Estimating and Mitigating Aging Effects in Routing Network of FPGAs

Behnam Khaleghi, Behzad Omidi, Hussam Amrouch *Member, IEEE*, Jörg Henkel *Fellow, IEEE*, and Hossein Asadi *Senior Member, IEEE*

***Abstract*—In this paper, we present a comprehensive analysis of the impact of aging on the interconnection network of *Field-Programmable Gate Arrays* (FPGAs) and propose novel approaches to mitigate aging effects on the routing network. We first show the insignificant impact of aging on data integrity of FPGAs, i.e., *Static Noise Margin* (SNM) and *Soft Error Rate* (SER) of the configuration cells, as well as we show the negligible impact of the mentioned degradations on the FPGA performance. As such, we focus on performance degradation of datapath transistors. In this regard, we propose a routing accompanied by a placement algorithm that prevents constant stress on transistors by evenly distributing the stress through the interconnection resources. By observing the impact of the signal probability on the aging of routing *buffers*, we enhance the synthesis flow as well as augment the proposed routing algorithm to converge the signal probabilities towards aging-friendly values. Experimental results over a set of industrial benchmarks and commercial-like FPGA architecture indicate the effectiveness of the proposed method with 64.3% reduction of stress duration in multiplexers and up to 45.2% improvement of the degradation of buffers. Altogether, the proposed method reduces the timing guardband by from 14.1% to 31.7%, depending on the FPGA routing architecture.**

I. Introduction

By providing promising advantages such as the flexibility to implement various digital systems, short time-to-market, and reduced *Non-Recurring Engineering* (NRE) cost, *Field-Programmable Gate Arrays* (FPGAs) have widely been used in a diverse range of use-cases from embedded systems to high-performance to safety-critical applications. These advantages, however, come at the cost of significant power and performance disadvantages compared to their *Application-Specific Integrated Circuit* (ASIC) counterpart [1]. To reclaim the aforementioned gap and keep up with the advances in ASICs, FPGA devices are continuously adopting the most recent transistor technologies. This, in turn, has been accompanied by pronounced reliability challenges, e.g., aging [2].

Aging is mainly caused by *Hot Carrier Induced Degradation* (HCID) and *Bias Temperature Instability* (BTI) phenomena which have a stronger effect compared with other aging mechanisms such as *Time-Dependent-Dielectric Breakdown* (TDDB) [3]. While HCID affects both nMOS and pMOS transistors [4], BTI manifests as *Negative* BTI (NBTI) which is well known to degrade pMOS transistors, and *Positive* BTI (PBTI) whose effect is intensified in nMOS transistors in deep-nano feature sizes, e.g., 22nm [5]. Aging occurs when a so-called *stress* phase (i.e., when $|V_{GS}| > 0$) is applied to the transistor. At the physical level, it induces interface and oxide traps at the *Si-dielectric* interface, which weakens the gate-bulk electric field. These defects are manifested as shifting two key parameters of transistors, i.e., increasing the magnitude of threshold voltage (V_{th}) and reducing the carrier mobility (μ) which, according to Equation 1, reduce the drain current (I_d) which leads to increased RC delay eventually. As Equation 2 shows the other factors than V_{GS} affecting the V_{th} degradation, including t the total time, λ or stress cycle which determines the ratio of total time the transistor is under stress, and T that accounts for the thermal cycle [6].

$$I_d \simeq \frac{\mu C_{ox}}{2}\frac{W}{L}(V_{GS} - (V_{th} + \Delta V_{th}))^2 \quad (1)$$

$$\Delta V_{th} \propto t^n \times \lambda^n \times e^{\frac{-E_a}{kT}} \quad (2)$$

In addition to performance degradation, aging also reduces the *Static Noise Margin* (SNM) of SRAM memory cells, which is defined as the maximum extrinsic DC voltage noise that an SRAM cell can tolerate without flipping its value. Degradation of SNM can lead to imperfect read or write operations or data corruption especially in presence of the intrinsic fluctuations in circuit operation conditions. In ASIC designs, a straightforward approach to mitigate aging is to add a *timing guardband* ΔT_{aging} corresponding to aging-induced degradations to the original clock cycle. That is, instead of running the circuit at T_{init}, a clock cycle of $T' = T_{init} + \Delta T_{aging}$ is adopted. Nevertheless, taking the intensified aging effects as well as other sources of uncertainty such as process variations in deep-nano CMOS technologies into consideration, undertaking a pessimistic timing guardband offsets the potential performance gain of the technology.

Various approaches have been proposed to mitigate the aging effects in ASIC circuits including (a) aging-aware standard cell-based synthesis [7], [8], (b) tight and runtime (adaptable) timing guardbanding rather than a worst-case guardband [8], [9], (c) critical path gate up-sizing [10], and (d) stress balancing to increase the *recovery* phase [11] whereby the aging phenomenon is reversed, i.e., the degraded characteristics of transistors begin to heal. Similarly, techniques based on balancing the stress duration of SRAM cells to mitigate aging effects have been proposed [12].

This work was supported by ICT Innovation Center of Sharif University of Technology and *Iran National Science Foundation* (INSF) under grant number 96006071, and in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 - spp1500.itec.kit.edu). B. Khaleghi, B. Omidi, and H. Asadi (corresponding author) are with the Sharif University of Technology, Tehran, Iran. H. Amrouch and J. Henkel are with the Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems (CES), Karlsruhe, 76131, Germany. E-mails: {behnam_khaleghi; omidi}@ce.sharif.edu, asadi@sharif.edu, {amrouch; henkel}@kit.edu.

Analysis and mitigation of aging in FPGA devices, however, is less considered. This is primarily because of the complexity of FPGA resources that encompass both complex datapath components (e.g., large multiplexers) and memory elements, as well as the dependency of the FPGA aging on the implemented design as it affects the stress duration on individual transistors. For instance, while it is adequate to measure/mitigate the aging of (near-) critical paths in an ASIC design [8], the concept of a fixed critical path does not apply for an FPGA device. That is, a resource in a non-critical path, or even an unused resource may undergo a constant stress in the currently implemented design and haphazardly reside in the critical path of the subsequent configurations (designs). Therefore, it is crucial to consider the aging of all used and unused resources rather than merely considering the critical path ones. For the same reason, in contrast to ASIC methods, increasing the size of critical path transistors during the design phase of FPGA device is not beneficial because different designs have different critical paths which will not necessarily be mapped on these upsized transistors. For the case of configuration bits, several previous studies have aimed to balance the stress duration of the SRAM cells of FPGAs [13], [14]. However, in contrary to typical memory arrays such as register file and cache memory [12], evenly balancing the stress duration of interconnect SRAM cells (e.g., by periodical inverting of the cells) in commercial FPGA devices is not feasible due to the specific structure of routing multiplexers of commercial devices. We discuss it further in Section III.

Accordingly, previous studies regarding FPGA aging have mainly aimed to measure or model the aging of these devices [15]–[18]. Few studies target mitigating the aging [13], [14], [19]–[23] either use abstract circuit models for building blocks which may make their methods impractical in commercial devices, and/or ignore the routing resources and solely focus on logic resources, i.e., *Look-Up Tables* (LUTs), while routing resources contribute to 80% of the total path delay in FPGA-based designs [24]–[26]. We discriminate our work by, for the first time, focusing on in-detail analyzing and mitigating the aging of routing resources along with considering circuit and aging models comparable to commercial ones. Our approach is based on reducing the stress (λ) of transistors, which can be employed together with other orthogonal methods such as thermal management.

Our novel contributions in this paper are as follows.
(1) We present a detailed analysis of the impact of aging on FPGA routing resources while considering FPGA architectures and building blocks of contemporary commercial devices. This work for the first time investigates the aging considering FPGA architecture and building resources similar to commercial devices, transistor sizing, and physics-based aging model that jointly considers BTI and HCID [27].
(2) According to our analysis that shows the insignificant impact of aging on data integrity of FPGA configuration cells, we mitigate the performance degradation of routing resources by using a novel routing algorithm accompanied by a placement algorithm that attempt together to balance the resource usage and reduce stress duration (λ) of device transistors.
(3) According to our analysis that reveals the impact of signal probability on aging degradation of routing buffers, we enhance the synthesis flow and augment the routing algorithm to converge the signal probabilities towards the value that causes minimum delay degradation.

We have exploited the latest version of COFFE [28] with 22nm transistor technology model to generate the SPICE netlist of FPGA resources and study the impact of aging by incorporating our physics-based aging model. The proposed mitigation method, including the constrained-mapping, routing algorithm, and signal imbalancing in the synthesis phase is then implemented with the aims of open-source VTR 7.0 toolset [29] that can model the FPGA architecture as well as includes the corresponding CAD flow from synthesis to routing. Experimental results show 64.3% enhancement of the routing multiplexers stress and up to 45.2% improvement in buffers guardbands. Together, 14.1% to 31.7% timing guardband reduction is achieved depending on the FPGA architecture.

The rest of this paper is organized as follows. Section II reviews the related studies. In Section III, we describe the routing architecture of state-of-the-art FPGAs. We articulate the proposed method in Section IV. Experimental setup and results are presented in Section V, and finally, Section VI concludes the paper.

II. Related Work

We categorize the previous work as the studies that (a) measure or model the aging in FPGAs and (b) aim to mitigate their aging which our proposed method also belongs to.

(a) Measuring and modeling aging: The majority of previous studies aim at measuring and estimating the impact of aging in FPGAs. In this regard, [18] proposes sensors by using the unused resources of FPGA for online (runtime) monitoring of aging impact. The principle of the sensor operation is based on detecting forbidden transitions in a specific interval close to the active clock edge which indicates the loss of signal integrity due to aging. Resolution of the sensor, however, is limited by the operating frequency and can be as low (i.e., inaccurate) as close to 80ps which may not detect aging degradation of designs with small critical path. In addition, in a typical design, there are manifold of near-critical paths which may become critical after the aging [8]. Such a method requires a large number of sensors, and hence unused resources on the FPGA, as well as placing and routing the sensors without disrupting the originally mapped design which is challenging.

[15] analyzes the impact of NBTI on the LUT delay assuming a 2-input LUT with a simple structure. While improvement in rise delay and degradation in fall delay is reported, the results cannot be generalized to the complicated structure of LUTs [28], especially considering the effect of transistor sizing on the behavior of aging [8].

