encounter different sets of signal transitions and different number of timing-critical signal transitions.

As shown in Fig. 1, all signal transitions at the inserted net will trigger the monitor. But only timing-critical signal transitions can be used to detect impending timing failures. So the monitoring efficiency is determined by the proportion of signal transitions that are timing critical. Compared to conventional approaches, *SlackProbe* identifies and inserts monitors at more timing-critical parts of the circuit. It is expected to have a larger fraction of timing-critical transitions and thus have higher monitoring efficiency.

The other difference between *SlackProbe* and conventional approaches is the monitoring observability. Due to logic masking, signal transitions passing through intermediate nets may be masked by side inputs of the logic gates and cannot propagate to the path endpoint. In this case, the monitor inserted at intermediate net will detect the signal transition and use it to detect potential timing failure due to any instances along that path, while the monitor inserted at endpoint will not see the transition. For instance, in the example shown in Fig. 1, the signal at node A switches from logic low to logic high during the second clock cycle. This signal transition can be detected by the monitor inserted at node A. Due to logic masking, however, this signal transition cannot propagate to path endpoint at node C (shown as $T1$ in Fig. 1). Therefore, a monitor inserted at path endpoint node C will miss this event and cannot use it to detect impending delay failures. Since all transitions at the path endpoints imply transitions at the intermediate nets but not vice versa, *SlackProbe* can detect more signal transitions, thus is expected to have higher monitoring coverage than conventional approaches. In this work, we define the monitoring observability as the average number of monitored nets per cycle.

In situ monitor relies on timing-critical signal transitions to monitor the critical path delay. For system that uses in situ monitor as speed sensors, this may limit the capability of monitoring dynamic variations as non-deterministic lag exists between the time when dynamic variation happens and the time when variation is captured (as opposed to some replica monitors [24], [25] which are used to monitor fast-varying variation). Therefore, higher monitoring observability means higher monitoring confidence and allows the system to monitor more fast-varying variation and achieve faster actuation response. For example, in the case of adaptive system, higher monitor confidence implies faster adaptation response (e.g., voltage/frequency adaptation) and less design margin for the reaction time. One exception is a razor-like system with recovery mechanisms as in [7], [22]where the purpose of monitoring is to capture real timing failures. Higher observability means more recovery events even though there may not be real timing errors (i.e., false positive). In this case, the unnecessary recovery may increase the overhead.

Since different circuit nets will have different probabilities of seeing timing-critical transitions, monitor location selection will affect the monitoring efficiency and observability. In order to evaluate the monitoring observability, other than the signal switching probability λ_i, we need to extract the signal critical switching probability γ_i for each node n_i.

Timing-aware verilog simulation can be used to extract γ_i, but is much slower than a pure functional simulation. In this work, we make an approximation based on the properties of the reduced graph as described in Section III-B. by assuming that a critical path/subpath is activated if there is an ordered sequence of switches for all nets along the path/subpath. To extract γ_i, we first run functional verilog simulation on the gate-level netlist and obtain the Value Change Dump (VCD) file. Then we apply our proposed algorithm (see Algorithm 1) for each simulated cycle and identify the sets of nodes in the reduced graph with regular switches and with timing-critical switches.

To obtain the switching probability λ_i and critical switching probability γ_i, we keep two counters for each node to record the total number of regular switches and timing-critical switches. For each simulated cycle, we increment the counters if corresponding flag is set. The corresponding switching probabilities will be the counter output divided by the total number of simulated cycles.

E. Clock Matching

Another difference brought by inserting monitors at intermediate nets is monitored flip-flop clock matching. Since *SlackProbe* can insert monitors at intermediate nets, monitors may be used to monitor multiple flip-flops. Depending on their relative locations in the clock tree, these flip-flops may

Algorithm 1 γ_i derivation

```
Input: VCD output for current simulation cycle
Output: $A[m_N]$ switching flags
$B[m_N]$ critical switching flags
for all $n_i$ do
    unset $A[i]$ and $B[i]$
end for
for each $n_i$ in VCD that belongs to the reduced graph do
    set $A[i]$
    if $n_i$ has no fan-in then
        set $B[i]$
    else if $B[j]$ is set for any $n_i$'s fan-in node $n_j$ then
        set $B[i]$
    end if
end for
```

have different potential clock skew variations, which should be accounted for by monitor delay margin in *SlackProbe*. Conventional approaches that use one monitor per flip-flops do not have the same issue.

