Automated Design Approximation to Overcome Circuit Aging

Konstantinos Balaskas, Georgios Zervakis, Hussam Amrouch, Member, IEEE,
Jörg Henkel, Fellow, IEEE, Kostas Siozios, Member, IEEE

Abstract—Transistor aging phenomena manifest themselves as degradations in the main electrical characteristics of transistors. Over time, they result in a significant increase of cell propagation delay, leading to errors due to timing violations, since the operating frequency becomes unsustainable as the circuit ages. Conventional techniques employ timing guardbands to mitigate aging-induced delay increase, which leads to considerable performance losses from the beginning of the circuit’s lifetime. Leveraging the inherent error resilience of a vast number of application domains, approximate computing was recently introduced as an aging mitigation mechanism. In this work, we present the first automated framework for generating aging-aware approximate circuits. Our framework, by applying directed gate-level netlist approximation, induces a small functional error and recovers the delay degradation due to aging. As a result, our optimized circuits eliminate aging-induced timing errors. Experimental evaluation over a variety of arithmetic circuits and image processing benchmarks demonstrates that for an average error of merely $5 \times 10^{-3}$, our framework completely eliminates aging-induced timing guardbands. Compared to the respective baseline circuits without timing guardbands (i.e., iso-performance evaluation), the error of the circuits generated by our framework is 1208x smaller.

Index Terms—Approximate Computing, Circuit Aging, Logic Synthesis, Genetic Algorithm

I. INTRODUCTION

Exploiting the inherent error resilience of a vast number of application domains, approximate computing is established as a promising design paradigm to boost the efficiency of our computing systems [1]. Such applications, e.g., image and digital signal processing [1-24] and neural networks [25, 26], are able to produce results of acceptable quality, while tolerating errors in the underlying computations. Concisely, approximate computing intelligently trades off implementation/computational accuracy for gains in other metrics (e.g., energy consumption) [1].

Leveraging this potential for efficiency improvement, approximate circuits design has gained significant research interest. Arithmetic circuits, like adders [2-4] and multipliers [5-7], are mainly targeted as they constitute the fundamental building blocks of DSP and error resilient applications [10]. In addition, complex approximate hardware accelerators are also proposed, such as coordinate rotation digital computer (CORDIC) [22], H.265/HEVC encoder [23, 24] and Deep Neural Networks [25, 26]. However, such approaches are application specific, limiting their applicability. It is noteworthy that approximate computing exacerbates the design complexity by introducing a new dimension in systems design, i.e., the error [1]. To mitigate the increased complexity of approximate design and enable circuit-independent approximations, several automated methodologies are presented [9-12].

Despite the significant energy benefits derived by approximate design, recent research works [17-20] have gone a step forward and demonstrated that approximate computing can be also employed to efficiently suppress circuit aging effects. Transistor aging phenomena, e.g., Bias Temperature Instability (BTI) and Hot-Carrier Injection (HCI), manifest themselves as degradations in the main electrical characteristics of transistors. The most important degradation is related to increased threshold voltage ($V_{th}$) [27], which in turn leads to a decrease in the drain current of the transistor in the ON state ($I_{ON}$). The latter increases significantly the transistor’s propagation delay. Therefore, the standard cells become slower over time [28]. Hence, circuits start to exhibit timing errors because the operating frequency becomes unsustainable as the circuit ages. Aging-induced timing errors, are by nature very large and catastrophic for the application’s accuracy since they mainly affect the most significant bits [19, 20]. In addition, they are unpredictable and hard to control since they depend on the sequence of inputs and the state of the system [10].

In order to keep aging effects at bay for the entire projected lifetime (e.g., 10 years), a timing guardband ($t_{GB}$) is included on top of the critical path delay at design time. As a result, timing violations are prevented, but performance is degraded, as the circuit is forced to operate at a lower frequency even at the early phases of its lifetime, when aging-induced degradations are still negligible. Timing guardbands are calculated at a worst case scenario, depriving the opportunity to operate the circuit at a higher frequency during the earlier part of its life cycle [18]. Several approaches have been proposed [29, 30] regarding $t_{GB}$ narrowing and aimed to minimize the associated performance losses. On the other hand, [17-21] apply approximate computing to reduce the circuit’s critical path delay such that operation at the maximum frequency (of the fresh exact circuit) is guaranteed throughout the circuit’s...
lifetime. Hence, aging-induced timing errors are eliminated by inducing smaller, deterministic, and predictable functional errors, reestablishing, thus, the reliability of the circuit [17]. However, despite their promising results, [17]–[21] target very simple topologies such as ripple-carry adders and array multipliers, limiting their exploitation and providing only some preliminary, non-conclusive results regarding the exploitation of approximate computing to suppress circuit aging.

