Stress-Aware Routing to Mitigate Aging Effects in SRAM-based FPGAs

Behnam Khaleghi[1], Behzad Omidi[1], Hussam Amrouch[2], Jörg Henkel[2], and Hossein Asadi[1]
[1] *Department of Computer Engineering, Sharif University of Technology, Tehran*
[2] *Chair for Embedded Systems, Karlsruhe Institute of Technology, Germany*

***Abstract*—Continuous shrinking of transistor size to provide high computation capability along with low power consumption has been accompanied by reliability degradations due to e.g., aging phenomenon. In this regard, with huge number of configuration bits, *Field-Programmable Gate Arrays* (FPGAs) are more susceptible to aging since aging not only degrades the performance, it may additionally result in corrupting the configuration cells and thus causing permanent circuit malfunctioning. While several works have investigated the aging effects in *Look-Up Tables* (LUTs), the routing fabric of these devices is seldom studied – even though it contributes to the majority of FPGAs' resources and configuration bits. Furthermore, there is a high prospect that errors in its state to propagate to the device outputs. In this paper, we first investigate aging effects in the routing fabric of FPGAs with respect to performance and reliability degradations. Based on this investigation, we enhance the conventional routing algorithm to mitigate the impact of aging by increasing the *recovery time* (i.e., the mechanism used to heal aging-induced defects) of transistors used in the routing resources. We examine our proposed method as reduction in stress time and required guardband to protect against aging in the routing fabric, as well as in improving the FPGA's lifetime. Our experiments show that the proposed method reduces the average stress time and aging-induced delay of routing resources by 41% and 18.3%, respectively. This, in turn, leads to improving the device lifetime by 130% compared to baseline routing. The proposed method can be applied by simple amending of conventional routing algorithms. Thus, it incurs negligible delay overhead.**

I. Introduction

To keep up with advances in *Application-Specific Integrated Circuits* (ASICs), the vendors of *Field-Programmable Gate Arrays* (FPGAs) are continuously adopting to new technologies to exploit higher performance opportunities with low power consumption, particularly considering the ever-existing performance, power, and area gap between FPGAs and ASICs [1], [2]. Shrinking the feature size, however, has been accompanied with reliability challenges such as aging phenomenon that is caused by mainly due to *Bias Temperature Instability* (BTI) and *Hot Carrier Induced Degradation* (HCID) [3]. While HCID affects both types of nMOS and pMOS transistors, BTI exhibits itself as *Negative* BTI (NBTI) and *Positive* BTI (PBTI). As contrary to PBTI which degrades nMOS transistors, NBTI degrades the electrical characteristics of pMOS transistors within a so-called *stress* phase in which a negative bias is applied on the transistor (i.e., $V_{GS} < 0$). These phenomena, at the physical level, induce interface and oxide traps at the *Si-dielectric* interface, which weaken the gate-bulk electric field. These defects are manifested as shifting two key parameters of transistors, i.e., increasing (magnitude of) threshold voltage (V_{th}) and reducing the carrier mobility (μ).

Aforementioned effects of aging result in increasing circuit's delay during lifetime, which causes errors due to timing violations if their effects have not been considered in the design time. Moreover, in memory elements, e.g., SRAM cells, aging reduces the drive strengths of transistors, resulting in data corruption because of decreased *Static Noise Margin* (SNM). Hence, FPGAs are more susceptible to aging because a) they have abundant resources that mostly remain unutilized and such resources are driven by a constant voltage which applies a permanent stress that is critical to aging and b) majority of the configuration SRAM cells hold a constant value (e.g., zero) for a long time until they are reconfigured, putting them into stress phase. In addition, overcoming the aging in FPGAs is more challenging because unlike ASIC designs, the notion of critical path does not apply for a FPGA platform. Mitigating the aging in an ASIC design necessitates alleviating the stress probability only in transistors of the critical path(s), since delay increase in the other paths does affect the circuit delay. However, in case some resources (e.g., a routing wire segment) degrade higher in FPGAs, even if they do not participate in the critical path or are unutilized, it is possible that the next designs critical paths reside on the same resources. Therefore, in FPGAs, resources with worst degradation are determinant, regardless they are used in current design or not.

While previous work have attempted to resolve aging effects in logic parts of FPGAs, i.e., *Look-Up Tables* (LUTs) [4]–[6], its impact on routing resources is far significant. Routing configuration cells contribute to more than 90% of total cells [7], [8], while for each circuit only very small portion of routing cells are configured to logical one [9]. This leads to a constant value in these cells (even after reconfiguration of the other circuits) and causes severe aging impact in majority of the configuration cells. In addition, in contrary with LUTs, SRAM cells of routing fabric are participated in path delay directly driving the gate of the routing transistors. Thus, the constant value in these cells not only incurs aging in them, but impairs the corresponding transistors, as well. Another reason that magnifies the aging of routing fabric is unutilization of routing resources (e.g., buffers) that incurs large stress time and delay degradation in these resources that manifests as FPGA's total performance degradation.

Our novel contributions within this paper are as follows:
(1) We first analyze and quantify the impact of aging in the routing fabric of SRAM-based FPGAs. For the first time, both the BTI and HICD are jointly considered.
(2) To mitigate aging effects, we propose a routing algorithm based on modifying the cost (i.e., routing priority) of possible routing paths in which the new design has minimum overlapping/equal configuration bits with the previous one, towards increasing the recovery time.
(3) Since majority of configuration bits are zero, the proposed

algorithm not only deals with the active bits (those which have a role in routing the design's used nets), it also periodically flips the major inactive bits during the configuration time to balance/mitigate aging stress there, as well.
(4) We elaborate and minimize the likelihood of ever-constant configuration bits (we call them *hotspot* aging points) that cause imbalance aging.