For *SlackProbe*, the monitors sharing same monitor should have small distance on the logic path graph. We expect their distance on the clock tree to be small as well. To evaluate and verify this, we include clock matching checking feature in *SlackProbe* implementation. For each of the monitors, we traverse downwards on the circuit graph and record the set of all downstream flip-flops and their common ancestor on the clock tree. In all experiments we ran, the maximum clock delay from the flip-flops' common ancestor to the flip-flops is at most 5% of the corresponding clock period. Therefore we expect clock matching to have little impact on *SlackProbe* delay margin.

F. Overall Monitor Insertion Cost Function

In [18], a cost function c_i for each net n_i is defined, and combines the monitor power cost and design perturbation cost as:

$$c_i = p_o + (\lambda_i p + l)d_i + a_t \cdot exp(-1 \times s_i) + a_r \cdot r_{n_i} \tag{9}$$

where parameters are chosen for correct normalization. In this paper, we derive a new cost function with monitor efficiency and observability awareness by normalizing the dynamic power cost with the critical switching probabilities γ_i as:

$$c_i = p_o + (\frac{\lambda_i}{\gamma_i} p + l)d_i + a_t \cdot exp(-1 \times s_i) + a_r \cdot r_{n_i} \tag{10}$$

where same parameter normalization is applied. In this cost function, power cost is calculated based on the power consumption per monitoring event, which represents the actual monitoring cost.

Monitor cost breakdown plots for a sample circuit node are shown in Fig. 6. The circuit node considered in Fig. 6 has $\lambda_i \approx 0.7$, $\gamma_i \approx 0.07$, $d_i \approx 0.1$ ns for both rising and falling transitions, $s_i \approx 0.05$ ns for both rising and falling transitions, neighborhood cell area utilization of about 0.8 and neighborhood routing utilization of 0.8. As discussed earlier, most but not all applications prefer higher monitoring observability. In the experiments, we will use the cost function in Equation (9) as our default cost function. The cost function in Equation (10) will be applied to show the improvement of *SlackProbe*'s monitoring efficiency and observability over conventional approaches.

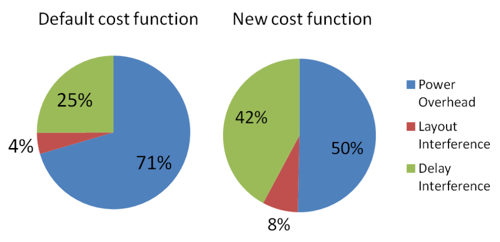


Fig. 6. Monitor cost breakdown examples for a sample circuit node.

VI. Experimental Results

A. Experiment Setup

To evaluate the effectiveness of our monitoring methodology and problem solution, we use three commercial processor benchmarks and implement them using a commercial sub-32nm process technology and libraries. The timing libraries are all at 0.81 V. The implementation is done using Cadence toolchain [26]. The implementation information is listed in Table I. Gate-level Verilog simulation and Algorithm 1 is used to obtain the net switch probabilities and the critical switching probabilities. In this work, we use Dhrystone [27] benchmarks for all three processors. For the input/benchmark dependence, relevant analysis and discussion can be found in [28].

Since different applications will have different requirements for the monitors, we have implemented three different monitor insertion methods:

- *Baseline:* This is the referenced baseline method which inserts a monitor at every critical path endpoint.
- *SlackProbe Event Detection:* This method aims at event detection. It allows the monitors to be placed at path intermediate nets. To ensure the detection of all switching events, it allows including additional nets in I_{n_i} only when they are connected to the monitor through inverter or buffer.
- *SlackProbe Speed Sensing:* This method aims at speed sensing. It allows the monitors to be placed at path intermediate nets and allows including nets in the fan-in cone regardless of the gate type.

