

# 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

**Abstract**— 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 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  $\sim 23X$ , power by  $\sim 5.5X$ , and area by  $\sim 2.8X$  compared to 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,  $\sim 22\%$  more delay variation is tolerated with a negligible energy overhead of less than  $\sim 1\%$ .

**Index Terms**—In-situ delay monitoring, variation resilience, timing error, timing speculation.

## I. INTRODUCTION

CMOS technology scaling has been the driving force of the semiconductor industry for decades. In the nano-scale era, however, the increased effects of variability on design robustness can nullify the benefits of technology scaling. A variety of physical phenomena, such as aging effects (*e.g.* hot carrier injection and bias temperature instability), voltage droops, process variations, etc., are 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 (ICs) are over-designed to improve their robustness by designing at worst-case corners. Fortunately, the cost of 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 run-time. Chip health monitoring is based on physical sensors [1], replica paths [2], or in-situ delay monitors [3] [4] [5] [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:

- 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 guard banding against variations, inspection window generation, and short path extension is removed. Thereby, power and area overhead of in-situ monitoring is low, compared to the state of the art (see *e.g.* [4] [7]).
- 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 block-based timing analysis, suitable for large circuits. The proposed method is  $\sim 2.8X$  faster for small circuits with  $\sim 100$  gates, and it is orders of magnitude faster for large circuits (more than  $\sim 1K$  gates).
- 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  $\sim 25\%$  more delay variation tolerance with negligible energy overhead compared to state of the art error masking techniques (see *e.g.* [4] [8]).

A prior version of this work was published in [9] where the design and insertion flows of within path in-situ delay monitors were introduced. This work 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 timing error 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.

H. Ahmadi Balef, K. Goossens, are with Eindhoven University of Technology, De Rondom 70, 5612 AP Eindhoven, The Netherlands (e-mails: h.ahmadi.balef@tue.nl, k.g.w.goossens@tue.nl). H. Fatemi is with NXP Semiconductors, High Tech Campus 46, 5656 Eindhoven, The Netherlands (e-mail: hamed.fatemi@nxp.com). J. Pineda de Gyvez is with both Eindhoven University and NXP Semiconductors (e-mail: j.pineda.de.gyvez@tue.nl).

In what follows, first, an overview of the state of the art is done in section II. Then, the proposed timing speculation 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 concludes the paper.

## II. PRELIMINARIES AND RELATED WORKS

In synchronous digital circuits, data propagates from primary inputs or from 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. 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 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 Figure 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 to G6, and ends into the capturing flip-flop. Conventionally, in-situ delay monitors are inserted at the end-point 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], 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 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

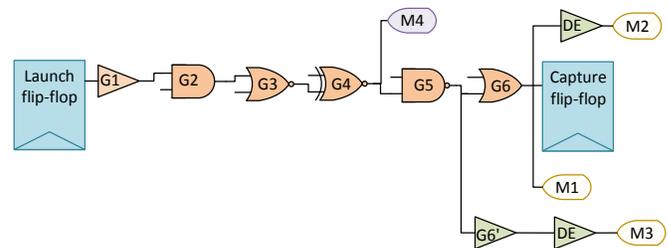


Figure 1- The insertion points of monitors along the timing paths of digital circuits for different in-situ delay monitoring techniques.

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 are accounted for by adding a pessimistic delay margin between the insertion point and the input of the monitor (see G6' in Figure 1). Note that the guard band is still applied by adding another delay margin before the monitor (see DE before M3 in Figure 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 Figure 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 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 end-point. 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

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. The authors of [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 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 e.g. [13]). Nevertheless, the clock stretching unit in those works are PLL-based and normally takes more than one clock cycle to take effect (see e.g. [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 its efficient micro-architectural implementation.

### III. THE PROPOSED TIMING SPECULATION SYSTEM

The proposed TS system consists of in-situ monitors, of an OR tree, and of a clock stretching unit, as shown in Figure 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 ( $W$  in Figure 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 Figure 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 XOR gate, and two buffers (B1 and B2 in Figure 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.