In order to evaluate the proposed method, we used VTR 7.0 toolset [10] – an open-source software suite that can model arbitrary FPGA architectures and provides CAD flow from HDL code to place and routing– along with MCNC benchmarks [11]. Experimental results showed that our proposed method reduces, on average, the stress time of configuration cells by 41%. In addition, by reducing the stress time of routing transistors, the proposed method improves the aging-induced delay by 18.3% which translates to average 130% improvement in device lifetime.

The rest of this paper is organized as follows. Section II reviews the related work. Section III discusses the motivation behind this paper. Section IV elaborates the proposed method. Details of experiments and results are provided in Section V, and finally, Section VI concludes the paper and presents the future work.

II. Previous Work

Previous research in the scope of aging in FPGAs can be categorized into two main classes. While the main aim of the first class is to measure or monitor the aging using sensors (i.e., aging detector circuits) that are mostly built based on FPGA resources [12], [13], the second class aims to mitigate its impact in FPGAs, mainly through alleviating the stress time. Our proposed work resides in the second class, which mitigates the aging effects in the routing fabric, e.g., SRAM cells and multiplexers.

Aging effects in different LUT structures, e.g., pass-gate and standard cells, are investigated in [4]. By performing SPICE simulation on two-input LUT implementations, the authors concluded that parameters such as LUT structure, input probability, and current and previous configuration bits have key effect on induced aging. In addition, they propose that the conventional all-zero configuration for unused LUTs may lead to worst aging. Sensitivity of two-input *Transmission-Gate* (TG) based LUT with respect to NBTI effect is analyzed in [6]. Based on the proposed observation, degradations can be mitigated in a circuit by alleviating the stress of critical inputs. In [14], the relation between process variation in FPGA (i.e., variability in original performance) and aging-induced delay variation is examined. It is found that process variation and aging degradations are uncorrelated. Thus, overall performance variation might increase as FPGA ages. To mitigate aging degradation in a design, it suggests incorporating sparse LUTs available in the same FPGA. However, this necessitates availability and proximity of such LUTs in order to impose minimum change in the global routing. The authors also propose using sparse routing resources, however, even partial re-routing a mapped circuit incurs additional delay that counteracts the possible gain of non-aged resources. Altering LUT configuration bits to mitigate the aging of LUT cells is proposed in [15]. In this method, configuration bits of a LUT can be shuffled (permuted) to have minimum overlapping with previous bits, putting the previously stressing transistors into recovery state (and vice versa). Accordingly, this method requires swapping the LUT inputs, as well, which can incur routing delay in depopulated crossbars. Nevertheless, while aging of configuration bits can impair their noise margin, it has no impact on LUT delay. An aging-aware placement which targets the runtime reconfigurable applications is proposed in [16]. This method enhances the aging by optimizing the stress distribution by evenly placing the accelerators (i.e., *runtime reconfigurable modules*) such that converges the overall toggle rate of the transistors within different logic blocks. A similar approach has also been exploited in [17] which uses different pre-defined configurations for runtime reconfigurable modules that attempts to balance the stress by determining the fraction of time each configuration should be used.

There are also few studies that analyze aging in the routing fabric. In this regard, the effects of aging on configurable resources of FPGA routing fabric has been investigated in [18]. Using SPICE simulations, the effect of wire length, cascaded switches, fan-out, and supply voltage on four different routing switches has been studied. The assumptions made in [18] about switch types and their connectivity, however, are not typically considered neither in industrial nor in academic FPGAs. The authors suggest pass-gate based and TG-based structures for technologies susceptible to NBTI and NBTI/BTI, respectively, but do not propose any approach for mitigating the aging. Finally, different reliability threats of FPGAs and approaches to mitigate them, including a bit inversion and shuffling method to mitigate the aging of configuration cells, is studied in [15]. For the unused multiplexers (i.e., multiplexers with undriven inputs and output) of routing fabric, this method suggests periodic flipping of configuration bits. The proposed method does not provide any approach for the other types of multiplexers (i.e., when any input or the output is driven). Furthermore, this method neglects the aging of multiplexers' transistors. For example, inverting the configuration bits of unused TG-based multiplexers exposes all of its pMOS transistors to aging. Nevertheless, efficiency of the proposed method has not been examined.

Among the manifold of studies that target mitigating aging of ASIC applications, [19]–[23] have focused on optimizing the aging in SRAM-based memories. These studies either attempt to improve the stress time of cells with bit rotating [19], [20] or bit inversion [21], [22]. Nonetheless, these methods impose considerable area and power overhead due to the

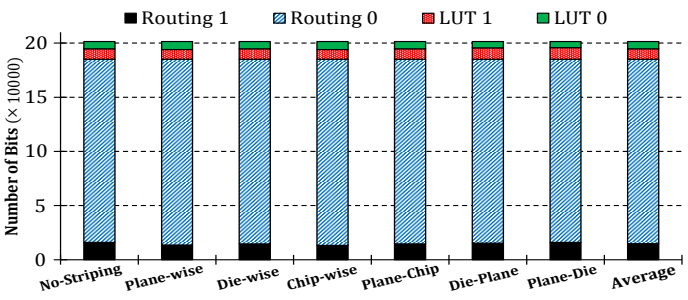

Fig. 1. Distribution of configuration bits in different FTL algorithms for a Virtex-II 2V80 device

