LUT Input Reordering to Reduce Aging Impact on FPGA LUTs

Mohammad Ebrahimi, Rezgar Sadeghi, and Zainalabedin Navabi, *Senior Member, IEEE*

Abstract—In this article, we propose a fine-grained FPGA aging mitigation method. Our method focuses on Look Up Tables (LUTs) on which Boolean functions are mapped. Based on our observations, for any configuration, even if it is carefully selected, a number of LUT transistors experience severe stress rates. Therefore, an algorithm is presented to select several alternative configurations for each LUT. Alternative configurations are obtained by LUT input reordering. These alternative configurations are rotationally loaded into the FPGA. Experimental results shows that our method achieves 263 and 14.1 percent Mean Time To Failure (MTTF) improvement for Hot Carrier Injection (HCI) and Bias Temperature Instability (BTI), respectively. Additionally, due to changing only local routings, our method imposes up to 1 percent performance overhead to the systems.

Index Terms—FPGA, aging mitigation, LUT input reordering, partitioning

1 Introduction

SRAM based FPGAs are widely used by digital system designers due to their configurability, time to market, cost, and high performance. A typical FPGA consists of an array of logic blocks called *configurable logic blocks* (CLBs) that are connected to each other through programmable routing resources. Each CLB consists of two *slices* and each *slice* contains several *look up tables* (LUTs) that may be configured to perform complex combinational functions and corresponding memory elements (*flip-flops*).

Fabricated in the state of the art silicon technology, FPGAs face reliability challenges such as aging. Bias Temperature Instability (BTI) and Hot Carrier Injection are two important phenomena causing accelerated transistor aging. Aging effects considerably reduce the lifetime of a chip by increasing the magnitude of the transistor threshold voltage and reducing the effective carrier mobility over time. The effect of aging on FPGA resources ranges from delay degradation to permanent failures in interconnects to metastability of SRAM cells containing configuration bits.

LUTs are one of the important resources in FPGAs that implement Boolean functions. LUTs have 6 available inputs (in *Xilinx* FPGAs) and can implement any Boolean function of up to 6 inputs. When programmed, because of difference in input activities, LUT transistors experience different stress [1]. This difference is significant and can be observed in almost all LUTs. This causes an FPGA to age unevenly. Therefore, a solution is required to distribute aging across LUT transistors. This will result in longer life of the FPGA and better performance of the circuit implemented on the FPGA.

In addition to stress distribution among LUT transistors, it is required to consider the worst case stressful transistors. This worst case stress affects the MTTF of the system. For any configuration, even if it is carefully selected, a number of transistors experience the worst case stress. Therefore, with a single configuration, the worst case aging cannot be reduced.

• *M. Ebrahimi, R. Sadeghi, and Z. Navabi are with the Electrical and Computer Engineering Department, University of Tehran, Tehran 14399-57131, Iran. E-mail: {ebrahimi.mo, rr.sadeghi, navabi}@ut.ac.ir.*

Manuscript received 20 Mar. 2019; revised 5 Dec. 2019; accepted 31 Jan. 2020. Date of publication 18 Feb. 2020; date of current version 7 Sept. 2020.
(Corresponding author: Mohammad Ebrahimi.)
Recommended for acceptance by D. Gizopoulos.
Digital Object Identifier no. 10.1109/TC.2020.2974955

In this paper, a fine-grained method is proposed to distribute stress on LUT transistors. For this purpose, instead of one configuration, several configurations, called *alternative configurations*, are specified for each LUT. These configurations are obtained by reordering and re-utilizing of LUT inputs. *Alternative configurations* implement the same Boolean function. Because of difference in LUT input activity rate imposed by the configurations, the stress of LUT transistors is different for each *alternative configuration*. If we select our configurations wisely, and use the configurations rotationally, the stress of LUT transistors can be more evenly distributed. Therefore, the most important step is to select *alternative configurations* among a pool of configurations.

The proposed method does not change the placement of stressful LUTs. Therefore, the simplicity in implementing the proposed method makes it practical, even for highly utilized FPGA designs. Additionally, input reordering involves changes to routing, albeit mostly local switch routing. This results in almost no impact on routing delay and very little impact on the delay of critical paths.

The rest of the paper is organized as follows. Section 2 explains the LUT structure we used and introduces the stress distribution problem in LUTs. Section 3 explains the details of some prior works. Section 4 presents the details of our proposed method. Experimental results are shown in Section 5. Finally, the paper is concluded in Section 6.

2 LUT Structure and Imbalanced Stress Issue

This section covers the background of our work, which consists of definition and structure of LUTs. This presentation is not complete and not necessarily exact, but it will be enough to justify the analysis that we require in this paper. We then discuss stress as it applies to transistor level structure of LUTs.

Various structures are presented for LUTs [2]. In this paper, we focus on the LUT structure shown in Fig. 1. The same principles are also applicable to other LUT structures. LUTs, shown in Fig. 1, have six independent inputs (A inputs, $A1$ to $A6$) and two independent outputs ($O5$ and $O6$). This structure has 64 configuration bits and can be configured either as a 6-input LUT or two 5-input LUTs. In the 6-input mode, A6 is used as the sixth input and O6 is the LUT output. In the 5-input mode, A6 is driven by a high signal [3], and O5 and O6 are the outputs of the LUT.