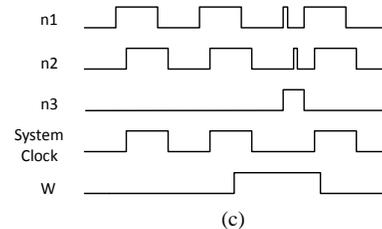
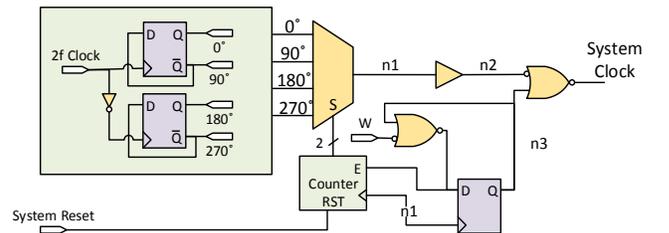
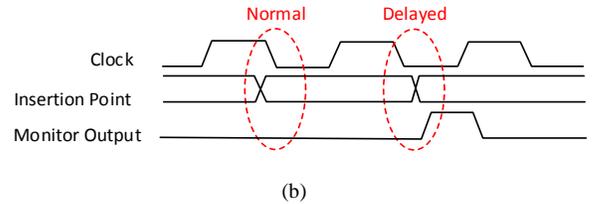
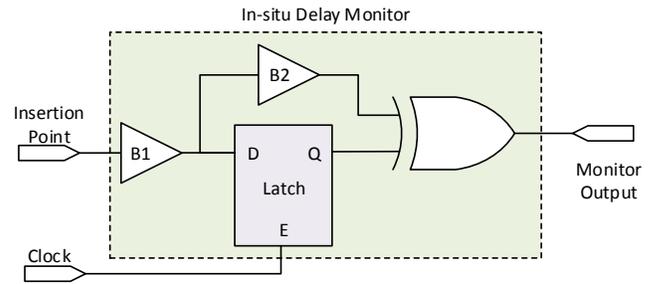
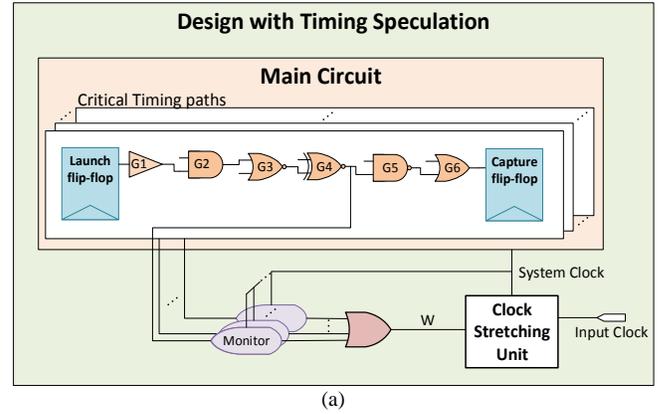


Figure 2- The proposed timing speculation technique. (a) High level view of the technique, (b) the design of in-situ monitor, and (c) the design of clock stretching unit.

#### B. Clock Stretching Unit

The proposed clock stretching unit is illustrated in Figure 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  $360^\circ/n$ . We considered four clock phases indicating that the clock phase can be shifted by  $90^\circ$ . The four phases are generated based on an input clock pulse whose frequency is  $2X$  greater than the clock frequency of the main design, as shown in Figure 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 Figure 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 Figure 2-(c)) and delaying the output of MUX (see n2 in Figure 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 *e.g.* [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 40nm technology. The proposed clock stretching unit has a power consumption of  $1.126\mu W$  at the nominal supply voltage (*i.e.* 1.1V) to generate a 200MHz clock pulse at the output. Furthermore, the area of the proposed circuit is  $96\mu m^2$ . 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,  $Slk_{max}$ , 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  $(t_{clock} - Slk_{max})$ . This definition is illustrated in Figure 3.

The timing graph of a digital circuit is a weighted directed

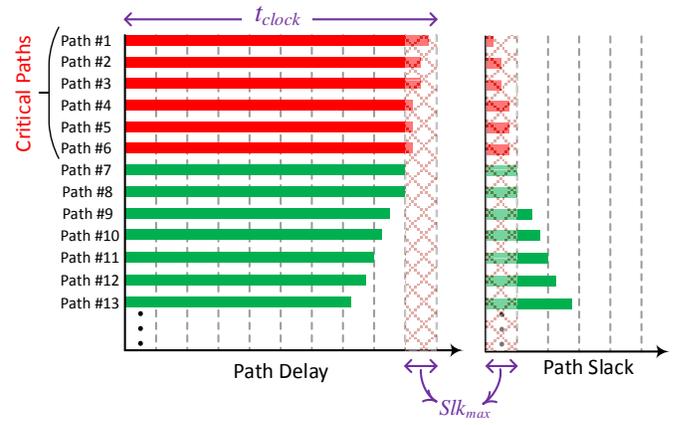


Figure 3- The definition of critical paths based on  $Slk_{max}$ .

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 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, 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

$$S_1 = \{v_i \in V(G) \mid d_i < t_{inp}\}, \quad (1)$$

and

$$S_2 = \{v_i \in V(G) \mid d_i \geq t_{inp}\}. \quad (2)$$

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 Figure 4, a timing graph and the cut are illustrated. The set of boundary vertices,  $S_{bnd} \subset S_1$ , is defined as

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

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 those vertices. The set of insertion points is therefore

$$S_{inp} = \{v_i \in V(G) \mid v_i \in S_{bnd} \wedge Slk_i < Slk_{max}\}, \quad (4)$$

where  $S_{bnd}$  is obtained from (3), and  $Slk_i \in SLK$ . Once the set of insertion points is identified from (4), the monitors are inserted

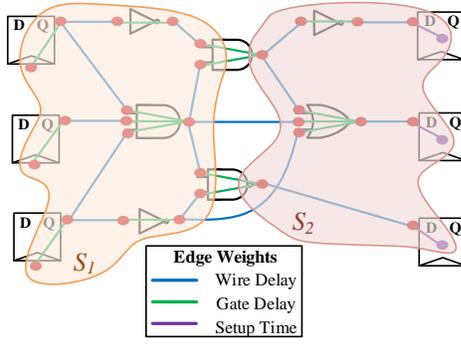


Figure 4- An example circuit graph showing the cut  $C=(S_1, S_2)$ .

at the corresponding gate pins of all vertices in  $S_{inp}$ , according to the flow which is explained in the next subsection.