The work presented in [16] measures the impact of aging on the LUT delay while considering different stress cycles (λ) in a commercial 65nm FPGA device. Impact of NBTI and especially PBTI, however, is insignificant in such technology node.

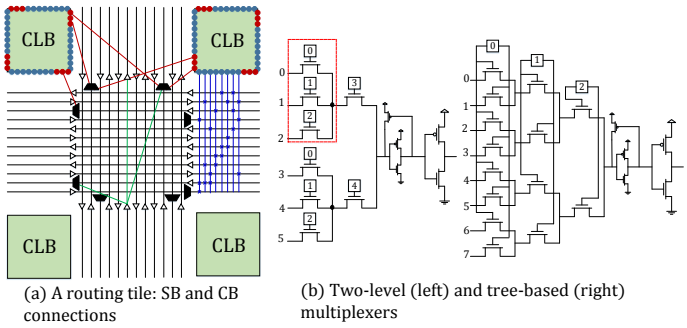
Finally, [17] investigates the impact of aging on the different implementation of FPGA routing resources, i.e., routing switches based on pass transistor, tri-state buffer, transmission gate, and multiplexer by considering different wire lengths, number of cascaded switches, and switch fanout. Nonetheless, the authors do not carefully consider the structure of FPGA resources which can lead to misleading outcome as follows. First, unlike academically supposed structures, contemporary devices only implement multiplexer-based switch boxes [30], [31]. In addition, this study assumes minimum width transistors for multiplexers and particularly their output buffer, while the size of transistors and buffers depends on other architectural parameters (see Section III) and plays a major role in the aging effect. Similarly, the length of wire and number of fanout driven by a switch multiplexer depends on the architecture.

(b) Aging mitigation: The studies in this category attempt to balance the stress cycle of transistors to alleviate the impact of aging. For this end, [13] has proposed to invert the configuration bits of routing multiplexers to balance the aging in SRAM cells. Nonetheless, as we show in this study, aging has basically an insignificant impact (i.e., $\sim 3\%$) on SNM and *Soft Error Rate* (SER) of FPGA configuration cells. Notice that, as shown in Fig. 1b, the configuration bits are directly connected to the gate of routing transistors so until they can provide a strong V_{dd}, their degradation does not affect the multiplexer delay.

The study in [19] measures the impact of aging on LUTs and a representative interconnect of a 65nm FPGA device. It reveals that unused LUTs are slightly affected by aging compared to those undergone a constant stress or high input signal frequencies. Thus, [19] suggests mitigating logic functions from heavily affected LUTs to the unused ones. This, however, requires unused LUTs to be in the same cluster of the used one as well as a fully flexible intra-cluster crossbar to avoid any overhead on the global routing that might compromise the achieved gain. Similarly, this study shows small delay increase for the unused routing resources (compared as to used resources) and proposes to use alternative routing paths for an aged design. This, however, highly depends on the existence of an alternative routing path with the same length for all near-critical paths.

[21] shows that various configurations of a LUT and its inputs probabilities imposes different stress duration on the LUT transistors. Thus, for a given LUT configuration and input probabilities, the authors find the input permutation (and corresponding configuration swapping) that leads to minimum delay increase. This requires the LUT input probabilities (which depends on application input) to be known in the configuration time. Furthermore, specifying a certain order (i.e., that with the minimum aging) for the connection of LUT inputs eliminates its intrinsic input permutation capability and incurs pressure on the global or local routing which incurs to performance overhead which may outweigh the potential aging improvement.

The study in [20] creates different configuration bits for runtime reconfigurable modules to provide fault tolerance and mitigate the aging. The minimal configurations bits are generated in a way that for any faulty cluster, there is at least one configuration that does not use that cluster. In addition, it determines the time fraction to use each configuration to balance the stress duration. Indeed, while the HCI induced degradation depends on the count of transistor triggering, BTI merely depends on the long-term stress cycle of transistors [32]. Thus, while a resource may not have any transition (triggering) during the runtime, the applied constant stress can impose high aging. Additionally, [20] manages the aging in cluster level and does not guarantee periodic use of cluster resources in a fine-grained manner (to eliminate the possibility of hotspot resources). It also does not consider routing resources. Similarly, [22] aims at reducing the aging by evenly placing runtime reconfiguration modules during the design runtime to converge the toggle rate of transistors. This study faces the same challenges of [20].



(a) A routing tile: SB and CB connections

(b) Two-level (left) and tree-based (right) multiplexers

Fig. 1. Routing tile and building blocks of island-style FPGAs

[14] alleviates the aging of SB multiplexers by periodically inverting the configuration bits. For this end, it assumes a three-input multiplexer with tree-based structure (see Section III). In each new configuration, [14] modifies the cost of using specific inputs of each SB in such a way the resulted configuration has large Hamming distance with the previous ones, which helps to balance the stress cycle of SRAM cells. Inversion of SRAM cells it not straightforward in commercial devices that employ two-level multiplexers wherein only two SRAM cells can hold logical one. This work also does not consider aging of routing buffers that contribute to a significant portion of FPGA delay.

III. FPGA Architecture

In order to better explain the concepts of the proposed method, it is essential to elaborate the routing architecture of contemporary FPGAs and transistor-level structure of their building blocks. Fig. 1 demonstrates the routing architecture and building blocks of contemporary FPGAs. State-of-the-art FPGAs have an island-style architecture in which the logic clusters, namely *Configurable Logic Blocks* (CLBs) are surrounded by routing resources [33], [34] as a two-dimensional array. According to Fig. 1a, *Switch Boxes* (SBs) and *Connection Blocks* (CBs) provide the routability between CLBs. The outputs of each CLB (made up of N LUTs, each of which has K inputs) are directly connected to several adjacent SB multiplexers. Accordingly, a net (i.e., the output of a source LUT) that is aimed to connect to a destination LUT inside a

particular CLB should be passed into routing network using the CB multiplexer of source CLB, routed using the network of connected SB multiplexers, and eventually passed to the destination CLB using its neighboring CB multiplexer.

1) CB and SB: CB multiplexers are responsible to input the global wires into the CLBs. There are I CB multiplexers for each CLB, one per each of I global input to CLB, uniformly arranged on four sides of the cluster for efficient routing. The connectivity factor of CB (F_{cin}) determines the size of its multiplexers, which is formulated in Equation 3. In this equation, W is the channel width, i.e., number of wires in a horizontal or vertical channel.

$$X_{CB} = W \times F_{cin} \tag{3}$$

As shown in Fig. 1a, SB multiplexers are located at the intersection of horizontal and vertical routing channels. Considering the number of branches (F_s, which is typically three), number of fanouts of each cluster output ($F_{cout} \times W \times Or$) that directly connect to the SBs, and wire segment length (L), the size of each SB multiplexer can be obtained using Equation 4

$$X_{SB} = \frac{L}{2} \times Or + L \times (F_s - 1) + 1 \tag{4}$$

2) Structure of Multiplexers: Instead of conventional tree-based structure, state-of-the-art FPGAs employ two-level *Pass Gate* (PG)-based multiplexers which have been shown to provide better area-delay efficiency [30], [31]. *Transmission Gate* (TG)-based multiplexer is an alternative for implementing two-level multiplexers which outperform the performance of PG-based FPGAs at the cost of a higher area [31]. In an X-input two-level multiplexer, the first level is composed of m one-level sub-multiplexers (i.e., m bunches) each of which has n inputs forming $n \times m = X$. To balance the area and delay of two-level multiplexer stages, typically $m = n \simeq \sqrt{X}$. Each of the multiplexer bunches is followed by an nMOS transistor in the second level. There are n shared SRAM cells for the first level each of which controls a particular nMOS transistor in every bunch. That is, as demonstrated in Fig. 1(b), SRAM_0 is shared in both bunches of the demonstrated 6-input two-level multiplexer. Thus, when this SRAM is *on* (holds logical *one*), the corresponding inputs in both bunches pass to the second level wherein SRAM_4 and SRAM_5 select the appropriate one. Hence, in a used multiplexer, only two of the $n + m$ SRAM cells (n SRAMs for the first level, and m SRAMs in the second level) can be *on*. That is why the SRAM cells of routing multiplexers cannot be arbitrarily inverted to balance the stress duration.

In addition to their efficiency, two-level multiplexers afford a high opportunity of aging relaxation compared to tree-based structures. Precisely, since only one transistor in each bunch is on, the stress probability of each transistor of a used two-level multiplexer is $\lambda = \frac{N_{used}}{N_{total}} = \frac{1}{\sqrt{X}}$. This provides higher recovery opportunity compared to $\lambda = 0.5$ in a tree-based multiplexer. More importantly, all transistors of an unused two-level multiplexer connect to *zero* gate voltage and are in the recovery phase.

3) Buffers: In order to effectively drive the load of wires, each routing multiplexer is followed by a large output buffer, particularly in the SB and cluster output multiplexers. Similar to the buffers used in standard cell libraries [35], the second stage of these buffers is typically larger than the first stage. Moreover, the pMOS to nMOS channel ratio in these buffers is customized to achieve a globally minimum area-delay cost [28], [31] and does not necessarily follow the ratio of minimum-size inverter. Because of the larger impact of NBTI, a conventional inverter undergoes higher aging degradation for a constant *zero*. Because of the relatively stronger impact of NBTI, the unequal size of buffer stages and the irregular ratio of each stage makes it impossible to estimate the impact of the signal value in FPGA buffers without performing an accurate circuit-level examination.

IV. The Proposed Method

In this section, we mitigate the FPGA aging. To this end, we first analyze the effect of aging of configuration cells on the reliability of FPGA, involving data integrity and delay. Based on our analysis, we target mitigating aging of routing multiplexers in Section IV-B which involves the proposed placement and routing method. In Section IV-C, we investigate the impact of the signal value on aging of routing buffers. Based on our examination, we modify the synthesis flow towards generating signals with less adverse impact (from delay perspective) on the aging of buffers.