The term stress is the condition under which the transistor degrades in some of its features. There are two types of stress related to the activity of the transistors. One *dynamic stress* that is due to the toggling rate of the transistors ($\alpha \times f$). Here, α and f are the switching activity and operating frequency, respectively. Another type is due to inactivity of transistors. Static stress is characterized by the duty cycle (Y), i.e., the percentage of operating time that a transistor is conducting. In other words, the static stress S_{stat} and dynamic stress S_{dyn} are equal to:

$$S_{stat} = Y_g \times P(V_{gs} = 1) \tag{1}$$

$$S_{dyn} = \alpha_g \times f, \tag{2}$$

where, $P(V_{gs} = 1)$ is the probability that $V_{gs} = 1$. Switching activity and duty cycle of an LUT input are referred to as input signal probabilities. To calculate static and dynamic stress of transistors, input signal probabilities are propagated through the LUT model. Fig. 2 shows the calculated duty cycle and switching activity of transistors of a 2-input LUT. The amount in parentheses indicates the duty cycle and switching activity of LUT inputs in Figs. 2a and 2b, respectively. Unused inputs are considered to be $'0'$ (duty cycle and switching activity are $'0'$). As can be seen, while some transistors are under high stress (red transistors, darker shades), others experience moderate

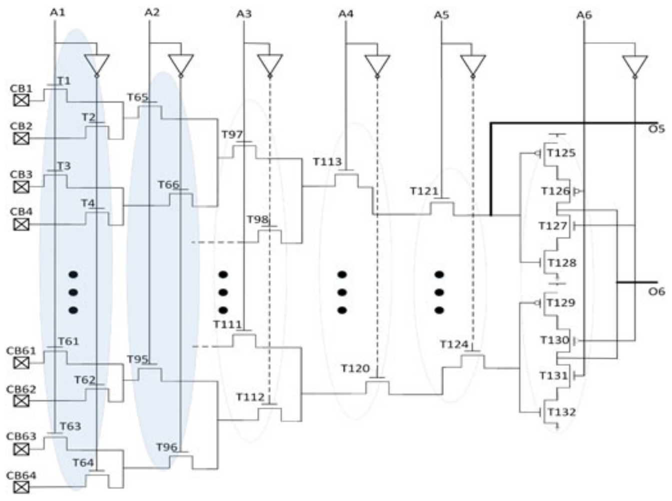

Fig. 1. LUT structure [2].

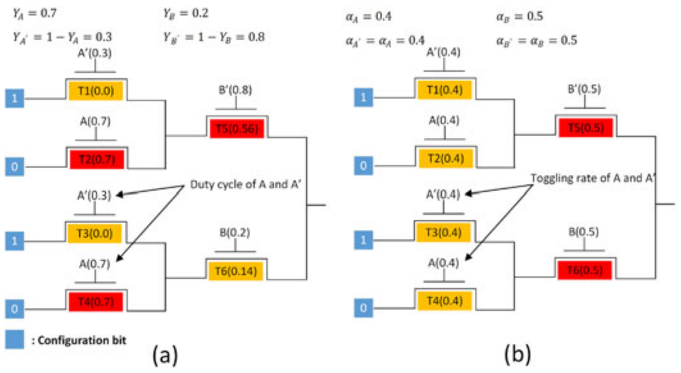

Fig. 2. Static and dynamic stress calculation.

and low stress (yellow transistors, lighter shades). Although the severity of this imbalance stress distribution is higher in LUTs whose inputs are only partially utilized, this situation is also observed in almost all fully input-utilized LUTs.

3 Related Work

Several aging mitigation methods have been proposed for aging mitigation on ASIC and FPGA based designs. Initially, a brief overview of the appropriate methods for ASIC designs is given and then the FPGA based solutions are discussed.

Authors in [4] presents a BTI mitigation method to minimize aging effects in the I-cache SRAM cells. In their method, two normal and recovery modes are defined for cache lines. In the recovery mode, the content of the target cache line is replaced by a predefined value that is optimal for aging mitigation of the I-cache line. A method to mitigate NBTI has been proposed, [5], in which the signal probability is modified using Input Vector Control (IVC) or Multiple Input Vector Control (MIVC) during the stand-by mode of the circuit. Authors in [6] presents a simulation framework called OldSpot that is able to analyze wearout and estimate the lifetime of a system at the unit level with arbitrary distribution of workloads, and therefore variations of temperature, voltage, frequency, and so on. In [7] a methodology and a circuit for the periodic monitoring of BTI induced aging in SRAM sense amplifiers is presented. According to the proposed method, periodic monitoring provides the ability to avoid SRAM failures by early detecting over-aged sense amplifiers (near failure) and then properly react in order to maintain the reliable memory operation.

Based on the granularity, the solutions proposed to mitigate the stress on FPGAs are divided into three categories. The first category focuses on methods providing solutions consider aging mitigation on the entire FPGA (coarse-grained). For example, in [8], authors propose a three-step post-synthesis stress-aware method, in order to reduce the impact of BTI in FPGA Look-up-Tables (LUTs) using the SAT-based Boolean Matching (BM) algorithm. Their method results in two configurations with flipped configuration bits for some LUTs. Authors in [9] presents a stress-aware placement algorithm that takes the degradation of reconfigurable resources into account. The method proposed in [10] introduces a routing accompanied by a placement algorithm that prevents constant stress on transistors by evenly distributing the stress through the interconnection resources. In [11] a high-level physical planning with reconfiguration strategy is presented in order to mitigate the aging-induced delay degradation in FPGA resources. The proposed solution is an offline framework composed of an aging aware floorplanner coupled with a proactive aging-aware reconfiguration policy which generates checkpoints aperiodically for runtime reconfiguration. Furthermore, the impact of BTI on SRAM cells in memory block has been studied in [12]. For example, in [12], a periodic flipping method is proposed for mitigating aging in SRAM-based memories.

