SlackProbe: A Low Overhead In Situ On-line Timing Slack Monitoring Methodology

Liangzhen Lai[†], Vikas Chandra[*], Robert Aitken[§], and Puneet Gupta[‡]
UCLA Electrical Engineering Dept.[†‡], Los Angeles; ARM Inc. R&D[*§], San Jose
E-mails: liangzhen@ucla.edu[†], vikas.chandra@arm.com[*], rob.aitken@arm.com[§], puneet@ee.ucla.edu[‡]

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 exclusively focus on monitoring path ending registers, which is not cost efficient from power and area perspectives.**

In this paper, we propose SlackProbe methodology, which inserts timing slack monitors like "probes" at a selected set of nets, including intermediate nets along critical paths. SlackProbe can significantly reduce the total number of monitors required at the cost of some additional delay margin. It 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.). Though we focus on monitor selection in this work, we give an example of using SlackProbe with adaptive voltage scaling.

Experimental results on commercial processors show that with 5% more timing margin, SlackProbe can reduce the number of monitors by 15-18X as compared to the number of monitors inserted at path ending pins.

I. Introduction

With variability increasing, 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 replica [4] makes the monitor tunable to reduce the mismatch of the real circuit and replica after manufacturing. 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, some 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. We observe that most of existing methods exclusively focus 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. The key contributions of this paper are as follows:

1) We propose a novel slack monitoring methodology allowing placing monitors at intermediate nets along circuit paths
2) 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

The rest of the paper is organized as follows: Section II gives an overview of the proposed monitoring methodology. Section III describes the path selection process and graph reduction. Section IV discusses the cost metrics for inserting monitors. Section V formulates and solves the monitor insertion problem. Section VI presents the experimental results. Section VII concludes the paper.

II. Monitoring Methodology 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 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 to monitor the delay of all critical paths passing through *A*. As shown in Fig. 1, in stead 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], [17]. 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 discussed in [17].

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.

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

978-3-9815370-0-0/DATE13/©2013 EDAA

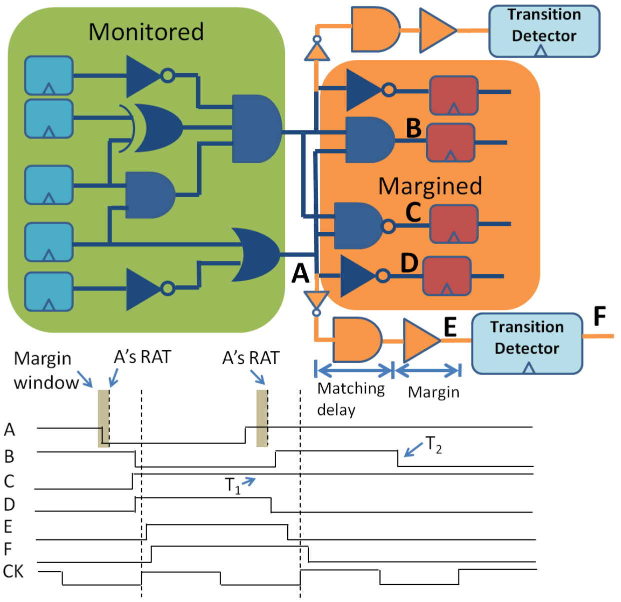


Fig. 1. SlackProbe working principle. As shown in the timing diagram, compared to inserting monitors at destination registers, the monitor inserted at A can monitor the path delay even when the transition does not propagate to the destination register (i.e. T_1 at C). But the monitor inserted at node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

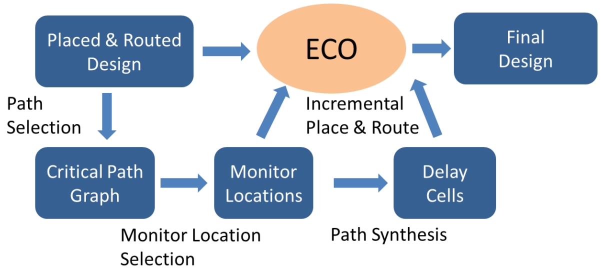


Fig. 2. Monitor Insertion Flow

The 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, the 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 [18] are applied when picking monitor locations to minimize the interference to the original design.

C. Possible Applications of the Monitors

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

One possible application is to use it as timing-failure event predictor and combine with some of the existing error resilience mechanisms like in [8], [19], [20]. In this case, the monitors have to capture all signal transition events that may result in timing errors.

Another possible application is to use it as speed sensor indicating whether current operation condition is close to possible timing failure. 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.

