Path Selection and Sensor Insertion Flow for Age Monitoring in FPGAs

Mohammad Ebrahimi[1], Zana Ghaderi[2], Eli Bozorgzadeh[2], Zain Navabi[1]

[1]Electrical and Computer Engineering Department
University of Tehran
{ebrahimi.mo, navabi}@ut.ac.ir

[2]Computer Science Department
University of California Irvine
{zghaderi, eli}@ics.uci.edu

***Abstract*— This paper presents a two-step aging-aware methodology for *Representative Critical Paths* (RCPs) selection from a large number of *Critical Paths* (CPs) in programmable logic devices. First, nomination of CPs is based on delay, temperature, and lexicographic function of duty cycle and switching activity filtering, which are the major causes in *Bias Temperature Instability* (BTI) and *Hot Carrier Injection* (HCI) aging mechanisms. Secondly, RCPs will be selected based on *Fan-out* (FO) and physical location of *Logic Blocks* (LBs) along a CP to decrease aging propagation and sensor distribution fairness, respectively. We then present a sensor insertion algorithm that will be used during design placement to avoid sensors inaccuracy. Implementation steps of sensor insertion are performed automatically with a limited human interaction. Higher aging-rate of RCPs than unselected CPs in our experiments demonstrates the effectiveness of the proposed methodology.**

Keywords*— *Aging, FPGA, path selection, placement.

I. INTRODUCTION

By aggressive down scaling semiconductor technology, designing a reliable system becomes challenging. BTI and HCI are two important phenomena causing accelerated transistor aging [1, 2], which increase the magnitude of the transistor threshold voltage and reduce the effective carrier mobility over time. Moreover, the rate at which supply voltage is scaling is lower than that of transistor size scaling. This results in an increase in current density and temperature, which causes acceleration in device degradation in future semiconductor technologies. Aging effects not only considerably reduce the lifetime of the chip, but also cause timing violations due to delay degradation (Δd) in transistor delay. Once the delay of *Critical Paths* (CPs) exceeds the clock period, the correct functionality of the circuit is affected and timing failures happen. Designers consider avoiding such a failure by on-chip critical path delay monitoring using low overhead on-chip sensors in reconfigurable architectures [2, 3].

The impact of aging and various solutions have been introduced in the context of ASIC-based design. However, limited research has been done for FPGA devices [4, 5]. As opposed to ASIC designs where designers are able to place sensors on a chip at the design time, placement of sensors to monitor critical paths in FPGAs needs to be customized for each design (rather than fixed locations). Each resource on FPGA can be potentially in a critical path of target applications due to different temperature and *Stress Rate* (SR) maps. Hence, sensor insertion and allocation need to be applied during place and route phase. Aging sensor insertion in FPGAs is challenged by various factors: 1) Due to high resource utilization locally (e.g., in a slice in Xilinx Virtex FPGA), sensor allocation at nearest location to a critical path may not be possible, and 2) Due to a large pool of critical paths in a design, it is impossible to allocate an aging sensor for each critical path, and hence, only a subset of CPs must be selected for age monitoring.

This paper presents a two-step methodology to find a list of *Representative Critical Paths* (RCPs) among CPs. At first, paths with delay values close to critical path delay are selected. This set of critical paths, named as *Pseudo Critical Path* (PCP) in the rest of this paper, are selected to have higher activity than CPs. Due to higher temperature and stress rate along PCPs, aging is manifested sooner on these paths. In addition, the delay overhead of sensors along PCPs does not cause timing violations compared to sensors inserted on critical paths due to their intrinsic timing slack.

In our proposed RCP selection algorithm, aging prone path candidates for age monitoring, are first selected. This is based on path delay (of PCPs), temperature, duty cycle, and switching activity. In the next step, a subset of candidates (RCPs) will be selected based on *Fan-out* (FO), and physical location of path endpoint in *Logic Blocks* (LBs). For Fan-outs, paths with larger FO will be selected. In other words, while path delays, temperature, duty cycles, and activities are given higher priorities to identify aging-prone paths, FOs and sensor distribution play a secondary role in selecting representative paths for age monitoring. We will then present a sensor insertion algorithm that will be used during placement phase to improve sensor accuracy to monitor the path delay. To the best of our knowledge this is the first work that considers different characteristics of implemented circuit to find reliable numbers of RCPs among a large number of CPs for aging monitoring in reconfigurable architectures.