In [13], solutions are demonstrated to counter each failure and degradation phenomena to increase the operating lifetime of the FPGAs. These solutions include load balancing to mitigate HCI and flipping configuration bits to reduce the NBTI effects on FPGAs. Authors in [14] propose workload-aware supply voltage assignment approach to reduce aging. They propose an NBTI and leakage co-optimization strategy based on an integer linear programming (ILP) approach to obtain the optimal input vector used when some modules of the design are in the standby mode. In [15], relocating logic functions to unused LUTs and rerouting signals to unused interconnect are performed to mitigate degradation in FPGAs.

The second group of FPGA aging methodologies deals with the fair distribution of stress among CLBs. This is a level below the first group discussed above and we refer to it as the medium-grained methods. These methods try to find alternative configurations for each configuration provided by the tool. In [16] and [17], authors propose a LUT level aging mitigation method. When programming an FPGA region, their method considers the previous aging of LUTs and estimates the amount of stress imposed by the current configuration. Then, based on this consideration, their method places accelerators on appropriate regions. However, in some cases, some prohibitive constraints are applied to avoid high stress imposition on some LUTs. Such methods distribute aging among LUTs. However, there is no mechanism for aging distribution among LUT transistors. On the other hand, because of changes in some paths due to some placement constraints, their method can increase involved path delays.

Transistor level aging mitigation methods fall into the third category (fined-grained). In [18], authors present a method that manipulates the configuration of used LUTs and their input signal probabilities to obtain a degradation-friendly LUT configuration. They consider the don't care bits of configuration bits (in partially utilized input LUTs) and select the best configuration among a pool of configurations to mitigate aging. In addition, they reorder LUT inputs to select the best configuration for fully utilized input LUTs. This method is interesting, however, it cannot be applied to commercial FPGAs because, in general, the synthesis tools (in this case *ISE* and *VIVADO*) do not allow to the unused configuration bits to be changed freely. Furthermore, based on our observations, for any configuration, even if it is carefully selected, a number of transistors experience the worst case stress. Furthermore, because of logic optimization process and power restrictions, unused inputs are connected to '0' in real FPGA implementations. Connecting such inputs to V_{dd} may cause routing congestion in actual implementations. In [15], using spare LUT inputs to obtain a degradation-friendly LUT

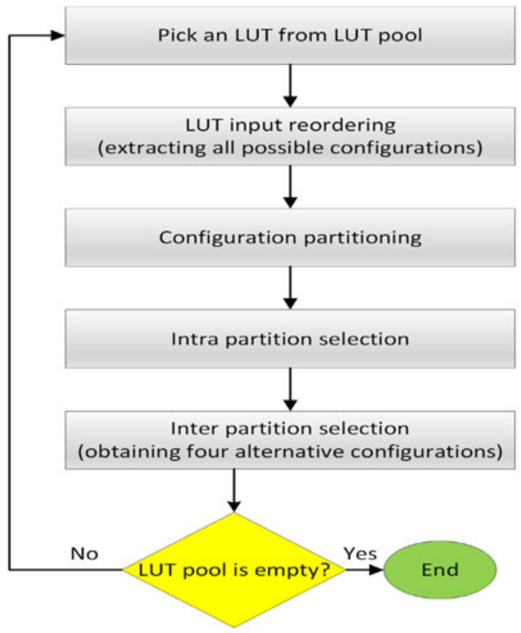
Pick an LUT from LUT pool
LUT input reordering
(extracting all possible configurations)
Configuration partitioning
Intra partition selection
Inter partition selection
(obtaining four alternative configurations)
No
LUT pool is empty?
Yes
End

Fig. 3. Overview of our method.

configuration is investigated. The paper concludes that using such a configuration does not lead to significant improvement in aging mitigation.

4 Proposed Method

The objective of our work is to come up with an aging-aware selection of *alternative configurations* for LUTs. Fig. 3 shows the overview of our method. First of all, an LUT is picked up from the set of design LUTs. Then, using LUT input reordering, all possible configurations for the LUT are extracted (Section 4.2). To select the alternative configurations, it is required to perform an exhaustive search in available possibilities. Because of the large search space size (discussed in Section 4.1), an exhaustive search is impossible. Therefore, a heuristic algorithm is used which is based on partitioning. Thus, the set of possible configurations is split into several partitions, where each partition consists of a set of configurations. Then a representative configuration is selected within each partition (Intra partition selection, Section 4.3.1) and then an exhaustive search is performed among these representatives (Inter partition selection, 4.3.2) and finally four alternative configurations are selected. The next subsection justifies the reason for using a heuristic algorithm.

4.1 Problem Size

The search space size of selecting alternative configurations becomes prohibitively large that a simple search is not possible. This Section illustrates this fact.

Fig. 4 shows the mapping of function variables ($\Sigma(a, b, c, d)$) to LUT inputs ($A_1 - A_6$). The center implementation (Fig. 4b) is

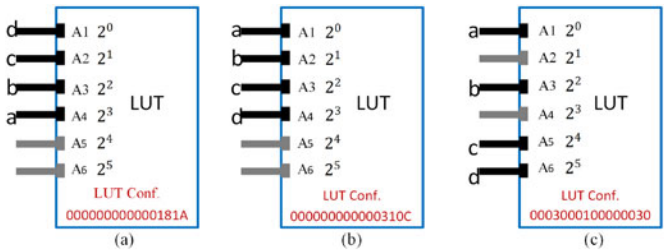

Fig. 4. LUT input reordering and corresponding configuration bits.

TABLE 1
The Size of the Search Space to Find the 4 Alternative Configurations Among all Possible LUT Configurations With n $(2 \leq n \leq 6)$ Active Inputs

	# of active inputs				
	2	3	4	5	6
srch. space	27405	8214570	$6.8e8$	$1.1e10$	$1.1e10$

achieved by reordering the LUT inputs of Fig. 4a. However, in Fig. 4c implementation, the active inputs are changed as well as their order. Input reordering changes the signal probabilities on inputs and also the configuration bits of the LUT that causes change in the transistor stress.