B. Results on Different Processor Benchmarks

For this experiment, the path selection is done through the opportunism window approach. We define the typical operating clock period as the clock period reported by timing analysis with typical process corner libraries. For example, in the case of Processor A as shown in Table I, the opportunism window size is the clock period difference between typical corner and slow corner, i.e., 0.22 ns. Delay margin ϵ is set to be 5% of the typical operating clock period. Table II summarizes

TABLE I
IMPLEMENTATION INFORMATION OF THE PROCESSOR BENCHMARKS

Processor	A	B	C
Gate count	10434	30296	58815
Register count	1191	3344	9516
Clock period at typical corner	1.01 ns	1.22 ns	1.43 ns
Clock period at slow corner	1.23 ns	1.48 ns	1.72 ns
Critical gate count	7246	21385	21630
Critical register count	852	2116	4195
Critical path endpoint count [2]	1256	2637	4993

TABLE II
EXPERIMENTAL RESULTS ON THE PROCESSOR BENCHMARKS

	Processor	A	B	C
Baseline	Monitor count	1256	2637	4993
	Normalized cost	12.16	8.95	14.05
SlackProbe Event Detection	Monitor count	148	480	510
	Normalized cost	1.34	1.69	1.57
	Normalized cost lower bound	1.34	1.59	1.53
SlackProbe Speed Sensing	Monitor count	113	311	374
	Normalized cost	1	1.07	1.08
	Normalized cost lower bound	1	1	1

the experimental results for the methods on different processor benchmarks. The total monitor cost are normalized with respect to the lower bound cost of *SlackProbe Speed Sensing*. In all three benchmarks, by allowing inserting monitors at path intermediate node and extra delay margin, the total number of monitors is reduced by almost an order of magnitude.

C. Impact of Opportunism Window

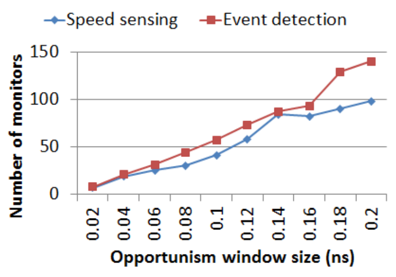
In order to evaluate the monitoring benefit/overhead trade-off, we pick processor A and apply different path selection criteria (i.e., different opportunism window sizes) to it.

Fig. 7 presents the results of different path selection criteria. For each case, the timing margin ϵ is kept at 5%. The paths are selected as critical if their delay at slow process corner falls into the corresponding opportunism window. The monitor count comparison of *SlackProbe* and baseline is shown in the log-scale plot in Fig. 8. Compared to the baseline method, our methods show on average 12X reduction in the number of monitors for event detection and 16X for speed sensing over all cases.

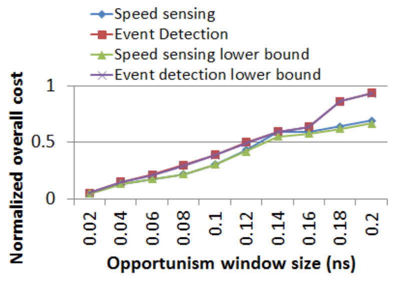
D. Results on Different Monitor Delay Margin

To show the trade-off between delay margin and monitor count, we also sweep the delay margin ϵ for processor A and plot the corresponding monitor count and monitor cost in Fig. 9. The path selection criteria are the same as that in Table II (i.e., defining the opportunism window with typical operating clock period obtained from typical process corner libraries). By allowing more timing margin, the number of monitors decreases for both methods. We also tried different weighting parameters of the monitor insertion cost (i.e., a_t, a_r etc.). Since monitor location selection depends more on the circuit topology, weighting parameters do not significantly

[2]The path endpoints include the circuit primary outputs as well. Some flip-flops use both D and scan-in pins multiplexers to select different data inputs. They are treated as different endpoints here.



(a) Monitor count



(b) Monitor cost

Fig. 7. Monitor count and cost for processor A with different opportunism window size

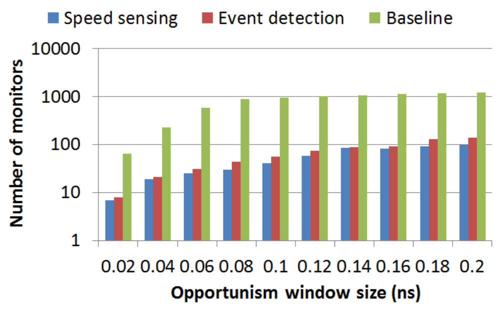


Fig. 8. Monitor count comparison between *SlackProbe* and baseline. The y-axis is plotted in log scale.