The rest of the paper is organized as follows: the previous works are explained in Section II. Section III presents the aging models followed by path selection methodology in Section IV.

Section V is dedicated to sensor placement. Experimental results for benchmark are shown in Section VI. Finally, Section VII concludes the paper.

II. Related work

Sensor-based online monitoring of aging-induced delay degradation in reconfigurable architectures have been proposed in [2, 3], while no strategy is presented for critical path selection and sensor insertion in FPGAs. Aging-aware path selection strategies for ASIC design have been proposed in [6, 7]. The approach proposed in [7] selects a small set of CPs considering the delay degradation of the entire circuit due to aging, and not the local causes of aging mechanisms such as temperature or stress rate. Two machine learning-based feature selection approaches are utilized in [6] to find RCPs for aging monitoring in ASIC design. Since the device level information of FPGAs is not available, these proposed methods cannot be deployed in the reconfigurable system design. Besides, these methods do not propose any placement strategy for sensor insertion to tackle the challenges in reconfigurable system design.

Some approaches against aging-induced delay degradation in reconfigurable systems at runtime are proposed in [8, 9]. The method [8] does not propose any path selection strategy and placement. To guide the online placement of modules on FPGAs, an algorithmic based approach determines the aging of a region instead of using sensors [10], which not only imposes performance overhead but also will not be as accurate as using sensors for online monitoring. Proposed methods for offline aging mitigation in reconfigurable systems [1,10,11] are out of scope of this paper.

III. Aging in Reconfigurable Architecture

Aging impact on transistors manifests as delay degradation (Δd). The delay degradation model of the BTI mechanism on transistor's switching delay (d) can be simplified as [12-14]:

$$\Delta d_{BTI}(t) = A_{BTI} \times (Y)^n \times (t)^n \times e^{(\frac{-E_a}{kT})} \times d_0 \quad (1)$$

Where d_0 is the pre-aged delay of transistor, A_{BTI} is technology dependent factor, t is the transistor age, Y is the duty cycle of transistor, T is temperature, E_a is activation energy, k is Boltzmann's constant, and n is constant depending on fabrication process. Delay degradation model of HCI mechanism on transistor's switching delay (d) can be simplified as [12-14]:

$$\Delta d_{HCI}(t) = A_{HCI} \times \alpha \times f \times (t)^{0.5} \times e^{(\frac{-E_b}{kT})} \times d_0 \quad (2)$$

Where d_0 is the pre-aged delay of transistor, A_{HCI} is technology dependent factor, t is the transistor age, α is the activity factor of transistor, f is clock frequency, T is temperature, E_b is activation energy, and k is Boltzmann's constant. According to Eq. 1 and Eq. 2, delay degradation in both BTI and HCI mechanisms is an exponential function of temperature and a nonlinear function of stress (duty cycle in BTI or switching activity in HCI). Since the transistor level information of the FPGA is not disclosed by vendors, we deployed the proposed method in [4] to calculate aging of nodes along a CP. Node is a basic logic element and its corresponding routing resources (Slice and NET in Xilinx Virtex). We assume that all transistors inside a node experience similar stress and temperature.

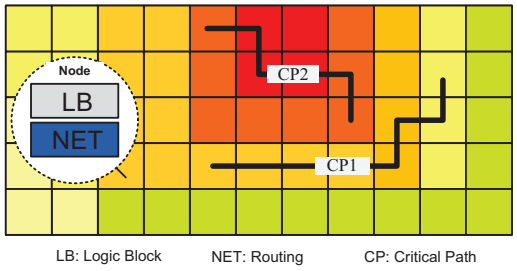


Fig. 1. Temperature map impact on critical path aging

As shown in Fig. 1, each node along a path may experience different temperatures (the darker the color, the higher the temperature), hence leading to different aging rates for the nodes along the path. For example, if all nodes have equal delay (one unit of time) and experience equal stress rate, path CP2's delay may exceed CP1's over time due to different temperature maps even if at the beginning $Delay_{CP1} > Delay_{CP2}$.