The total number of ways for mapping a 4-variable function to 6-input LUT inputs is permutation of 6 that is P(6,4) =360. The purpose of the selection algorithm is to pick up a set of four alternative configurations (the reason of selecting 4 alternative configurations will be discussed in Section 4.4 in details). Therefore, the search space of selecting among all possible variable mappings is a combination of 4 from 360 that equals 688,233,310. Table 1 shows number of selections for picking configurations among all possible variable mappings for Boolean functions with 2 up to 6 variables mapped to a 6-input LUT.

The best result can only be obtained by an exhaustive search algorithm. However, it is impossible with such a large search space. For two-active input LUTs we can still perform an exhaustive search, but for LUTs with a larger number of active inputs a heuristic search based on partitioning will be done.

4.2 Partitioning Possible Configurations

To reduce the search space, this paper proposes a partitioning-based algorithm which initially places LUT configurations with similar effects on stress of transistors into a partition. Then, within each partition, the best configuration is selected (*Intra partition selection*). This is regarded as the partition's representative. Then, an exhaustive search among all these representatives is performed, which results in selection of four configurations (*Inter partition selection*) that have the optimal criterion value that will be presented in Section 4.3.2.

The partitioning procedure is performed based on LUT inputs. As shown in Fig. 1 that shows the structure of an LUT, the LUT inputs are connected to the gates of a different number of transistors, for example inputs A1 and A2 are connected to 64 and 32 transistors, respectively. As can be seen, each input puts a direct stress on the transistors it is connected to (e.g., A1 on transistors 1-64), and indirectly affects stress on transistors in the next stages. The first two stages whose transistors are driven by A1 and A2 directly affect the stress of 96 transistors (64+32=96). For a given Boolean function, any LUT configuration that connects same Boolean variables to A1 and A2 LUT inputs are considered as part of the same group and are placed in the same partition.

Consider an LUT to which a 6-variable Boolean function is mapped. For such an LUT, the number of partitions is equal to the number of ways of mapping two Boolean function variables among ($I_1 - I_6$) to A_1 and A_2. As the order does matter, the total number of such mappings (i.e., the number of partitions) is equal to 30 (P(6,2) =30). Additionally, the number of configurations inside a partition is equal to the number of ways of mapping of the four remaining variables to A_3, A_4, A_5, and A_6 (P(4,4)=24).

For a 6-input Boolean function with 720 configurations (i.e., 30 $\times$ 24) as above, stress rate on individual transistors in an LUT for every alternative configuration are shown in Fig. 5. In Fig. 6 we have rearranged LUT configurations according to the partitioning described, i.e., connecting Boolean variables to A1 and A2. Vertical slices in Fig. 6 correspond to the 30 partitions mentioned above in which they impose similar stresses on LUT transistors. Each slice is composed of 24 different configurations.

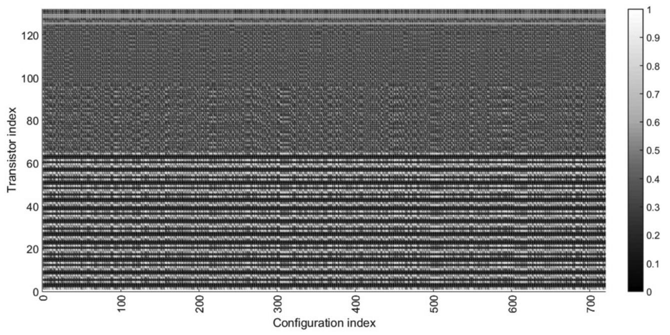

Fig. 5. Stress of transistors in an LUT for every alternative configuration (before partitioning).

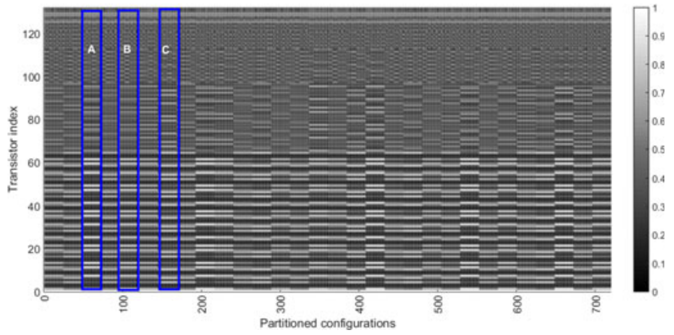

Fig. 6. Stress of transistors in an LUT for partitioned configurations.

For other Boolean function sizes, we basically follow the same procedure. The partitioning procedure for a 5-active input LUT is the same as that of 6-active input LUTs. Partitioning for 3 and 4-active input LUTs are slightly different. Consider a 4-active input LUT. Mapping to A_1 and A_2 of LUT to the four Boolean variables can be done in four possible ways.

1) A_1 and A_2 are utilized
2) A_1 is utilized and A_2 is not used
3) A_1 is not used and A_2 is utilized
4) No variable is mapped to A_1 and A_2

The number of partitions of case 1 above is the number of possible mappings of four variables to A_1 and A_2 (i.e., $P(4,2) = 12$). The number of configurations of these partitions equals the number of ways in which two remaining variables can be mapped to A_3, A_4, A_5, and A_6 (i.e., $P(4,2) = 12$). Therefore, case 1 has 12 partitions each of which includes 12 configurations.

In the same way, the number of partitions are 4, 4, and 1 for cases 2, 3, and 4, respectively. Each of these partitions includes 24 configurations.