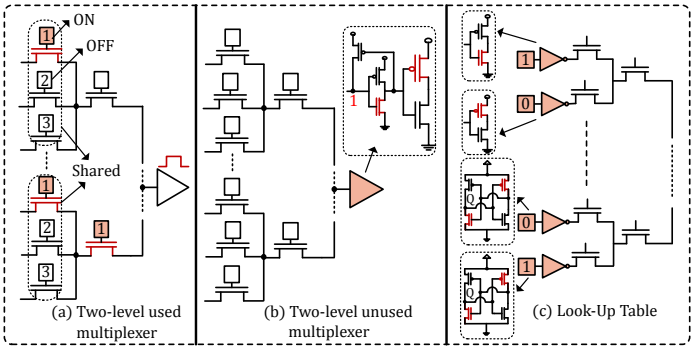

Fig. 2. Two-fold degradation in routing resources

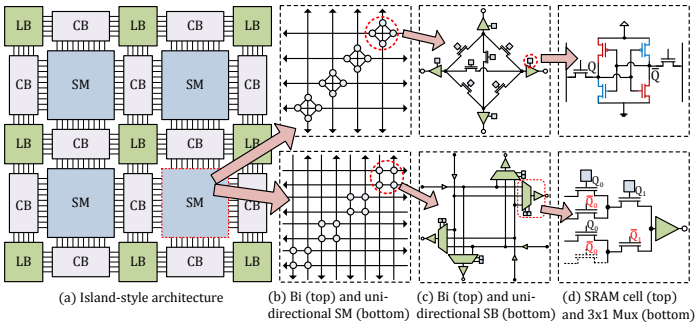

Fig. 4. FPGA architecture and structures used to employ the proposed method

support circuitries (e.g., Barrel shifters for rotating) or write schemes since increasing the data hamming distance in order to balance the stress time increases the dynamic power.

III. Motivation

The abundance of routing configuration cells coupled with low utilization of these cells makes them particularly susceptible to aging. To clarify the scenario, we have implemented sample *Flash Translation Layer* (FTL) algorithms, which are employed in *Solid-State Drives* (SSD) controllers, on a minimum size Virtex-II device. Fig. 1 illustrates the distribution of inter-cluster routing and LUT configuration bits of these algorithms, obtained by probing their bitstream [24]. As represented in this figure, routing configuration cells contribute to more than 90% of total cells (92% on average), while for each circuit only 8% of routing cells are configured to logical "1", compared to 41% in LUTs[1]. As a rule of thumb, average *Duty Cycle* (DC, the ratio of stress time to the total time) of routing configuration cells can be estimated as:

$$DC_{routing} = P_{0\to0} + P_{1\to1} = 0.92 \cdot 0.92 + 0.08 \cdot 0.08 = 0.85$$

This number is higher than $DC_{LUT} = 0.52$, which means it will impose higher aging impact on the routing resources[2]. In addition, in contrary with LUTs, SRAM cells of routing fabric are directly involved in path delay. This concept is demonstrated in Fig. 2 for conventional two-level routing multiplexers, and fully-encoded tree-based LUT [25]. Constant

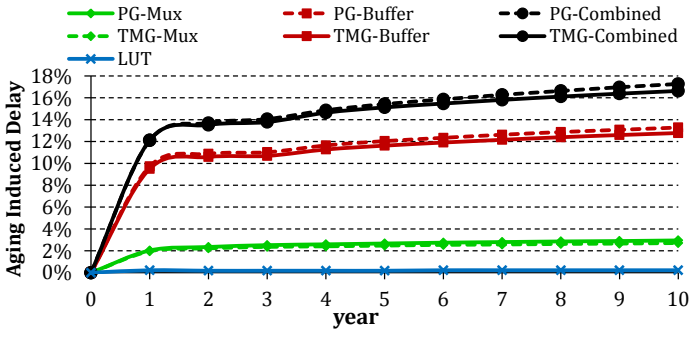

Fig. 3. Aging-induced delay in switch box (multiplexer and buffer) and LUT

configuration bits in routing multiplexers not only degrade these cells, but impose high stress interval in corresponding transistors (Fig 2.a). In an unused multiplexer (Fig 2.b), all SRAMs are configured to zero, and hence, do not age the associated *nMOS* transistors. However, in this scenario, the output buffer is under a ceaseless stress. In addition, configuration cells of LUTs are separated from the pass-gates by an isolating inverter to improve speed and robustness (e.g., by preventing flowing of rush current from pass-tree back to SRAM cells). While constant value in LUT configuration bits degrades the cells and associated inverters, it does not influence the delay since the inverters can still drive a strong voltage.

To validate our observation, aging-induced degradation over 10 years for a pass-gate (PG) and transmission-gate (TMG) based switch-box multiplexer along with its output buffer, and a tree-based LUT is demonstrated in Fig. 3 [3]. It can be observed that routing buffers have highest proportion in delay increase while they are under the same stress interval. As expected, aging in configuration cells of LUTs has little impact on its delay. Observations in this section establish the skeleton of the proposed method which will be detailed next.

IV. Proposed Method

A. Preliminary

Similar to commercial FPGAs as Xilinx Virtex IV [26] and Altera Stratix II [27], we use the popular island-style architecture (shown in Fig. 4) to implement our proposed method. We also target both the multiplexer-based (used in commercial FPGAs) and buffer-based switch boxes to examine the proposed method. For the multiplexers, here we exploit tree-based structure since it requires only two SRAM cells.

