Timing speculation with optimal in situ monitoring placement and within-cycle error prevention

Timing Speculation With Optimal *In Situ* Monitoring Placement and Within-Cycle Error Prevention

Hadi Ahmadi Balef, Hamed Fatemi, Kees Goossens, and José Pineda de Gyvez, *Fellow, IEEE*

***Abstract*— In this paper, a timing speculation technique with low-overhead *in situ* delay monitors placed along critical paths is presented. The proposed insertion of monitors enables timing error prevention within the same clock cycle. Compared to other techniques, the design cost per monitor in our technique is low because no additional gates for the guard banding, inspection window generation, and short path extension are required. We benchmarked our approach on an ARM Cortex M0. The insertion strategy reduces the number of monitors by up to ∼23×, power by ∼5.5×, and area by ∼2.8× compared to the traditional *in situ* monitoring techniques that insert monitors at the flip-flops. The timing error correction uses a global clock stretching unit to prevent errors within one cycle. With the proposed error prevention technique, ∼22% more delay variation is tolerated with a negligible energy overhead of less than ∼1%.**

***Index Terms*— *in situ* delay monitoring, timing error, timing speculation (TS), variation resilience.**

I. Introduction

THE CMOS technology scaling has been the driving force of the semiconductor industry for decades. In the nanoscale era, however, the increased effects of variability on design robustness can nullify the benefits of the technology scaling. A variety of physical phenomena, such as aging effects (e.g., hot carrier injection and bias temperature instability), voltage droops, and process variations, are the sources of variability. Variability effects on electrical parameters of a circuit are manifested as delay variation. Delay variation may cause erroneous capturing of data at the flip-flops, i.e., a timing error happens. If the timing error is not masked properly, a timing failure occurs in the system.

Traditionally, integrated circuits are over-designed to improve their robustness by designing at the worst-case corners. Fortunately, the cost of the worst-case design can be reduced by employing timing speculation (TS) techniques. TS techniques are based on chip health monitoring and masking of timing errors at runtime. Chip health monitoring is based on physical sensors [1], replica paths [2], or *in situ* delay monitors [3]–[6]. In contrast to other monitoring techniques, *in situ* monitoring allows for a truly in-system monitoring. The main challenges of *in situ* monitoring TS techniques are the prohibitive design overhead, limited observability to variation effects, design intrusion, and complexity and overhead of the error masking mechanism.

In this paper, we propose a new *in situ* monitoring technique with a low number of monitors and improved observability to delay degradation. The following are our contributions.

1) An *in situ* monitoring strategy is proposed, in which the monitors are inserted at intermediate points along timing paths. With the proposed technique, the design overhead due to the guard banding against variations, inspection window generation, and short path extension is removed. Thereby, power and area overhead of *in situ* monitoring are low, compared to the state of the art (see [4] and [7]).
2) A method to identify a selective set of insertion points is introduced that instead of relying on a list of all critical timing paths (which requires path-based timing analysis), it relies on the block-based timing analysis, suitable for large circuits. The proposed method is ∼2.8× faster for small circuits with ∼100 gates, and it is orders of magnitude faster for large circuits (more than ∼1000 gates).
3) An error prevention technique based on stretching the clock pulse globally within a cycle is introduced. This clock stretching depends on the outputs of the *in situ* monitors. This approach yields ∼25% more delay variation tolerance with negligible energy overhead compared to the state-of-the-art error masking techniques (see [4] and [8]).

A prior version of this paper was published in [9], where the design and insertion flows of within-path *in situ* delay monitors were introduced. This paper extends [9] in several ways. The design of the monitor and its insertion algorithm are improved to lower the power per monitor and to find the insertion points more effectively. Recall and precision metrics from binary classification are employed to show the tradeoffs of insertion point choices against speed constraints. A new clock stretching unit is presented that allows the timing error

Manuscript received August 9, 2018; revised November 21, 2018; accepted January 17, 2019. Date of publication February 14, 2019; date of current version April 24, 2019. This work was supported in part by EU CATRENE RESIST under Grant 2015/CATRENE/CT217 and in part by EU ECSEL SILENSE under Grant 2017/ECSEL/00737487. *(Corresponding author: Hadi Ahmadi Balef.)*

H. Ahmadi Balef and K. Goossens are with the Department of Electrical Engineering, Eindhoven University of Technology, 5612 AP Eindhoven, The Netherlands (e-mail: h.ahmadi.balef@tue.nl; k.g.w.goossens@tue.nl).

H. Fatemi is with NXP Semiconductors, High Tech Campus, 5656 AE Eindhoven, The Netherlands (e-mail: hamed.fatemi@nxp.com).

J. Pineda de Gyvez is with the Department of Electrical Engineering, Eindhoven University of Technology, 5612 AP Eindhoven, The Netherlands, and also with NXP Semiconductors, 5656 AE Eindhoven, The Netherlands (e-mail: j.pineda.de.gyvez@tue.nl).

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

Digital Object Identifier 10.1109/TVLSI.2019.2895972

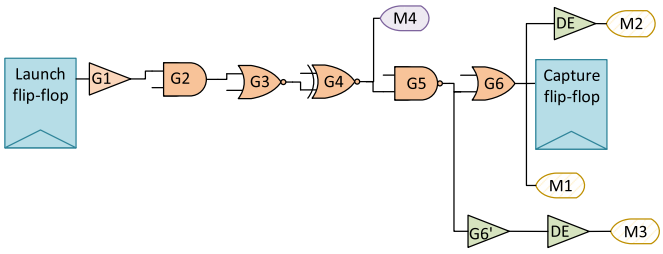

Fig. 1. Insertion points of monitors along the timing paths of digital circuits for different *in situ* delay monitoring techniques.

prevention within a cycle by delaying the capturing edge of the clock based on the collected monitoring data with an OR tree. Furthermore, the experiments are expanded to assess the proposed technique more extensively.

In what follows, first, an overview of the state of the art is done in Section II. Then, the proposed TS technique is explained in Section III. The new method to insert the monitors is explained in Section IV. The selection of insertion points is discussed in Section V. In Section VI, the experimental results of applying the proposed ideas to some benchmark circuits and an industrial design are provided and discussed. Finally, Section VII provides the conclusion.

II. Preliminaries and Related Works

In synchronous digital circuits, data propagate from primary inputs or sequential elements, through combinational logic, to another or the same sequential gate or the primary outputs of the circuit. Assuming that the sequential logic gates are positive edge-triggered flip-flops, the timing path starts from the clock pin of the launching flip-flop, continues through the timing arcs of some logic gates, and ends into the data pin of a capturing flip-flop. The path delay is the sum of the propagation delay of each timing arc in the path. Within one clock period, the maximum data arrival time to a circuit node is the maximum delay from the clock pin of a launching flip-flop to the node. The maximum delay of all the timing paths that end into a capture flip-flop is, therefore, equal to the maximum data arrival time to the data input pin of the flip-flop. Note that the maximum delay of each timing path should not be more than one clock period t_{clock}. Otherwise, erroneous data may be captured, i.e., a timing error may happen. The timing analysis is performed to find the propagation delay of the paths and to make sure that a timing violation does not happen. Basically, the delay of all the paths could be calculated in a path-based timing analysis. Alternatively, in a block-based timing analysis, the maximum data arrival time is calculated from summation and maximum operations to find the maximum delay of all paths ending in each flip-flop. The block-based timing analysis is a practical method because its runtime is much less than the path-based approach.

In Fig. 1, an example of a timing path and the possible insertion points of the *in situ* monitors are illustrated. The timing path, starts from the launching flip-flop, goes through the combinational logic gates G1–G6, and ends into the capturing flip-flop. Conventionally, *in situ* delay monitors are inserted at the endpoint of the timing paths [4] (see M1). Hence, a timing error is detected by sensing if the data arrival is too late, i.e., a setup time violation occurs at the capture flip-flop. To avoid flip-flop metastability and to avoid the complexity and overhead of error correction [8], the guard banding is applied between the detection at the monitors and the actual timing error in the main flip-flops. The guard banding is performed by adding a delay margin between the insertion point and the input of the monitor (see DE before M2). In fact, DE makes the slack of M2 to be less than the one of the capturing flip-flop which means that there is a guard band between the monitor triggering and the timing failure in the main design. Note that inserting the monitors at every flip-flop adds a huge design cost due to the large number of added monitors. Observe as well that the detection of timing degradation depends on the excitation of the monitored paths, i.e., the paths which end into the monitors. The monitored paths are not always excited since path excitation is data dependent. Therefore, *in situ* monitoring may have limited observability to variability effects. In [7], the monitors are inserted at intermediate points to reduce the number of monitors. In [7], the unmonitored part of the paths is accounted for by adding a pessimistic delay margin between the insertion point and the input of the monitor (see G6′ in Fig. 1). Note that the guard band is still applied by adding another delay margin before the monitor (see DE before M3 in Fig. 1). This adds to the cost of each inserted monitor. To reduce the design overhead per monitor, the monitor is, therefore, preferably inserted almost at the end of the paths to reduce the additional gates per monitors. An inspection window is also required in such methods to define the time in which any transition should be flagged as timing error. However, this adds to the design cost per monitor. Besides, inserting the monitors almost at the end of the path reduces the chance to prevent the error within one cycle and adds to the number of required monitors. For a proper guard banding, the insertion point of the monitor can be selected such that the slack of the monitor is made less than the slack of the main design without additional gates (see M4 in Fig. 1).