As discussed, the number of configurations in the partitions of the above four cases are not the same. To be consistent with the 6-variable case of 30 partitions, and for easier handling of the above cases, we split the configurations of the above four cases such that each partition will have an equal number of configurations that is 12. This way, we will end up with 12, 8, 8, and 2 partitions for the above cases. Each partition has 12 configurations. As before, the total number of partitions for 4-variable Boolean functions remains 30, as with the 6-variable Boolean functions. Similar to the above, for 3-variable Boolean functions after splitting the partitions, we will have a total of 30 partitions each with 4 member configurations.

4.3 Heuristic Algorithm

In the above, we showed how mapping of 6, 5, 4, and 3 variable functions to 6-input LUTs are grouped into 30 partitions with 24, 24, 12, and 4 configurations. In the algorithm that we discuss here, we show how only one configuration is selected among the configurations of a partition, i.e., 1 from 24, 24, 12, or 4, and then selecting four configurations among the possible 30. The former is referred to as *Intra partition selection*, and the latter is the *Inter partition selection*. The *Inter partition selection* selects four configurations to rotate between these four stress optimized configurations, as discussed in Section 4.3.2.

4.3.1 Intra Partition Selection

At this step, a representative configuration is selected from each partition. This selection is based on the stress standard deviation of LUT transistors in each configuration ($\sigma(C_i)$). Standard deviation is a suitable indicator of the amount of stress distribution in a configuration. Therefore, any configuration that has the minimum $\sigma(C_i)$ value is added to the list of representative configurations. Standard deviation is defined as:

$$\begin{aligned}\sigma(C_i) &= \sum_{j=1}^{m} \mid S_{i,j} - \mu_i \mid \\ with \quad \mu_i &= \frac{1}{m}\sum_{j=1}^{m} S_{i,j},\end{aligned} \tag{3}$$

where C_i is a configuration in a partition, m is the number of LUT transistors (m = 132), μ_i is the average stress of LUT transistors, and $S_{i,j}$ represents the stress that C_i imposes to the jth transistor of the LUT.

Algorithm 1 describes the selection procedure of representative configurations. This algorithm takes the list of LUTs and all possible configurations of each LUT obtained by input reordering, and gives the list of alternative configurations for each LUT as output. First, partitioning is performed for a selected LUT (line 3). Then, the *Intra partition selection* (lines 5-14) must be performed to obtain the list of representative configurations ($ReprList$). This list is set initially empty (line 4).

In the *Intra partition selection*, for each partition, initially the minimum standard deviation value (Min_std) is set to infinity (line 6). Then, inside each partition, standard deviation of each configuration (Eq. (3)) are calculated. If this value is less than the current value of Min_std (line 8), the Min_std and temporary representative configuration Tmp_Repr are updated with the current value of $\sigma(C_i)$ and C_i, respectively (lines 9 and 10). Once this procedure is performed for all configurations inside a partition, the Tmp_Repr becomes the representative configuration of that partition, and is added to $ReprList$ (line 13). The next step is the *Inter selection* procedure described below.

4.3.2 Inter Partition Selection

After extraction of representative configurations at the *Intra partition selection*, we have a $ReprList$ for each LUT. A $ReprList$ is equal to:

$$ReprList = \{RC_1, RC_2, \ldots, RC_{30}\} \tag{4}$$

where RC_i is the **R**epresentative **C**onfiguration of the **i**th partition. The *Inter partition selection* procedure selects a 4-member subset of $ReprList$, using an exhaustive search. Note that member replication is allowed.

The criterion to use is the next key issue that we are discussing here. The proposed algorithm must consider the worst stress scenario as well as the stress distribution. This is because the Mean Time to Failure (*MTTF*) of LUTs is affected by its most vulnerable transistor (i.e., the most stressful transistor). In other words, a criterion is required that considers both the distribution of stress and the worst case stress on the LUT transistors. Therefore, the standard deviation and the maximum stress of LUT transistors, as the index of distribution and worst stress, contribute to the *Inter partition selection* criterion (*Std_Max*).

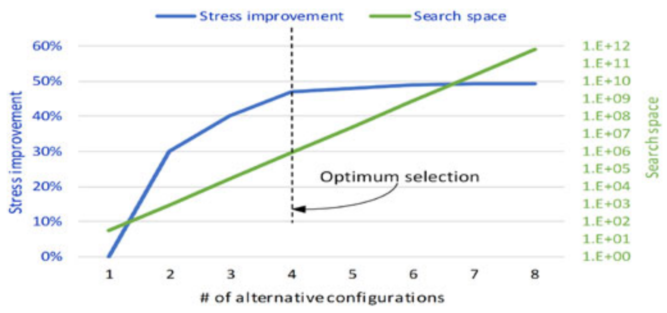

Fig. 7. Trade-off between in stress distribution improvement and the search space size.

Algorithm 1. Selection of alternative configurations

```
1: Inputs:
      LUT list{$LUT$}, Configuration matrix{$\overrightarrow{C}$}
2: Outputs:
      Sets of alternative configurations {$Set_i$}
3: partition=Partitioning();
4: $ReprList = \emptyset$
      /*********Intra selection*********/
5: for each $partition_i \in$ partition do
6:    $min\_std \leftarrow \infty$
7:    for each $C_i \in partition_i$ do
8:       if $\sigma(C_i) \leq Min\_std$ then
9:          $Min\_std \leftarrow \sigma(C_i)$; ▷ Eq. (3)
10:         $Tmp\_Repr \leftarrow C_i$;
11:      end if
12:   end for
13:   $ReprList.Add(Tmp\_Repr)$
14: end for
      /*********Inter selection*********/
15: $Std\_Max_{opt} \leftarrow \infty$
16: for each 4-member subset of ReprList do
17:    $set \leftarrow pick\_set(ReprList)$
18:    if $Std\_Max(set) \leq Std\_Max_{opt}$ then
19:       $Std\_Max_{opt} \leftarrow Std\_Max(set)$
20:       $Set_{opt} \leftarrow set$;
21:    end if
22: end for
```