### B. Integration in the Design Flow

The monitors are conventionally added to the design after the place and route step (see *e.g.* [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 Figure 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 non-critical 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 (ECO) 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 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  $Slk_{max}$ , with a given  $n_{mon}$ , we obtain a path coverage in terms of  $Slk_{max}$ . Intuitively, by increasing  $n_{mon}$ , a higher  $Slk_{max}$  is possible since more paths are monitored. Alternatively, to monitor all of the critical paths according to a specific  $Slk_{max}$ , a prior analysis of the design is required to identify  $n_{mon}$  according to  $Slk_{max}$ . The analysis is in fact going through the flow and identifying the set of insertion points in the design, without monitors and targeting  $Slk_{max}$ .  $n_{mon}$  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 Figure 5.

In this work, the conventional worst case design approach for

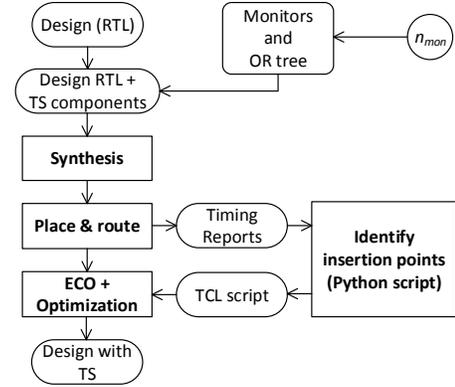


Figure 5- The proposed implementation flow.

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  $Slk_{max}$  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  $Slk_{max}$ ) and the slack of the monitors are made less than the slack of the main design (*i.e.* with 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 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

Notice 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 40nm 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 Figure 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 the monitor  $t_{inp}$ . In this figure,  $t_{full}$  and  $t_{inp}$  are normalized to clock period  $t_{clock}$  and  $0.5 \times t_{clock}$ , respectively. The dashed lines in the figure 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

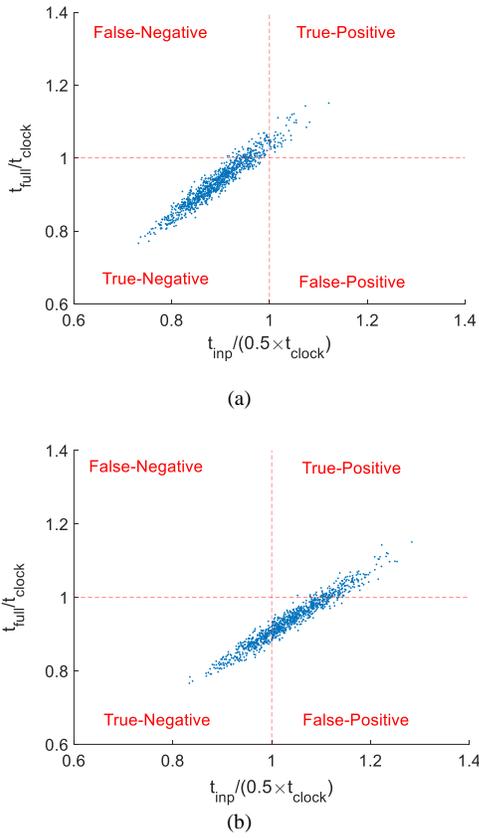


Figure 6- Scatter plot obtained from Monte Carlo simulations of a critical path delay of an industrial design showing  $t_{full}$  versus  $t_{inp}$  for the cases where (a)  $t_{inp} = 0.5 \times t_{clock}$ , and (b)  $t_{inp} = 0.55 \times t_{clock}$ .

sample then occurs in one of these regions

- *True-Negative region*:  $t_{full} < t_{clock}$  and  $t_{inp} < 0.5 \times t_{clock}$ . Therefore, the monitor truly detects no timing violation in the design.
- *True-Positive region*:  $t_{full} > t_{clock}$  and  $t_{inp} > 0.5 \times t_{clock}$ . Therefore, the monitor truly detects timing violation in the design.
- *False-Negative region*:  $t_{full} > t_{clock}$  and  $t_{inp} < 0.5 \times t_{clock}$ . Therefore, the monitor falsely detects no timing violation in the design.
- *False-Positive region*:  $t_{full} < t_{clock}$  and  $t_{inp} > 0.5 \times t_{clock}$ . Therefore, the monitor falsely detects timing violation in the design.