B. Proposed Algorithm

Since the optimal DC for a SRAM cell is 0.5 (otherwise one of the nMOS-pMOS pairs will be weak spot), we attempt to invert the utmost configuration bits during each device reconfiguration. This periodic inversion of SRAM cells not only yields their minimum degradation, but also changes the corresponding transistors' state —from stress to recovery or vice versa— and makes the stress/recovery uniform among different resources. To achieve this, during configuring the FPGA with a new design, we first exploit our routing algorithm

[1] On average, 90% of LUTs in the FPGA device are utilized in these benchmarks.

[2] $DC_{LUT} = 0.41 \cdot 0.41 + 0.59 \cdot 0.59 = 0.52$.

[3] Simulation details are explained in Section V.

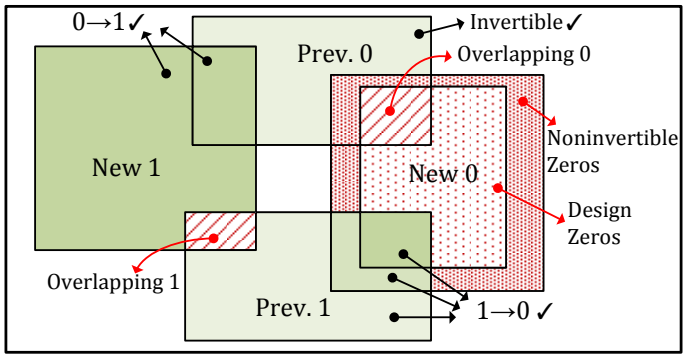

Fig. 5. Distribution of configuration bits in two consecutive designs

in which the segments with highest *configuration hamming-distance* with the previous design take the minimum routing cost (which turns in higher priority), while the most overlapping (equal configuration bits) segments get the maximum cost to avoid using overlapping configuration bits. These segments in a multiplexer-based routing fabric are the multiplexer's selected input, and in a buffer-based structure, the active terminals (e.g., *East* to *South*) are determined as segments. As an example, for a two-input multiplexer with previous configuration bits as Q_0Q_1=10, the highest priority is 01 while both 00 and 11 configurations have the second priority, and eventually, the least priority is for 10. For buffer based switches, the condition is simple because each segment is controlled by a unique SRAM cell, thus, for a previously used segment (Q=1) the current priority will make it off (Q=0), and vice versa. Determining the costs depends on the routing algorithm, and in our case, it is obtained empirically; while specifying large costs for undesirable segments enforces the routing tool to entirely avoid these paths, assigning small costs might ignore the priorities.

Each design consists of a manifold of *used* ones and zeros that refer to the configuration bits of the used multiplexers in the multiplexer-based architecture. For the previous and new designs, these configuration bits are distinguished by "Prev. 0", "Prev. 1", "New 0", and "New 1", as shown in Fig. 5. All other bits (white spaces) are the default zero configuration bits of unused resources. Accordingly, the first step of the proposed algorithm attempts to reduce the "overlapping 0" and "overlapping 1" configuration bits and increase the "0 $\rightarrow$ 1" and "1 $\rightarrow$ 0" regions (i.e., cells holding 0-value in the previous design that are configured to 1 in the new circuit and vice versa) between consecutive designs.

In the second step, the proposed algorithm inverts the *safe* zero bits, i.e., unused (zero) configuration cells in the new design that also hold zero in the previous circuit, whether in its used cells ("Prev. 0") or unused cells (white spaces). Note that current unused zero cells that were holding one in the previous design ("Prev. 1") should not be inverted. Furthermore, there is a set of zero configuration bits in the new design that belong to unused multiplexers that cannot be inverted. These bits are distinguished by *Noninvertible Zeros* in Fig. 5. This happens whenever a multiplexer is not used but one or some of the wires passing through its inputs is used in the design. In such scenarios, changing the configuration of multiplexers does not affect the circuit functionality but it imposes dynamic power overhead since the output of the multiplexers will be driven by an active net.

Therefore, the proposed algorithm changes the configuration bits of the unused multiplexers in the new design (that have 00 configurations which comprise more that 75% of the configurations) according to their previous configurations. If it was previously Q_0Q_1=00, it will change to 11 if the multiplexer input corresponding to 11 configuration (the first input in Fig. 4.d) is not driven by any net, otherwise, it changes to 01 or 10. Similar procedure applies to other conditions, e.g., when Q_0Q_1 is equal to 11 in the previous design, the current 00 configuration is the best choice.

In addition, as revealed by our experiments, even after applying the proposed algorithm, there is a minority of configuration cells that hold the same constant value between several consecutive reconfigurations. This makes these cells and their corresponding resources as unknown aging hotspots that necessitates incurring the maximum guardband to address the aging problem. We resolve this issue by keeping track of the overlapping configuration cells, i.e., $Config_1 \cap Config_2 \cap Config_3 \cap ...$ and forcing them to be inverted after a threshold value. Indeed, when the number of these cells falls below a specific ratio of the *used configuration bits* of the new design (e.g., 8% of the used bits which corresponds to 2.5% to 3% of the total configuration bits), the proposed algorithm entirely inverts these cells and re-route the design. Typically, after four reconfigurations, this condition occurs. It is noteworthy that this threshold value is also achieved empirically. High threshold values turn off larger number of paths and hence it imposes higher delay overhead and requires more additional memory if these bits are kept in configuration memory. On the other hand, low threshold value keeps these cells under stress for larger intervals.