In this work, we propose for the first time an automated, circuit-agnostic design framework that generates aging-aware approximate combinational circuits. Our proposed and implemented framework employs a genetic algorithm to concurrently apply, in a coordinated manner, wire-by-wire [31] and wire-by-constant [9] replacements, managing to completely eliminate the aging induced timing guardband while minimizing the induced logic error. Unlike state-of-the-art aging-aware combinatorial circuit design automation techniques, our framework enables applying aging-aware approximation and by introducing a static error at design time, it efficiently eliminates timing errors due to aging. Still, state-of-the-art techniques for aging-aware combinational circuit design are orthogonal to our work and can be applied synergistically. Our framework operates on the circuit’s post-synthesis gate-level netlist and thus it is circuit-agnostic and can be seamlessly integrated into any standard design flow. In addition, our framework provides a scalable and time-efficient solution since it employs high-level delay and error models and thus it fully exploits the inherent parallelism of genetic algorithms. Unlike the state-of-the-art in aging-driven approximation that examines only very simple topologies and applies handcrafted solutions, we evaluate our framework over four speed-optimized adders and multipliers from the industry-level Synopsys DesignWare library as well as five image processing benchmarks implemented with DesignWare components. Our evaluation demonstrates that by inducing a negligible functional error (in terms of Normalized Mean Error Distance (NMED)), aging guardbands can be removed for the entire projected lifetime.

Our novel contributions within this paper are as follows:

1. We present the first automated design framework for applying aging-aware approximation on combinational circuits.
2. This is the first work that applies wire-by-wire and wire-by-constant approximation concurrently and in a coordinated manner and enables delay-optimized netlist approximations.
3. We demonstrate that for an average error (NMED) of merely $5 \times 10^{-3}$, our framework eliminates the performance loss due to aging guardbands, while aging-induced timing errors are suppressed for the entire lifetime of 10 years.
4. We demonstrate that for the projected lifetime, compared to the baseline circuits without timing guardbands, the error (NMED) of the approximate circuits generated by our framework is 1208x smaller.

II. RELATED WORK

Targeting to alleviate the increased complexity introduced by approximate hardware design, many approximate computing works focus on the automation of the approximate circuit generation. In [15], a methodology to create approximate designs by identifying the approximation “don’t cares” within a quality constraint circuit and using them for logic simplification is proposed. The authors in [31] utilize similarities between nodes in a circuit implementation and introduce a substitution technique to produce approximate and quality configurable designs in an automated manner. Quality configurable circuits are also generated in [13], through an approximation design framework that identifies specific logic islands to be isolated at runtime, depending on a given quality constraint. Netlist modification with approximate standard cells is presented in [16], where a heuristic model is employed for the generation of approximate arithmetic circuits in reasonable design time. In [8] and [32], cartesian genetic programming is used to generate approximate multipliers and adders in an automated manner. The authors in [33] extend high-level synthesis (HLS) tools and apply variable-to-constant approximation to generate approximate circuits that satisfy real-time constraints. The authors in [9] proposed an automated framework for generating approximate arithmetic circuits that achieve considerable energy savings. In [9], gate-level pruning of the post-synthesis netlist is applied by removing netlist’s gates based on their significance and activity. The works proposed in [12] and [10] extend [9] and enable runtime reconfigurability [12] or boost energy gains [10]. In [12] the approximated gates are not pruned but replaced by switches, while in [10] a voltage-driven netlist pruning scheme is proposed, aiming to minimize the voltage supply value. In [34], a library of approximate functional units combined with analytical models are used to build an automated framework for HLS of approximate accelerators. All the aforementioned works target mainly power and not delay optimization and do not consider aging-specific approximations.