In [10] and [11], the temporal middle point of the critical path is monitored to check if a delay increase to the insertion point causes a transition after half of the clock period. Observe that in the presence of variability effects, the critical path ranking may change. Therefore, the critical timing paths are those paths whose delay is longer than a specific value smaller than the clock period. Note that the insertion points must be identified based on the timing report of all paths. To obtain a list with all critical paths, a path-based timing analysis is required which is not feasible for medium-to-large size circuits due to the complexity of the analysis [12]. Furthermore, in [10] and [11], no delay margin is added for the guard banding because it is assumed that the delay of monitored and unmonitored parts of the paths degrade in a similar fashion. To prevent errors, the output of the monitor should be connected to the capture flip-flop to enable time borrowing. The error prevention is thereby performed by local clock stretching at each endpoint. Therefore, all capturing flip-flops must be redesigned to have a clock stretching circuitry inside. Besides, stretching the clock period locally at each flip-flop may cause hold time

violations. This adds to the hold time margin of all flip-flops with clock stretching capability. Furthermore, clock stretching may accumulate along consequent pipeline stages depending upon the consequent time borrowings, which eventually may cause a setup time violation at one of the flip-flops. Therefore, a local error prevention strategy may cause timing problems due to both setup and hold time violations in the next pipeline stage from skewing the clock between flip-flops. Londono and Pineda [6] propose a TS technique with within-cycle global clock stretching, in which the occurrence of a timing error is predicted with a transition detector along the timing path. The insertion point of the monitor is selected such that the clock manipulation is performed before the end of the clock cycle. However, the *in situ* monitoring technique in [6] does not take into account the variability of the unmonitored part of the path. Furthermore, it requires to generate an inspection window which adds to the overhead of the approach. The use of a global clock stretching in combination with *in situ* monitoring has been considered in other works too (see [13]). Nevertheless, the clock stretching unit in those works is phase-locked loop-based and normally takes more than one clock cycle to take effect (see [14]). Recently, an all-digital clock stretching unit has been proposed in [15] which is based on a delay line to provide different clock phases. We propose a new clock stretching unit that is more power and area efficient compared to the circuit proposed in [15] due to its efficient microarchitectural implementation.

III. Proposed Timing Speculation System

The proposed TS system consists of the *in situ* monitors, an OR tree, and a clock stretching unit, as shown in Fig. 2(a). The monitors are connected to a set of insertion points along critical timing paths. Since the insertion of monitors is at intermediate points along the paths, a (potential) delay degradation can be detected before the end of the clock cycle. The OR tree collects the outputs of the monitors into one bit [W shown in Fig. 2(a)]. The clock stretching unit delays the start of the next clock cycle depending upon the output of the OR tree to make sure that the delayed paths still fit into the current cycle. Therefore, the monitoring, the collection of monitor outputs with the OR tree, and the error prevention should fit into one clock cycle.

A. In Situ Delay Monitor

The design of the *in situ* delay monitor is shown in Fig. 2(b). The monitor detects delay degradation by checking if any late data transition occurs during the second half of the clock period. Therefore, the insertion points are the potential points where the maximum data arrival is less than half of the clock period, in the absence of variation effects. The monitor consists of one latch with active high clock, one XORgate, and two buffers [B1 and B2 shown in Fig. 2(b)]. The data value at the insertion point is captured by the latch at the negative clock edge. The XOR gate then flags any transition which occurs during the second half of the clock cycle. A delayed arrival of data to the insertion point implies delay degradation, and it is detected by the monitor. Buffer B1 minimizes the loading effect of the monitor on the insertion point. Buffer B2 is selected such that the delays to the inputs of the XOR gate are made the same to avoid glitches.

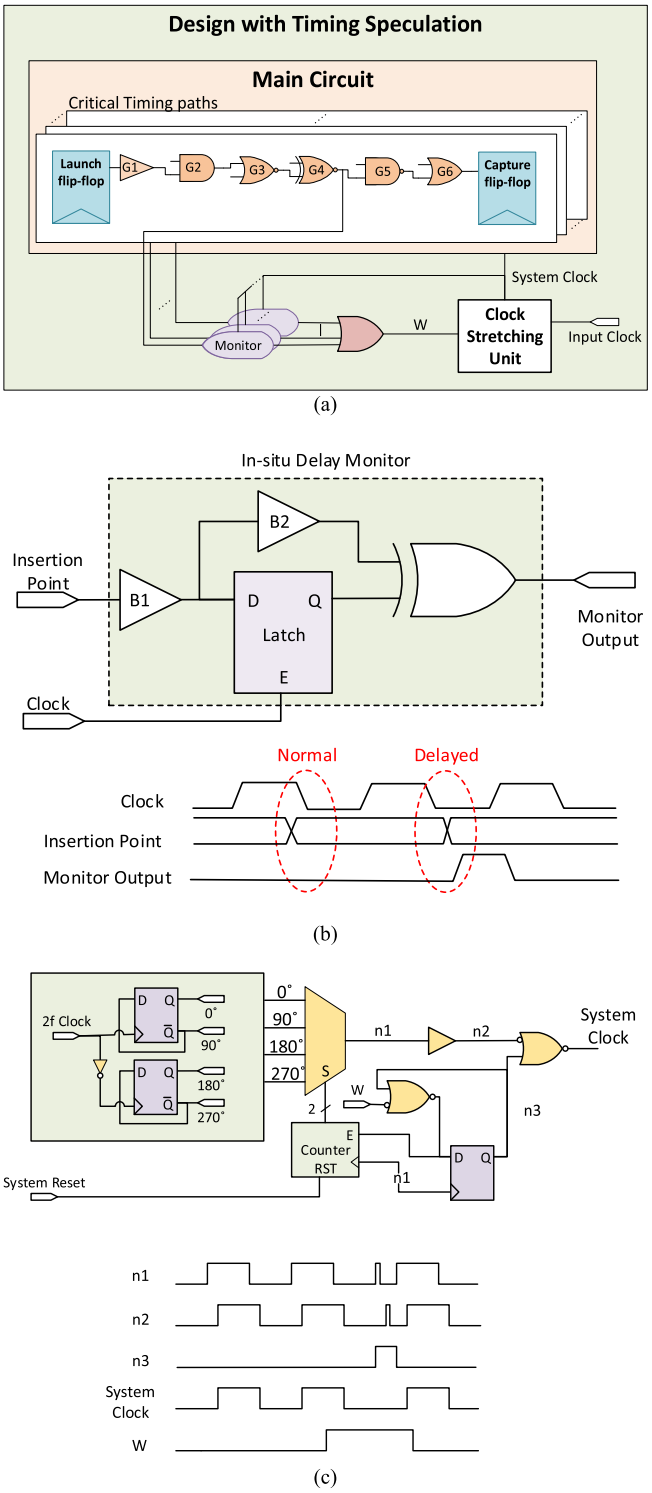

Fig. 2. Proposed TS technique. (a) High level view to the technique. (b) Design of the *in situ* monitor. (c) Design of the clock stretching unit.