True-Negative and True-Positive detection is 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

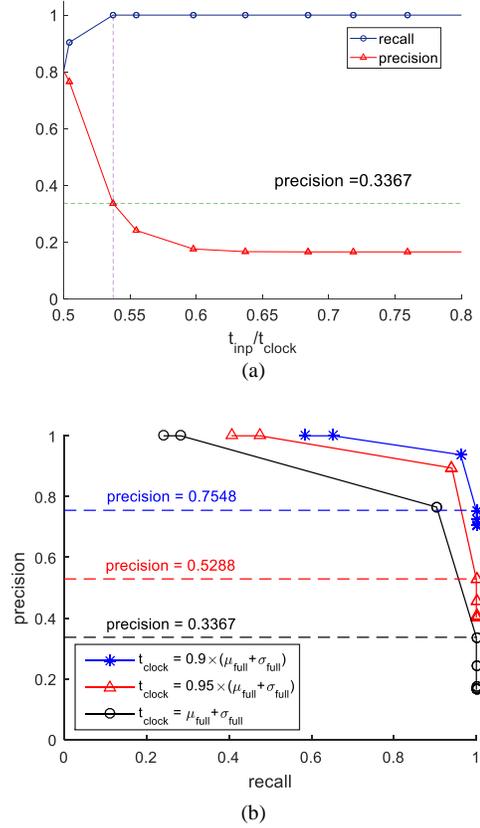


Figure 7- The effect of  $t_{inp}$  on the *precision* and *recall* metrics. (a) *precision* and *recall* versus  $t_{inp}$  normalized to  $t_{clock} = \mu_{full} + \sigma_{full}$ . The maximum *precision* subject to no False-Negative prediction (*recall*=1) is 0.3367. (b) *precision* versus *recall*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

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 end-point the delay to the insertion point is higher, *i.e.* there are more samples with  $t_{inp} > 0.5 \times t_{clock}$ . 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 Figure 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  $0.5 \times t_{clock}$  and  $0.55 \times t_{clock}$ , respectively. By inserting the monitors at the points which are after the temporal middle point of the clock period (*i.e.*  $t_{inp} > 0.5 \times t_{clock}$ ), False-Negative cases are avoided at the cost of more False-Positive ones. In both plots, the clock period is selected as  $t_{clock} = \mu_{full} + \sigma_{full}$ , where  $\mu_{full}$  and  $\sigma_{full}$  are the average and the standard deviation of 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]

$$precision = \frac{N_{TP}}{N_{TP} + N_{FP}}, \quad (5)$$

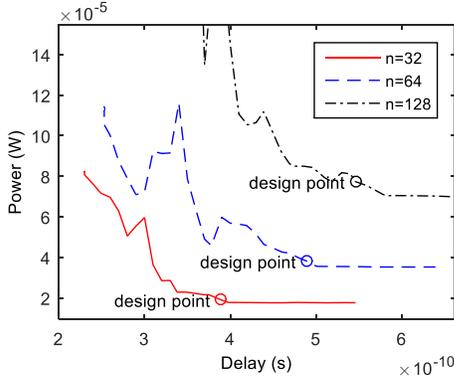


Figure 8- The power versus delay of OR Tree considering different number of inputs,  $n$ . The design point is selected considering 10% increase in the power compared to the power when the delay is relatively long.

and

$$recall = \frac{N_{TP}}{N_{TP} + N_{FN}}, \quad (6)$$

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 Figure 7-(a), recall and precision are plotted versus  $t_{inp}$  considering  $t_{clock} = \mu_{full} + \sigma_{full}$ . To avoid False-Negative samples, based on (6)  $t_{inp}$  is selected such that  $recall = 1$ , i.e.  $t_{inp} \approx 0.54 \times t_{clock}$ . The obtained *precision* at this insertion point is 0.3367. There is a trade-off between *recall* and *precision* as  $t_{inp}$  is varied. This trade-off is illustrated in Figure 7-(b). In Figure 7-(b), *precision* versus *recall* is plotted for three choices of the clock period: the worst case value,  $t_{clock} = \mu_{full} + \sigma_{full}$ , 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 the figure. 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 another 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 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 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

Table 1- The runtime of path-based and the proposed method to identify monitor insertion points, considering  $Slk_{max} = 0.1 \times t_{clock}$ , for different ICT benchmark circuits synthesized for 1GHz 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

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 Figure 2-(b)). The monitoring data is thereby stable after  $t_{inp}$  to be collected with the OR tree into one bit which triggers the clock stretching unit. Since the clock selection should be done before the end of cycle,

$$t_{inp} + t_{OR} + t_{sel} < t_{clock}, \quad (7)$$

where  $t_{OR}$  is the delay of OR tree and  $t_{sel}$  is the delay of clock selection (i.e. to generate enable signal of Counter in 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 40nm technology considering distinct maximum delays as the constraints. The power versus delay of the OR tree with different number of inputs are plotted in Figure 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 390ps to 550ps, depending on the number of inputs. Due to the design of monitors,  $t_{inp}$  should be more than 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 ~1GHz.

## VI. EXPERIMENTAL RESULTS

In this section, 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 40nm technology. Cadence Genus, Innovus, and Incisive were employed for frontend design, backend design, and netlist simulation, respectively.

### A. Computation 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 ICT benchmark circuits. The path based method requires the list of all paths whose slack is less than  $Slk_{max}$ . To get the list of the paths, path-based timing

Table 2 - The 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 ( $\mu m^2$ )	15646

analysis is performed. The insertion points are then identified based on the arrival time to the intermediate points of all reported timing paths. On the other hand, with the proposed method, the set of insertion points are obtained, given the maximum arrival time to all circuit points through 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  $Slk_{max}=0.1 \times t_{clock}$ . All the circuits were synthesized with a 1GHz 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 1, 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  $\sim 2.8X$  to  $\sim 152X$  more than our method, depending on the circuit structure and size.