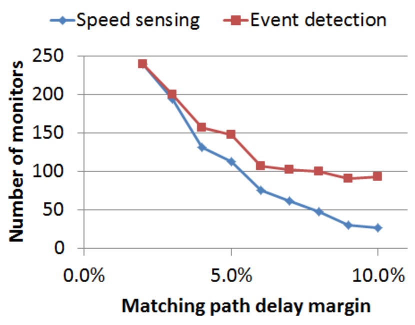
change the total number of monitors. However, they do affect the monitor locations locally, especially for the speed sensing case, where different monitor location candidates can have similar critical path coverage but different monitor insertion cost.

In all experiments, our proposed solution achieves results equal or very close to the theoretical lower bound.

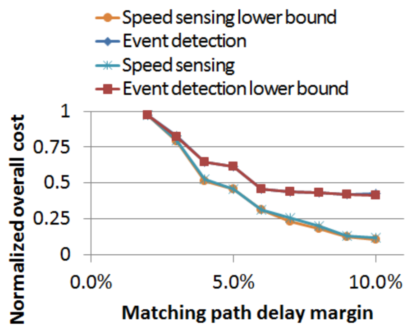
E. Extrapolation for Low Voltage Scenarios

In the experiments, timing libraries at 0.81 V is used. A lower supply voltage can potentially increase the total amount of variation. A sample circuit path delay at slow corner with different supply voltages is shown in Fig. 10. When the supply voltage decreases from 0.8V to 0.7V, the delay difference between typical corner and slow corner increases by about 2X (approximately from 25% to 50% when normalized with the delay at typical corner).

The magnified variation will scale up the worst-case design margin (see Fig. 3) by a similar amount. This has two impacts for *SlackProbe* under the same design setup. First, for the



(a) Monitor count



(b) Monitor cost

Fig. 9. Monitor count and cost vs. delay margin for processor A

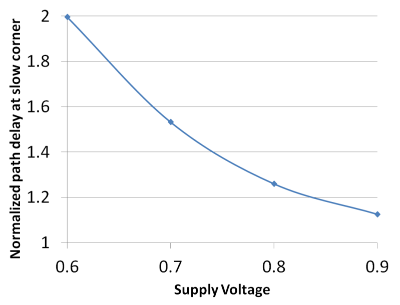


Fig. 10. Delay of a circuit path at slow corner. All path delay numbers are normalized to the corresponding delay at typical corner. The variation has larger impact at lower supply voltage.

same set of monitored paths, the equivalent opportunism window size will increase with magnified variation, which will increase the potential monitoring benefit and makes the monitoring scheme more viable. Second, the same monitor candidate location will require more delay margin. Therefore, with the same amount of delay margin (ϵ), the set of monitor candidate locations will be less, which results in similar effect as reducing ϵ as discussed in Section VI-D and in Fig. 9. This will increase the number of monitors and reduce the corresponding advantage of SlackProbe over conventional endpoint monitoring schemes.

F. Monitoring Efficiency and Observability

In order to evaluate and compare the monitoring efficiency and observability of *SlackProbe* and conventional approaches, we pick processor A and simulate it with Dhrystone [27] benchmarks. For each simulated cycle, we use the proposed method as in Algorithm 1 to identify the set of nets with signal

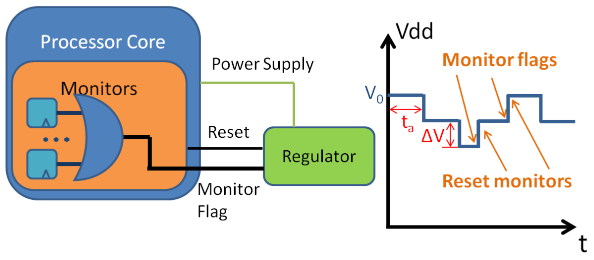


Fig. 11. Example application: the monitors give a one bit flag to the voltage regulator, the regulator reset the monitors after a corresponding voltage adaptation operation

transitions and the set of nets with timing-critical transitions. λ_i and γ_i are updated accordingly in order to evaluate the monitoring efficiency and to evaluate the results of different cost functions. A counter is used to record the number of nets whose delay is monitored in the current cycle (i.e., a timing-critical transition passes through this net and hits a monitor), which can be used to evaluate the monitoring observability.