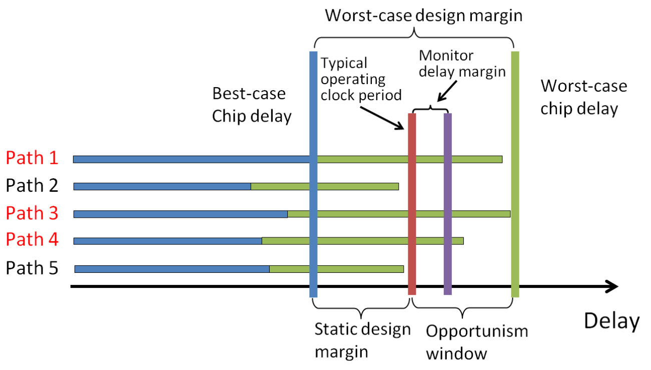


Fig. 3. Opportunism window is the margin saving comparing to worst-case design. Monitor delay margin is the delay margin of the delay matching chain which will be discussed in detail in Section IV-A.

III. Path Selection and Graph Reduction

A. Path Selection Criteria

Given a placed and routed design, the first step is to identify the part of the design that may be timing critical. Depending on the application, different criticality criteria may be applied for the selection process.

One possible path selection method is to define a targeting typical operating clock period and corresponding *opportunism window* as in Fig. 3. The circuit is designed to operate at the typical operating clock period if there are no monitor flags. If there is a path whose delay exceeds the typical operating clock period, the circuit should be able to adapt accordingly with the help of the monitors. Therefore, all circuit paths whose worst-case delay falls into the opportunism window should be classified as timing critical and require monitors. This 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 preassumed correlation. The size of opportunism window will affect the total number of circuit paths to be monitored. The monitor delay margin will affect the possible monitor locations. So there is a trade-off between the monitoring benefit (defined by opportunism window and monitor delay margin) and monitoring overhead. We will discuss and explore this trade-off in later sections and experiments.

Another way to select the paths with slack smaller than a fixed amount at various process corners and ambient conditions. Similarly, statistical methods as in [21] can also be used. This method can be used to account for correlation in the variation and reduce the pessimism with the help of some pre-known information from the applications.

B. Circuit Graph Reduction

With the path selection criteria, critical and non-critical parts can be identified from the circuit graph. We delete the non-critical pins and gates from the original circuit graph and obtain a reduced circuit graph. The critical paths in the original circuit graph are still preserved in the reduced graph. With this

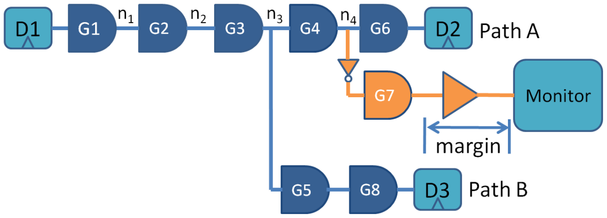


Fig. 4. Example of path-monitor pairs

property, we can analyze the monitor insertion problem on the reduced circuit graph only. Unless otherwise specified, circuit graph discussed in the rest of the paper is the reduced graph.

IV. Monitor Cost Metrics and Analysis

A. Delay Margin Cost

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 a 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 etc.) are allowed.
3) The path has no joint instances with the monitor. In this case, we consider that the monitor has no effect on the path.

2) Delay Margin for Monitors

Though different paths may require different delay margins, each monitor will have only one margin. The 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, theoretically the best case delay of the delay chain (i.e. n_4 to the monitor) should match the worst case delay of the original path (i.e. $G6$). 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. The equivalent delay margin in this case equals the delay of $G6$ at slow process corner (i.e. delay of the delay chain at typical process corner) minus the delay of $G6$ at typical process corner. This margin is considered as the delay uncertainty of $G6$.

Methods as in [3] or [4] can also be applied to synthesize a replica-like or tunable delay path.

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.

Given this 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 $P_k \cap I_{n_i} \neq \emptyset$.

B. 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. Picking 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. In this work, power overhead is modeled as $p_o + (\lambda_i p + l)d_i$, 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.

C. Design 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 [18], we use a linear model to evaluate the design interference cost of inserting a monitor at net n_i as:

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

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.

D. Monitor Insertion Cost and Constraints

Based on the cost metrics, the overall cost of inserting a monitor at net n_i is

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

where parameters are chosen for correct normalization. The monitor insertion constraint is defined as in Section IV-A.

V. Problem Formulation and Solution

A. 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:

$$y_i = \sum_{n_j \in O_{n_i}} x_j \tag{1}$$

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 (1) can be represented as matrix representation $\mathbf{y} = \mathbf{Q}\mathbf{x}$.

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

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

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:

$$\begin{aligned} \text{minimize:} \quad & \mathbf{c}^T \cdot \mathbf{x} \\ \text{subject to:} \quad & \mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{x} \geq \mathbf{1} \\ & x_i \in \{0, 1\} \end{aligned} \tag{3}$$

B. Problem Conversion

Solving (3) directly is hardly tractable for any reasonable size of circuit because of the large number of paths. We will convert the problem into a solvable one.