Equation (5) shows the stress imposed by a 4-member set of representative configurations to an LUT transistor. A set includes RC_i, RC_j, RC_k, and RC_l ($1 \leq i, j, k, l \leq 30$). The effect of stress is additive. Hence, in Eq. (5), the total stress is the sum of stress under individual configurations. As each configuration is used in a quarter of time, coefficient $\frac{1}{4}$ is used.

$$\mathbf{S}_{set} = \frac{1}{4}[S_{RC_i} + S_{RC_j} + S_{RC_k} + S_{RC_l}]. \tag{5}$$

Equation (6) shows the stress standard deviation of a 4-member set of representative configurations ($\{RC_i, RC_j, RC_k, RC_l\}$). $S_{set}(i)$ represents the stress of i^{th} ($1 \leq i \leq m$, m represents number of LUT transistors). ($\sigma(set)$) and the worst case stress ($Max(set)$) are of different scales. To contribute equally, the standard deviation ($Std\_Max(set)$) is given w coefficient. Each LUT has an individual w constant value. Using this constant $Std\_Max(set)$ for the 4-member set is shown in Eq. (7).

$$\sigma(\mathbf{set}) = \sum_{i=1}^{m} |\, \mathbf{S}_{set}(i) - \mu' \,| \qquad with \quad \mu' = \frac{1}{m}\sum_{i=1}^{m} \mathbf{S}_{set}(i) \tag{6}$$

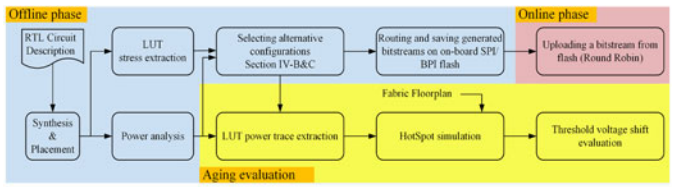
Offline phase
Online phase
RTL Circuit Description
LUT stress extraction
Selecting alternative configurations Section IV-B&C
Routing and saving generated bitstreams on on-board SPI/BPI flash
Uploading a bitstream from flash (Round Robin)
Fabric Floorplan
Synthesis & Placement
Power analysis
LUT power trace extraction
HotSpot simulation
Threshold voltage shift evaluation
Aging evaluation

Fig. 8. Experimental flow, generating alternative bitstreams, and evaluating the threshold voltage shift.

$$Std\_Max(set) = \sigma(\mathbf{set}) + w * Max(\mathbf{set}) \tag{7}$$

The *Inter partition selection* details are shown in Algorithm 1 (lines 15-22). Initially, the Inter partition selection criterion (Std_Max_{opt}) is set to infinity (line 15). Then, the exhaustive search is performed (lines 16-22). For this purpose, first, a 4-member set (*set*) is chosen up from the *ReprList* (line 17). If the value of *Std_Max* of the *set* is lower than the Std_Max_{opt} (line 18), the Std_Max_{opt} and set_{opt} are updated with *Std_Max* and *set* (lines 19 and 20). At the end, set_{opt} will contain the *alternative configurations*.

4.4 Four Alternative Configurations

As mentioned, in the *Inter partition selection*, four alternative configurations are selected for each LUT. The more alternative configurations, the more improvement will be in stress distribution. However, increasing the number of alternate configurations will result in increased search space and runtime. The search space at the *Inter partition selection* is equal to 30^n. Where 30 is the number of partitions and n is the number of *alternative configurations*. Additionally, the n bitstreams must be stored on the on-board flash memory to be loaded rotationally into the FPGA. As n increases, the required memory will increase. Therefore, there is a tradeoff between stress distribution improvement and runtime/memory requirements.

Fig. 7 shows the percentage of stress distribution improvement of LUTs (left Y axis) and the search space size (right Y axis) for the different number of alternative configurations. As can be seen, more than 4 alternative configurations do not result in a significant stress distribution. However, the search space will increase exponentially. Additionally, typical flash memories can store 4 bitstreams of the largest FPGAs. Therefore, value of 4 is an appropriate selection.

5 Experimental Evaluation

The proposed method is implemented on *Xilinx Kintex XC7K325T*. In this FPGA, Boolean functions are implemented on 6-input LUTs. In order to calculate the stress of LUT transistors, we employ the LUT transistor model presented in [2], patented by *Xilinx*. For the studies of this section we have considered four applications with different FPGA utilizations. Applications include JPEG decoder, BM3D (an image denoising method), OFDM receiver, and an A5/1 attack algorithm.