IV. Path Selection Methodology

A. Path Characteristics

Aging-induced delay degradation along each path originates from the delay degradation of each node along it. Each node ages differently from other nodes for various reasons. The aging in each node depends on functional complexity in logic delay as well as Fan-out (FO) in corresponding net delay. In addition, given that temperature and stress on chip varies from one region to another, the temperature and stress at each node varies as well. In this section, we summarize the aging-related path characteristics that are deployed for representative aging-prone critical path selection algorithm.

Considering Fig. 1, assume each CP i consists of m_i nodes. Each node i can be interpreted as a sextuple $(d_{ij}, T_{ij}, \alpha_{ij}, Y_{ij}, Ph_i, FO_{ij})$ where d is delay, T is temperature, α is the activity factor, Y is duty cycle, Ph is physical location, and FO is Fan-out. These parameters play an important role in the aging rate and are computed as follow:

1) Delay

The total delay of a critical path is equal to:

$$\boldsymbol{Delay\ (CP\ i)} = \sum_{j=1}^{m_i} d_{ij} \quad (3)$$

Due to resource limitation and exponential number of paths, it is impossible to monitor all of the circuit's paths. In addition, power and area overhead for each sensor should be taken into account. Hence, we need to select paths which are more prone to aging and the aging-induced delay degradation along those paths are more likely to cause timing violations. The pseudo-

critical paths are those with delay in the range of α% to β% of minimum clock period. These bounds may change based on the sensor delay to avoid performance loss, implemented circuit delay, and number of CPs.

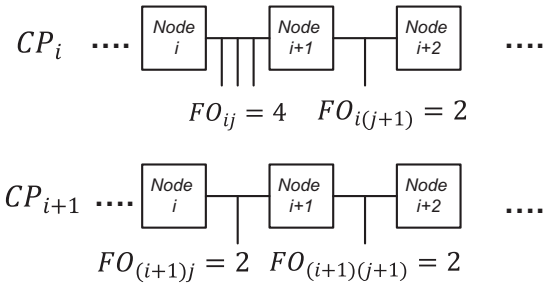


Fig. 2: Impact of Fan-out (FO) for path selection

2) Temperature

The exponential impact of temperature on both BTI and HCI mechanisms are unavoidable. In order to compute the aging rate at each node along CPs, we need to consider the temperature in the corresponding region. Therefore, the temperature map of implemented design on FPGA is extracted adapting HotSpot tool [15]. Unlike ASIC design, temperature map of the chip changes by different configurations on the FPGA. This leads to the fact that FPGA resources on FPGA based on the implemented configuration experience different aging rates at different times. As shown in Eq. 4, the temperature of a path is the average temperature of its nodes.

$$\boldsymbol{Temp(CP\ i)} = \frac{\sum_{j=1}^{m_i} T_{ij}}{m_i} \quad (4)$$

3) Stress Rate

Based on Eq. 1 and Eq. 2 *Stress Rate (SR)* is equal to Y in BTI mechanism and equal to $\alpha \times f$ in HCI mechanism. BTI is a static mechanism and is triggered when transistor in ON. HCI is a dynamic mechanism and occurs when transistor toggles. In both cases, transistor is under stress and hence, aging causes delay degradation. Each node along a path may experience different stress rates which results to different stress maps among different configurations implemented on the FPGA. This consideration comes from the fact that paths close to critical path delay but with relatively lower stress rate may not be prone to aging as aggressively as others. As shown in Eq. 5 each path's stress rate is a lexicographical function of activity rate ($\alpha * f$) and duty cycle (Y). C_1 and C_2 are chosen based on relation impact of BTI and HCI on aging-induced delay (C_2 is three times greater than C_1).

$$\boldsymbol{Stress\ (CP\ i)} = \frac{\sum_{j=1}^{m_i} C_1 \times (\alpha_{ij} \times f) + C_2 \times Y_{ij}}{m_i} \quad (5)$$

4) Fan-out

In addition to closeness between CPs, another factor to be considered in our path selection algorithm is Fan-out (FO) of nodes along the path. Assume CP_i and CP_{i+1} in Fig. 2 experience similar aging rate while based on Eq. 6 $FO\ (CP_i)$ is larger than $FO\ (CP_{i+1})$ ($FO\ (CP\ i) = 6$). The probability of aging propagation in paths with larger FO is higher. FO calculated based on Eq. 6.