It is also noteworthy that we assume several reconfigurations take place in an FPGA device. The proposed method does not apply for the cases that a device is being configured for once, unless it is reconfigured periodically (e.g., once a year) with the same design to alleviate the degradation. For these cases, considering the fact that any design has various potential place and route mappings, one can produce several configurations (place and routing steps) for a single design using the proposed algorithm and try to periodically reconfigure the device with different mappings of the same design in order to mitigate the aging effects. Therefore, in these cases, it is only required to periodically reconfigure the same design using the proposed algorithm. The proposed method is summarized in Algorithm 1.

V. Experimental Setup and Results

A. General Setup

We perform the experiments using VTR 7.0 toolset with 20 largest MCNC benchmarks. The place and route (VPR) tool is modified to generate the bitstream after routing each circuit based on the value of SRAM cells. Therefore, for each unused segment in buffer-based switch boxes, a zero bit is assigned, while the used segments take one. For multiplexer-based implementation, the select bits corresponding to each input are assigned as shown in Fig. 4.d (so the first input takes Q_0Q_1=11). A *subset* structure for *Switch Matrices* (SMs) is

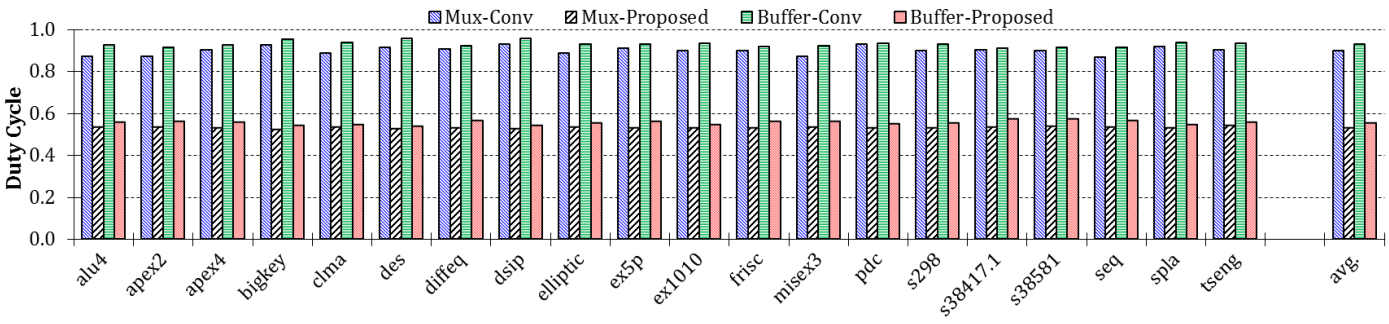

Fig. 6. Comparing the stress time (duty cycle) of configuration cells of the proposed method and baseline (denoted by Conv)

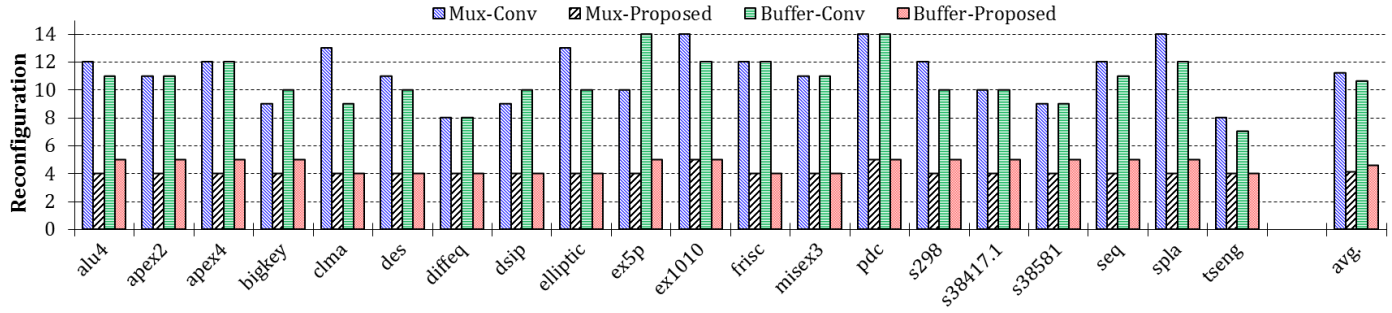

Fig. 7. Minimum number of reconfigurations that all of the configuration cells are inverted at least once

Algorithm 1: Aging-aware Routing Algorithm

Input: $Config_{prev}$: Previous design configuration bits
Input: $Netlist_{new}$: New placed design
Input: $Config_{overlap}$: Overlapping configuration bits
Output: $Config_{new}$: New design configuration bits

for *each* $Mux_i \in Config_{prev}$ **do**
  $cost_i \leftarrow determine\_priority(Mux_i)$;
$Config_{new} \leftarrow route(Netlist_{new}, cost)$;
for *each* $Mux_i \in Config_{new}$ **do**
  if $Mux_i = 00$ **then**
    $Mux_i \leftarrow$
    $newConfig(Config_{prev}, Netlist_{new})$;
update $Config_{new}$;
$Config_{overlap} \leftarrow Config_{overlap} \cap Config_{new}$;
if $Config_{overlap} < threshold$ **then**
  for *each* $Mux_i \in Config_{overlap}$ **do**
    $Mux_i \leftarrow invert(Mux_i)$;
  update $cost$;
  $Config_{new} \leftarrow route(Netlist_{new}, cost)$;

assumed which also has been employed in commercial FPGAs [10]. Since VPR maps the circuits to device with the minimum logic array and routing channel width by default, in both the proposed method and baseline we set the channel width to 1.2X of the obtained minimum to provide the tool with sufficient flexibility to route the circuits, particularly considering the fact that in commercial FPGAs, the utilized routing resources are considerably below the available resources [9]. Transistor sizes for both multiplexer-based and buffer-based structures are directly obtained from VTR repository, while the buffer sizes are derived by reverse engineering the buffer area used in *L(wire segment length)*$= 1$ architecture files in which the first stage of the buffer is minimum sized. Finally, we assume a typical minimum-size six-transistor SRAM cell is employed in FPGAs [25], [28].