Fig. 8 shows the experimental evaluation flow of the proposed method. The implementation includes online and offline phases. In the offline phase, after receiving the RTL description of the target circuit, the design is synthesized and its elements are placed on the target FPGA. At this point, the Boolean functions of all LUTs can be extracted. Then, using *Xilinx Power Estimator (XPE)*, the activity rates of all the circuit nets, including LUT inputs, are extracted. Following this, alternative configurations for every LUT are considered and partitioning of configurations are done as discussed in the paper. At this stage, we have enough information to select alternative configurations using our selection procedure. With this, we generate four bitstreams and save them on the on-board flash. What comes next is during the operation of the application circuit on the FPGA. The on-board flash has four complete alternatives for the entire FPGA. In each configuration the LUTs utilized in the

TABLE 2
Resource Utilization on *Xilinx Kintex XC7K325T*

	JPEG	BM3D	OFDM	A5/1	Avail. res.
LUTs	8%	43%	57%	75%	203800
FFs	2%	21%	24%	46%	407600
DSPs	2%	9%	65%	79%	840

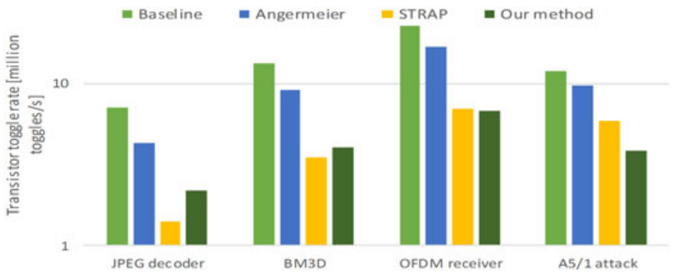

Fig. 9. Comparing dynamic stress in target applications.

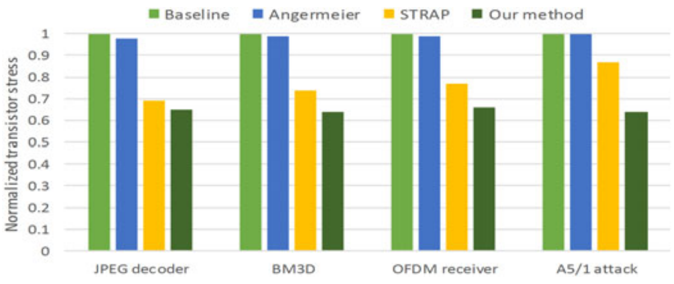

Fig. 10. Comparing static stress in target applications.

circuit have one of their alternative configurations. In a rotational fashion, one of the four configurations will be loaded into the FPGA.

For aging evaluation, in addition to stress calculation of transistors, it is required to estimate the temperature of transistors. To obtain temperature, we have to extract the power trace of LUTs. For this purpose, architectural simulation is performed to extract execution and idle profile of FPGA regions. The power trace and the fabric floorplan of the FPGA are then fed into Hotspot [19] to obtain the temperature of each LUT, which is necessary to evaluate the threshold voltage shift.

5.1 Results

Table 2 shows the FPGA resource utilization for the target applications. Choosing applications with different sizes on FPGAs help us to demonstrate the efficiency of our method. For example, JPEG decoder, as a small design, utilizes 8 percent of 203800 available LUTs. This value is 75 percent for A5/1 attack algorithm which is considered as a relatively large design.

Fig. 9 shows the worst case dynamic stress reduction, measured in million toggles/s, for the above applications. Similarly, Fig. 10 demonstrates the normalized worst case static stress. In addition to the baseline design and our method, two state-of-the-art methods presented by Angermeier *et al.* [9]) and STRAP [16], are shown in this Figure. The former method is a coarse-grained aging mitigation technique, whereas the latter is considered as a medium-grained method. As shown, all methods reduce the worst case stress. For example, for the JPEG decoder application, Angermeier [9] reduces the worst case toggling rate by 39 percent as compared with the baseline. STRAP [16], as the closest competitor, achieves the best reduction in the worst case toggling rate (80 percent). This compares with 69 percent for our method. However, increase in the design size affects the efficiency of competitors. For example, for A5/1 attack algorithm, dynamic stress reduction for the STRAP [16] is 51 percent, compared with 68 percent dynamic stress reduction in our method. Results for the static stress reduction follow a similar pattern.

TABLE 3
Worst Case Stress and MTTF Improvement, of Our Work, [16], [9] and Baseline

Method	Stress reduct.[%]		MTTF impr.[%]	
	Dyn.	Stat.	HCI	BTI
Ange.	29	1	49	1
STRAP	68.5	24	251	10
Our work	69.5	36	263	14.1

TABLE 4
Stress Distribution Among Transistors of a 3-Active Input LUT

Method	Standard deviation	
	Dynamic	Static
Baseline	0.237	0.458
Our method	0.133	0.294
Best selection	0.130	0.275

The reason of different reduction of stress for STRAP [16] and Angermeier [9] is related to the dependency of their methods on the size and density of the implemented designs. As the utilized number of LUTs increases, the chance of mapping high stress Boolean functions to unutilized LUTs decreases. Additionally, denser implementations impose more restrictions to the placer for finding low stress LUTs for the stressful Boolean functions. In contrast, our method is independent of the size of implemented designs. Our method avoids LUT routing congestion by only changing LUT local routings for already placed LUTs. For all designs, the worst stress reduction is about 69 and 36 percent for dynamic and static stress, respectively.

For the reliability improvement, we calculate MTTF. It is assumed that a device fails when ΔV_{th} of any internal transistor exceeds 50 percent of its initial value (V_{th0}). Table 3 shows the average reduction for the worst dynamic and static stress for Angermeier [9], STRAP [16] and our method. Based on these reductions, MTTF improvement is calculated and shown in the last two columns. Relative to the baseline, our method improves the MTTF by about 263 percent and 14.1 percent for HCI and BTI, respectively. In comparison with the closest competitor, i.e., STRAP [16], MTTF improvement of our method is 12 and 4.1 percent for HCI and BTI, respectively.

Table 4 compares stress distribution of our method with the baseline and best selection. To show the stress distribution, we use *standard deviation*. Since the runtime for LUTs with 4 and more inputs is too high, the results for a 3-active input LUT are shown. As can be seen, our method improves aging distribution (standard deviation) by 44 and 36 percent for dynamic and static stress, respectively. Additionally, our method result is very close to the best selection (1.02 and 1.06 percent, respectively). To extract the best selection result for the 3-active input LUT, an exhaustive search has been run for one hour on an Intel(R) Core(TM) i7 3.4 Ghz.