ALGORITHM 1. RCP selection

Input: Critical paths list {CP}, Delay matrix $\vec{d}$, Activity matrix $\vec{\alpha}$, Duty cycle matrix $\vec{Y}$, Power consumption matrix $\vec{P}$, Clock frequency f, Fan-out matrix $\overrightarrow{FO}$, Path endpoints list {Ph}

Output: List of representative critical paths {RCP}

```
1.  RCPlist = {};
2.  for each Path_i ∈ {CP} do
3.      D_i ← CalDelay (Path_i, d⃗);                    // Eq. 3
4.      Stress_i ← CalStress (Path_i, α_i, f, Y_i);     // Eq. 5
5.      FO_i ← CalFO(Path_i, FO⃗);                       //Eq. 6
6.  End for
7.  delay_range ← CalDelayRange();
8.  for each Path_i ∈ {CP} do                          // Extract PCPs
9.      if Path_i-delay ∈ delay_range then
10.         RCPlist.Add(Path_i);
11. End for
12. T⃗ ← FindTemp (P⃗);                                 // Call HotSpot
13. for each Path_i do
14.     Temp_i ← CalAvgTemp(Path_i, T⃗);                // Eq. 4
15. End for
16. T_th ← CalTempThreshold();
17. for each Path_i ∈ RCPlist do
18.     if Temp_i < T_th then
19.         RCPlist.Remove(Path_i);
20. End for
21. Stress_th ← CalStressThreshold();
22. for each Path_i ∈ RCPlist do
23.     if Stress_i < stress_th then
24.         RCPlist.Remove(Path_i);
25. End for
26. FO_th ← CalFOThreshold();
27. for each Path_i ∈ RCPlist do
28.     if FO_i < FO_th then
29.         RCPlist.Remove(Path_i);
30. End for
31. RCPlist.Remove (Ph);
```

$$\boldsymbol{FO\ (CP\ i)} = \sum_{j=1}^{m_i} FO_{ij} \quad (6)$$

5) Physical location

Due to high number of such paths and the sensor overhead, only a subset of the paths is selected as representative paths. To fairly distribute the limited number of sensors on the chip, physical location of nodes along the path, especially the node that contains the endpoint *flip-flop*, is required to be considered. Location filtering is for avoiding redundant and unnecessary sensor. Someone may argue all the filtering phases are not for aging mechanism such as location but it needs to be considered to further optimize the number of required aging sensors.

B. Path Selection Algorithm

Algorithm 1 shows the pseudo code for the proposed critical path selection method. The algorithm inputs are the activity matrix $\vec{\alpha}$, duty cycle matrix $\vec{Y}$, power consumption matrix $\vec{P}$, clock frequency f, and matrix of each node Fan-out along each critical path $\overrightarrow{FO}$. The objective of this algorithm is to find the near minimum number of RCPs. The algorithm is decomposed of two major steps. First, finding the aging-prone paths (finding PCPs, temperature filtering, and stress rate

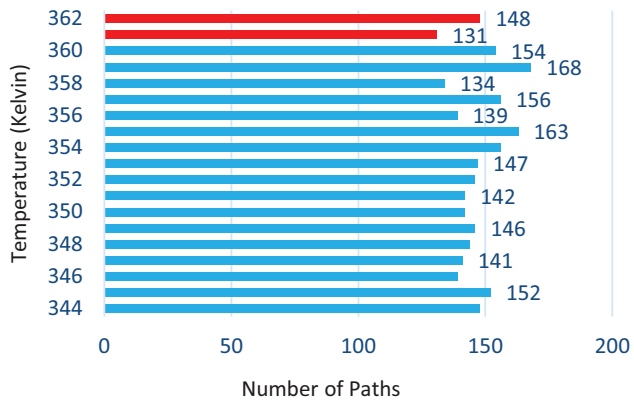


Fig. 3. S38417 benchmark temperature histogram

filtering), followed by finding representative critical paths for sensor insertion (Fan-out and endpoint physical location filtering).