B. Clock Stretching Unit

The proposed clock stretching unit is illustrated in Fig. 2(c). In the clock stretching unit, the output clock is stretched by

increasing the phase of the clock at the output dynamically, i.e., the phase of the clock is increased by a fixed degree every time that the output of the OR tree is high at the end of the cycle. This phase shifting is performed by circular selection from n different clock phases, i.e., the clock phase can increase by $360°/n$. We considered four clock phases indicating that the clock phase can be shifted by 90°. The four phases are generated based on an input clock pulse whose frequency is $2\times$ greater than the clock frequency of the main design, as shown in Fig. 2(c). The phase selection is performed using a MUX gate whose select signal is driven by a 2-bit counter. The 2-bit counter upcounts every time the output of the OR tree is high at the end of the cycle. Hence, if at least one monitor predicts a timing violation, the clock stretching unit delays the upcoming clock positive edge by one quarter of the clock period to avoid a setup time violation. One can consider a more fine-tuned clock stretching by having more clock phases. Note that clock stretching happens globally (i.e., at all flip-flops), thereby no risk of hold time violation. Since the output of the MUX is used for the counter, a narrow pulse will appear at the output clock [see $n1$ in Fig. 2(c)]. Glitch suppression is thereby required to have a clean clock signal at the output. This is done by generating the appropriate glitch suppression signal [see $n3$ in Fig. 2(c)] and delaying the output of MUX [see $n2$ in Fig. 2(c)].

Clock stretching is employed in adaptive clocking systems. However, not all adaptive clocking systems are suitable for within-cycle error prevention because many of them require more than one cycle to take effect. In [16], clock gating is employed for clock manipulation. However, gating one clock cycle means shifting the clock edge by one cycle which causes performance loss. In other works, (see [15]) the clock is manipulated by shifting the edge for less than a cycle to reduce the performance loss when preventing timing errors. However, those techniques rely on a delay line which imposes higher hardware overhead and element-wise variation sensitivity compared to our clock stretching technique. We implemented the proposed clock stretching unit in the 40-nm technology. The proposed clock stretching unit has a power consumption of 1.126 μW at the nominal supply voltage (i.e., 1.1 V) to generate a 200-MHz clock pulse at the output. Furthermore, the area of the proposed circuit is 96 μm^2. Therefore, the proposed circuit is much more power and area efficient compared to the circuit proposed in [15].

IV. Selective Points for In Situ Monitoring

In this section, our method to identify the set of insertion points and the integration of the proposed technique in the design flow are explained.

A. Identifying the Insertion Points

As was mentioned before, we define critical timing paths as those paths whose delay is longer than a specific value. Therefore, we define the maximum monitored slack, $\text{Slk}_{\max}$, as the constraint for selecting the critical timing paths to be monitored, i.e., the critical paths are all the paths whose delay is more than ($t_{\text{clock}} - \text{Slk}_{\max}$). This definition is illustrated in Fig. 3.

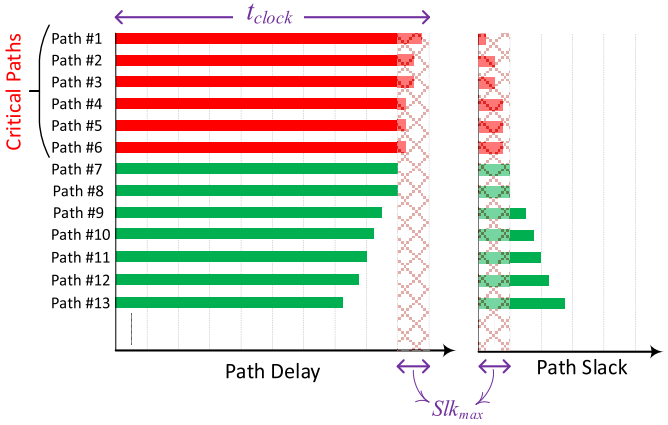

Fig. 3. Definition of the critical paths based on $\text{Slk}_{\max}$.

The timing graph of a digital circuit is a weighted directed graph G. The set of vertices in G representing the ports and instance pins is $V(G)$, and the set of directed edges in G representing the timing arcs is $E(G)$. The timing arc from vertex $v_i \in V(G)$ to vertex $v_j \in V(G)$ is represented by $e_{ij} \in E(G)$. Let the vector $D \in \mathbb{R}^{n\times 1}$ contain the maximum data arrival time to a vertex $v_i \in V(G)$. The maximum data arrival time to vertex $v_i \in V(G)$ is represented as $d_i \in D$. Industrial timing analysis tools [17] can report the most critical path that includes any specific nodes, based on the block-based timing analysis. We employ this report to find the maximum data arrival time to each circuit node, i.e., to find D. Furthermore, based on this report, we can find the minimum slack of the paths which are crossing each specific circuit node. Therefore, the block-based timing analysis also returns the vector SLK $\in \mathbb{R}^{n\times 1}$, where SLK contains the minimum slack of the paths that include each vertex $v_i \in V(G)$.

A cut $C = (S_1, S_2)$ is created based on D, where

$$S_1 = \{v_i \in V(G) | d_i < t_{\text{inp}}\} \tag{1}$$

and

$$S_2 = \{v_i \in V(G) | d_i \geq t_{\text{inp}}\} \tag{2}$$

where t_{inp} is the maximum delay from the launch flip-flop to the monitor insertion point. Note that during the clock tree synthesis, the design tool attempts to make the delay from the clock ports to the clock pin of all flip-flops the same. Naturally, in a more accurate analysis, clock skew should also be included. In the example of Fig. 4, a timing graph and the cut are illustrated. The set of boundary vertices, $S_{\text{bnd}} \subset S_1$, is defined as

$$S_{\text{bnd}} = \{v_i \in V(G) | e_{ij} \in E(G), v_i \in S_1, v_j \in S_2\} \tag{3}$$

where S_1 and S_2 are obtained from (1) and (2), respectively. The vertices in S_{bnd} are candidates for insertion of the monitors. A screening is performed on S_{bnd} to find the vertices which are in the critical paths, and to insert the monitors at

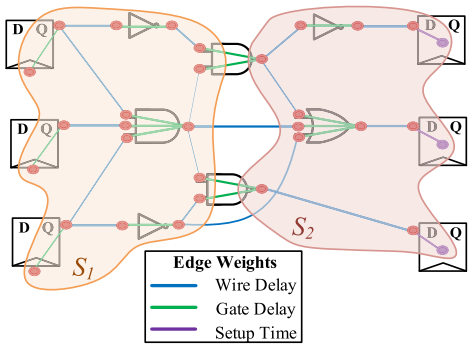

Fig. 4. Example circuit graph showing the cut $C = (S_1, S_2)$.

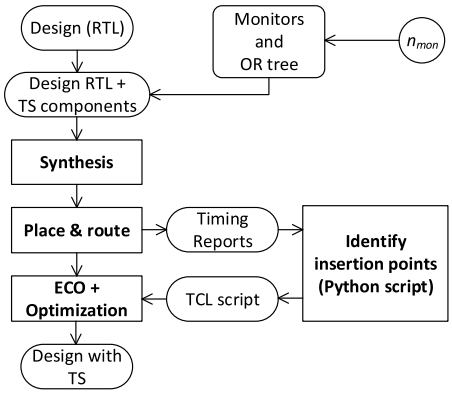

Fig. 5. Proposed implementation flow.

those vertices. Therefore, the set of insertion points is

$$S_{\text{inp}} = \{v_i \in V(G) | v_i \in S_{\text{bnd}} \text{^} Slk_i < Slk_{\max}\} \quad (4)$$

where S_{bnd} is obtained from (3), and $\text{Slk}_i \in \text{SLK}$. Once the set of insertion points is identified from (4), the monitors are inserted at the corresponding gate pins of all vertices in S_{inp}, according to the flow which is explained in Section IV-B.

B. Integration in the Design Flow

The monitors are conventionally added to the design after the place and route step (see [6] and [7]). However, such an implementation flow perturbs the layout and the timing of the main design [7]. To minimize the perturbation, we propose the implementation flow illustrated in Fig. 5. In the proposed flow, the monitors are added at the synthesis stage with dangling insertion point inputs. Yet, the clock input of the monitors is connected such that their loading effect is taken into account during the clock tree synthesis. The implementation with unconnected monitors then goes through synthesis and back-end design steps. The insertion points of the monitors are identified based on the timing report, when timing closure is achieved after the place and route step. The monitors are then truly inserted into the design by connecting them to the identified insertion points (i.e.,S_{inp}). Note that with this approach, there is no loading on the noncritical paths, and that the critical paths which are used to determine S_{inp} are still the true critical paths of the circuit after this step. The insertion is done by generating a TCL script which contains engineering change order commands. Then, optimization is performed to compensate for the loading effects of the monitors on the timing closure of the main circuit. The maximum delay to the monitors is set to t_{inp}, to preserve the guard banding and to avoid over-optimization of the circuit during the final optimization. Hence, in the proposed implementation flow, the number of monitors n_{mon} is decided at the front end. Since the monitored paths are the critical paths identified by $\text{Slk}_{\max}$, with a given n_{mon}, we obtain a path coverage in terms of $\text{Slk}_{\max}$. Intuitively, by increasing n_{mon}, a higher $\text{Slk}_{\max}$ is possible since more paths are monitored. Alternatively, to monitor all of the critical paths according to a specfic $\text{Slk}_{\max}$, a prior analysis of the design is required to identify n_{mon} according to $\text{Slk}_{\max}$. The analysis is, in fact, going through the flow and identifying the set of insertion points in the design, without monitors and targetting $\text{Slk}_{\max}$. n_{mon} is then identified based on the number of identified insertion points. After this analysis, n_{mon} is determined, and the monitors are added according to the flow that is shown in Fig. 5.