The authors in [35] introduced a technique to optimize the circuit against aging by capturing potential paths that may become critical after aging, and then applied iteratively tighter timing constraints on them to force the synthesis tool to optimize these paths. Nevertheless, the structure of paths may change and thus different gates will be used that might be more susceptible to aging. [36] proposed to redesign the library cells through adjusting the $W_{PMOS}/W_{NMOS}$ ratio to keep the rise and fall delays balanced. However, one cannot expect a unique sizing for each gate/cell. Moreover, to mitigate aging effects, there are many techniques (e.g., [37]) that aim to balance the duty cycle. In general, such techniques are orthogonal to our work. When complex designs like processors are targeted, aging-aware circuit design techniques often analyze the impact of aging on the critical path only, e.g., [38]. However, aging may switch a path from critical to uncritical and vice versa, as it has been demonstrated in [19]. Other works, e.g., [39], proposed to consider the top $x\%$ of the critical path. This might not be feasible in realistic designs since the number of paths within the top 5% may reach $>10^7$ [35]. In practice, determining an $x$ such that it is guaranteed that the path that may become critical after aging is included is not trivial.

The authors in [19] proposed the integration of degradation-aware cell libraries in the design flow to effectively mitigate aging effects and enable fine-grained timing guardbands.
In [18], the authors introduce the concept of aging-aware approximation. [18] applies precision scaling on the inputs of a ripple carry adder, demonstrating that approximation can mitigate the aging-induced timing errors. [18] demonstrated that an approximate-aware circuit to overcome aging provides around 11% - 13% less area, energy and delay overheads compared to state-of-the-art aging-aware combinatorial circuit design technique. Despite encouraging results even for 10 years of aging, their methodology lacks the guarantee of eliminating timing guardbands. Extending [18], a dynamic approach to substitute timing errors caused by aging with computation approximation errors was presented in [17]. A detailed monitoring scheme of the critical path allowed for dynamic adaptation of the inserted approximations whenever timing violations would be detected. The methodology however is circuit-specific and also introduces small guardbands for mismatch correction between the monitoring system and the actual circuit. Both [18] and [17] are applied on very simple topologies, i.e., the slow ripple carry adder and array multiplier, limiting their exploitation. [40] employs reconfigurable approximate circuits to compensate for aging-induced timing violations. By monitoring the performance degradation of a subject circuit throughout its lifetime with an aging sensor (i.e. a Canary FF in their implementation), the authors propose switching from accurate to approximate operation mode when timing violations are detected. The efficiency of [40] heavily relies on the reconfigurability of the employed approximate circuits and the respective trade-offs they offer in terms of accuracy and performance. Similar to [17], [18], the approach of [40] is evaluated only on a simple adder circuit. In [21], a dynamic approach is proposed that trades off pessimistic thermal timing guardbands for quality and delay configurable approximate circuits. The runtime reconfigurable designs extend over two approximation levels and introduce notable speedups with minimal quality loss. However, significant overheads in terms of area are introduced. Finally, [20] applies dynamic approximation through input compression to eliminate timing guardbands. Nevertheless, the approach presented in [20] is specific for DNN accelerators.

Our work distinguishes from existing state of the art as follows: This is the first automated framework that applies aging-aware approximation. Our framework is circuit-agnostic and can seamlessly extend any typical design flow by operating on the synthesized gate-level netlist. Our framework completely eliminates aging-induced timing guardbands and doesn’t add any area or power overheads.

III. AGING-AWARE CELL LIBRARY CHARACTERIZATION

In order to estimate how transistor aging ultimately impacts the delay of a circuit’s path, we employ aging-aware standard cell libraries. Those libraries are characterized so that every transistor exhibits the worst-case aging-induced degradation (i.e., threshold voltage increase). This provides us with maximum delay increase that circuits’ paths might exhibit under aging for the entire projected lifetime. As mentioned earlier, transistor aging phenomena such as BTI and HCI, induce different types of defects during the lifetime. Those defects accumulate at the interfacial layer Si-SiO₂ and within the transistor dielectric. Over time, they weaken the electric field and hence result in an increase in the transistor threshold voltage ($\Delta V_{th}$). To estimate the latter, we employ a physics-based aging model validated against measurements [41], [42]. To capture worst-case aging, we consider a maximum shift in the transistor threshold voltage of 50mV, which is considered a critical degradation in many industrial applications [43]. In this work, we consider the 45nm technology node and we employ the open-source 45nm standard cell library [44] along with the Predictive Technology Model (PTM) [45], [46] to model the underlying pMOS and nMOS transistors. The industry standard compact model BSIM6 [47] for planar transistors was used in Synopsys HSPICE simulations during the library characterization.