Based on the fact that sensor inserting on *Pseudo Critical Paths* (PCPs) does not lead to performance loss (no increase on CP delay), our proposed algorithm concentrates on PCPs instead of critical paths. The pseudo-code in line 8-11 identifies PCPs. $delay_{range}$ is the range of critical path delay allowed for selecting CPs in the first round. The delay upper bound is computed by the negation of critical path delay by the sensor delay and the lower bound delay is defined by the user (85% of critical path delay).

At the next filtering, we consider temperature at each node along the selected paths (line 12) which is a dominant factor in both BTI and HCI mechanism. The *FindTemp()* function calls HotSpot tool [15] for extracting the circuit's temperature map. Temperature will be calculated at node level as shown in Fig. 1, a path may route through different temperature zones, which leads to different aging rates along a path. If average temperature along $Path_i$ (line 14) is less than a threshold (T_{th}), the path will be filtered and removed from the candidates (lines 17-20). Selecting the appropriate value for T_{th} is a crucial factor that heavily depends on temperature distribution of paths and varies from application to application. For example, Fig. 3 shows the path temperature distribution of S38417 benchmark.

Among the paths, 131 paths have average temperature close to 361 and 148 paths have average temperature of 362 (Kelvin). T_{th} is set to 361 Kelvin to include both sets of paths with high temperature. In other words, aging rate of these paths (with respect to exponential effect of temperature) can potentially exceed the others. After this filtering, CPs are further filtered based on their stress (lines 22-25), another important factor in both BTI and HCI mechanisms. In order to filter appropriate paths, value of $Stress_{th}$ is chosen based on stress factor distribution of paths (similar procedure as temperature filtering). In the last stage, to recognize the representative CPs, those paths that consist of nodes with high FO will be chosen (lines 27-30). Finally, considering the location of endpoints, the paths with same endpoints but one will be removed from the final list, also at this filtering phase the spatial distribution

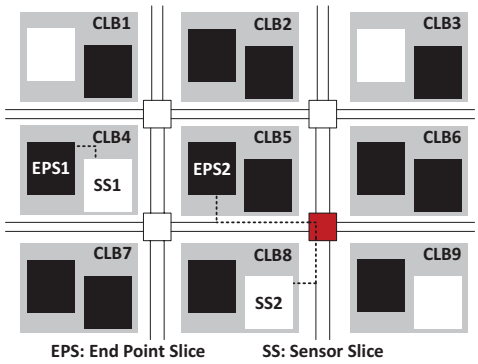


Fig 4: Sensor placement

V. Sensor Placement

of the sensors will be considered to fairly distribute them around the implemented circuit's bounding box (line 31).

A. Sensor Placement Algorithm

The proposed algorithm is based on a greedy local search algorithm that localizes unused slices near the endpoint slice (EPS) including the destination *flip-flop* of the path under consideration for sensor insertion. We explain the algorithm using an example shown in Fig. 4. The used resources are colored as black (Slices) or red (Switch boxes) and the unused resources are left as white. Finally, Resources labeled SSi, are utilized to place an aging sensor (SS).

In the placement algorithm, the best slice for sensor insertion is when the other slice in the endpoint CLB is empty (CLB4 situation in Fig 3). When the other slice in the endpoint CLB has been occupied, one of the unused slices in the nearby CLBs will be selected as the sensor slice (SS2 in CLB8 is utilized to place the sensor for EPS2). In this case, the general interconnect through a switch box is utilized (red resource in Fig 3). We assume an aging sensor occupies only one slice [2, 3].