In this paper, the conventional worst-case design approach for setup (i.e., slow corner) is followed and the monitors are added based on timing reports in the slow corner. Inserting them at the points with a maximum arrival time of more than half of a clock cycle results in negative slack for the monitors, while the main design is still signed off with zero/positive slack. Note that in the typical corner all setup slacks increase, that the monitors' slacks also increase, but that the slack of the monitors is still less than the slack of the main design to maintain the guard banding. Furthermore, since the ranking of the most critical paths can change across corners, we select more than one path to be monitored by defining $\text{Slk}_{\max}$ as the criteria for path selection for monitoring. Therefore, one could insert the monitors in other timing corners provided that the monitored paths are selected properly (i.e., with proper selection of $\text{Slk}_{\max}$), and the slack of the monitors are made less than the slack of the main design (i.e., with the proper selection of insertion points).

V. Analysis of Insertion Point Selection

In this section, the selection of t_{inp}, as the main design knob in the proposed *in situ* monitoring technique, is discussed. First, we discuss the effect of t_{inp} on the guard banding against variations. Moreover, the design of the OR tree and its delay as another constraint for t_{inp} are discussed in this section.

A. Guard Banding

Note that when the monitor is inserted at a temporal intermediate point of the timing path, it does not capture the effect of timing spread on the remaining logical gates which are located after the insertion point. To investigate this effect, we performed 1000 Monte Carlo simulations on the SPICE netlist of a critical path of an industrial design in the 40-nm technology. The Monte Carlo simulation is performed to capture the effect of technology-dependent local and global process variations on the gate delays of the path. In Fig. 6(a), the scatter plot of delay samples is shown for a full path delay t_{full} versus the delay of the path up to the insertion point of

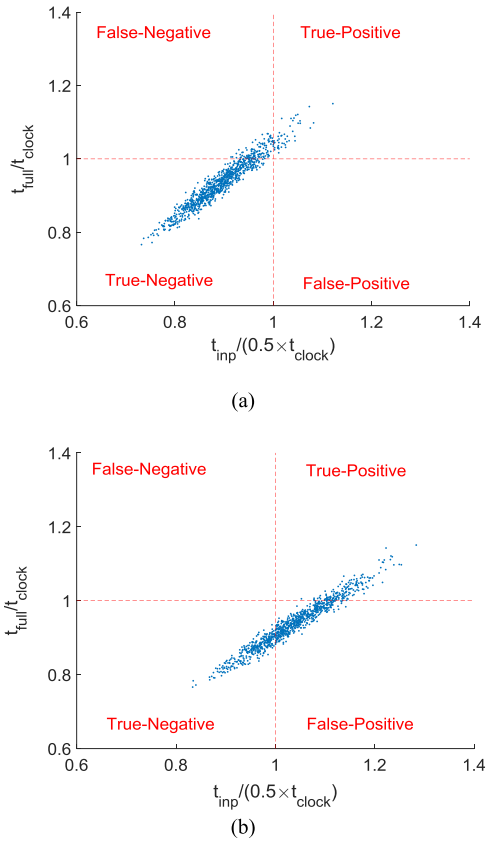

Fig. 6. Scatter plot obtained from the Monte Carlo simulations of a critical path delay of an industrial design showing t_{full} versus t_{inp} for the cases where (a) $t_{\text{inp}} = 0.5 \times t_{\text{clock}}$ and (b) $t_{\text{inp}} = 0.55 \times t_{\text{clock}}$.

the monitor t_{inp}. In Fig. 6(a), t_{full} and t_{inp} are normalized to clock period t_{clock} and $0.5 \times t_{\text{clock}}$, respectively. The dashed lines in Fig. 6(a) divide the space into four regions according to the timing checks performed on t_{inp} and t_{full}. These checks are at half of the clock cycle for t_{inp}, and at the end of the clock cycle for t_{full}. Each sample then occurs in one of these regions.

True-Negative

1) *Region*: $t_{\text{full}} < t_{\text{clock}}$ and $t_{\text{inp}} < 0.5 \times t_{\text{clock}}$. Therefore, the monitor truly detects no timing violation in the design.

True-Positive

2) *Region*: $t_{\text{full}} > t_{\text{clock}}$ and $t_{\text{inp}} > 0.5 \times t_{\text{clock}}$. Therefore, the monitor truly detects timing violation in the design.

False-Negative

3) *Region*: $t_{\text{full}} > t_{\text{clock}}$ and $t_{\text{inp}} < 0.5 \times t_{\text{clock}}$. Therefore, the monitor falsely detects no timing violation in the design.

False-Positive

4) *Region*: $t_{\text{full}} < t_{\text{clock}}$ and $t_{\text{inp}} > 0.5 \times t_{\text{clock}}$. Therefore, the monitor falsely detects the timing violation in the design.

True-negative and true-positive detections are acceptable, while false-negative and false-positive detections are not. False-negative detection must be avoided because in this case the gate delays after the insertion points may increase, but the monitors would falsely show that the system is robust. On the other hand, false-positive detection can be acceptable for chip health monitoring since it reflects delay degradation of some gates while the full path still has enough margin. In fact, during normal circuit operation, only true-negative detections are acceptable, and as the circuit degrades, it is preferred to have false-positive detections before true positive ones, to have a guard band between the detection of delay degradation with the monitors and the actual timing violation in the design. Note that this guard banding is implemented without additional design cost, by selecting the insertion points properly.

By inserting the monitor at a point closer to the endpoint, the delay to the insertion point is higher, i.e., there are more samples with $t_{\text{inp}} > 0.5 \times t_{\text{clock}}$. Therefore, the points are shifted to the right in the scatter plot, thereby more samples fall in the false-positive region. The scatter plots shown in Fig. 6(a) and (b) are for the cases in which the insertion point is selected such that the average value of t_{inp} is about $0.5 \times t_{\text{clock}}$ and $0.55 \times t_{\text{clock}}$, respectively. By inserting the monitors at the points which are after the temporal middle point of the clock period (i.e., $t_{\text{inp}} > 0.5 \times t_{\text{clock}}$), false-negative cases are avoided at the cost of more false-positive ones. In both plots, the clock period is selected as $t_{\text{clock}} = \mu_{\text{full}} + \sigma_{\text{full}}$, where μ_{full} and σ_{full} are the average and the standard deviation of the full path delay samples, respectively.

B. Recall and Precision Analysis

From a chip health perspective, and for a more thorough investigation of the effect of t_{inp} on the relevance of the monitors' outputs, we employ the precision and recall metrics from binary classification, defined as [18]

$$\text{precision} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \tag{5}$$

and

$$\text{recall} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}} \tag{6}$$

where N_{TP}, N_{FP}, and N_{FN} are the number of samples which are in the true-positive, false-positive, and false-negative regions, respectively. In Fig. 7(a), recall and precision are plotted versus t_{inp} considering $t_{\text{clock}} = \mu_{\text{full}} + \sigma_{\text{full}}$. To avoid false-negative samples, based on (6), t_{inp} is selected such that recall $= 1$, i.e., $t_{\text{inp}} \approx 0.54 \times t_{\text{clock}}$. The obtained precision at this insertion point is 0.3367. There is a tradeoff between recall and precision as t_{inp} is varied. This tradeoff is illustrated in Fig. 7(b). In Fig. 7(b), precision versus recall is plotted for three choices of the clock period: the worst case value, $t_{\text{clock}} = \mu_{\text{full}} + \sigma_{\text{full}}$, and two better than worst-case values, being 5% and 10% shorter than the worst-case clock period. The maximum precision considering recall $= 1$ is also shown in Fig. 7(b). As shown, a better than worst-case clock period results in a higher precision for the monitors. This is because with a shorter clock period more positive cases (i.e., timing violations) happen, and therefore this increases N_{TP} (i.e., higher precision). On the other hand, a longer clock period indicates that even though the monitor flags the delay increase, there is enough slack margin to avoid the timing violation at