For aging-related simulations, we have used high performance 22nm High-K models from PTM [29]. In order to jointly model the BTI and HCID, we employ recent physics-based model that is obtained for 22nm technology considering various duty cycles ranging from 0 to 1 [3], i.e., a pMOS/nMOS transistor is under 0% and 100% stress time (from total operation time), respectively.

B. Stress Reduction

Fig. 6 compares the duty cycle of configuration cells in the proposed method with the baseline (i.e., no inverting approach) for both the buffer-based and multiplexer-based architectures. These results are obtained by placing and routing each benchmark with different seed numbers which keeps the same FPGA array size but results in different placement and routing, i.e., different circuits. By reconfiguring the FPGA with the same benchmark but different placement and routing, we can also examine the effectiveness of the proposed method in mitigating the aging of a long-lasting design by periodical reconfiguring and inverting. By reducing the stress time by 41% on average, the proposed method could achieve $DC = 0.53$ and $DC = 0.56$ for the multiplexer-based and buffer-based architectures, respectively, which is very close to the optimum duty cycle (i.e., DC=0.5).

In addition, as mentioned previously, there are a set of hotspot configuration bits that hold their constant 0 or 1 value even after several different reconfigurations which extremely

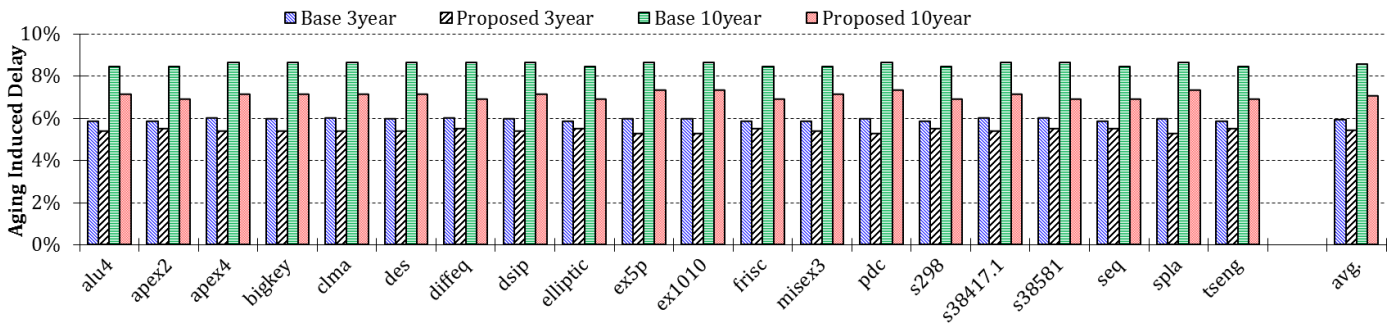

Fig. 8. Aging-induced delay in the baseline and proposed method after 3 and 10 years

degrades them and corresponding transistors. The minimum number of different reconfigurations in which all of the configuration bits are inverted at least once is shown in Fig. 7. For each circuit, the number is obtained by placing and routing with different initial placements (which results in different routing, as well). Accordingly, these hotspot cells have $DC = 0.95$, on average. In the proposed algorithm, all of the configuration cells are inverted after at most four reconfigurations, however, as discussed in Section IV-B, we smooth the stress interval of these cells by forcing them to hold the inverse value in the next reconfigurations.

C. Aging Mitigation

We compare the efficiency of the proposed method in enhancing the aging-induced delay in both average and worst-case scenarios. In the first scenario, we find the average stress time of multiplexer transistors in consecutive reconfigurations for each benchmark. Furthermore, to find the aging in buffers, we calculate the signal probabilities of each benchmark using [30]. Fig. 8 compares the aging-induced delay in routing switch-boxes (and associated wire segment) in the baseline and proposed algorithm after 3 and 10 years which is obtained by HSpice simulations using the degraded V_{th} and μ models. According to this figure, the proposed method can mitigate the aging effects by 8.70% and 17.52% after 3 and 10 years, respectively. In long-term, the efficiency of the proposed method increases since the effect of high stress intervals exacerbates in the long time. It is noteworthy that, theoretically, considering an optimum balanced stress and recovery between all routing resources, e.g., assuming $DC = 0.5$ for the buffers, the maximum achievable aging mitigation is 11.15% and 19.66% after 3 and 10 years, respectively. Therefore, the efficiency of the proposed method in aging mitigation is 89%, compared with the optimum solution.

In the worst-case scenario, we consider the hotspot resources, e.g., multiplexers or buffers driven by a constant signal in the long-time. Fig. 9 illustrates the improvement of the proposed method over the baseline in term of reducing the required timing guardband over 3 and 10 years. The timing required guarband is reduced by 9.72% and 18.32%, respectively, which implies to 93% efficiency of the proposed method.