Monitoring efficiency and observability results are listed in Table III. The monitor types marked with "(opt)" are the ones using the new cost functions as in Equation (10).

Compared to baseline, *SlackProbe* using default cost function in Equation (9) selectively inserts monitors at nets that have lower λ_i (thus with lower power overhead). Over all cases, *SlackProbe* has up to 1.5X higher criticality ratio (i.e., $\frac{\gamma_i}{\lambda_i}$) than baseline. SlackProbe using new cost function in (10) favors inserting monitors at nets that have both lower λ_i and higher γ_i (thus with higher monitoring efficiency), which further boosts the criticality ratio to up to 1.9X higher than baseline. A larger delay margin (ϵ) will give more flexibility in selecting monitor location, which results in higher criticality ratio.

Because monitors inserted at intermediate nets cannot monitor the nets after the inserted location, *SlackProbe* without observability awareness can only monitor similar or less number of nets compared to baseline. However, with the new cost function, *SlackProbe* is able to monitor up to 32% more nets than baseline. Larger delay margin (ϵ) allows monitors to be inserted further away from path endpoints, which results in less monitored nets (thus lower monitoring observability). Event detection can detect more signal transitions than speed sensing, which makes it achieve higher monitoring observability.

G. Practical and Implementation Considerations

One major concern with in situ monitoring is the implementation feasibility and potential overhead and disturbance due to the additional instances and wiring. To explore the implementation overhead and find out possible implementation issues of the monitor insertion, we pick processor A and implement the complete monitor insertion on it.

The target application model is shown in Fig. 11. The system will start with a safe supply voltage (V_0) and always wait for certain time (t_a) before making the voltage adjustment

[3]If monitor is inserted at path endpoint, no extra delay margin is required for delay matching.

TABLE III
MONITORING EFFICIENCY AND OBSERVABILITY RESULTS

Monitor type	Delay margin (ϵ)	Monitor count	Average λ_i	Average γ_i	Criticality ratio $\frac{\gamma_i}{\lambda_i}$	Average monitored nets per cycle
Baseline	N.A.[3]	1256	0.746	0.068	9.11%	317.4
Speed sensing	5%	113	0.311	0.042	13.61%	295.7
Speed sensing(opt)	5%	147	0.443	0.089	20.20%	420.8
Event detection	5%	148	0.303	0.048	15.88%	314.1
Event detection(opt)	5%	216	0.383	0.073	19.02%	376.5
Speed sensing	8%	48	0.329	0.076	22.99%	284.4
Speed sensing(opt)	8%	45	0.568	0.151	26.52%	357.1
Event detection	8%	100	0.459	0.089	19.28%	317.6
Event detection(opt)	8%	100	0.558	0.124	22.21%	360.2

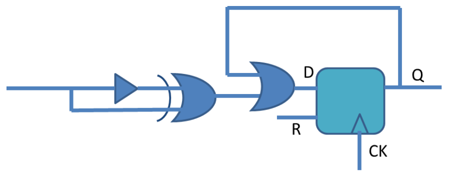


Fig. 12. Implemented monitor structure: the flag is sticky with an external reset

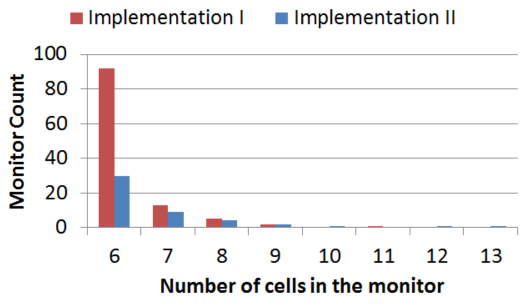


Fig. 13. Monitor cell count distribution: most monitors are inserted very close to the path endpoints, thus do not need extra cells other than the minimum required 6 cells.

of ΔV if no monitor flag is generated. In such system, the value of t_a and ΔV will depend on the monitoring observability and delay margin (ϵ). The monitors are designed to give a one bit sticky flag which can be reset externally. For implementation simplicity, we pick the monitor structure that consists of only standard cells as shown in Fig. 12.[4]

Monitor flag signals are connected through an OR tree and gives the one bit flag signal as processor primary output. Therefore, for each monitor, there are at least six gates in it including the first minimum size inverter, four cells as in Fig. 12 and one OR gate in the OR tree.