A. Aging in Configuration Cells

Aging affects both SNM and SER of an SRAM cell which may increase its vulnerability to data corruption due to, respectively, intrinsic source of errors (e.g., voltage fluctuations and noise), and extrinsic sources such as particle strikes. In addition to data corruption, aging can cause timing errors in an SRAM-based memory array, e.g., a processor cache, because of increased read time. SRAMs, however, are employed differently in FPGAs. In contrary to a memory array in which SRAM cells drive bit-lines during the read operation, such concept of read operation does not apply for resources in FPGAs. That is, as shown in Fig.1b, SRAMs nodes are directly connected to multiplexer transistors, and hence, are continuously being read. Therefore, the conventional read failure either as timing violation or even data corruption during the read/sense operation does not take place in FPGA configuration cells.

Aging, however, may deteriorate the hold SNM of an SRAM cell, i.e., the SNM corresponding to the cell when it maintains the data, as it is in the case of FPGA configuration cells. In this regard, Fig. 2 demonstrates the degradation of the hold SNM of an SRAM cell considering different λ and over 10 years. The SNM data is obtained empirically by an industrial partner [27], [32]. λ of an SRAM cell is defined as the ratio of time the cell holds logical *one*. Due to the symmetric structure of SRAM cell that comprises two back-to-back inverters, for an SRAM cell we have $SNM(\lambda) = SNM(1 - \lambda)$. This is because whether stress cycle is λ or it is $1 - \lambda$, the stress ratio of transistors in one of the inverters is λ and $1 - \lambda$,

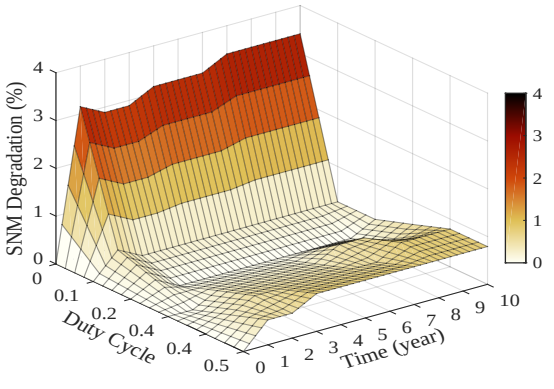


Fig. 2. Degradation of SRAM hold SNM due to aging. Data are derived from [27].

and in the other inverter is $1-\lambda$ and λ which causes the same effect. Therefore, here only the SNM for $\lambda \leq 0.5$ is represented. As λ converges *zero* (or *one*), SNM degradation increases because one pair of the pMOS-nMOS transistors will undergo a severe stress (note that the case of SRAM cell is different from transistor case whereby smaller λ means less aging). Nonetheless, aging has a slight impact on hold SNM of the SRAM cell on overall. Based on this figure, in the worst case, only 3.4% reduction in hold SNM is observed.

Extrinsic source of errors, on the other hand, such as radiation particles that deposit an electrical charge on SRAM nodes can corrupt SRAM data whenever the induced charge exceeds a threshold value, referred to as Q_{crit}. Since aging weakens the transistors transconductance ($g_m \propto I_d$), a reduction in SRAM Q_{crit} and hence, an increase in the SER is expected. Nonetheless, using a realistic distribution of energy and flux of neutrons, [27] has revealed that aging slightly shifts the Q_{crit} distribution which increases the failure probability by only 2.4%.

The reduction in transconductance of SRAM transistors, however, does not affect the multiplexer delay. This is because the SRAM cells are connected to the gate of transistors; hence, no driving current flows through them when an input of a routing multiplexer switches. That is, until the SRAM cells provide strong V_{dd} (which is not influenced by aging), delay of the multiplexer remains intact. We demonstrate this in Fig. 3 for both pass-gate (with and without voltage boosting) and tranmission-gate based architectures. The experimental setup for these results follows the same detailed in Section V. As shown in this figure, SRAM aging affects the delay of PG- and TG-based multiplexers by less than 0.2% which is due to the small voltage fluctuations of SRAM cells.

It is worth to remind that optimal balancing of SRAMs stress cycle (i.e., $\lambda = 0.5$) basically is not practical in two-level multiplexers. As explained in Section III, in a two-level multiplexer, one of the SRAMs of the first level and another one in the second level are *on* ($Q = 1$). Therefore, in a $m \times n$ multiplexer, $\frac{1}{n}$ and $\frac{1}{m}$ of the SRAMs at the first and second stage hold *one*. This leads to $\lambda_{L1} = 1 - \frac{1}{n}$ and $\lambda_{L2} = 1 - \frac{1}{m}$ which becomes close to $\lambda_{worst} = 1$ as the size of multiplexer increases.

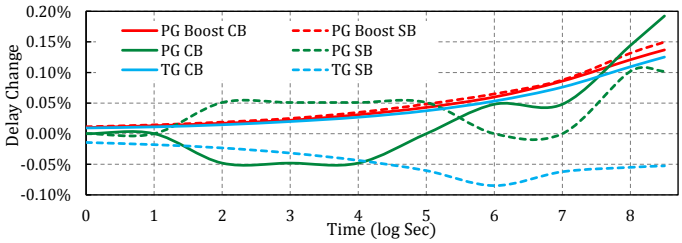


Fig. 3. Impact of SRAM aging on the delay of SB and CB multiplexers

B. Multiplexers

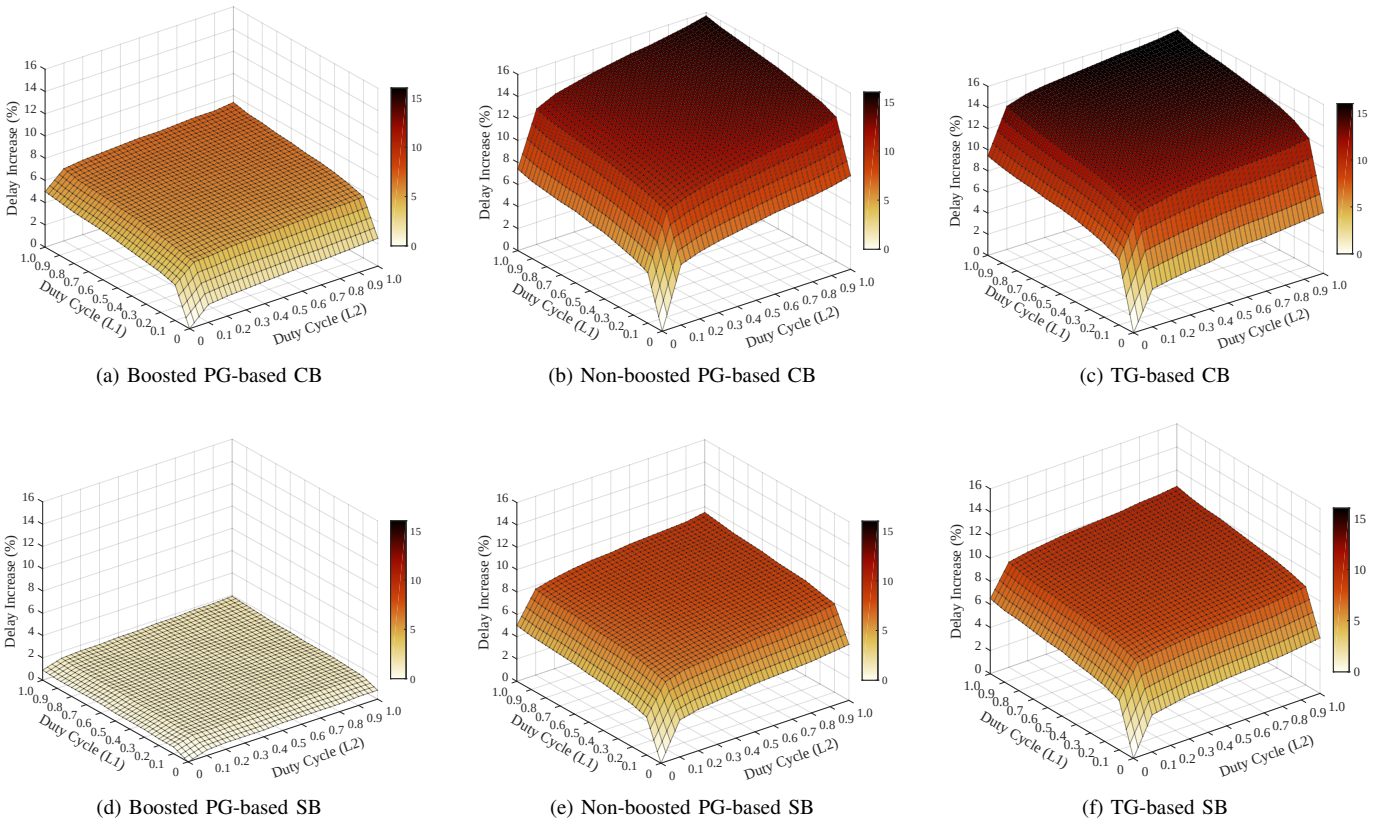
The *on* SRAMs in each level of the multiplexers degrade all corresponding nMOS transistors at the first stage. This includes one transistor at each of m bunches in the first level, and a single transistor at the second stage. Therefore, λ of a transistor at the first and the second stage is, $\frac{1}{n}$ and $\frac{1}{m}$, respectively. For large input multiplexets, these stress probabilities might be considered small enough to cause degradation. However, it is possible for a transistor to be connected to an *on* SRAM in consecutive designs, which is a common issue [14].

Fig. 4 demonstrates the impact of aging (after 10 years) on the delay of CB and SB multiplexers for various stress level (λ) on the transistors of the first (L1) and second (L2). Here, only the aging of the multiplexer transistors is considered. Aging of the output buffers is studied later. The experimental setup for these results follows the same detailed in Section V.

The following observations can be made from the figure.
(a) Unlike the SRAM cells, aging of the transistors of routing multiplexers causes significant performance degradation. Hence, alleviating the aging of multiplexers is vital.
(b) With the same level of degradation, L1 and L2 transistors in certain architectures increase the multiplexer delay differently. This needs to be considered when giving priority to stress reduction.
(c) Whether L1 or L2 transistor, the rate of delay increase with respect to stress cycle, λ, is exponential. It has a spike as λ increases from 0 to 0.2. Afterward, the slope of delay reduces and saturates in the tail distribution.

In the following, we incorporate the aforementioned observations and explain the proposed placement and routing algorithm.