B. Sensor Placement Challenge

Although, the placement algorithm can finally place each aging sensor, but its distance from the corresponding endpoints of the path is a crucial factor. The sensor must be placed on a slice in the same CLB of the destination slice or in the worst case, on a slice of adjacent CLBs. Placing the sensor far from the destination slice leads to a significant increase in the delay of the RCP which is monitored and affects the correct functionality of the sensor. In such a situation we should reserve a slice for the sensor during place and route phase followed by our proposed sensor insertion flow.

VI. Experiments

A. Insertion Flow and Setup

The implementation of sensor insertion is divided to several steps (Fig. 5):

1) The benchmark circuits are synthesized using synthesis tools (ISE 14.7). The synthesizer generates post route information without considering aging effects.

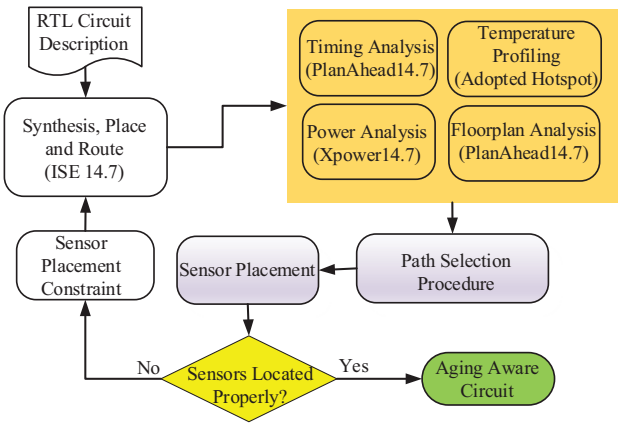


Fig 5. Sensor insertion flow

2) Timing analysis is performed using *Xilinx PlanAhead* tool in order to obtain timing information including path delays. The path delays are extract using a timing analysis tool. For example, *PlanAhead* tool can generate a timing report file (.twr) containing all of the logic objects and interconnects and their associated delay.

3) Activity rates and duty cycle of circuits are extracted using a power analyzer tool. *Xilinx Power Analyzer* (XPA) receives implemented design description in ISE (using the .ncd file generated after the place and route phase) and simulation activity files (.saif or .vcd) generated by a simulation tool (ISim or Modelsim). The XPA generates a power report file (.pwr) containing signal and I/O activity rates. Path stress rates are calculated based on net activity rates extracted from power report file using a C-based tool.

4) Our proposed critical path selection and sensor placement are applied to design.

5) PlanAhead tool provides user various placement constraints to be applied during place-and-route tools. Using this tool, we reserve some locations in order to place the aging sensors.

B. Results and Discussion

In aging model, the values for A_{BTI} and A_{HCI} are chosen such that the maximum delay degradation in 5 years is 15% in worst case (PMOS transistors always ON, the maximum frequency (AR=500 MHz) at temperature 380 Kelvin). The target FPGA device is Virtex6 at 40 nm technology.

To evaluate the impact of proposed filtering, five different benchmarks are selected from different applications. The characteristic of each benchmark is shown in Table I. For each benchmark the number of selected RCPs is extracted by proposed algorithm 1. Table I shows that the number of paths filtered in each filtering phase of the algorithm. The results show that the number of paths are reduced significantly at each filtering phase and the impact of each filtering phase varies from one benchmark to another. For example, in AES, the number of paths (delay filtering in Table I) is very high, but the number of pseudo-critical paths (PCPs) is only 1122. The other four filtering characteristics (temperature, stress rate, FO, and physical location) reduce the number of selected paths and at the end, 38 sensors are required to monitor aging for this circuit.

Similarly, results are shown for S15850, S38417, B20, and FIR. The number of sensors gradually increases when the circuits' sizes are increasing. For S15850, S38417, and B20 the numbers of critical paths that are required to be monitored are 16, 32, and 28, respectively. For example B20 is a larger circuit with higher number of critical paths at first phase of filtering (delay) but due to low stress rate and temperature of nodes (due to its behavior) most of the critical paths are filtered out at temperature and stress filtering.