In this experiment, we implement two versions of the monitor insertion. The implementation results are summarized in Table IV. The final layout of implementation II is shown in Fig. 14. The overhead may be further reduced by using simpler or customized monitors as in [14], [17], [19].

During monitor insertion, because of the additional load from the monitors and other ECO timing disturbance, the original design is slowed down by about 5%. However, this only happens around the selected nets, which can be recovered through simple optimization like incremental sizing and threshold voltage assignment. In the experiments, we incrementally optimize the design after the monitor insertion so that it still meets the same delay target.

[4]This monitor is used only for evaluation purpose. It may not work for signal transitions with small glitches. A proper monitor design (e.g., TDTB [19]) can avoid the glitch issues.

The ECO also affects the clock network. We perform detailed analysis on the clock network before and after ECO for Implementation II. After ECO, the total number of clock sinks increases by 48, which is the number of monitor inserted. The number of clock tree levels remains at 3, but the number of clock buffers increases from 25 to 29. Maximum local clock skew slightly increases from 11.9 ps to 14.9 ps.

The other side effect of the ECO timing disturbance is that some new nets become timing critical. This may introduce unmonitored critical paths, which will require additional delay margin, pessimism in path selection or further timing optimization. The slow-down due to additional load from the monitors will only affect the nets that are monitored already. However, other timing changes such as ECO routing and clock skew changes may introduce unmonitored critical paths. In this implementation, we found two unmonitored critical paths due to the clock skew changes. A more careful monitor selection and less intrusive insertion can help prevent it.

VII. CONCLUSIONS

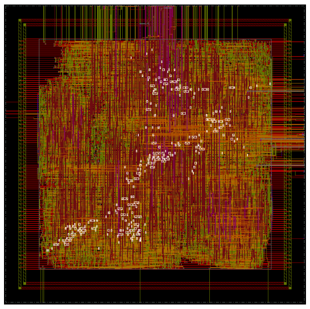
In this paper, we have proposed SlackProbe, a novel timing slack monitoring methodology of inserting monitors at both path ending nets and path intermediate nets. Experimental results on commercial processors show that with 5% additional timing margin, *SlackProbe* can reduce the number of monitors by 12-16X as compared to the number of monitors inserted at path ending pins. *SlackProbe* can also improve the monitoring efficiency by up to 1.9X and improve the monitoring observability by up to 32%, as compared to approaches that monitor path endpoints only.

REFERENCES

[1] P. Gupta *et al.*, “Underdesigned and opportunistic computing in presence of hardware variability,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012, Keynote Paper.
[2] A. Drake *et al.*, “A distributed critical-path timing monitor for a 65nm high-performance microprocessor,” in *Proc. IEEE International Solid State Circuits Conference*, feb. 2007.

TABLE IV
IMPLEMENTATION RESULTS ON PROCESSOR A

	Implementation I	Implementation II
Target delay margin	5%	8%
Number of monitors	113	48
Additional instances	711	327
Instances per monitor	6.3	6.8
Additional power overhead	13.65%	6.38%


(a) Layout with monitor instances highlighted

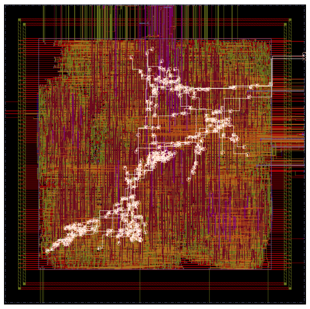

(b) Layout with both monitor instances and wires highlighted

Fig. 14. Final layout of processor A with the inserted monitors placed and routed

[3] T.-B. Chan *et al.*, “DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators,” in *IEEE International Symposium on Quality Electronic Design*, march 2012.

[4] J. Tschanz *et al.*, “Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance,” in *VLSI Circuits, Symposium on*, june 2009.