After estimating the worst-case $\Delta V_{th}$, we modify the transistor model and then we perform library characterization using Synopsys SiliconSmart tool flows [48]. To this end, we employ the SPICE netlists of standard cells including parasitics and the degraded transistor models. For every standard cell, we consider $7 \times 7$ input signal slews and output load capacitances, similar to commercial standard cell libraries.

The generated standard cell libraries contain the manner in which all cells are impacted by aging-induced degradation. The open source 45nm PDK [44] was used for both the baseline as well as the aging-aware libraries, thus enabling direct comparisons and evaluations. Our libraries are fully compatible with existing standard tool flows for logic synthesis and timing analysis. Therefore, they can be directly deployed without any modification and utilized to extract how the delay of circuits’ paths will be affected by transistor aging.

IV. PROPOSED FRAMEWORK

In this section, our proposed aging-aware approximation framework (illustrated in Fig. 1), is described. Our framework takes as input the post-synthesis gate-level netlist of the examined circuit. Without any loss of generality, if the netlist is not available, an EDA tool is used to synthesize the Behavioural/RTL description of the circuit and generate the gate-level netlist. The reason for operating over the post-synthesis netlist is to make our framework automated and circuit-agnostic since post-synthesis netlists feature a regular structure and our framework need only to comprise a grammar to detect the gates of the considered technology library. This netlist will

---

**Fig. 1:** Abstract overview of our proposed framework that applies aging-aware approximation.
be referred hereafter as baseline netlist. Using our aging-aware library, we perform a Static Timing Analysis (STA) to obtain the timing information of the respective aged circuit. Next, based on the obtained aging-aware timing analysis, we run our optimization phase to generate the approximate netlist. The goal of our optimization is to identify the respective approximations so that i) the critical path delay (CPD) of the aged approximate netlist is less or equal to the CPD of the baseline fresh (i.e., w/o aging) netlist and ii) the error of the approximate netlist is minimized:

\[
given \text{fresh BL netlist} \text{ find aged AX netlist} \\
s.t. \text{CPD(aged AX netlist)} \leq \text{CPD(fresh BL netlist)} \quad (1) \\
\text{and } \min \text{(Error(aged AX netlist))}
\]

In other words, the approximate netlist can operate from day zero (fresh) until the end of the projected lifetime (50mV \(\Delta V_{th}\) degradation\[^1\][43]) at the CPD of the fresh baseline netlist with error guaranties, i.e., no aging-induced timing errors will occur. Hence, the applied approximation completely eliminates the aging-induced timing guardbands. In addition, the error of the approximate netlist is constant (and predictable since our framework introduces a functional/logic error) for the entire projected lifetime, i.e., the output of the approximate netlist is not affected by aging-induced degradations. Such a degradation (i.e., \(\Delta V_{th} = 50\text{mV}\)) can be reached after 10 years, or much earlier at around 3 years and less, depending on the operating condition of the circuit (e.g., workload induced stress, operating temperature, voltage, etc.) [49]. Still, our technique is orthogonal to the specific degradation level since the only modification required is to characterize our standard cell library at the desired degradation level. To generate the approximate netlist our framework uses a library of netlist approximation techniques and employs a genetic algorithm. Genetic algorithms are proven to deliver close to optimal solutions in a fast manner, due to their inherent parallelism. Hence, note that our framework applies a static approximation, at design time, and by inducing a small error, it generates approximate circuits that are by definition designed to tolerate such a \(\Delta V_{th}\) degradation, as (1) reveals. Finally, the obtained approximate netlist is synthesized to exploit all the optimizations performed by the synthesis algorithms.

A. Netlist Approximation Library

In our analysis, the approximation library employed by our framework comprises two state-of-the-art techniques: wire-by-wire [31] and wire-by-constant [9] replacements. Both techniques have been extensively used in design methodologies involving high level synthesis. However, to the best of our knowledge our work is the first to employ them concurrently and in a coordinated manner, to enable delay-optimized netlist approximations. Nevertheless, our framework is not bound to only these techniques and can be extended to support other gate-level netlist-specific approximation techniques. Netlist approximations are facilitated by the fact that every combinational circuit can be represented by a Directed Acyclic Graph (DAG), in which the nodes are the netlist’s gates and the edges are the wires that connect the gates. For example, Fig. 2a presents the DAG of a simple netlist, derived from a boolean expression. The timing information of the aged circuit is carried over to the DAG nodes. Hence, the critical path delay can be estimated as the maximum sum of gate delays among all paths in the DAG (annotated by red lines in Fig. 2a). Thus, netlist approximation techniques are easily applied, by just manipulating the DAG of the baseline netlist.