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 given $x_i \in \{0, 1\}$. 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, 0 otherwise. If we replace $\mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{x} \geq \mathbf{1}$ with $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{1}$ in (3), the new optimization is equivalent to the original one.

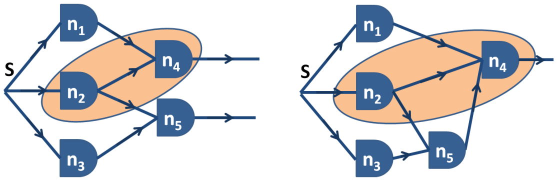
Then we relax the constraint on x_i as $x_i \geq 0$. [1] This gives us an LP problem with its dual as:

$$\begin{aligned} \text{maximize:} \quad & \mathbf{1}^T \cdot \mathbf{f} \\ \text{subject to:} \quad & \mathbf{A}^T \cdot \mathbf{f} \leq \mathbf{c} \\ & \mathbf{f} \geq 0 \end{aligned} \tag{4}$$

If we add two dummy nodes n_s and n_t as the beginning and ending nodes for all paths, (4) becomes similar to a maximum flow problem with $\mathbf{f}$ being the dedicated flow along each path.

Converting (3) into the LP problem (4) still does not reduce the problem size. To make the problem solvable, we will convert (4) into a formulation with edge-based variables and constraints. We denote $\mathbf{e} = [... \; e_{i,j} \; ...]^T$ as the total 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. The objective in (4) is equivalent to maximizing the total flow out from n_s.

If we look at the i-th row of the constraint $\mathbf{A}^T \cdot \mathbf{f} \leq \mathbf{c}$, on the left is the sum of all path flows that pass through 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 nets in I_{n_i} are connected (see example in Fig. 5(a)). If we group I_{n_i} as one single 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 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 edge flow that goes into I_{n_i}.



(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 $n_5 \in I_{n_4}$.

The converted edge-based formulation becomes:

$$\begin{aligned} \text{maximize:} \quad & \sum_{\forall i} e_{s,i} \\ \text{subject to:} \quad & \forall n_i, \sum_{\forall j} e_{j,i} = \sum_{\forall k} e_{i,k} \\ & \forall n_i, \sum_{n_j \notin I_{n_i}, n_k \in I_{n_i}} e_{j,k} \leq c_{n_i} \\ & \mathbf{e} \geq 0 \end{aligned} \tag{5}$$

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 (5) takes less than a minute.

C. Problem Solution

Because of the relaxation on x_i, the result of (5) will be a lower bound of that in (3).

To derive an exact solution of the monitor locations, we take the solution of (5) 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 (3). 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.

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 implementation is done using Cadence toolchain [22]. The implementation information is listed in Table I. To calculate the monitor insertion cost for each net, we obtain the net switching probabilities through running dhrystone [23] benchmarks.

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.

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

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 I
IMPLEMENTATION INFORMATION OF 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

TABLE II
EXPERIMENTAL RESULTS ON THE PROCESSOR BENCHMARKS

- *SlackProbe Speed Sensing:* This method aims at speed sensing. It allows the monitor 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 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. Delay margin ϵ is set to be 5% of the typical operating clock period. Table II summarizes 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. Results on Different Path Selection Criteria

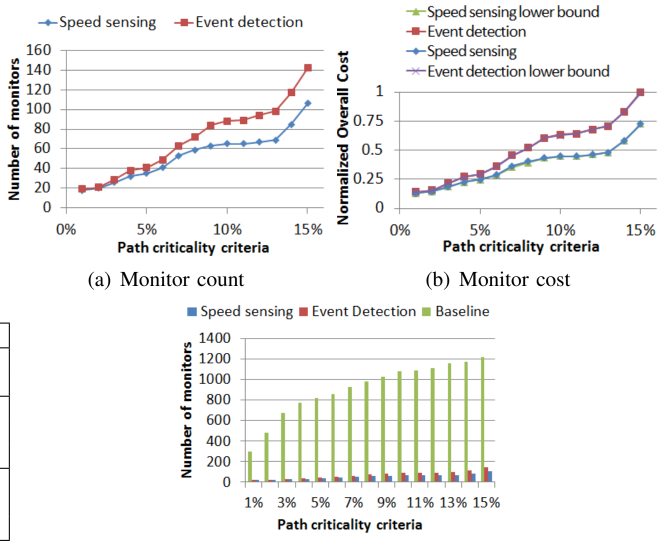
Depending on the application context, different path selection criteria may be applied. To evaluate the proposed method over different path selection criteria, we pick processor A and apply different path selection criteria to it.