D. Lifetime Improvement

In order to interpret the stress reduction and aging-induced delay mitigation of the proposed method to the enhancement

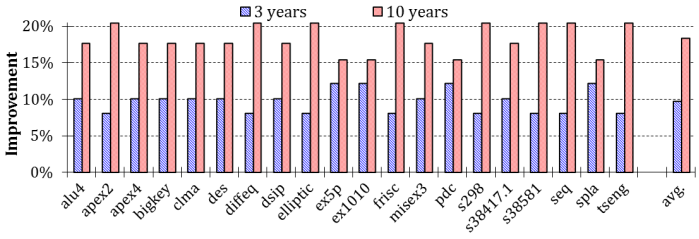

Fig. 9. Mitigating the aging-induced delay considering hotspot (maximally-aged) resources

in the device lifetime, we assume the term $\Delta delay$ as the maximum tolerable aging-induced delay increase that is considered during the design time by setting a timing guardband. We denote the corresponding threshold increase by ΔVth_{fail}. Therefore, for a specific ΔVth_{fail} value, we calculate −using the threshold model and improved stress time− the time that the proposed method takes to break this guardband. For example, if a device is supposed to work for 5 years in the baseline method, the corresponding ΔVth_{fail} would be equal to 44.6mV (i.e., the degradation reaches to 44.6mV after 5 years), however, due to reducing the stress time, the proposed method reaches this degradation after 11.9 years, so our method increases the lifetime by 6.9 years.

Fig. 10 shows the lifetime of the proposed method with respect to the given baseline device lifetimes (i.e., with different degradation thresholds). The stress time used to calculate the ΔVth is averaged among all the benchmarks in the baseline and the proposed method. Our proposed method on average could increase the device lifetime by 130%.

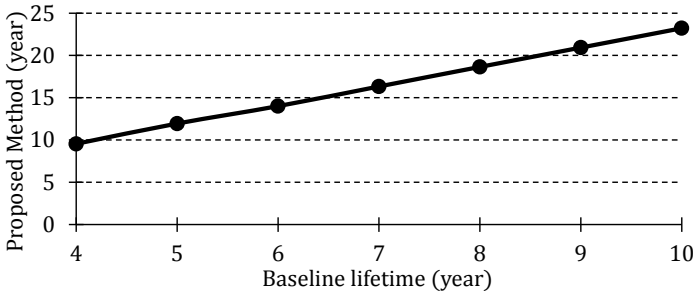

Fig. 10. Comparing the lifetime of the proposed method and baseline

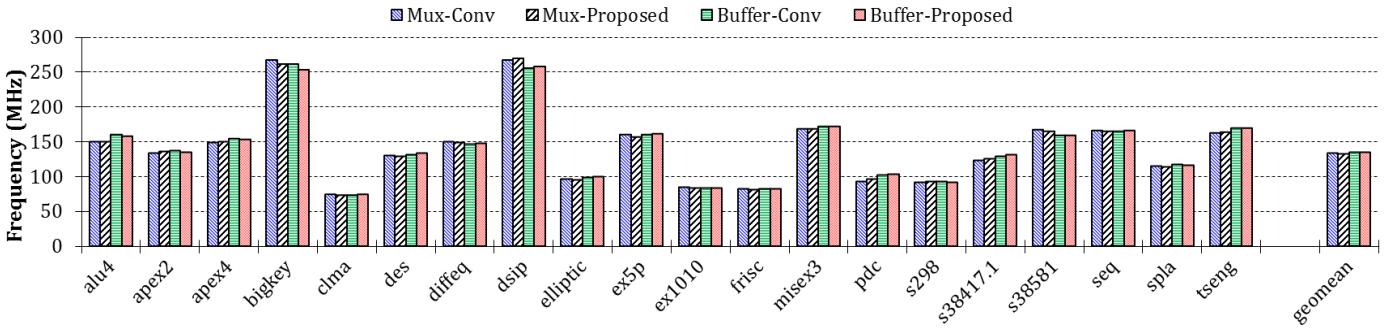

Fig. 11. Comparing the benchmarks frequency of the proposed method with the baseline

E. Delay Overhead

To investigate the possible delay overhead of the proposed method, we obtained the average frequency of each benchmark during the reconfigurations. As demonstrated in Fig. 11, the proposed method incurs 0.2% performance overhead in the multiplexer-based architectures, which is very negligible. Nonetheless, in the buffer-based architecture, the proposed method does not impose any delay overhead. The difference of the obtained frequencies for each benchmark during consecutive reconfigurations is due to the different seed numbers of benchmarks, i.e., different placements, which can reduce or even improve the performance. Therefore, due to the abundant routing resources in FPGAs, the proposed algorithm generally does not affect the performance.

VI. Conclusion and Future Work

In this paper, we proposed aging mitigation technique to alleviate the aging effects in the routing resource of FPGAs by enhancing the routing algorithm to alleviate the stress time by evenly using the routing resources and inverting the configuration cells. As demonstrated by the experimental results, our proposed method reduces the stress time of configuration cells to 0.53 which is close to optimal value. In addition, aging-induced delay was reduced by 18.3% which is interpreted to 130% increase in the device lifetime. In this work, for the sake of brevity, we focused on a specific type of routing architectures, i.e., subset switch matrices with short segment lengths. As an extension of this work, we will incorporate the aging of logic resources and consider a more comprehensive set of architectures and different routing resource structures to precisely investigate and mitigate the aging in commercial FPGAs.

Acknowledgement

This work is supported by the AICT Innovation Center of Sharif University of Technology and in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 - spp1500.itec.kit.edu).

References

[1] "7 series fpgas overview," Data Sheet, Xilinx, May 2015.

[2] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 203–215, 2007.

[3] H. Amrouch, V. M. van Santen, T. Ebi, V. Wenzel, and J. Henkel, "Towards interdependencies of aging mechanisms," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*. IEEE, 2014, pp. 478–485.