Wire-by-constant substitutes a wire in the netlist by a constant logic value. In wire-by-constant replacement, the approximated wire (DAG edge) is disconnected from the output port of its parent gate (DAG node) and instead, it is driven by a constant ‘1’ or ‘0’. By replacing a wire with a constant, we break the signal propagation through that wire and thus, all the timing paths comprising that wire, become potentially faster. Fig. 2b depicts an example of wire-by-wire approximation. In this example, the replacement of wire \(w_{10}\) by the constant ‘1’ is demonstrated, resulting in

in our work, we protect the circuit against a \(V_{th}\) increase of a maximum 50mV, which is considered to be a critical level of degradation \[^1\][43].

\[^1\][43]
13% error rate. All paths through the edge $U6 \rightarrow U8$ are interrupted, allowing for an acceleration of the updated critical path (by 10.6%) through the rest of the nodes.

Wire-by-wire replaces a wire (namely approximated wire) in the netlist with another wire in the netlist (namely approximation wire). In this replacement, the approximated wire (DAG edge) is disconnected from the output port of its parent gate (DAG node) and instead, it is connected to the respective output port of the parent gate (DAG node) of the approximation wire. By applying wire-by-wire replacement, if the approximation wire is closer to the DAG root than the approximated wire, then all the timing paths comprising the approximated wire potentially become faster. Fig. 2c depicts an example of wire-by-wire approximation, where $w9$ replaces $w12$. Paths which previously comprised the edge $U5 \rightarrow U6$ are now significantly accelerated by omitting the propagation delay of gates $U6$ and $U8$. Again, the DAG shows a CPD delay reduction of 10.6%.

B. Approximation Candidates

To apply wire-by-wire and wire-by-constant replacement, for each wire in the netlist, we need to identify a set of possible approximation candidates. The extracted candidates will be next used in our optimization phase to generate the approximate netlist.

To obtain efficient approximation candidates, we use a circuit simulator and run a functional simulation of the given baseline netlist. During the performed simulation, we capture the value of each wire in the netlist at every cycle. Then, for each wire $i$, we calculate the percentage of cycles that it was ‘0’ (namely $T^i_0$) and the respective percentage that it was ‘1’ (namely $T^i_1$) [9]. In addition, for each wire $i$ and $j$, so that $j$ is closer to the DAG root than $i$, we calculate the similarity $S^i_j$ between the wires $i$ and $j$. By similarity, we define the percentage of cycles in which $i$ and $j$ feature the same value. For each wire $i$, we select as approximation candidate ($AX_i$) the value 0, 1, or $j$, $\forall j$, that introduces the smallest error, i.e., features the highest score $T^i_0, T^i_1$, or $S^i_j, \forall j$:

$$AX_i = (\alpha, \gamma) \in \{(k,T^i_k), \forall k\} \cup \{(j,S^i_j), \forall j\},$$

$$\gamma = \max \{T^i_0, T^i_1\} \cup \{S^i_j, \forall j\}$$  \hspace{1cm} (2)

The largest the $\gamma$ value is in (2), the smallest the introduced error rate at wire $i$ becomes. For example, $T^i_0 = 0.95$ means that for 95% of the time the wire $i$ was 0. Hence, by replacing $i$ with 0, we obtain accurate results for the wire $i$ at 95% of the time. Similarly, if $S^i_j = 0.95$, then the wires $i$ and $j$ have the same value for 95% of the time and replacing $i$ with $j$ results in 5% error rate at $i$. To identify the approximation candidates we select the replacement with the highest similarity. In cases where both wire-by-constant and wire-by-wire replacement techniques produce equally adequate approximation candidates, i.e., significant similarity score, our algorithm prefers to apply wire-by-constant replacement. By substituting a wire with a constant value, the signal propagation passing through that wire is completely cut off, leading thus to higher delay reduction than any wire-by-wire replacement. In cases where more than one wire-by-wire replacement techniques produce significant similarity score, the wire that is closer to the DAG root (i.e., higher delay reduction is achieved) is selected. In case of a tie, one of these wires is randomly selected.