Finding empty slices for sensors determined for each circuit, is the next step. Table II shows the output of greedy search algorithm representing number of sensor slices which placed in the same CLB including endpoint slices (Case1) and the adjacent CLBs (Case2). In our experiments, there was always an available slice for sensor insertion either in the same CLB or adjacent CLB. Hence, there was no need to apply placement constraints and resynthesize the circuit in selected benchmarks (Case3).

C. Evaluation of Proposed Filtering

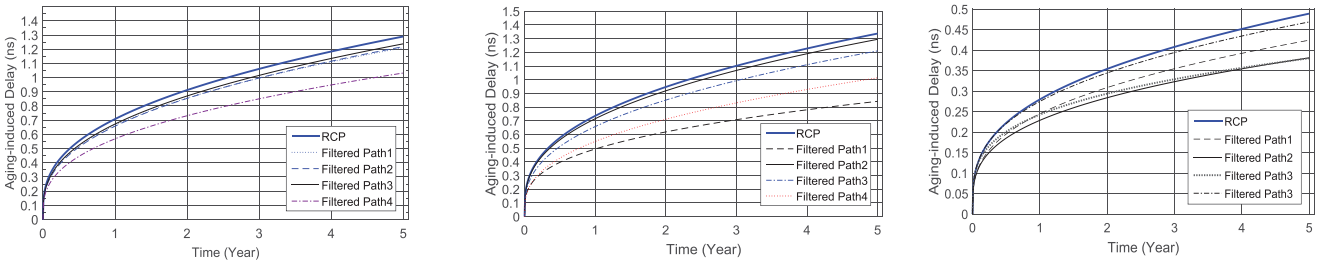
To evaluate the effectiveness of the proposed method, we compared aging rates of RCPs with those paths that are filtered out and removed from our proposed algorithm. As shown, in Fig. 6, aging rate of the last selected RCP is higher than aging rate of the filtered paths. In other words, RCPs will be aged sooner than the filtered paths. Hence, the inserted aging sensors detect aging-induced delay degradation of circuit sooner while inserting sensors on removed paths will only cost to the circuit without any meaningful usage.

Fig. 7 shows aging rate of the last selected RCP in comparison with aging rate of the top-ranked critical paths (based on delay including main critical path, CP1). This result shows aging rate of the last RCP is higher than them, although its pre-aged delay is lower. Consequently, aging manifestation on these RCPs happens sooner than that of the top-ranked critical paths. Hence, by monitoring RCPs we are being able to detect aging before any timing violations.

Table I. Number of sensors after applying filtering algorithm

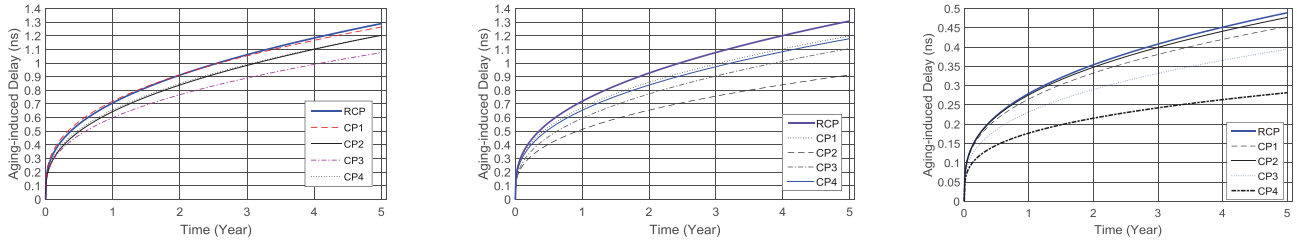
Benchmark			S15850	S38417	B20	FIR	AES
# of LUTs			582	1997	1912	108	3974
# of FFs			431	1387	430	400	261
# of Slices			245	929	557	83	1282
CLK frequency (MHz)			166	125	101	100	136
# of remained paths before each step	Step 1	D*	12322	31359	58326	10235	12568
		T‡	516	2804	6716	200	1122
		SR§	130	292	1754	182	757
	Step 2	FO$	82	121	735	72	301
		Ph**	71	60	291	53	263
	# of Sensors		**16**	**32**	**28**	**17**	**38**