1) Aging-aware Placement: State-of-the-art FPGAs supply an abundant number of logic resources, ranging in number of LUTs from 46K to 354K in Xilinx Virtex-6 [33], and 326K to 1,139K in Virtex-7 FPGA families [36], which is significantly higher than the requirements of the majority of applications. In such cases that the number of design blocks is considerably less than the available FPGA blocks, the FPGA CAD flow attempts to encompass the logic blocks in close proximity to avoid using global, long wires. This causes non-uniform utilization of FPGA logic clusters and hence interconnection resources, because the placed clusters utilize the adjacent interconnection resources for routing. Therefore, in a coarse-grained approach, we first aim to balance the utilization probability of the clusters by defining *placement bounding boxes*



(a) Boosted PG-based CB (b) Non-boosted PG-based CB (c) TG-based CB

(d) Boosted PG-based SB (e) Non-boosted PG-based SB (f) TG-based SB

Fig. 4. Delay degradation of CB and SB two-level multiplexers (with output buffer) with respect to the aging of the first (L1) and second level (L2) transistors

or modifying the original placement of designs. According to the size of the design and placement of the previous designs, the size and location of bounding boxes vary in consecutive configurations to help to balancing the resource utilization and transistors stress.

Fig. 5 demonstrates the proposed method to equalize the probability of resource utilization. Fig. 5b shows how the original circuit in Fig. 5a has been forced to be placed in the left upper-hand corner of the device. As mentioned before, the original circuit in Fig. 5a is automatically placed densely to shorten the global nets. Thus, defining a temperate bounding box shown in Fig. 5b do not congest the routing, especially considering the ample routing resources, i.e., channel tracks. We examine the impact of the bounding box on the circuit delay in Section V. Notice that the bounding box could be defined in, for example, the center of the device. However, to provide proximity to IO pads to avoid potential delay overhead, the bounding boxes should be defined in a way to include device sides. Therefore, the proposed method places the designs on device corners (top-left, top-right, and so on) repetitively.

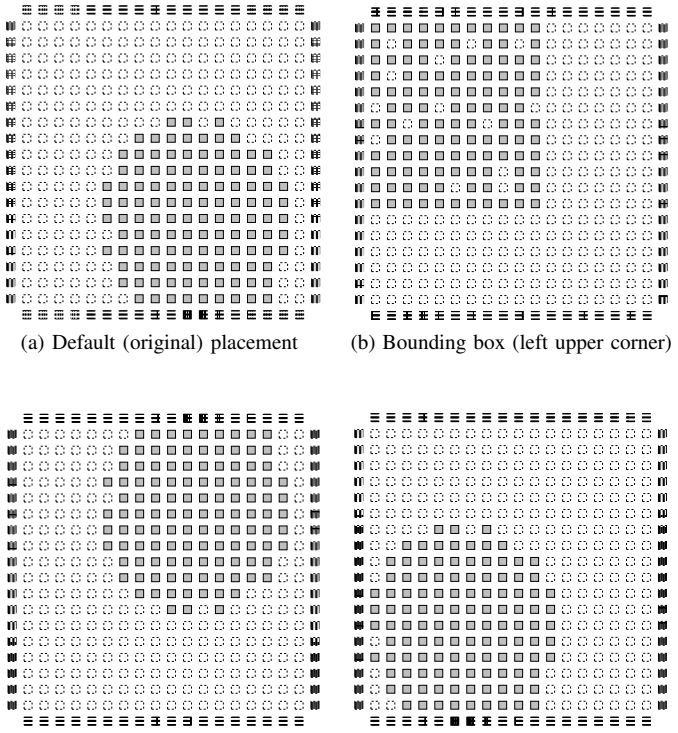
The minimum length of the bounding box (in terms of logic cluster) to provide sufficient logic clusters can be determined by Equation 5 in which n_{LUT} is the number of design LUTs, N is the number of LUTs in each cluster, and $k_{bb} \geq 1$ is an tuning constant to provide placement flexibility.

$$Length(bb) = \lceil \sqrt{\frac{n_{LUT}}{N}} \rceil \times k_{bb} \quad (5)$$

In addition to defining bounding boxes, a mapped design can be moved within the device to specific locations by flipping and/or shifting it without disturbing its original mapping, which keeps the design characteristics intact. Fig. 5c illustrates the original design of Fig. 5a, flipped with respect to the horizon. We should note that the purpose of the proposed placement method is different than previous studies such as [20], [22] that create multiple alternative configurations of a design that occupy different regions of FPGAs in order to reduce the toggle rate of transistors. Even though such approaches mainly target logic resources alone, keeping the resources unused does not guarantee aging reduction. The resources in unused regions could be connected to constant DC voltage which causes higher degradation even compared to used regions with high toggle rate [19].

2) Aging-aware Routing: The proposed placement method provides a high-level balancing, but it does not guarantee an optimal fine-grained balancing. For designs that occupy less than $\frac{1}{4}$ of resources, the proposed placement can guarantee a $\lambda_{max} = \frac{1}{4}$ by iteratively using different four quarter of designs whereby in each configuration only one region is utilized. However, such upper-bound for large designs cannot be assured since the overlapping region might be permanently under stress. Even for small designs, the effectiveness of placement is limited by the fact that when placing a design in a specific region, it has no fine-grained control of resources within the region, so some resources might be used constantly.

To address the aforementioned issue, we propose our aging-



(a) Default (original) placement (b) Bounding box (left upper corner)

(c) Flipping (horizontal) (d) Shifting (shift left)

Fig. 5. Modifying the FPGA placement to uniform utilization of resources

aware routing algorithm that controls the utilization of routing *transistors* and their stress rate in the finest granularity, i.e., per transistor. By using the proposed aging-aware placement followed by the proposed routing algorithm, the eventual goal is to optimally balance the long-term transistors stress rate which yields the value formulated by Equation 6, whereby $\bar{\alpha}_{logic}$ denotes the average utilized FPGA footprint and depends on the designs, and $\bar{\alpha}_{routing}$ indicates the utilization rate of routing resources (within the used regions) that can be as low as 20% [37]–[39]. $\frac{1}{\sqrt{X}}$ stands for the stress probability of a multiplexer transistor.

$$\lambda_{min} = \bar{\alpha}_{logic} \times \bar{\alpha}_{routing} \times \frac{1}{\sqrt{X}} \tag{6}$$

The routing algorithm of state-of-the-art FPGAs is based on the *PathFinder* algorithm [40] that performs multiple routing iterations by re-routing the nets using different paths in each iteration to achieve exclusive use of each resource as well as to reduce the delay. PathFinder controls the congestion-delay trade-off of the nets by their timing criticality. That is, the timing critical nets are routed using shortest paths even if it increases the congestion. The routing iterations are done until all resource overuses are eliminated while at the end of each iteration, the cost of overusing the resources will increase in order to eventually obtain a legal routing. The criticality of a connection from source block i to destination block j is determined using Equation 7 wherein $slack(i,j)$ and D_{max} (maximum delay of the nets) are updated based on timing analysis of the current iteration [41].

$$Crit(i,j) = 1 - \frac{slack(i,j)}{D_{max}} \tag{7}$$

In the next iteration, the cost of using a resource r for the i to j source-destination pair is obtained by Equation 8.

$$\begin{aligned} Cost(r) = & \overbrace{Crit(i,j) \cdot delay(r)}^{timing\ cost} \\ & + \overbrace{\big(1 - Crit(i,j)\big) \cdot \big(b(r) + h(r)\big) \cdot p(r)}^{congestion\ cost} \end{aligned} \tag{8}$$

In this equation, $delay(r)$ in the timing part denotes the delay of the particular resource r (e.g., SB multiplexer). In the second term (congestion), $b(r)$ is the base cost of the node r and is equal to $delay(r)$ in the initial versions of PathFinder [41]. $h(r)$ is the historical congestion of the node and increases in each iteration if the node is overused. Finally, $p(r)$ stands for the current congestion of the resource r; it is equal to 1 if using the resource r does not lead to overuse and increases with the number of overuses. $p(r)$ also depends on the number of iterations as it grows rapidly when the number of iterations increases.

We aim to modify $Cost(r)$ in a way that it accounts for the aging of a resource in the greatest detail, i.e., transistor level. For this end, we assign a variable for each transistor to keep track of its aging, λ. Note that an X-input multiplexer requires only $2\sqrt{X}$ values. Afterward, we update the timing term of Equation 8 to account for the degradation caused by increased λ of transistors. As it is observed in Section IV-B, the sensitivity of certain architectures to the aging of transistors of the second level is less than that of the first level. In addition, as λ diverges from 0, the multiplexers delay increases exponentially. Therefore, as it is formulated in Equation 9, we augment the timing cost of resource r with a factor of '$1 + k_{mux} \times (\sqrt{\lambda_{L1}} + k_{L2}\sqrt{\lambda_{L2}})$' to account the degradation of its active transistors whenever resource r is used. k_{mux} is a calibrating constant obtained empirically to trade off between the pressure for balancing by avoiding the use of highly stressed transistors and routability of the critical path nets. k_{L2} determines the weight of L2 transistors and is equal to 1 in the architectures with the same sensitivity to L1 and L2, and is equal to 0.5 (according to the observed slope of L1 and L2) wherein L1 is more critical.

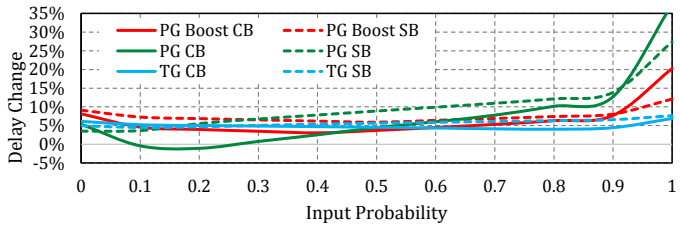
$$Cost_{timing}(r)' = Cost_{timing}(r) \cdot \big(1 + k_{mux}(\sqrt{\lambda_{L1}} + k_{L2}\sqrt{\lambda_{L2}})\big) \tag{9}$$

The λ of each transistor is updated before configuration of the next design based on the current configuration. It needs the operation time of the current design and total operation time of the FPGA device to be known which can be handled in the software level. λ_{new} of each transistor (trn) is updated according to Equation 10. T_{new} and T_{old} represent the total operating time of FPGA, respectively, with and without considering the operating time of current design. According to this equation, the operating duration of transistor trn in the current design ($T_{new} - T_{old}$) is multiplied by its logical gate voltage $V_{SRAM}(trn)$ that returns 0 or 1 to obtain the stress duration of the transistor in the current design. Then, it is summed up with the previous total stress duration of the transistor, $T_{old} \cdot \lambda_{old}(trn)$ and is divided to the total operating time to achieve the new λ.