Table 5 shows the frequency degradation after applying the aging mitigation methods. As shown, with the increase in the size of a design, frequency overhead increases for Angermeier [9] and STRAP [16]. However, our method imposes negligible frequency overhead, which is again due to no change in placement of LUTs in our method. Additionally, our method only affects local switch routing with negligible effects on critical path delays. However,

TABLE 5
Frequency Overhead [%]

	Initial[MHz]	Ange. [9]	STRAP [16]	Our work
JPEG	50	8.08	6.24	0.00
BM3D	110	10.88	7.91	1.00
OFDM	200	14.76	11.17	0.88
A5/1	100	16.13	12.82	0.78

the competitors require to apply some constraints on the placement of Boolean functions which affects the critical path delays.

6 Conclusion

We have presented an FPGA aging mitigation method that only deals with individual FPGA LUTs and avoids elaborate routing of FPGA resources. For mapping function variables to an LUT, a transistor model from *Xilinx* is used and all aging evaluations are done on this basis. Using such evaluations, alternative configurations for every FPGA LUT are found, and are partitioned into groups of configurations with similar transistor stress. We use a two-step algorithm for selection of a configuration from every partition and then selection four final configurations as best alternatives for every LUT. The four alternatives for every LUT are put into four FPGA configuration bitstreams that are rotationally loaded into the FPGA. Our results reveal that using this method, 263 and 14.1 percent MTTF improvement can be achieved for HCI and BTI aging.

References

[1] S. Kiamehr, A. Amouri, and M. B. Tahoori, "Investigation of NBTI and PBTI induced aging in different LUT implementations," in *Proc. Int. Conf. Field-Programmable Technol.*, 2011, pp. 1–8.

[2] M. Chirania, "Look up table with relatively balanced delays," US Patent 7471104B1, Dec. 30, 2008.

[3] User Guide (UG 474). *7 Series FPGAs Configurable Logic Block*. Xilinx,1.8 Ed., San Jose, CA, Sep. 27, 2016.

[4] R. Nazari, N. Rohbani, H. Farbeh, Z. Shirmohammadi, and S. G. Miremadi, "A^2CM^2: Aging-aware cache memory management technique," in *Proc. IEEE CSI Symp. Real-Time Embedded Syst. Technol.*, 2015, pp. 1–8.

[5] Y. Wang *et al.*, "On the efficacy of input vector control to mitigate NBTI effects and leakage power," in *Proc. 10th Int. Symp. Quality Electron. Des.*, 2009, pp. 19–26.

[6] A. Roelke, X. Guo, and M. Stan, "OldSpot: A pre-RTL model for fine-grained aging and lifetime optimization," in *Proc. IEEE 36th Int. Conf. Comput. Des.*, pp. 148–151, 2018.

[7] H.-M. Dounavi, Y. Sfikas, and Y. Tsiatouhas, "Periodic monitoring of BTI induced aging in SRAM sense amplifiers," *IEEE Trans. Device Materials Rel.*, vol. 19, no. 1, pp. 64–72, Mar. 2019.

[8] Z. Ghaderi, N. Bagherzadeh, and A. Albaqsami, " STABLE: Stress-aware boolean matching to mitigate BTI-induced SNM reduction in SRAM-based FPGAs," *IEEE Trans. Comput.*, vol. 67, no. 1, pp. 102–114, Jan. 2018.

[9] J. Angermeier *et al.*, "Stress-aware module placement on reconfigurable devices," in *Proc. 21st Int. Conf. Field Programmable Logic Appl.*, 2011, pp. 277–281.

[10] B. Khaleghi, B. Omidi, H. Amrouch, J. Henkel, and H. Asadi, "Estimating and mitigating aging effects in routing network of FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 3, pp. 651–664, Mar. 2019.

[11] Z. Ghaderi and E. Bozorgzadeh, "Aging-aware high-level physical planning for reconfigurable systems," in *Proc. 21st Asia South Pacific Des. Automat. Conf.*, 2016, pp. 631–636.

[12] M. Shafique, M. U. K. Khan, and J. Henkel, " Content-aware low power configurable aging mitigation for SRAM memories," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3617–3630, Dec. 2016.

[13] S. Srinivasan *et al.*, "Toward increasing FPGA lifetime," *IEEE Trans. Dependable Secure Comput.*, vol. 5, no. 2, pp. 115–126, Apr.–Jun. 2008.

[14] Y. Yu *et al.*, "NBTI and power reduction using a workload-aware supply voltage assignment approach," *J. Electron. Testing*, vol. 34, no. 1, pp. 27–41, 2018.

[15] E. Stott, J. S. J. Wong, and P. Y. K. Cheung, "Degradation analysis and mitigation in FPGAs," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2010, pp. 428–433.

[16] H. Zhang *et al.*, "STRAP: Stress-aware placement for aging mitigation in runtime reconfigurable architectures," in *IEEE/ACM Int. Conf. Comput.-Aided Des.* , 2016, pp. 38–45.

[17] H. Zhang, L. Bauer, M.A. Kochte, E. Schneider, H.-J. Wunderlich, and J. Henkel, " Aging resilience and fault tolerance in runtime reconfigurable architectures," *IEEE Trans. Comput.*, vol. 66, no. 6, pp. 957–970, Jun. 2017.

[18] P. M. Rao *et al.*, "Altering LUT configuration for wear-out mitigation of FPGA-mapped designs," in *Proc. 23rd Int. Conf. Field Programmable Logic and Appl.*, 2013 pp. 1–8.

[19] W. Huang *et al.*, "Hotspot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.