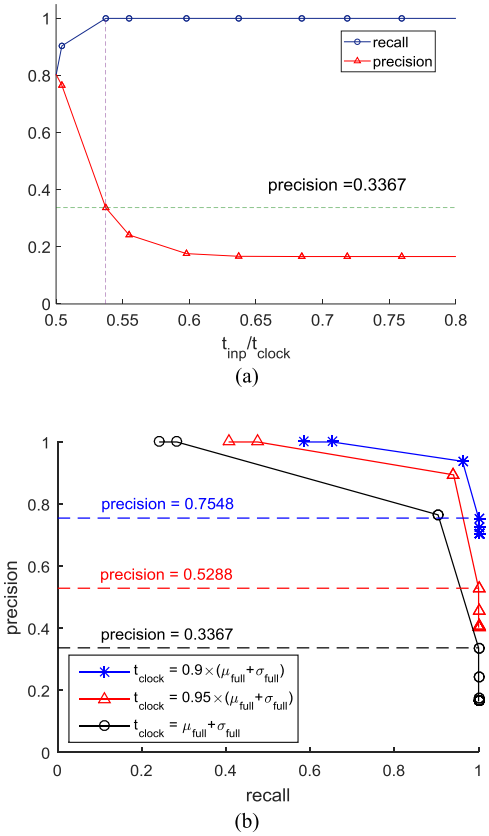

Fig. 7. Effect of t_{inp} on the precision and recall metrics. (a) Precision and recall versus t_{inp} normalized to $t_{clock} = \mu_{full} + \sigma_{full}$. The maximum precision subject to no false-negative prediction (recall = 1) is 0.3367. (b) Precision versus recall considering a different t_{clock}. In a better than worst case design (smaller t_{clock}), the precision of the monitor outputs increases.

the expense of higher N_{FP} (i.e., lower precision). Since in the proposed TS technique the clock is stretched in reaction to the output of the monitors, lower precision causes unnecessary performance penalty. Note that we kept recall = 1 because this work targets robustness (we guarantee functionality in safety critical applications despite malfunctions) during operation in the field (in real time). To reduce the cost of recall = 1 that is the lower precision, the clock can be made faster. In this way, the precision increases and the faster clock can compensate for the performance loss. A better than worst-case clock choice can thereby alleviate the performance penalty of variation resilience with our technique.

C. Inspection Window and OR Tree Delay

As introduced in Section III-A, the inspection window of the monitor is the second half of the clock cycle. During the inspection window, the output of monitor changes if there is any toggling at the insertion point after the negative clock edge. The insertion point is selected in the worst-case corner based on the maximum delay which causes the last toggling at this point. Therefore, the maximum delay to the output of monitor t_{mon} is the sum of maximum delay to the insertion point and the delay of buffers and XOR gate in the monitor [see Fig. 2(b)]. The monitoring data are thereby stable after t_{inp} to be collected with the OR tree into one bit which triggers

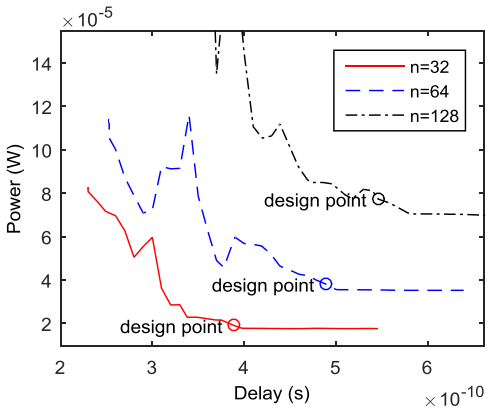

Fig. 8. Power versus delay of OR tree considering different number of inputs n. The design point is selected considering 10% increase in the power compared to the power when the delay is relatively long.

the clock stretching unit. Since the clock selection should be done before the end of cycle

$$t_{inp} + t_{or} + t_{sel} < t_{clock} \tag{7}$$

where t_{or} is the delay of the OR tree and t_{sel} is the delay of the clock selection (i.e., to generate enable signal of counter in the clock stretching unit). The OR tree delay together with the clock period thereby put a constraint on the maximum value for t_{inp}. We synthesized a 32, 64, and 128 inputs OR tree with an industrial CMOS 40-nm technology considering distinct maximum delays as the constraints. The power versus delay of the OR tree with different number of inputs are plotted in Fig. 8. The design point is selected such that the power of the OR tree circuit is 10% more than the power when the delay constraint is relatively long. Therefore, the delay of the OR tree is in the range of 390–550 ps, depending on the number of inputs. Due to the design of monitors, t_{inp} should be more than the half of the clock period. Therefore, the OR tree delay value indicates that according to (7), the proposed TS technique is suitable for the designs with the speed constraint of less than ∼1 GHz.

VI. Experimental Results

In this section, the experimental results are provided, and the effectiveness of the proposed technique is evaluated. This is done considering the computational efficiency of the proposed insertion method, the number of monitors and design overhead of the proposed monitoring scheme, and the variation resilience achieved with the proposed error prevention technique. The experiments were performed using an industrial CMOS 40-nm technology. Cadence Genus, Innovus, and Incisive were employed for frontend design, backend design, and netlist simulation, respectively.

A. Computational Efficiency of Monitor Insertion Method

To show the computational efficiency for identifying the insertion points, we benchmarked it against the path-based method using six synthesized ITC benchmark circuits. The path-based method requires the list of all paths whose slack

TABLE I

RUNTIME OF PATH-BASED AND THE PROPOSED METHOD TO IDENTIFY MONITOR INSERTION POINTS, CONSIDERING $\text{Slk}_{\text{max}} = 0.1 \times t_{\text{clock}}$, FOR DIFFERENT ICT BENCHMARK CIRCUITS SYNTHESIZED FOR 1-GHZ CLOCK FREQUENCY

Benchmark	Number of Gates	Path-based runtime (s)	Proposed method runtime (s)
B03	76	14.918	2.496
B04	266	19.605	5.665
B05	438	2026.936	13.300
B07	228	14.386	5.166
B09	104	21.071	2.930

TABLE II

IMPLEMENTED ARM CORTEX M0 DESIGN SPECIFICATIONS

Target Frequency	200MHz
Number of gates	8441
Number of flip-flops	841
Number of IOs	136
Power (mW)	2.333
Area (μm^2)	15646

is less than Slk_{max}. To get the list of the paths, the path-based timing analysis is performed. The insertion points are then identified based on the arrival time to the intermediate points of all the reported timing paths. On the other hand, with the proposed method, the set of insertion points is obtained, given the maximum arrival time to all circuit points through the block-based timing analysis. We employed both methods to find the insertion points of the monitors to cover the same set of critical timing paths for $\text{Slk}_{\text{max}} = 0.1 \times t_{\text{clock}}$. All the circuits were synthesized with a 1-GHz clock frequency. Depending on the circuit structure and size, the list of paths which should be monitored can be long or short. If the list of paths is long, the path-based method requires more time and effort to report it. In Table I, the circuit size in terms of the number of gates and the runtime of the path-based method as well as the runtime of our method are shown. Based on the results, the runtime of the path-based method is $\sim 2.8\times$ to $\sim 152\times$ more than our method, depending on the circuit structure and size.

B. Number of Insertion Points

The proposed technique is verified for *in situ* monitoring in an ARM Cortex M0 processor. The design specifications are provided in Table II for a clock period of 5 ns. The target speed of the design affects the topology of the circuit and its timing graph. Furthermore, the number of monitors varies depending on the specified t_{inp}. To investigate the effect of f_{clock} and t_{inp} on the number of monitors, t_{inp} is varied from 50% to 100% of the clock period, for a design implemented with different target frequencies of 150, 200, and 250 MHz. Fig. 9 shows the results of this experiment. A slower design is more relaxed, and the number of paths which are critical is less. Therefore, less monitors are required to monitor critical paths in a slower design. However, note that although a design with lower target speed generally requires less number of monitors, since the circuit topology may change, it can happen that for a specific

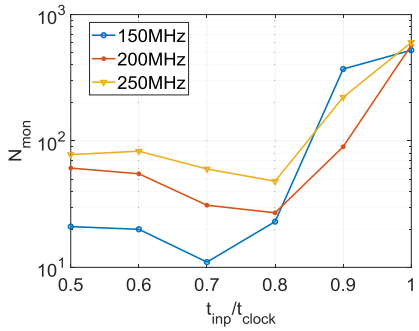

Fig. 9. Number of monitors versus delay up to insertion point normalized to clock period of the ARM Cortex M0. The core was designed targeting different frequencies, considering $\text{Slk}_{\text{max}} = 0.1 \times t_{\text{clock}}$.