### B. The 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 2 for a clock period of 5ns. 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 150MHz, 200MHz, and 250MHz. Figure 9 illustrates 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  $t_{inp}/t_{clock}$  ratio the number of required monitors is high

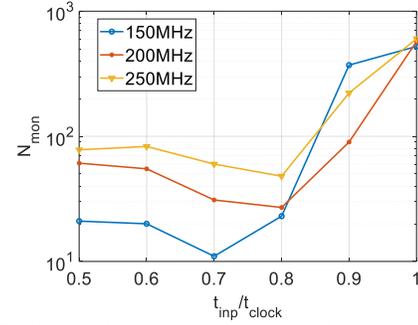


Figure 9- The number of monitors versus delay up to insertion point normalized to clock period of the ARM Cortex M0. The core was designed targeting different frequencies, considering  $Slk_{max}=0.1 \times t_{clock}$ .

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

To show the efficiency of the proposed monitor insertion technique compared to end-point monitoring, we swept  $Slk_{max}$ . The results for the 200MHz design are shown in Figure 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  $\sim 23X$  compared to the traditional end-point monitoring, as shown in Figure 10.

### C. Adding the monitors to the design

Based on the analysis discussed in the previous subsection, we added 64 monitors to cover  $Slk_{max}>0.05 \times t_{clock}$ . 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

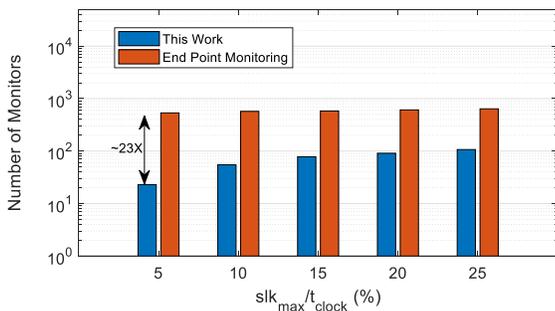


Figure 10- Comparison between the number of monitors with our technique ( $t_{inp}=0.7 \times t_{clock}$ ) and the number of monitored end-points for an ARM Cortex M0 processor considering different  $Slk_{max}$  values normalized to clock period.

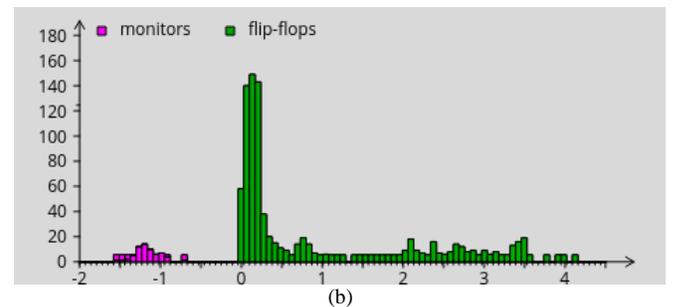
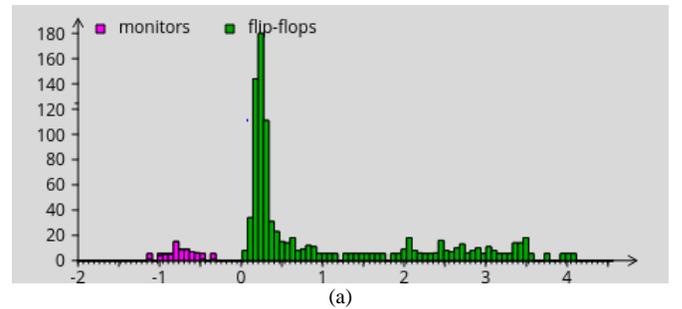


Figure 11- The histogram of slacks of monitors as well as the main flip-flops for ARM Cortex M0 design targeting 200MHz speed with 64 monitors considering (a)  $t_{inp}=0.6 \times t_{clock}$ , and (b)  $t_{inp}=0.7 \times t_{clock}$ .

Table 3 - Design overhead, monitored slack, and the number of monitored flip-flops with the proposed technique employed in an ARM Cortex M0 designed with a 200MHz 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

flip-flops. In the other technique, 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 3. The end point monitoring approach has the least coverage. Since the number of covered end points 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 end-point monitoring, when compared to our technique. In the other technique, the monitors are inserted almost at the end of the timing paths ( $t_{inp}=0.9 \times t_{clock}$ ), which results in a reduced number of monitors. This allows to achieve a higher coverage of critical end-points compared to the end-point 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 is significantly more. Based on the observed results, with our technique, power and area overhead is about 5.5X and 2.8X less, respectively.

As was mentioned before 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  $t_{inp} = 0.5 \times t_{clock}$ , then the worst-case slack of the monitors is zero since the latches inside the monitors close at half of the clock cycle. If  $t_{inp} > 0.5 \times t_{clock}$  the worst-case slack of the monitors is negative w.r.t.  $0.5 \times t_{clock}$ , 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  $t_{inp} > 0.5 \times t_{clock}$ . In Figure 11, the slack histograms of the monitors and the main flip-flops are illustrated, considering  $t_{inp}=0.6 \times t_{clock}$  and  $t_{inp}=0.7 \times t_{clock}$ . 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 Figure 11. Therefore, False-Positive error predictions are higher when a higher  $t_{inp}$  is used for monitor insertion.