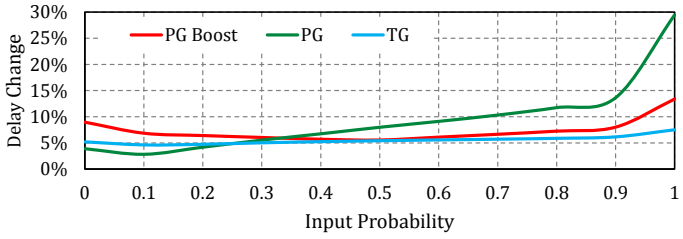
$$\lambda_{new}(trn) = \frac{(T_{new} - T_{old}) \cdot V_{SRAM}(trn) + T_{old} \cdot \lambda_{old}(trn)}{T_{new}} \tag{10}$$

C. Routing Buffers

As explained in Section III, the asymmetric transistor sizing and stage ratio, together with the unequal impact of aging

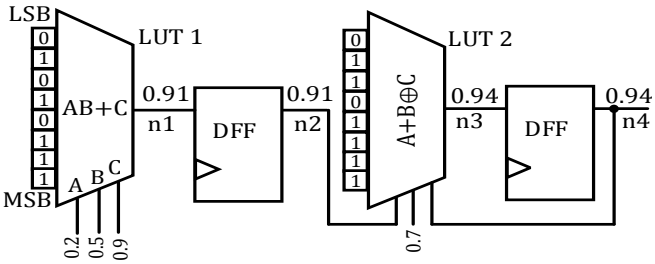


(a) Impact of the buffer aging on CB and SB delay

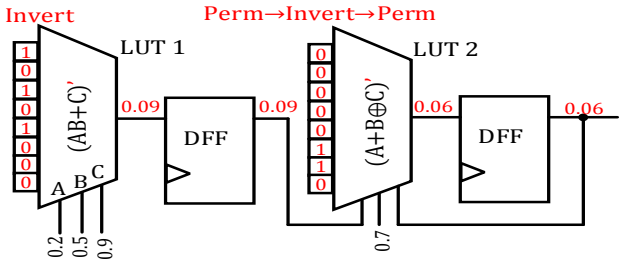


(b) Average impact of the buffer aging on routing delay

Fig. 6. Impact of the buffer aging on routing delay of different architectures



(a) Initial configuration and signal probabilities



(b) Modified configuration and resulted signal probabilities

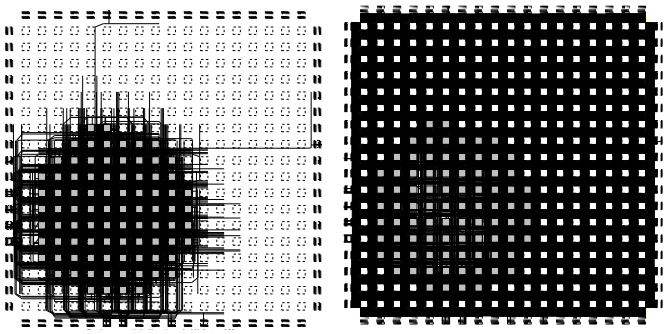
Fig. 7. Modifying the synthesis follow to imbalance the signal probabilities

on pMOS and nMOS transistors cause the routing buffers to have different levels of degradation for different probabilities of input signals. Notice that for a logical *one* input, the nMOS of the first stage (N1) and the pMOS of the second stage inverter (P2) are degraded. While for a *zero* input, the pMOS of the first stage (P1) and the nMOS of the second stage (N2) will be aged.

Fig. 6a shows the impact of aging on the CB and SB buffers of an FPGA considering different signal probabilities p_s, the probability of a signal to be logical one. The experimental setup for these results follows the same detailed in Section V. For the sake of brevity, only the aging values after 10 years have been illustrated and for each value of p_s, the maximum value of rise and fall delay is plotted since the impact of aging on the increase (or reduction [8]) of a gate delay is not necessarily similar. As it is shown in this figure, different architectures and even different resources in the same architecture have been affected differently by aging. By considering the average relative contribution of SBs and CBs in the total delay of a design, we show the average impact of buffer aging on routing delay of different architectures in Fig. 6b. According to this figure, PG-based boosted architecture (PG Boost) has minimum aging degradation for $p_s \simeq 0.5$ while non-boosted PG- and TG-based architectures undergo minimum aging for p_s around 0.1. Our investigations over the synthesized benchmarks show that around 70% of the configuration bits of the utilized LUTs hold *zero* which gives an intuition of imbalancing the signals toward 0.1 in the favor of non-boosted PG- and TG-based architectures. However, the signal probabilities do not necessarily follow the average value of configuration bits and are a function of LUT input values, as well.

To this end, we propose the following to imbalance the signal probabilities of the used buffers toward $p_s = 0.1$. First, we rely on an activity estimation tool, namely ACE 2.0 [42] to obtain the signal probabilities of a synthesized circuit. Fig. 7a demonstrates an example sub-circuit with the calculated signal probabilities. Thereafter, we write an in-house script laying over ACE 2.0 tool that iterates on the circuit netlist and flips the configuration bits of LUTs with $p_s > 0.5$. As shown in Fig. 7b, p_s of net $n1$ is changed from 0.91 to 0.09. This required permuting the bits of all of fanout LUTs in order to maintain the correct functionality. It is noteworthy than unlike ACE 2.0 that conducts simulations to calculate the signal probabilities of the sequential feed-back parts of circuits, we iteratively solve a non-homogeneous recurrence relation to obtain the p_s of such LUTs. We should note that the purpose of the proposed approach (negation and permutation) is different than previous studies that try to maximally inverse the SRAM cells to balance their aging. Our goal is to change the signal probabilities, which does not necessarily mean to negate all of SRAM cells. For instance, in Fig. 7b only half of the SRAM cells in LUT 2 are changed.

To control the p_s of the unused buffers, we route a constant *zero* fake net from one or some of the unused LUTs that have been configured to all *zero*. Similar to the placement bounding box constraint which is supported by commercial devices in detail [33], routing of such fake nets is facile by either building in-house scripts [43] using *Xilinx Design Language* (XDL) [44] or using already developed tools such [45]. Fig. 8 demonstrates an FPGA before and after the routing of constant (here, *zero*) fake nets. The gray blocks represent the used clusters. Based on the fact that the proposed aging-aware routing discussed in Section IV-B2 attempts to evenly use the routing multiplexers (and hence their associated routing buffers), the expected value for the long-term p_s of buffers denoted as $p_{s,expt}$ can be estimated as Equation 11 where $\bar{\alpha}_{logic} \times \bar{\alpha}_{routing}$ denotes the utilization probability of a buffer and $\bar{p_s}$ is the average signals probability. We take a similar approach for the boosted PG-based architecture for which the optimal p_s value is 0.5. Albeit, we imbalance the signals (both the used and unused) consecutively towards *zero* and *one* to



(a) Original (default) routing (b) Routing the constant nets

Fig. 8. FPGA layout without (a) and with (b) the *zero* nets

achieve a long-term average of 0.5.

$$p_{s,expt} = \bar{\alpha}_{logic} \times \bar{\alpha}_{routing} \times \bar{p_s} \tag{11}$$

We further calibrate the cost function of the routing resources in Equation 9 to account for the aging of the output buffer. Equation 12 represents the new cost function that takes the aging of routing buffers into account in which $p_{s,opt}$ is the optimal signal probability of the architecture (either 0.1 or 0.5) and $p_s(r)$ is the long-term signal probability of the resource r, updated at each configuration analogous to Equation 10. k_{buf} is a calibrating constant that denotes the sensitivity of architecture to deviating from $p_{s,opt}$ which is higher for non-boosted PG architecture as was shown in Fig. 6b.

$$Cost_{timing}(r)'' = Cost_{timing}(r)' \cdot (1 + |p_s(r) - p_{s,opt}|)^{k_{buf}} \tag{12}$$

D. Discussion

The proposed placement algorithm keeps track of the placed design and places the successive designs on the different corners of FPGA device consecutively. Thus, after, e.g., application A is placed starting the top left coordinate $(0, 0)$ of the device with a bounding box area defined in Equation 5, the next application (e.g., application B) will be placed starting from the top right coordinate $(0, n_{col})$ whereby n_{col} is the number of device logic columns. The upper bound of λ in a conventional placer is one ($\lambda_{max} = 1$), because it is possible for the conventional placer to place all applications in a certain region, e.g., all encompass the top left corner of the device. However, the proposed placer attempts to evenly distribute the resource utilization. In the long term, λ_{max} of the proposed placement can be estimated according to Equation 13.

$$\lambda_{max} = \frac{\sum_{i=1}^{N} max(\frac{1}{4}, \frac{n_{CLB}(i)}{n_{FPGA}})}{N} \tag{13}$$

In this equation, λ_{max} simply shows the long-term utilization probability of resources. The placement, itself, cannot guarantee a λ_{max} smaller than $\frac{1}{4}$ since every four times, it places a design in each of regions. Even if the design occupies less than $\frac{1}{4}$ of resources, the placer chooses the most corner clusters to provide adjacency to IO blocks. Thus, the minimum of λ_{max} is $\frac{1}{4}$ (at least for the corner clusters). The term $\frac{n_{CLB}(i)}{n_{FPGA}}$ indicates the utilization probability of resources, in which $n_{CLB}(i)$ is the number of clusters in application i, n_{FPGA} is the total number of device clusters, and N is the number of designs that have been placed over time. According to this equation, even designs occupying a large area of FPGA still take advantage of the proposed placement as it avoids successive utilization of regions and imposing permanent aging. Note that this is an upper-bound for λ and the proposed routing algorithm attempts to further reduce it as available routing resources are far more than required resources. Hence, utilization of a region does not mean all routing resources of that region are occupied.