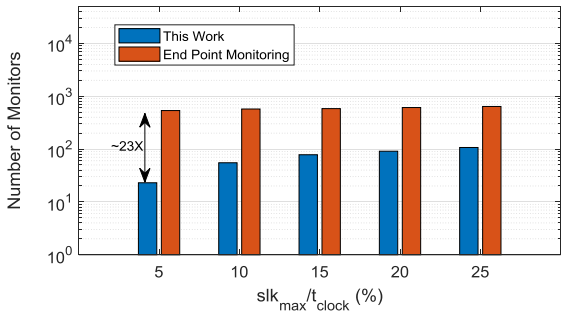

Fig. 10. Comparison between the number of monitors with our technique ($t_{\text{inp}} = 0.7 \times t_{\text{clock}}$) and the number of monitored endpoints for an ARM Cortex M0 processor considering different Slk_{max} values normalized to clock period.

$t_{\text{inp}}/t_{\text{clock}}$ ratio, the number of required monitors is high with a lower target speed. This can be observed at $t_{\text{inp}} = 0.9 \times t_{\text{clock}}$ shown in Fig. 9. Furthermore, depending on t_{inp}, there are a minimum number of monitors for a given design. The results shown in Fig. 9 are obtained considering $\text{Slk}_{\text{max}} = 0.1 \times t_{\text{clock}}$.

To show the efficiency of the proposed monitor insertion technique compared to endpoint monitoring, we swept Slk_{max}. The results for the 200-MHz design are shown in Fig. 10. One can see that the number of *in situ* monitors increases when Slk_{max} increases. Thus, the effectiveness of the proposed method is higher when Slk_{max} is smaller. With the proposed method, the number of monitors is reduced by up to $\sim 23\times$ compared to the traditional endpoint monitoring, as shown in Fig. 10.

C. Adding the Monitors to the Design

Based on the analysis discussed in the Section VI-B, we added 64 monitors to cover $\text{Slk}_{\text{max}} > 0.05 \times t_{\text{clock}}$. To compare our technique with similar methods in terms of design overhead, we also implemented two alternative techniques. In one of the techniques, the monitors are added at the capturing flip-flops. In the other techniques, the monitors are inserted almost at the end to reduce the number of monitors. In all cases, the number of monitors is fixed (64 monitors). The area and power overhead, as well as the covered slack and the number of monitored flip-flops are shown in Table III. The endpoint monitoring approach has the least coverage.

TABLE III

DESIGN OVERHEAD, MONITORED SLACK, AND NUMBER OF MONITORED FLIP-FLOPS WITH THE PROPOSED TECHNIQUE EMPLOYED IN AN ARM CORTEX M0 DESIGNED WITH A 200-MHz SPEED TARGET

Technique based on insertion point of monitors	End-point	SlackProbe [8]	This work
t_{inp}	t_{clock}	$0.9\times t_{clock}$	$0.6\times t_{clock}$
Number of monitors	64	64	64
Power Overhead (%)	10	16	2.9
Area Overhead (%)	5.05	8.8	3.15
Slk_{max}/t_{clock} (%)	2.8	5.2	8.5
Number of Monitored Flip-flops	64	467	543

Since the number of covered endpoints is equal to the number of monitors, more monitors are required to cover the same list of critical paths. However, due to the additional delay elements and the shadow flip-flop, the design overhead is also too much for 64 monitors with endpoint monitoring, when compared to our technique. In the other techniques, the monitors are inserted almost at the end of the timing paths ($t_{\text{inp}} = 0.9 \times t_{\text{clock}}$), which results in a reduced number of monitors. This allows achieving a higher coverage of critical endpoints compared to the endpoint monitoring. Moreover, in addition to the monitor cells, more delay elements for the remainder of the paths are required. Therefore, the power and area of this technique are significantly more. Based on the observed results, with our technique, power and area overhead are about 5.5× and 2.8× less, respectively.

As was mentioned in Section IV, the monitors are truly inserted in the design when their inputs are connected to the identified insertion points. The slack of the monitors is determined after connecting the monitor inputs, depends on t_{inp} in the selection of insertion points. If $t_{\text{inp}} = 0.5 \times t_{\text{clock}}$, then the worst-case slack of the monitors is 0 since the latches inside the monitors close at half of the clock cycle. If $t_{\text{inp}} > 0.5 \times t_{\text{clock}}$, the worst-case slack of the monitors is negative with respect to $0.5 \times t_{\text{clock}}$, i.e., the monitors fail earlier than the main flip-flops in the presence of timing degradation. Therefore, the design is guard banded with the monitors if $t_{\text{inp}} > 0.5 \times t_{\text{clock}}$. In Fig. 11, the slack histograms of the monitors and the main flip-flops are illustrated, considering $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$ and $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$. The shown histograms verify that higher t_{inp} results in lower slacks for the monitors. Note that the shown histogram is from a timing analysis in the slow corner. In the typical corner, the bins shift to higher slack values. The monitors must be inserted such that in a normal operation (i.e., typical corner), the slack is positive to avoid false-positive error predictions. As the delays of the circuit components degrade, the slacks decrease. Since the slack of the monitors is smaller than the slack of the main flip-flops, the monitors flag the timing degradation before a timing violation occurs at the main flip-flop. With a higher t_{inp}, the monitors start flagging earlier because their slack is smaller, as shown in Fig. 11. Therefore, false-positive error predictions are higher when a higher t_{inp} is used for monitor insertion.

We performed a netlist simulation for 10 000 cycles and obtained the number of cycles in which at least one monitor flags a warning (i.e., error prediction signal). The simulation

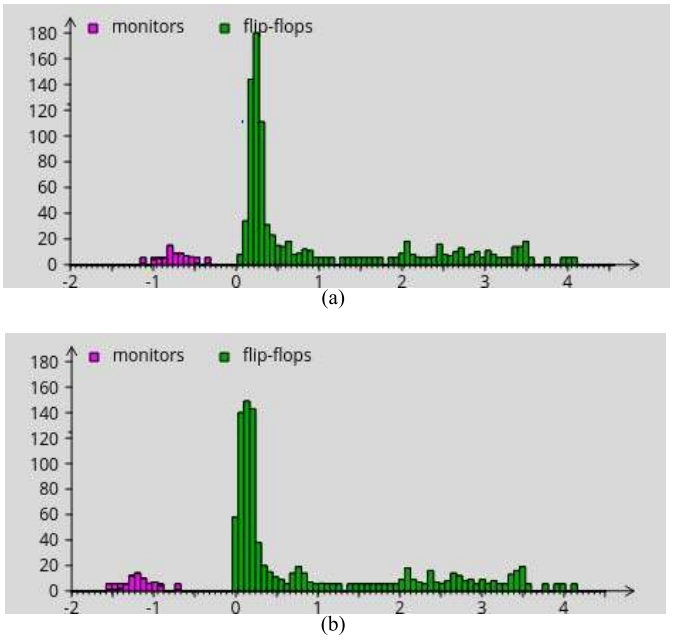

Fig. 11. Histogram of slacks of monitors as well as the main flip-flops for ARM Cortex M0 design targeting 200-MHz speed with 64 monitors considering. (a) $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$. (b)$t_{\text{inp}} = 0.7 \times t_{\text{clock}}$.

testbench runs image binarization, finite-impulse response filter, infinite-impulse response filter, and while (1) applications on the processor. Timing annotation is performed at the typical corner, and delay scaling is applied to show the delay degradation effect. To examine the effect of t_{inp} on the guard banding of monitors against variations, two cases of $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$ and $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$ were considered for monitor insertion. Fig. 12 shows the corresponding number of warnings. The ratio of the delay scaling factor for which a timing violation happens, to the smallest delay scaling factor for which the monitors generate a warning signal, reflects the guard banding of the monitors against delay variation. As shown in Fig. 12(a), with $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$, the design is guard banded by 1.38×. Similarly, it can be observed from Fig. 12(b) that with $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$, the design is guard banded by 1.5×. Therefore, with larger t_{inp}, more guard banding is added with the monitors.