[4] S. Kiamehr, A. Amouri, and M. B. Tahoori, "Investigation of nbti and pbti induced aging in different lut implementations," in *Field-Programmable Technology (FPT), 2011 International Conference on*. IEEE, 2011, pp. 1–8.

[5] K. Ramakrishnan, S. Suresh, N. Vijaykrishnan, M. J. Irwin, and V. Degalahal, "Impact of nbti on fpgas," in *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*. IEEE, 2007, pp. 717–722.

[6] E. Stott, P. Sedcole, and P. Y. Cheung, "Modelling degradation in fpga lookup tables," in *Field-Programmable Technology, 2009. FPT 2009. International Conference on*. IEEE, 2009, pp. 443–446.

[7] H. Asadi, M. B. Tahoori, B. Mullins, D. Kaeli, and K. Granlund, "Soft error susceptibility analysis of sram-based fpgas in high-performance information systems," *Nuclear Science, IEEE Transactions on*, vol. 54, no. 6, pp. 2714–2726, 2007.

[8] H. Asadi and M. B. Tahoori, "Analytical techniques for soft error rate modeling and mitigation of fpga-based designs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 12, pp. 1320–1331, 2007.

[9] B. Khaleghi, A. Ahari, H. Asadi, and S. Bayat-Sarmadi, "Fpga-based protection scheme against hardware trojan horse insertion using dummy logic," *Embedded Systems Letters, IEEE*, vol. 7, no. 2, pp. 46–50, 2015.

[10] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed *et al.*, "Vtr 7.0: Next generation architecture and cad system for fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, p. 6, 2014.

[11] S. Yang, *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.

[12] M. D. Valdes-Pena, J. Fernandez Freijedo, M. J. M. Rodriguez, J. J. Rodriguez-Andina, J. Semiao, I. M. Cacho Teixeira, J. P. Cacho Teixeira, and F. Vargas, "Design and validation of configurable online aging sensors in nanometer-scale fpgas," *Nanotechnology, IEEE Transactions on*, vol. 12, no. 4, pp. 508–517, 2013.

[13] C. Leong, J. Semião, I. C. Teixeira, M. B. Santos, J. P. Teixeira, M. Valdes, J. Freijedo, J. J. Rodriguez-Andina, and F. Vargas, "Aging monitoring with local sensors in fpga-based designs," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*. IEEE, 2013, pp. 1–4.

[14] E. Stott, J. S. Wong, and P. Y. Cheung, "Degradation analysis and mitigation in fpgas," in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*. IEEE, 2010, pp. 428–433.

[15] S. Srinivasan, R. Krishnan, P. Mangalagiri, Y. Xie, V. Narayanan, M. J. Irwin, and K. Sarpatwari, "Toward increasing fpga lifetime," *Dependable and Secure Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 115–127, 2008.

[16] H. Zhang, M. A. Kochte, E. Schneider, L. Bauer, H.-J. Wunderlich, and J. Henkel, "Strap: Stress-aware placement for aging mitigation in runtime reconfigurable architectures," in *Proceedings of the IEEE/ACM*

International Conference on Computer-Aided Design. IEEE Press, 2015, pp. 38–45.

[17] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, C. Braun, M. E. Imhof, H.-J. Wunderlich, and J. Henkel, "Module diversification: Fault tolerance and aging mitigation for runtime reconfigurable architectures," in *Test Conference (ITC), 2013 IEEE International*. IEEE, 2013, pp. 1–10.

[18] A. Amouri, S. Kiamehr, and M. Tahoori, "Investigation of aging effects in different implementations and structures of programmable routing resources of fpgas," in *Field-Programmable Technology (FPT), 2012 International Conference on*. IEEE, 2012, pp. 215–219.

[19] S. Kothawade, K. Chakraborty, and S. Roy, "Analysis and mitigation of nbti aging in register file: An end-to-end approach," in *Quality Electronic Design (ISQED), 2011 12th International Symposium on*. IEEE, 2011, pp. 1–7.

[20] H. Amrouch, T. Ebi, and J. Henkel, "Stress balancing to mitigate nbti effects in register files," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2013, pp. 1–10.

[21] S. Wang, T. Jin, C. Zheng, and G. Duan, "Low power aging-aware register file design by duty cycle balancing," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 546–549.

[22] T. Siddiqua and S. Gurumurthi, "Recovery boosting: A technique to enhance nbti recovery in sram arrays," in *2010 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2010, pp. 393–398.

[23] M. Shafique, M. U. K. Khan, O. Tüfek, and J. Henkel, "Enaam: energy-efficient anti-aging for on-chip video memories," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 101.

[24] A. Upegui and E. Sanchez, "Evolving hardware by dynamically reconfiguring xilinx fpgas," in *Evolvable Systems: From Biology to Hardware*. Springer, 2005, pp. 56–65.

[25] C. Chiasson and V. Betz, "Should fpgas abandon the pass-gate?" in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*. IEEE, 2013, pp. 1–8.

[26] "Virtex-4 platform FPGA user guide," User Guide, Xilinx, December 2008.

[27] "Stratix-2 platform FPGA hand book," Hand Book, Altera, April 2011.

[28] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for deep-submicron FPGAs*. Springer Science & Business Media, 2012, vol. 497.

[29] (2013 (accessed March 20, 2016)) Predictive technology model (ptm). [Online]. Available: http://ptm.asu.edu/

[30] J. Lamoureux and S. J. Wilton, "Activity estimation for field-programmable gate arrays," in *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*. IEEE, 2006, pp. 1–8.