The proposed placement algorithm can be further augmented by providing aging information for the placer. The timing-driven placers use the following cost function during their simulated-annealing based placement defined in Equation 14 [46].

$$\Delta cost = \gamma \cdot \frac{\Delta cost_{timing}}{cost_{timing}} + (1 - \gamma) \cdot \frac{\Delta cost_{wire}}{cost_{wire}} \tag{14}$$

In this equation, $\Delta cost_{timing}$ and $\Delta cost_{wire}$, respectively, indicate the difference of the timing and wiring costs between the new and previous placements, which are normalized to compensate the difference in their relative magnitudes. γ is a weight factor to trade-off between the timing and wiring costs. If $\Delta cost < 0$, the placer accepts the new placement and moves forward further enhancing. We define the aging cost during placement as follows.

$$cost_{aging}(i, j) = \frac{\sum_{\forall\, r \in paths\ from\ i\ to\ j} \lambda_r}{N_r} \tag{15}$$

In Equation 15, the aging cost between source node i and destination node j is calculated by taking average of stress cycle (λ_r) of all resources of potential *shortest* paths between the two nodes. Fig. 9 demonstrates an example from source S to destination D in which the switch box multiplexers in all possible shortest paths are distinguished (by black color). The rest of multiplexers are either diverging from destination D, or do not reside in the shortest paths. Therefore, while in the placement stage it is not exactly known which path the router will choose, the defined cost function instructs the placer to place the nodes in such a way that, during routing, resources with less aging will be chosen. That is, the potential paths between i and j will encompass the less-aged resources. The final $cost_{aging}$ is the sum of $cost_{aging}(i, j)$ for all available connections in the design. Eventually, we redefine the placement $\Delta cost$ as Equation 16, wherein $\gamma_2 \geq 0$ indicates the relative importance of $cost_{aging}$.

$$\Delta cost = \gamma \cdot \frac{\Delta cost_{timing}}{cost_{timing}} + (1-\gamma) \cdot \frac{\Delta cost_{wire}}{cost_{wire}} + \gamma_2 \cdot \frac{\Delta cost_{aging}}{cost_{aging}} \tag{16}$$

V. Experimental Setup and Results

In this section, we first detail the experimental setup used for the evaluations, including the toolset, architectural parameters

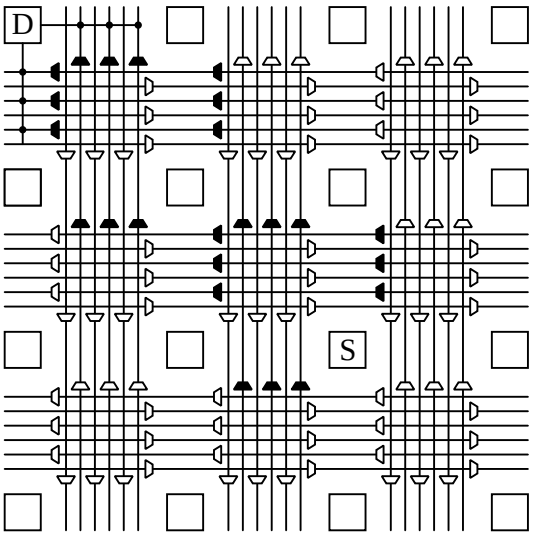


Fig. 9. SBs (black) located in potential shortest paths from source S to destination D.

and benchmark set. Afterward, using a third-party (i.e., trial) benchmark suite, we adjust the tuning parameters attributed to the placement constraint and cost function of the routing algorithm, explained already in Section IV-B and Section IV-C. Finally, we evaluate the effectiveness of the proposed method in alleviating the aging of routing resources and compare it with the baseline.

A. General Setup

We use the latest version of COFFE [28] to generate the circuit-level SPICE netlist of the routing resources. To do this, we feed it with the 22nm High-K transistor technology model from *Predictive Technology Model* (PTM) [47] (the architectural parameters are summarized in Table I). Similar to [31], we use a channel width of $W = 320$ in which the largest target benchmarks can be efficiently routed. Segment length of $L = 4$ that provides the most efficient interconnect amongst the fixed-length architectures [41] has been chosen. The other parameters (e.g., $K = 6$ according to commercial FPGAs [33], [36]) are adopted from VTR repository [29] which have been shown to provide high performance with affordable area. COFFE takes the resistive and capacitive parasitics of the global and local interconnects carefully into account and iteratively attempts to obtain a globally efficient architecture (i.e., transistor sizing) according to the user constraint, which we have set to minimize the area-delay product. As we consider different types of architectures, i.e., boosted and non-boosted PG-based as well as TMG-based FPGAs, we generate the SPICE netlist (along with the delay characteristics) of different resources in the mentioned architectures. We consider an operating voltage of $V_{dd} = 0.8V$ for all architectures while for the boosted architecture the supply voltage of SRAM cells have been set to $V_{SRAM} = 1.0V$. To model the aging, we employ an accurate physics-based aging model that jointly considers BTI and HCID (the degradations in both V_{th} and μ) [27]. Therefore, the results referring to the impact of aging in Section IV are obtained using the SPICE netlist provided by COFFE (with minor modifications) and simulating it with the modified transistor model that considers aging degradations. Table II summarizes the size of multiplexers and transistors generated by COFFE. The sizes of transistors are represented as multiple of minimum width transistor which is 45nm. For instance, width of pMOS (nMOS) transistors in the first and second stage of CB buffer in boosted-PG based architecture is 450nm (90nm), and 675nm (135nm), respectively.

TABLE I
ARCHITECTURAL PARAMETERS USED IN COFFE

Parameter	Value	Parameter	Value
K	6	F_s	3
N	10	F_{cin}	0.2
W	320	F_{cout}	0.1
L	4	X_{local}	$\frac{N+I}{2} = 25$
I	40	Or	1

TABLE II
TRANSISTOR SIZING GENERATED BY COFFE

	Boosted PG	PG	TMG
SB multiplexer size	12	12	12
CB multiplexer size	64	64	64
SB L1, L2	4, 5	5, 7	3, 3
CB L1, L2	2, 3	2, 3	1, 2
SB inverters ratios	14/5, 73/21	6/7, 52/15	6/9, 50/35
CB inverters ratios	10/2, 15/3	3/3, 10/2	3/5, 13/6

We implement the proposed method by using VTR 7.0 open-source toolset [29], including (a) ODIN-II [48] for initial synthesizing of the benchmarks from Verilog to BLIF format, (b) Berkeley ABC [49] to map the benchmarks of the previous step to LUT-6 and also to implement the proposed method to modify the signal probabilities (see Fig. 7), and (c) *Versatile Place and Route* (VPR) in order to map, pack, place, and route the benchmarks (already modified in the previous step by the proposed synthesis algorithm) to the user-defined island-style FPGA architectures by implementing the aging-aware placement and routing algorithms. We discriminate a total of 20 benchmarks including the largest benchmarks of VTR repository [29], IWLS'2005 [50], and five different types of processors: DSP, FFT, 5- and 6-stages RISC, and the Berkeley RISC-V processor [51]. The average number of LUTs and flip-flops of the benchmarks are, respectively, 15,553 and 6,560 where the largest benchmark (*mcml*) consists of 59,952 LUTs and 22,685 flip-flops. Therefore, we assume a 90×90 array device with total of 81,000 LUT-6 (close to Xilinx XC6VLX130T device [33] with 80,000 LUT-6) which can encompass the largest benchmark. We should note that the experiments in Section IV do not depend on FPGA size as the architectural parameters of routing network is independent of FPGA array size. In other words, for a particular FPGA architectural parameters, transistor sizing for routing network is fixed among all FPGA sizes.

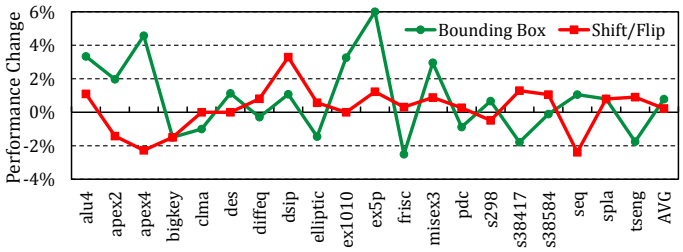


Fig. 10. Impact of bounding box and shift/flip placement constraints on the performance

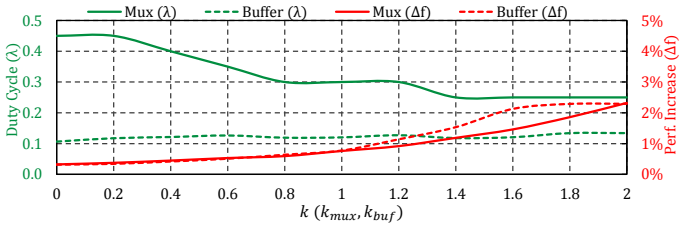


Fig. 11. Impact of k_{mux} and k_{buf} on the stress cycle of routing multiplexers and buffers (left axis) versus average performance degradation of the designs (right axis)

B. Calibration of Parameters

The proposed aging-mitigation methods introduced in Section IV involve defining a placement bounding box (according to Equation 5) or using shift and/or flip to restrict the placement boundaries to relax the aging in a coarse-grained manner, as well as modifying the routing algorithm to account for the multiplexers and buffers aging, according to Equation 9 and Equation 12. To calibrate the cost-function parameters in the mentioned equations (to avoid being restricted in the aforementioned specific benchmarks), we employ 20 largest MCNC benchmarks [52] as trial or tuning benchmarks.

In this regard, first, we examine the impact of bounding box and shift/flip based placement on the designs' performance. To do this, as the baseline and without any constraint, we map all the benchmarks on an 18×18 device with $W = 130$ that can encompass all of MCNC benchmarks. Using the same device, we then map the designs to an arbitrary corner using the minimum bounding box (i.e., $k_{bb} = 1$ in Equation 5). In addition, we examine the efficiency of shift/flip by shifting and/or flipping the original mapping of the design to the nearest corner (see Fig. 5c and Fig. 5d). Based on the results shown in Fig. 10, on average, the bounding box and shift/flip placement techniques do not impair the design performance wherein, respectively, 0.8% and 0.2% improvement in designs performance can be also observed. Note that such small changes in the performance of the designs are due to the heuristic nature of the placement. An advantage of shift/flip-based placement is maintaining the original mapping of the design, which can be the point of interest for partially reconfiguring designs since the relative placement of the blocks is preserved. The bounding box-based placement, however, precludes scattered placement of the blocks (when the device is larger than the mapped design) and assures minimum intersection between consecutive designs. Therefore, in our experiments, we opt the (minimum) bounding box solution, i.e., $k_{bb} = 1$.