D. Timing Error Prevention System

The proposed error prevention system is implemented by connecting the output of the OR tree, which collects monitor outputs, to the clock stretching unit. If an error is predicted by a given monitor, the next clock cycle starts with a delay (i.e., the clock edge is shifted) to avoid the timing error. In Fig. 13, the simulation waveforms showing the output of the OR tree (W) and the system clock are illustrated. The clock edge is shifted by 1.25 ns ($= t_{\text{clock}}/4$) if W is high at the end of cycle. This means that the execution time of a task on the processor is delayed if an error prevention is performed during its execution. On the other hand, the effective power consumption decreases with error prevention due to clock stretching as it decreases the effective frequency

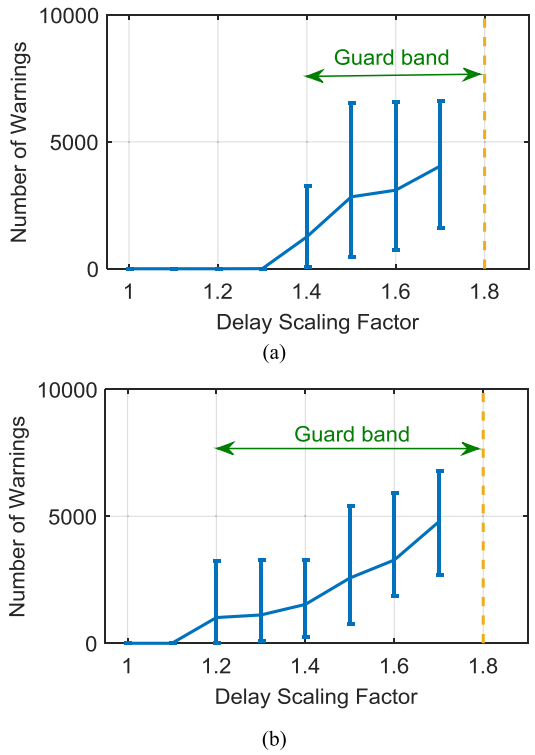

Fig. 12. Number of cycles with warnings generated by the monitors versus the delay degradation factor. Results are of four applications obtained based on a netlist simulation for 10k cycles with timing annotation at the typical corner and scaling the delays. The insertion points of 64 monitors are identified considering (a) $t_{\text{inp}} = 0.55 \times t_{\text{clock}}$ and (b) $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$.

of the design. For instance, if at least one monitor flags a warning in each cycle, the clock is stretched for every cycle resulting in an effective clock period which is 25% longer. Therefore, the circuit works with a 20% lower clock frequency and consequently the power consumption decreases. Furthermore, putting less guard banding by having smaller t_{inp} prevents unnecessary error preventions during error-free operation thereby the performance penalty is less. In Fig. 14(a) and (b), the speed, power, and energy overhead of the proposed error prevention system is illustrated with $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$ and $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$, respectively. With $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$, no clock stretching occurs up to 1.3× delay scaling. If the delay is scaled to higher values, the clock stretching occurs in some cycles and the speed and energy overhead starts to increase. Note that the power overhead reduces due to the decreased effective clock frequency from stretching the clock. For $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$, the clock stretching starts from a lower delay scaling factor of 1.1×, because the slack of the monitors is less, i.e., the monitors are more pessimistic. It can be observed that with the proposed error prevention scheme, the system can tolerate more delay degradation compared to the original one. The resilience with $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$ is less compared to $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$, because according to (7) with higher t_{inp}, the output of the monitors becomes false negative for a lower delay scaling factor, i.e., the monitor misses to capture the late transition at the insertion point. Overall, the results show that with the proposed TS technique,

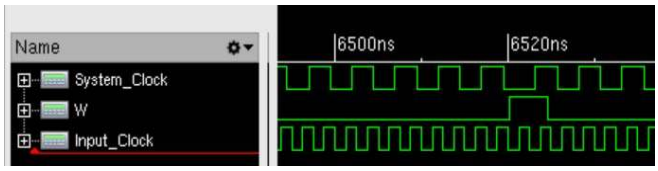

Fig. 13. Netlist simulation waveforms showing system clock being stretched by the clock stretching unit in response to a warning signal W generated by the monitors.

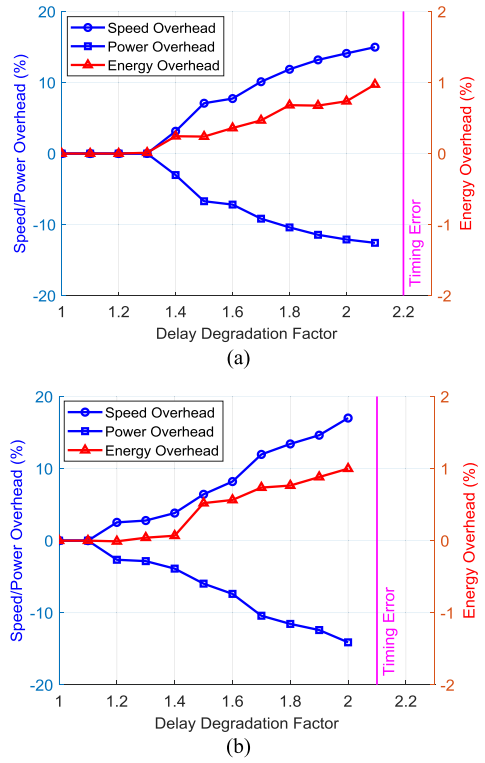

Fig. 14. Speed, power, and energy penalty due to clock stretching versus the delay degradation obtained from a netlist simulation of 10k cycles with 'timing annotation in the typical corner. The insertion points of 64 monitors are identified considering (a) $t_{\text{inp}} = 0.6 \times t_{\text{clock}}$ and (b) $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$.

the resilience is increased by up to ∼22% with a negligible energy overhead of less than 1%.

E. Scalability of the Technique

To assess the proposed methodology on a larger design, we applied it to an ARM Cortex M3 processor which has more than ∼40 000 cells in the 40-nm technology (post place and route) targeting 200-MHz speed. Similar to the results of Fig. 10, Fig. 15(a) shows the ones for the ARM Cortex M3. It can be observed that the proposed technique requires less monitors than the endpoint monitoring technique. Assuming that $\text{Slk}_{\text{max}} = 0.05 \times t_{\text{clock}}$ is intended, with the endpoint monitoring technique ∼2000 monitors are required while with our technique, the same slack coverage is obtained with less than 300 monitors even for $t_{\text{inp}} = 0.7 \times t_{\text{clock}}$. Therefore, compared to the endpoint monitoring technique, more than 7× reduction in the number of monitors is achieved for the same path coverage ($\text{Slk}_{\text{max}} = 0.05 \times t_{\text{clock}}$). Furthermore, the power

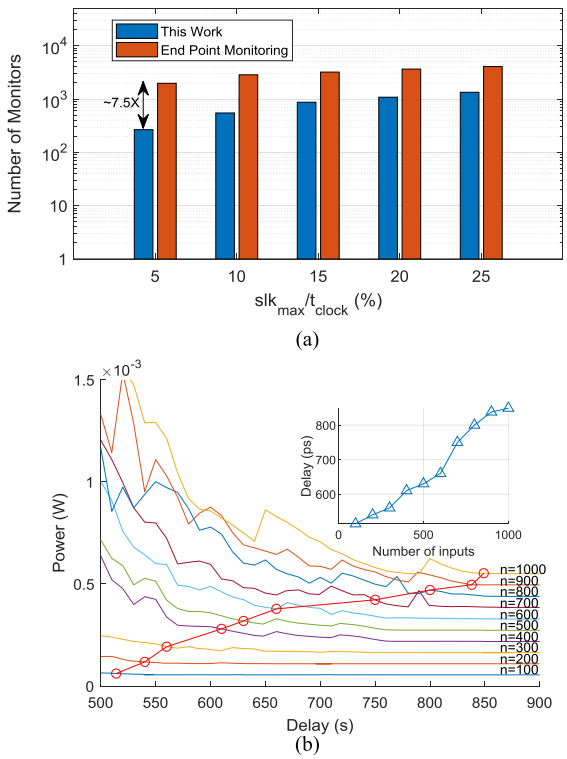

Fig. 15. Scalability of technique. (a) Comparison between the number of monitors with our technique ($t_{\text{inp}} = 0.7 \times t_{\text{clock}}$) and the number of monitored endpoints for an ARM Cortex M3 processor considering different Slk_{max} values normalized to clock period. (b) Delay of OR tree versus the number of inputs based on the power-delay characteristic.

and area overhead of our *in situ* monitoring technique are ∼3% and ∼1.9%, respectively.