We performed a netlist simulation for 10K cycles and obtained the number of cycles in which at least one monitor flags a warning (i.e. error prediction signal). The simulation 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  $t_{inp}=0.6 \times t_{clock}$  and  $t_{inp}=0.7 \times t_{clock}$  were considered for monitor insertion. Figure 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 Figure 12-(a), with  $t_{inp}=0.6 \times t_{clock}$  the design is guard banded by 1.38X. Similarly, it can be observed from Figure 12-(b) that with  $t_{inp}=0.7 \times t_{clock}$  the design is guard banded by 1.5X.

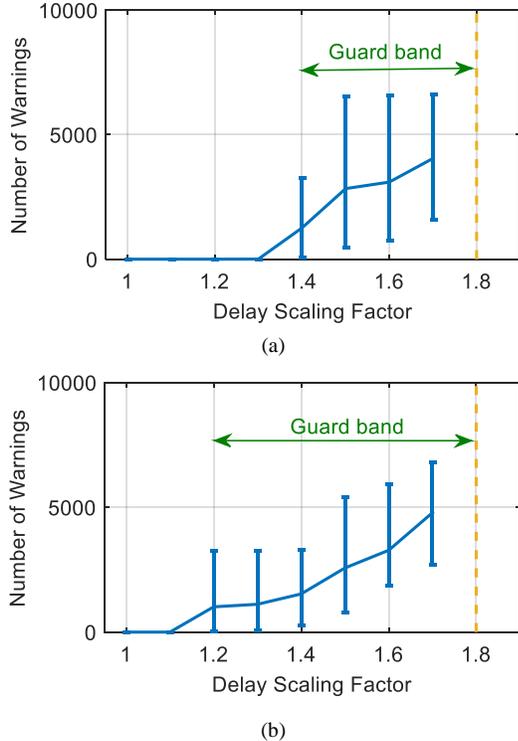


Figure 12- The number of cycles with warnings generated by the monitors versus the delay degradation factor. Results are of four applications obtained based on a netlist simulation for 10K cycles with timing annotation at the typical corner and scaling the delays. The insertion points of 64 monitors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

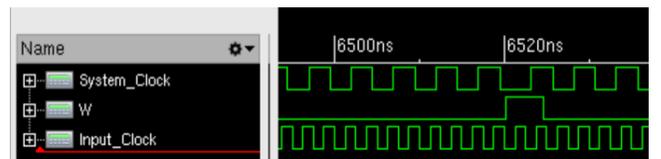


Figure 13- Netlist simulation waveforms showing System Clock being stretched by the clock stretching unit in response to a warning signal, W, generated by the monitors.

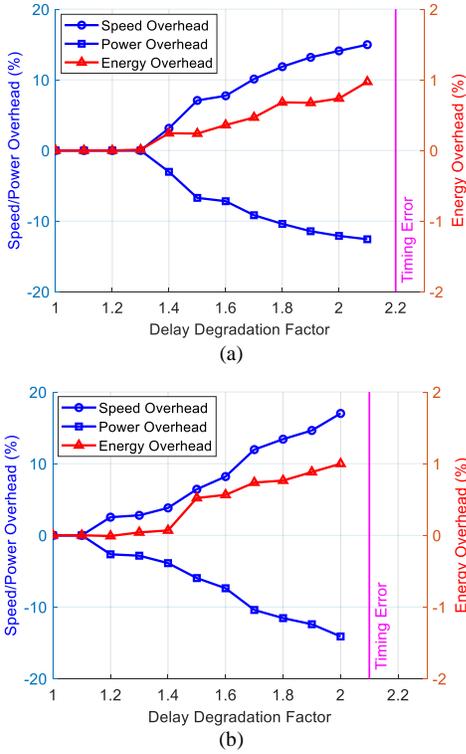


Figure 14- Speed, power, and energy penalty due to clock stretching versus the delay degradation obtained from a netlist simulation of 10K cycles with timing annotation in the typical corner. The insertion points of 64 monitors are identified considering (a)  $t_{imp}=0.6\times t_{clock}$ , and (b) 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

Therefore, with larger  $t_{imp}$  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 Figure 13, the simulation waveforms showing the output of the OR tree (W) and the system clock are illustrated. The clock edge is shifted by 1.25ns ( $=t_{clock}/4$ ) 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 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  $t_{imp}$  prevents unnecessary error preventions during error free operation thereby the performance penalty is less. In Figure 14- (a) and (b) the speed, power, and energy overhead of the proposed error prevention system is illustrated with  $t_{imp}=0.6\times t_{clock}$  and  $t_{imp}=0.7\times t_{clock}$ , respectively. With  $t_{imp}=0.6\times t_{clock}$ , no clock stretching occurs up to 1.3X delay