To adjust the k_{mux} and k_{buf} which control the cost of reusing the transistors and buffers, we place and route the benchmarks iteratively on the device corners with bounding boxes, starting from *alu4* to *tseng* in a clockwise manner. Larger values for k_{mux} and k_{buf} maximize aging relaxation (by avoiding the consecutive use of a specific resource) but on the other hand impose higher pressure on the routing and increase the routing delay.

Fig. 11 represents the impact of different cost function factors, i.e., k_{mux} and k_{buf}, on the stress cycle of the routing multiplexer transistors and buffers. Note that x-axis has a base of 3.0 and 2.0 for k_{mux} and k_{buf} to be able to show the same range for both parameters. According to this figure, we opt $k_{mux} = 3.8$ $(3.0 + 0.8)$ since it provides an acceptable stress cycle ($\lambda \simeq 0.3$) with negligible performance degradation $\leq 0.6\%$. As mentioned earlier, larger values of k_{mux} lead to smaller λ values at the cost of considerable performance degradation. Unlike the multiplexer transistors, the optimum value of buffers λ is not necessarily *zero* (actually, it is either 0.1 or 0.5 depending on the architecture). Thus, for each k_{buf}, we show the maximum $|p_s(r) - p_{s,opt}|$ among all routing buffers (r). According to Fig. 11, we choose $k_{buf} = 2.0$ $(2.0 + 0.0)$ as it outcomes the smallest worst-case wherein the upper-bound of $|p_s(r) - p_{s,opt}|$ is 0.11, with only 0.32% reduction of the performance.

C. Results

After tuning the cost function parameters using the trial benchmarks, we examine the efficiency of the proposed method by conducting the experiments using the large test benchmarks explained in Section V-A. For the baseline case, we run (i.e., place and route) the 20 benchmarks consecutively and extract the stress cycle of the multiplexers and buffers after each benchmark. An equal runtime for all benchmarks is assumed while different runtime for each benchmark is also covered by Equation 10. For the proposed method, we define the benchmarks bounding boxes as a clockwise manner on the FPGA corners, followed by the proposed aging-aware routing. As already mentioned in Section IV-C, signal probabilities of the buffers are controlled by inverting the signals of the used buffers (if necessary) and routing the constant *zero*/*one* signals for the unused ones. Notice that since different architectures have different cost function parameters under the proposed method (hence, different aging result is expected), the experiments are performed separately for all architecture including boosted PG, typical PG, and TMG.

1) Impact on Performance: First, we investigate the impact of the proposed method on the FPGA performance to ensure that it does not impose performance overhead. Fig. 12 compares the performance of the proposed method and baseline using the test benchmarks. Note that due to the heuristic nature of FPGA placement algorithm, the initial placement plays a considerable role ($\pm 5\%$) in determining the post-routing timing characteristics. To compensate this effect in comparing the proposed routing algorithm, we conduct the experiments

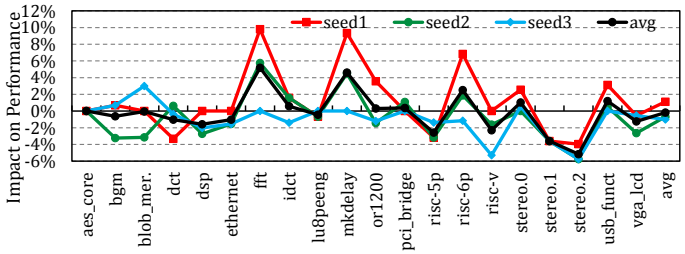


Fig. 12. Performance impact of the proposed method with respect to the baseline for different placement seeds

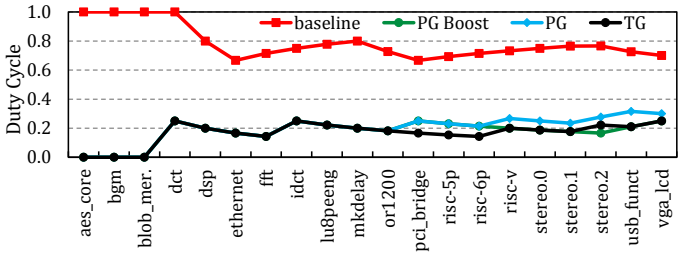


Fig. 13. Stress cycle of the worst multiplexer among the baseline and the architectures with the proposed method

using different initial placements (denoted by different seed numbers) to offset the role of initial placement. These seed numbers are passed to the pseudo-random generator function used in placement algorithm to generate different initial random placements [46]. We perform the experiments using 10 different seed numbers (for the sake of simplicity, only three are shown in Fig. 12). On average, only a trivial 0.01% degradation of the performance is noticed. Therefore, while the performance characteristic of each benchmark changes because of modifying its placement and routing, the average impact of the proposed method on the performance is negligible. Note that arbitrary oscillations in the performance of each benchmark is expectable. This is because the overall efficiency of the proposed algorithms is the same as the baseline. Hence, changing the initial placement changes the performance of each benchmark by a few percentage. This has nothing to do with the size of the designs. It is expected that increasing the number of runs (i.e., trying more seed numbers) will converge the average performance differences of the proposed method and the baseline in each benchmark towards zero.

2) Multiplexers Aging: After finding the multiplexer with the worst aging (upon running all benchmarks), we trace back the stress cycle of this worst-case multiplexer during the consecutive running of the benchmarks, which is shown in Fig. 13. Assuming the TG-based architecture, the worst-case multiplexer is not used in the first three benchmarks (*aes_core*, *ngm*, and *blob_mer.*), so its $\lambda = 0$ after these benchmarks. However, it is used in the forth benchmark (*dct*), making its λ equal to $\frac{0\times3+1}{4} = 0.25$. As shown by the results, the proposed method has reduced the multiplexers worst stress cycles from 0.7 in the baseline down to 0.25, i.e., a 64.3% reduction. Such reclaiming of the stress cycle enhances the aging-induced delay degradation of the multiplexers by 19.2%, 17.9%, and 16.3% in, respectively, boosted PG-, regular PG-, and TMG-based architectures.

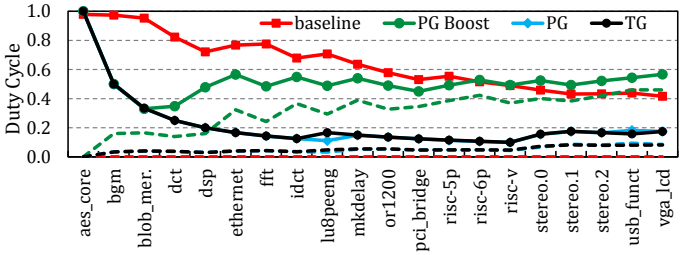


Fig. 14. Maximum and minimum stress cycle of the buffers in the baseline and the proposed method

3) Buffers Aging: Fig. 14 demonstrates the maximum (lines with marker) and minimum (dashed lines) stress cycle of the buffers for the baseline and the proposed method. Similar to the multiplexers, since the baseline does not implement any mitigation technique, stress cycles of the buffers are independent of the underlying architecture. That is, p_s of boosted PG-, non-boosted PG-, and TMG-based architectures is equal in the baseline case. However, since $p_{s,opt}$ of the architectures is different, we implement the proposed method separately for each architecture. According to Fig. 14, without any aging mitigation technique, p_s ranges from 0 to 0.42 after running 20 benchmarks. That is $[0.46, 0.57]$ for the boosted PG-, $[0.085, 0.174]$ for the regular (non-boosted) PG-, and $[0.081, 0.174]$ for the TG-based architecture. The worst p_s of each architecture can be obtained from Fig. 6. Comparing the delay corresponding to the worst p_s of each range reveals that the proposed signal imbalancing method reduces the aging-induced delay of the buffers by 33.8%, 45.2%, and 10.4% in the boosted PG-, non-boosted PG-, and TMG-based architectures. As an example, the worst signal probability for the boosted PG-based architecture is $p_s = 0$ for $p_s \in [0, 0.42]$ which results in $\Delta delay = 9\%$. However, the proposed method bounds the p_s to $[0.46, 0.57]$ which reduces the maximum $\Delta delay$ to 5.9%.

VI. Conclusion

In this paper, we analyzed the impact of aging on the reliability and delay degradations of FPGA routing resources in details. Based on our investigations, we aimed to reduce the stress duration of multiplexer transistors by balancing the use of interconnect resources in a coarse-grained manner by using constrained mapping without deteriorating the overall performance, followed by a fine-grained (i.e., intra-resource) method by proposing a stress-aware routing algorithm. By demonstrating the fact that different signal probabilities have different impact on aging-induced delay degradation, we modified the synthesis flow to converge the long-term probabilities of the FPGA routing tracks towards the most aging-friendly values (which can be different for various architectures). Experimental results show 64.3% reduction of the multiplexers stress duration which, in turn, enhances their aging-induced degradations by up to 19.2%. This is up to 45.2% enhancement for the routing buffers. Overall, 31.7%, 31.1%, and 14.1%

improvement in the aging guardband of routing network achieved for FPGAs with boosted PG-, regular PG- and TMG-based routing architectures. While most of the previous work have focused on balancing the aging of SRAM cells, our experiments revealed that aging has a negligible impact on SNM and SER of configuration SRAM cells. Furthermore, aging of SRAM cells basically does not influence the performance of FPGA.

REFERENCES

[1] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 26, no. 2, pp. 203–215, 2007.

[2] J. Henkel, L. Bauer, J. Becker, O. Bringmann, U. Brinkschulte, S. Chakraborty *et al.*, "Design and architectures for dependable embedded systems," in *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2011 Proceedings of the 9th International Conference on*. IEEE, 2011, pp. 69–78.