[5] D. Fick *et al.*, “In situ delay-slack monitor for high-performance processors using an all-digital self-calibrating 5ps resolution time-to-digital converter,” in *Proc. IEEE International Solid State Circuits Conference*, feb. 2010.

[6] X. Wang *et al.*, “Path-RO: a novel on-chip critical path delay measurement under process variations,” in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008.

[7] D. Ernst *et al.*, “Razor: a low-power pipeline based on circuit-level timing speculation,” in *IEEE/ACM International Symposium on Microarchitecture*, dec. 2003.

[8] S. Das *et al.*, “RazorII: In situ error detection and correction for pvt and ser tolerance,” *IEEE Journal of Solid State Circuits*, jan. 2009.

[9] H. Fuketa *et al.*, “Adaptive performance compensation with in-situ timing error prediction for subthreshold circuits,” in *IEEE Custom Integrated Circuits Conference*, sept. 2009.

[10] B. Rebaud *et al.*, “Digital timing slack monitors and their specific insertion flow for adaptive compensation of variabilities,” in *International Conference on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation*, ser. PATMOS’09, 2010.

[11] M. Eireiner *et al.*, “In-situ delay characterization and local supply voltage adjustment for compensation of local parametric variations,” *IEEE Journal of Solid State Circuits*, july 2007.

[12] M. Kurimoto *et al.*, “Phase-adjustable error detection flip-flops with 2-stage hold-driven optimization, slack-based grouping scheme and slack distribution control for dynamic voltage scaling,” *ACM Trans. Des. Autom. Electron. Syst.*, 2010.

[13] M. Agarwal *et al.*, “Circuit failure prediction and its application to transistor aging,” in *IEEE VLSI Test Symposium*, may 2007.

[14] B. Das *et al.*, “Warning prediction sequential for transient error prevention,” in *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, oct. 2010.

[15] T. Sato *et al.*, “A simple flip-flop circuit for typical-case designs for dfm,” in *IEEE International Symposium on Quality Electronic Design*, march 2007.

[16] K. Hirairi *et al.*, “13% power reduction in 16b integer unit in 40nm cmos by adaptive power supply voltage control with parity-based error prediction and detection (pepd) and fully integrated digital ldo,” in *Proc. IEEE International Solid State Circuits Conference*, feb. 2012.

[17] S. Kim *et al.*, “Razor-lite: A side-channel error-detection register for timing-margin recovery in 45nm soi cmos,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*. IEEE, 2013, pp. 264–265.

[18] L. Lai *et al.*, “Slackprobe: A low overhead in situ on-line timing slack monitoring methodology,” in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013, pp. 282–287.

[19] K. Bowman *et al.*, “Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance,” *IEEE Journal of Solid State Circuits*, jan. 2009.

[20] J. Lee *et al.*, “Eco cost measurement and incremental gate sizing for late process changes,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 1, p. 16, 2012.

[21] M. Choudhury *et al.*, “Time-borrowing circuit designs and hardware prototyping for timing error resilience,” *Computers, IEEE Transactions on*, vol. 63, no. 2, pp. 497–509, 2014.

[22] M. Fojtik *et al.*, “Bubble razor: Eliminating timing margins in an arm cortex-m3 processor in 45 nm cmos using architecturally independent error detection and correction,” *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 66–81, 2013.

[23] L. Xie *et al.*, “Representative path selection for post-silicon timing prediction under variability,” in *Proc. ACM/IEEE Design Automation Conference*, june 2010.

[24] K. Bowman *et al.*, “All-digital circuit-level dynamic variation monitor for silicon debug and adaptive clock control,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 9, pp. 2017–2025, Sept 2011.

[25] A. Drake *et al.*, “Single-cycle, pulse-shaped critical path monitor in the power7 microprocessor,” in *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*, Sept 2013, pp. 193–198.

[26] [Online]. Available: http://www.cadence.com

[27] R. P. Weicker, “Dhrystone: a synthetic systems programming benchmark,” *Commun. ACM*, Oct. 1984.

[28] M.-L. Li, P. Ramachandran, U. R. Karpuzcu, S. Hari, and S. V. Adve, “Accurate microarchitecture-level fault modeling for studying hardware faults,” in *High Performance Computer Architecture. IEEE International Symposium on*. IEEE, 2009, pp. 105–116.