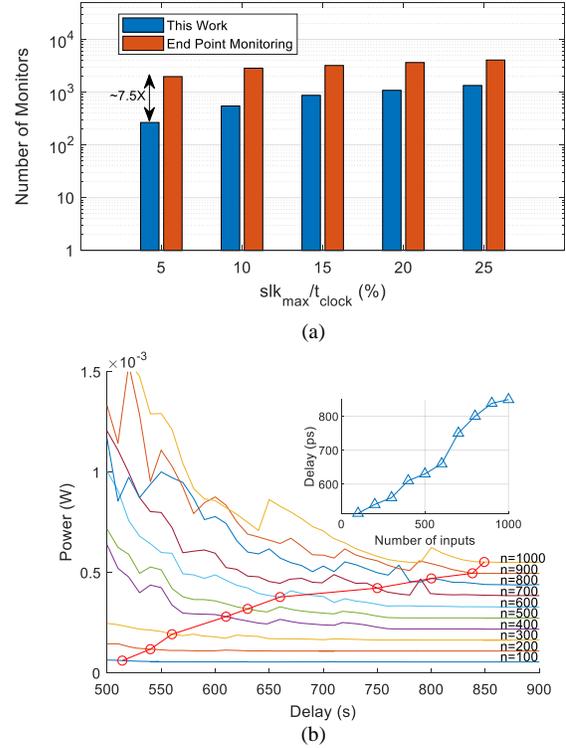


Figure 15- The scalability of technique. (a) Comparison between the number of monitors with our technique ( $t_{imp}=0.7\times t_{clock}$ ) and the number of monitored end-points for an ARM Cortex M3 processor considering different  $Slk_{max}$  values normalized to clock period. (b) Delay of OR tree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

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  $t_{imp}=0.7\times t_{clock}$ , the clock stretching starts from a lower delay scaling factor of 1.1X, 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  $t_{imp}=0.7\times t_{clock}$  is less compared to  $t_{imp}=0.6\times t_{clock}$ , because according to (7) with higher  $t_{imp}$  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 the resilience is increased by up to  $\sim 22\%$  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  $\sim 40K$  cells in 40nm technology (post place and route) targeting 200MHz speed. Similar to the results of Figure 10, Figure 15- (a) displays the ones for the ARM Cortex M3. It can be observed that the proposed technique requires less monitors than the end point monitoring technique. Assuming that  $Slk_{max}=0.05\times t_{clock}$  is intended, with the end point monitoring technique  $\sim 2000$  monitors are required while with our

technique, the same slack coverage is obtained with less than 300 monitors even for  $t_{imp}=0.7\times t_{clock}$ . Therefore, compared to the end point monitoring technique, more than 7X reduction in the number of monitors is achieved for the same path coverage ( $Slk_{max}=0.05\times t_{clock}$ ). Furthermore, the power and area overhead of our in-situ monitoring technique are  $\sim 3\%$  and  $\sim 1.9\%$ , 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 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 Figure 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 1ns. According to (7) and assuming  $t_{sel}$  is negligible, for  $t_{OR}=1\text{ns}$  and  $t_{mon}\approx t_{imp}=0.6\times t_{clock}$  the lower boundary for  $t_{clock}$  is 2.5ns, i.e. the maximum speed according to (7) is 400MHz. On the other hand, with almost at the end point monitoring (e.g.  $t_{mon}\approx t_{imp}=0.9\times t_{clock}$ ), for the same OR tree delay the maximum achievable speed is 100MHz (according to (7) and assuming negligible  $t_{sel}$ ). Therefore, our timing speculation 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 timing speculation 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 2.8X faster compared to techniques that rely on path-based timing analysis. Moreover, the number of required monitors is up to 23X less compared to the traditional end-point monitoring. The power and area overheads of monitoring delay variation in the ARM Cortex M0 design is reduced by 5.5X and 2.8X, respectively, with our technique compared to the other in-situ monitoring techniques. Furthermore, the variation resilience is improved by  $\sim 22\%$  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 end point monitoring ( $\sim 7.5X$ ). 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.

## ACKNOWLEDGEMENT

This research has been partially supported by EU grant CATRENE RESIST (2015/CATRENE/CT217), and EU grant ECSEL SILENSE (2017/ECSEL/00737487).

## REFERENCES

- [1] M. Agarwal, B. C. Paul, M. Zhang and S. Mitra, "Circuit Failure Prediction and Its Application to Transistor Aging," in *Proceedings of IEEE VLSI Test Symposium (VTS'07)*, 2007.
- [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 *Proceedings of IEEE Symposium on VLSI Circuits (VLSIC'09)*, 2009.
- [3] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," in *Proceedings of IEEE VLSI Test Symposium (VTS'99)*, 1999.
- [4] S. Das, C. Tokunaga, S. Pant, W. H. Ma, S. Kalaiselvan, K. Lai, D. M. Bull and D. T. Blaauw, "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 44, pp. 32-48, 2009.
- [5] M. Choudhury, V. Chandra, K. Mohanram and R. Aitken, "TIMBER: Time Borrowing and Error Relaying for Online Timing Error Resilience," in *Proceedings of Design Automation and Test in Europe Conference and Exhibition (DATE'10)*, Dresden, 2010, pp. 1554-1559.
- [6] S. M. Londono and J. Pineda de Gyvez, "A Better-Than-Worst-Case Circuit Design Methodology Using Timing-Error Speculation and Frequency Adaptation," in *Proceedings of IEEE International SOC Conference*, Niagara Falls, NY, 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, pp. 1168-1179, 2014.
- [8] D. Fick, N. Liu, Z. Foo, M. Fojtik, J. s. Seo, D. Sylvester and D. Blaauw, "In Situ Delay-Slack Monitor for High-Performance Processors Using An All-Digital Self-Calibrating 5ps Resolution Time-to-Digital Converter," in *Proceedings of IEEE International Solid-State Circuits Conference - (ISSCC)*, San Francisco, CA, 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 *Proceedings of the 28th ACM Great Lakes Symposium on VLSI*, Chicago, 2018, pp. 213-218.
- [10] V. Mahalingam, N. Ranganathan and R. Hayman, "Dynamic Clock Stretching for Variation Compensation in VLSI Circuit Design," *ACM Journal on Emerging*