Since in a larger design, the number of required monitors increases, the OR tree should also scale up accordingly. According to (7), the sum of the error prediction delay, OR tree delay, and clock selection delay should fit into one clock cycle. Hence, the OR tree delay can become a bottleneck when the design is too big or the target speed is too high. In Fig. 15(b), the delay of the OR tree is shown based on its power-delay characteristic considering different input counts. It can be seen that for 1000 inputs, the delay of the OR tree is less than 1 ns. According to (7) and assuming t_{sel} is negligible, for $t_{\text{or}} = 1$ ns and $t_{\text{mon}} \approx t_{\text{inp}} = 0.6 \times t_{\text{clock}}$, the lower boundary for t_{clock} is 2.5 ns, i.e., the maximum speed according to (7) is 400 MHz. On the other hand, with almost at the endpoint monitoring (e.g., $t_{\text{mon}} \approx t_{\text{inp}} = 0.9 \times t_{\text{clock}}$), for the same OR tree delay, the maximum achievable speed is 100 MHz [according to (7) and assuming negligible t_{sel}]. Therefore, our TS technique allows higher clock speed not only because it allows a reduction in the number of monitors (i.e., lower OR tree delay) but also because it leaves more room for the OR tree delay when inserting the monitors deeper inside the timing paths.

VII. Conclusion

A new TS system is proposed in which the monitors are inserted along timing paths and timing errors are predicted within one clock cycle by checking if the delay of the first part of a path has increased such that an undesired transition happens during the second half the clock cycle. For an ARM Cortex M0 design, we found that the insertion points of the monitors are identified with a novel technique which is more than 2.8× faster compared to techniques that rely on the path-based timing analysis. Moreover, the number of required monitors is up to 23× less compared to the traditional endpoint monitoring. The power and area overheads of monitoring delay variation in the ARM Cortex M0 design are reduced by 5.5× and 2.8×, respectively, with our technique compared to the other *in situ* monitoring techniques. Furthermore, the variation resilience is improved by ∼22% with our error prevention technique. The proposed error prevention technique is based on a global clock stretching module connected to the output of an OR tree which collects the monitor outputs. The energy overhead of the proposed error prevention technique is less than 1% which is negligible compared to the state-of-the-art error recovery techniques. For designs with higher gate count, the number of monitors' scales up but according to our experience with ARM Cortex M3, the number of monitors with our technique is less compared to endpoint monitoring (∼7.5×). Furthermore, our *in situ* monitoring technique is more compatible with within-cycle error prevention since it allows inserting the monitors deeper inside paths and thereby makes OR tree delay constraint more relaxed.

References

[1] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *Proc. 25th IEEE VLSI Test Symp. (VTS)*, May 2007, pp. 277–286.

[2] J. Tschanz, K. Bowman, S. Walstra, M. Agostinelli, T. Karnik, and V. De, "Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance," in *Proc. IEEE Symp. VLSI Circuits (VLSIC)*, Jun. 2009, pp. 112–113.

[3] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *Proc. IEEE VLSI Test Symp. (VTS)*, Apr. 1999, pp. 86–94.

[4] S. Das *et al.*, "RazorII: In situ error detection and correction for PVT and SER tolerance," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[5] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken, "TIMBER: Time borrowing and error relaying for online timing error resilience," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2010, pp. 1554–1559.

[6] S. M. Londoño and J. Pineda de Gyvez, "A better-than-worst-case circuit design methodology using timing-error speculation and frequency adaptation," in *Proc. IEEE Int. SOC Conf.*, Niagara Falls, NY, USA, Sep. 2012, pp. 15–20.

[7] L. Lai, V. Chandra, R. C. Aitken, and P. Gupta, "SlackProbe: A flexible and efficient *in situ* timing slack monitoring methodology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 8, pp. 1168–1179, Aug. 2014.

[8] 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 Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2010, pp. 188–189.

[9] H. A. Balef, H. Fatemi, K. Goossens, and J. Pineda de Gyvez, "Effective In-Situ chip health monitoring with selective monitor insertion along timing paths," in *Proc. 28th ACM Great Lakes Symp. (VLSI)*, Chicago, IL, USA, 2018, pp. 213–218.

[10] V. Mahalingam, N. Ranganathan, and R. Hayman, "Dynamic clock stretching for variation compensation in VLSI circuit design," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 8, pp. 16:1–16:13, Aug. 2012.

[11] M. Nejat, B. Alizadeh, and A. Afzali-Kusha, "Dynamic flip-flop conversion: A time-borrowing method for performance improvement of low-power digital circuits prone to variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2724–2727, Nov. 2015.

[12] H. Cherupalli and J. Sartori, "Graph-based dynamic analysis: Efficient characterization of dynamic timing and activity distributions," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, Nov. 2015, pp. 729–735.
[13] K. Chae and S. Mukhopadhyay, "A dynamic timing error prevention technique in pipelines with time borrowing and clock stretching," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 74–83, Jan. 2014.
[14] S. Guntur, G. Al-Kadi, R. I. Mechtildis, P. Meijer, J. Hoogerbrugge, and H. Fatemi, "Clock selection circuit and method," U.S. Patent 8 836 379 B2, Sep. 16, 2014.
[15] W. Shan *et al.*, "A low overhead, within-a-cycle adaptive clock stretching circuit with wide operating range in 40-nm CMOS," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 65, no. 11, pp. 1718–1722, Nov. 2018.
[16] J. Zhou *et al.*, "HEPP: A new in-situ timing-error prediction and prevention technique for variation-tolerant ultra-low-voltage designs," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Singapore, Nov. 2013, pp. 129–132.
[17] *Cadence Tempus (TM) Timing Signoff Solution*, Cadence Design Systems, San Jose, CA, USA, 2018.
[18] M. Shah and N. Japkowicz, *Evaluating Learning Algorithms: A Classification Perspective*, Cambridge, U.K.: Cambridge Univ. Press, 2011.

Hadi Ahmadi Balef received the B.S. degree in electrical engineering from the Amirkabir University of Technology, Tehran, Iran, and the M.S. degree in electronics engineering from the University of Tehran, Tehran. He is currently working toward the Ph.D. degree in the Electronic Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands.

His current research interests include low-power and reliable digital integrated circuits design.

Hamed Fatemi received the B.Sc. degree from the Electrical and Computer Engineering Department, University of Tehran, Tehran, Iran, in 1998, the M.Sc. degree from KNT, Tehran, in 2001, and the Ph.D. degree in computer architecture from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2007.

He is currently the Innovation Leader/Department Manager at NXP Semiconductors, Eindhoven, The Netherlands. He has authored or coauthored over 25 U.S. patents, scientific publications, and presentations. His current research interests include the low-power design, multiprocessors, heterogeneous and reconfigurable systems, and variability tolerance design.

Kees Goossens was with NXP Semiconductors, Eindhoven, where he focused on real-time networks on chip for consumer electronics, on-chip communication protocols, and memory management. He is currently a Full Professor of Real-Time Embedded Systems with the Electronic Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. He focuses on composable (virtualized), predictable (real-time), low-power embedded systems, and supporting multiple models of computation. He investigated the formal verification of hardware, in particular by using semi-automated proof systems in conjunction with formal semantics of hardware description languages such as ELLA and VHDL. He led the team that defined the Aethereal network on chip for consumer electronics. He has authored over 24 patents, four books, and 175 articles, with four paper awards.

Prof. Goossens has been an Editorial Board Member of the *ACM Transactions on Design Automation of Electronic Systems* since 2009, and an Associate Editor of the *Journal of Design Automation of Embedded Systems* (Springer) since 2006. He has been a Guest Editor for several special issues on networks on chip.

José Pineda de Gyvez (F'09) received the Ph.D. degree from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1991.

He was a Tenured Faculty Member with the Department of Electrical Engineering, Texas A&M University, College Station, TX, USA. He is currently a Fellow at NXP Semiconductors, Eindhoven. He also holds the professorship "Resilient Nanoelectronics" at the Department of Electrical Engineering, Eindhoven University of Technology. He has authored numerous papers in the fields of testing, nonlinear circuits, and low power design, coauthored several books, and a number of U.S. granted patents.

Dr. Gyvez has served as an Associate Editor for several IEEE Transactions and is continuously involved in technical program committees of scientific symposia.