* Delay filtering
‡ Temperature filtering
§ Duty cycle and Activity rate filtering
$ Fan-out filtering
** Physical location filtering



a. B20 b. S38417 c. S15850

Fig. 6. Aging-rate comparison of last selected RCP and top feiltered out paths



a. B20 b. S38417 c. S15850

Fig. 7. Aging-rate comparison of last selected RCP and top critical paths

Table II. Number of Sensors Placed by the Algorithm

	S15850	S38417	B20	FIR	AES
Case1	9	25	15	14	30
Case2	7	7	13	3	8
Case3	0	0	0	0	0

VII. Conclusion and Future Works

In this paper, we proposed an aging-aware critical path selection methodology for online aging monitoring using sensors in reconfigurable architecture. Our objective is to reduce the number of required sensors and to avoid unnecessary sensor insertion due to its cost and overhead. We first present a novel representative critical path selection based on major causes in aging mechanisms such as temperature, duty cycle and switching activity. Furthermore, to avoid performance loss due to sensor insertion a performance-aware placement is proposed. To the best of our knowledge this is the first work that considers path selection for aging monitoring in FPGAs, considering not only aging effects, but also delay, physical location and impact of aging on net delay paths.

References

[1] Z. Ghaderi and E. Bozorgzadeh, "*Aging-aware High-level Physical Planning for Reconfigurable Systems*", in Asia and South Pacific Design Automation Conference (ASPDAC), 2016.

[2] A. Amouri, and M. Tahoori, "*A Low-Cost Sensor for Aging and Late Transitions Detection in Modern FPGAs*", in Field Programmable Logic and Applications (FPL), pp. 329-335, 2011.

[3] M. Valdes-Pena and et. Al, "*Design and validation of configurable online aging sensors in nanometer-scale FPGAs*", IEEE Transactions on Nanotechnology (TN), 2013.

[4] E. Stott and et. Al, "*Degradation in FPGAs: measurement and modelling*", in Field programmable gate arrays (FPGA), pp. 229-238, 2010.

[5] K. Zick and et. Al, "*On-line sensing for healthier FPGA system*", in Field programmable gate arrays (FPGA), pp. 239-248, 2010.

[6] F. Firouzi and et. Al, "*Aging-and Variation-Aware Delay Monitoring Using Representative Critical Path Selection*", ACM Transactions on Design Automation of Electronic Systems (TODAES) 2015.

[7] S. Wang and et. Al, "*Representative critical reliability paths for low-cost and accurate on-chip aging evaluation*", in International Conference on Computer-Aided Design (ICCAD), pp. 736-741, 2012.

[8] C. Leong and et. Al, "*Aging monitoring with local sensors in FPGA-based designs*", in Field Programmable Logic and Applications (FPL), pp. 1-4, 2013.

[9] J. Angermeier and et. Al, "*Stress-aware module placement on reconfigurable devices*", in Int. Conf. on Field Programmable Logic and Applications (FPL), pp. 277-281, 2011.

[10] E. Stott and et. Al, "*Improving FPGA reliability with wear-leveling*", in Field Programmable Logic and Applications (FPL), pp. 323-328, 2011.

[11] A. Bsoul and et. Al, "*Reliability-and process variation-aware placement for FPGA*", in Design, Automation and Test in Europe (DATE), pp. 1809-1814, 2010.

[12] A. Amouri, and et. Al, "*High-level aging estimation for FPGA-mapped designs*", in Field Programmable Logic and Applications (FPL), pp. 284-291, 2012.

[13] N. Mitsumasa and et. Al, "*On estimation of NBTI-induced delay degradation*", In Test Symposium (ETS), pp. 107-111, 2010.

[14] B. Sarvesh, and et. Al, "*Predictive modeling of the NBTI effect for reliable design*", in Custom Integrated Circuits Conference (CICC) pp. 189-192, 2006.

[15] K. Skadron and et. Al, "Temperature-aware microarchitecture," in International Symposium on Computer Architecture (ISCA), pp. 2–13, 2003.