*Technologies in Computing Systems (JETC)*, vol. 8, pp. 16:1-16:13, 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 Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 23, pp. 2724 - 2727, 2014.
- [12] H. Cherupalli and J. Sartori, "Graph-based Dynamic Analysis: Efficient Characterization of Dynamic Timing and Activity Distributions," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD'15)*, Austin, TX, 2015, pp. 729-735.
- [13] K. Chae and Saibal Mukhopadhyay, "A Dynamic Timing Error Prevention Technique in Pipelines With Time Borrowing and Clock Stretching," *IEEE Transactions on Circuits and Systems I*, 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". United States of America Patent US8836379B2, 16 09 2014.
- [15] W. Shan, L. Wan, X. Liu, X. Shang, W. Dai, S. Shao, J. Yang and L. Shi, "A Low Overhead, Within-a-cycle Adaptive Clock Stretching Circuit with Wide Operating Range in 40nm CMOS," *IEEE Transactions on Circuits and Systems II*, vol. 65, no. 11, pp. 1718-1722, Nov. 2018.
- [16] J. Zhou, X. Liu, Y.-H. Lam, C. Wang, K.-H. Chang, J. Lan and M. Je, "HEPP: A new in-situ timing-error prediction and prevention technique for variation-tolerant ultra-low-voltage designs," in *2013 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Singapore, 2013 , pp. 129-132.
- [17] *Cadence Tempus (TM) Timing Signoff Solution*, Cadence Design Systems, 2018.
- [18] M. S. Nathalie Japkowicz, *Evaluating Learning Algorithms: A Classification Perspective*, Cambridge University Press, 2011.

#### BIOGRAPHIES



**Hadi Ahmadi Balef** received the B.S. degree in electrical engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, the M.S. degrees in electronics engineering from the University of Tehran, Tehran, Iran. He is currently working towards the Ph.D. degree in the Electronic Systems

group in the department of Electrical Engineering of Eindhoven University of Technology, Eindhoven, The Netherlands. His research interests include low power and reliable digital integrated circuits design.



**Hamed Fatemi** received the BSc and MSc degrees from the Electrical and Computer Engineering Department of the University of Tehran, Tehran, Iran, and KNT, and the PhD degree in computer architecture from the Eindhoven University of Technology, Eindhoven, The Netherlands in 1998, 2001, and 2007, respectively. He is

Innovation lead / Department manager at NXP Semiconductors. His research interests include the areas of low-power design, multi-processors, heterogeneous and reconfigurable systems, and variability tolerance design. Fatemi has authored and co-authored more than 25 US patents, scientific publications and presentations.



**Kees Goossens** is Full Professor of Real-time Embedded Systems in the Electronic Systems group of the Department of Electrical Engineering at Eindhoven University of Technology (TU/e). He focuses on composable (virtualized), predictable (real-time), low-power embedded systems, supporting multiple models of computation. At Topic

Embedded Products, Goossens works on real-time dependable dynamic partial reconfiguration in FPGAs. Early in his research, 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. At NXP Semiconductors, Goossens worked on real-time networks on chip for consumer electronics, then on on-chip communication protocols and memory management. He led the team that defined the Aethereal network on chip for consumer electronics. Goossens has been editorial board member for the ACM Transactions on Design Automation of Electronic Systems (TODAES) since 2009 and associate editor for the Springer Journal of Design Automation of Embedded Systems (DAEM) since 2006, and he has been guest editor for several special issues on networks on chip. He is author of 24 patents, and he has published four books and over 175 articles, with four paper awards.



**Jose Pineda de Gyvez (F'09)** received the Ph.d. degree from the Eindhoven University of Technology, The Netherlands, in 1991. He was a tenured Faculty member in the Department of Electrical Engineering at Texas A&M University, USA. Currently, he holds the position of Fellow at NXP Semiconductors. He also holds the

professorship "Resilient Nanoelectronics" in the Department of Electrical Engineering at the Eindhoven University of Technology, The Netherlands. Pineda has served as Associate Editor in several IEEE Transactions and is continuously involved in technical program committees of scientific symposia. Pineda has published numerous papers in the fields of testing, nonlinear circuits, and low power design, He has co-authored several books, and a number of US granted patents.