[3] J. Keane, X. Wang, D. Persaud, and C. H. Kim, "An all-in-one silicon odometer for separately monitoring hci, bti, and tddb," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 817–829, 2010.

[4] Y. M. Randriamihaja, V. Huard, X. Federspiel, A. Zaka, P. Palestri, D. Rideau *et al.*, "Microscopic scale characterization and modeling of transistor degradation under hc stress," *Microelectronics Reliability*, vol. 52, no. 11, pp. 2513–2520, 2012.

[5] K. Joshi, S. Mukhopadhyay, N. Goel, and S. Mahapatra, "A consistent physical framework for n and p bti in hkmg mosfets," in *Reliability Physics Symposium (IRPS), 2012 IEEE International*. IEEE, 2012, pp. 5A–3.

[6] M. A. Alam and S. Mahapatra, "A comprehensive model of pmos nbti degradation," *Microelectronics Reliability*, vol. 45, no. 1, pp. 71–81, 2005.

[7] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Nbti-aware synthesis of digital circuits," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 370–375.

[8] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, "Reliability-aware design to suppress aging," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*. IEEE, 2016, pp. 1–6.

[9] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno *et al.*, "Active management of timing guardband to save energy in power7," in *proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011, pp. 1–11.

[10] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, "Negative bias temperature instability: Estimation and design for improved reliability of nanoscale circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 4, pp. 743–751, 2007.

[11] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti, "Combating aging with the colt duty cycle equalizer," in *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2010, pp. 103–114.

[12] H. Amrouch, T. Ebi, and J. Henkel, "Resi: Register-embedded self-immunity for reliability enhancement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 5, pp. 677–690, 2014.

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

[14] B. Khaleghi, B. Omidi, H. Amrouch, J. Henkel, and H. Asadi, "Stress-aware routing to mitigate aging effects in sram-based fpgas," in *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*. IEEE, 2016, pp. 1–8.

[15] 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.

[16] M. Naouss and F. Marc, "Modelling delay degradation due to nbti in fpga look-up tables," in *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*. IEEE, 2016, pp. 1–4.

[17] 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.

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

[19] 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.

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

[21] P. M. Rao, A. Amouri, S. Kiamehr, and M. B. Tahoori, "Altering lut configuration for wear-out mitigation of fpga-mapped designs," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*. IEEE, 2013, pp. 1–8.

[22] 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 *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 38–45.

[23] Z. Ghaderi, N. Bagherzadeh, and A. Albaqsami, "Stable: Stress-aware boolean matching to mitigate bti-induced snm reduction in sram-based fpgas," *IEEE Transactions on Computers*, 2017.

[24] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-d fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 216–229, 2007.

[25] 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," *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2714–2726, 2007.

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

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

[28] S. Yazdanshenas and V. Betz, "Automatic circuit design and modelling for heterogeneous FPGAs," in *International Conference on Field Programmable Technology (FPT)*. IEEE, 2017, pp. 9–16.

[29] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk *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.

[30] D. Lewis, E. Ahmed, G. Baeckler, V. Betz, M. Bourgeault, D. Cashman *et al.*, "The stratix ii logic and routing architecture," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. ACM, 2005, pp. 14–20.

[31] 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.

[32] H. Amrouch, J. Martin-Martinez, V. M. van Santen, M. Moras, R. Rodriguez, M. Nafria *et al.*, "Connecting the physical and application level towards grasping aging effects," in *Reliability Physics Symposium (IRPS), 2015 IEEE International*. IEEE, 2015, pp. 3D–1.

[33] "Virtex-6 fpga configurable logic block," User Guide, Xilinx, February 2012.

[34] "Stratix iv device handbook," Handbook, Altera, January 2016.

[35] (2016) Nangate Open Cell Library. [Online]. Available: http://www.nangate.com/

[36] "7 series fpgas data sheet: Overview," Data Sheet, Xilinx, March 2017.

[37] Z. Seifoori, B. Khaleghi, and H. Asadi, "A power gating switch box architecture in routing network of sram-based fpgas in dark silicon era," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 1342–1347.

[38] Z. Ebrahimi, B. Khaleghi, and H. Asadi, "Peaf: A power-efficient architecture for sram-based fpgas using reconfigurable hard logic design in dark silicon era," *IEEE Transactions on Computers*, vol. 66, no. 6, pp. 982–995, 2017.

[39] Z. Seifoori, Z. Ebrahimi, B. Khaleghi, and H. Asadi, "Introduction to emerging sram-based fpga architectures in dark silicon era," *Advances in Computers*, 2018.

[40] L. McMurchie and C. Ebeling, "Pathfinder: a negotiation-based performance-driven router for fpgas," in *Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays*. ACM, 1995, pp. 111–117.

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

[42] 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.

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

[44] C. Beckhoff, D. Koch, and J. Torresen, "The xilinx design language (xdl): Tutorial and use cases," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on*. IEEE, 2011, pp. 1–8.

[45] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings, "Rapidsmith: Do-it-yourself cad tools for xilinx fpgas," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*. IEEE, 2011, pp. 349–355.

[46] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for fpgas," in *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*. ACM, 2000, pp. 203–213.

[47] (2013 (accessed August 13, 2017)) Predictive technology model (ptm). [Online]. Available: http://ptm.asu.edu/

[48] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, "Odin ii-an open-source verilog hdl synthesis tool for cad research," in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*. IEEE, 2010, pp. 149–156.

[49] A. Mishchenko *et al.*, "ABC: A system for sequential synthesis and verification," *URL http://www. eecs. berkeley. edu/˜ alanmi/abc*, 2007.

[50] C. Albrecht, "Iwls 2005 benchmarks," in *International Workshop for Logic Synthesis (IWLS): http://www. iwls. org*, 2005.

[51] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The risc-v instruction set manual, volume i: Base user-level isa," *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62*, 2011.

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

Behnam Khaleghi received his B.Sc. and M.Sc. degrees in computer engineering from *Sharif University of Technology* (SUT), Tehran, Iran, in 2013, and 2016, respectively. He is currently working as a research assistant in the *Data Storage, Networks, & Processing* (DSN) Laboratory at the Department of Computer Engineering, SUT. He spent the summer 2014 and 2015 as a research assistant at the Chair for Embedded Systems in the Karlsruhe Institute of Technology. His research interests include reconfigurable architectures and computer-aided design. He has two Best Paper Nominations at the DAC'17, and DATE'17.

Behzad Omidi received the B.Sc. and M.Sc. degrees in computer engineering from Shahed University and SUT, Tehran, Iran, in 2014 and 2016, respectively. He has been with the DSN Laboratory at the Department of Computer Engineering, SUT, as a researach assistant for three years. His current research interests include reconfigurable computing and reliability.

Hussam Amrouch received the Ph.D.(*summa cum laude*) degree from Karlsruhe Institute of Technology (KIT), Germany, in 2015. He is currently a Research Group Leader with the Chair for Embedded Systems, KIT where he is in charge of the Dependable Hardware Research Group. His main research interests are design for reliability, thermal-aware VLSI design, modeling, and mitigating aging effects at the device/circuit levels. He holds five HiPEAC Paper Awards. He has three Best Paper Nominations at the DAC'16, DAC'17, and DATE'17 for his work on reliability. He currently serves as an Associate Editor of *Integration, the VLSI Journal*.

Jörg Henkel received the Diploma degree and the Ph.D. (*summa cum laude*) degree from the Technical University of Braunschweig. He was a Research Staff Member with NEC Laboratories, Princeton, NJ, USA. He is currently the Chair Professor for Embedded Systems with the Karlsruhe Institute of Technology. His research is focused on co-design for embedded hardware/software systems with respect to power, thermal, and reliability aspects. He has received six Best Paper Awards throughout his career from, among others, the ICCAD, ESWeek, and DATE. For two consecutive terms he served as the Editor-in-Chief of the ACM Transactions on Embedded Computing Systems. He is currently the Editor-in-Chief of the IEEE Design & Test Magazine and is/has been an Associate Editor of the major ACM and IEEE Journals. He has led several conferences as a General Chair including the ICCAD, and ESWeek and serves as a Steering Committee Chair/Member for leading conferences and journals for embedded and cyber-physical systems. He coordinates the DFG Program SPP 1500 Dependable Embedded Systems and is a Site Coordinator of the DFG TR89 Collaborative Research Center on Invasive Computing. He is the Chairman of the IEEE Computer Society, Germany Chapter.

Hossein Asadi (M'08, SM'14) received the B.Sc. and M.Sc. degrees in computer engineering from the SUT, Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2007.

He was with EMC Corporation, Hopkinton, MA, USA, as a Research Scientist and Senior Hardware Engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. From 2001 to 2002, he was a member of the Sharif Rescue Robots Group. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently a tenured Associate Professor. He is the Founder and Director of the *Data Storage, Networks, and Processing* (DSN) Laboratory, Director of Sharif *High-Performance Computing* (HPC) Center, the Director of Sharif *Information and Communications Technology Center* (ICTC), and the President of Sharif ICT Innovation Center. He spent three months in the summer 2015 as a Visiting Professor at the School of Computer and Communication Sciences at the Ecole Poly-technique Federele de Lausanne (EPFL). He is also the co-founder of HPDS corp., designing and fabricating midrange and high-end data storage systems. He has authored and co-authored more than eighty technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and reconfigurable and dependable computing.

Dr. Asadi was a recipient of the Technical Award for the Best Robot Design from the International RoboCup Rescue Competition, organized by AAAI and RoboCup, a recipient of Best Paper Award at the 15th CSI International Symposium on *Computer Architecture & Digital Systems* (CADS), the Distinguished Lecturer Award from SUT in 2010, the Distinguished Researcher Award and the Distinguished Research Institute Award from SUT in 2016, and the Distinguished Technology Award from SUT in 2017. He is also recipient of Extraordinary Ability in Science visa from US Citizenship and Immigration Services in 2008. He has also served as the publication chair of several national and international conferences including CNDS2013, AISP2013, and CSSE2013 during the past four years. Most recently, he has served as a Guest Editor of IEEE Transactions on Computers, an Associate Editor of Microelectronics Reliability, a Program Co-Chair of CADS2015, and the Program Chair of CSI National Computer Conference (CSICC2017).