Fig. 6 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 slack at typical corner is smaller than the specified percent of the clock period. Compared to the baseline method, our methods show on average 15X reduction in the number of monitors for event detection and 18X for speed sensing over all cases.

Since we did not have access to aging libraries, we did not do the experiments of inserting monitors as aging sensors. But it can be inferred from Fig. 6, depending on the aging effect and expected life time, e.g. if worst-case delay degradation due to process variation and aging is 15%, path selection will be similar to the corresponding 15% point in Fig. 6.

D. Results on Different Monitor Delay Margin

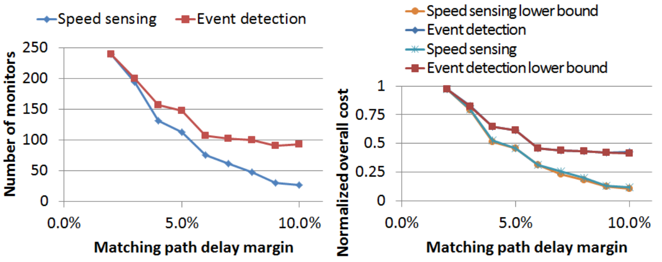
To show the trade-off between delay margin and monitor count, we also sweep the delay margin ϵ for processor A



(a) Monitor count (b) Monitor cost

(c) Comparison with baseline

Fig. 6. Monitor count and cost for processor A with different criticality criteria (i.e. extract paths with slack less than the specified percent of clock period at typical corner)



(a) Monitor count (b) Monitor cost

Fig. 7. Monitor count and cost vs. delay margin for processor A

and plot the corresponding monitor count and monitor cost in Fig. 7. The path selection criteria is 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 reduces for both methods. We also try to combine different weighting parameters of the monitor insertion cost (i.e. a_t, a_r etc.). Since the monitor location selection depends more on the circuit topology, the weighting parameters do not significantly change the total number of monitors. But 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. Implementation Issues

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 targeting application model is shown in Fig. 8. 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 consist of only standard cells as shown in Fig. 9. The monitor flags 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

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

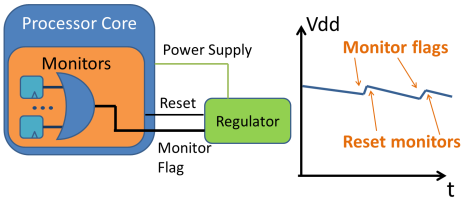


Fig. 8. Example application: the monitors give a one bit flag to the voltage regulator, the regulator reset the monitors after a corresponding voltage adaptation operation

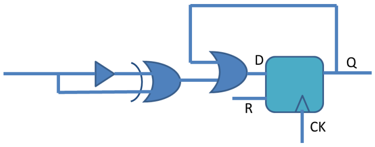


Fig. 9. Implemented monitor structure: the flag is sticky with an external reset

including the first minimum size inverter, four cells as in Fig. 9 and one OR gates in the OR tree.

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, the slow down is only 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.

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. But 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.

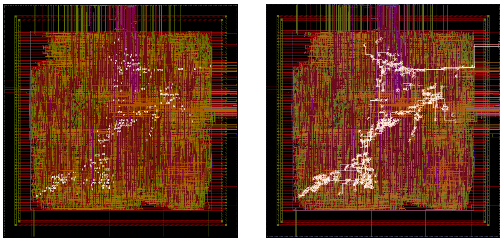
In this experiment, we implement two versions of the monitor insertion. The implementation results are summarized in Table III. The final layout of implementation II is shown in Fig. 10. The layout overhead could be further reduced by using simpler or customized monitors as in [14], [17].

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% additonal timing margin, our methods can reduce the total number of monitors by 15-18X compared to the total number of critical path ending pins. We also demonstrate the implementation feasibility and overhead through an example of using SlackProbe with adaptive voltage scaling. Future work will incorporate the monitors in more applications as in [1] and improve the monitor location selection and insertion to be less intrusive.

	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%

TABLE III
Implementation results on processor A



(a) Layout with monitor instances highlighted

(b) Layout with both monitor instances and wires highlighted

Fig. 10. Final layout of processor A with the inserted monitors placed and routed

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.

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

[18] J. Lee *et al.*, "Incremental gate sizing for late process changes," in *Proc. IEEE International Conference on Computer Design*, oct. 2010.

[19] M. Choudhury *et al.*, "Timber: Time borrowing and error relaying for online timing error resilience," in *IEEE/ACM Design, Automation and Test in Europe*, march 2010.

[20] M. Fojtik *et al.*, "Bubble razor: An architecture-independent approach to timing-error detection and correction," in *Proc. IEEE International Solid State Circuits Conference*, feb. 2012.

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

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

[23] R. P. Weicker, "Dhrystone: a synthetic systems programming benchmark," *Commun. ACM*, Oct. 1984.