Considering only one approximation candidate for each wire (as in (2)), efficiently limits the size of the design space defined by netlist approximation using wire-by-wire and wire-by-constant techniques. Assuming $n$ wires in the baseline netlist, the size of the approximation space is $2^n$ (i.e., each wire can be either accurate or replaced by its approximation candidate). Alternatively, for each wire $i$, we can select a set of approximation candidates:

$$AXS_i = \{k | T^i_k > C, k \in [0,1]\} \cup \{j | S^i_j > C, \forall j\},$$  \hspace{1cm} (3)

with $C$ a user defined constant (e.g., $C = 0.80$). This approach would give more flexibility to our optimization algorithm and enable more fine-grained approximation. However, assuming $|AXS_i| > 1$, using (3) would explode the design space.

C. DAG

Our framework uses a DAG to represent the input netlist (i.e., input circuit) and perform the approximations. We use Python to parse the Verilog description of the post-synthesis gate-level netlist and generate the circuit’s DAG. Each gate in the netlist is represented by a DAG node and each wire by a DAG edge. Using the timing information obtained by the aging-aware STA, we annotate the DAG with the respective delays. Although, aging degradation is workload dependent, when we run the STA, in order to ensure guardband elimination even under worst-case scenarios, we consider the highest degradation due to aging, i.e., $50\mu V VTH$ degradation, for all the gates in the netlist (details in Section III). For each node in the DAG, we store its name (e.g., U1), its type (e.g., NAND), and its delay. Those information are directly obtained from the Verilog description of the netlist and the STA.

As aforementioned, the employed approximation techniques are applied by directly manipulating the DAG. After each approximation, the delay of the approximate netlist is obtained by traversing the respective approximate DAG [12]. In addition, after each approximation, we translate the DAG to a C function [13]. To perform this conversion, we use the stored information for each node and we generate the C representation by hierarchically parsing the approximate DAG. Each node is converted to the corresponding C function w.r.t the node type. Since C code is serial, whereas Verilog is concurrent, the order in which the DAG nodes are written in the C function, is determined by the topological ordering of the nodes in the DAG. The DAG edges (wires) are used as the inputs and outputs of the C function. Leveraging this DAG-to-C conversion, we simulate the functionality of the approximate DAG in a very fast manner and we obtain a precise estimation of its error using large input datasets. Note that, the goal of our framework is to completely eliminate the aging-induced
Algorithm 1: Heuristic Optimization Pseudocode

Inputs: 1. List of approximation candidates: C, 2. Delay target, 3. DAG, 4. Accurate netlist c-file: Netlist.c, 5. Simulation inputs: Inputs.txt, 6. GA parameters

Output: Optimal approximate netlist

1. # Create random chromosomes, weighted by the critical path
2. Initialize population;
3. for Generations do
4. # Produce new generation
5. Keep fittest solution;
6. while NewGeneration incomplete do
7. Select parents;
8. Produce offsprings;
9. Add to new generation;
10. end
11. # Select the PopulationSize fittest chromosomes
12. Calculate offsprings fitness;
13. Merge parent and offspring populations;
14. Keep PopulationSize fittest chromosomes;
15. Update MutationProbability if necessary;
16. end
17. return Population[0];

Function CalcFitness(Chromosome, DelayTarget C, DAG, Netlist.c Inputs.txt):

18. Decode: Chr ←→ C ;
19. Apply approximation on DAG;
20. Isolate nodes with constant output;
21. Update critical path;
22. if Delay > DelayTarget then
23. return 0;
24. end
25. Create approximate netlist;
26. Calculate error;
27. return 1/Error;

Enables Algorithm [1] to explore solutions of diminished delay from the first generation. Throughout the standard iterative process, the Genetic Algorithm selectively produces offsprings that satisfy the delay constraint until the next generation is populated (lines 13-25). Leveraging the inherent parallel nature of the genetic algorithm, this process can be run on multiple threads concurrently, thus efficiently utilizing the underlying hardware to speedup the optimization phase. An elitist approach is implemented (line 20), as the parent chromosomes of highest fitness are propagated intact to the next generation. Depending on the new population’s diversity, the mutation probability can be adaptively tuned to increase exploration (line 12).

The core of our optimization algorithm is the fitness calculation of approximate solutions (15-26). For any solution, the respective bitstring is decoded into wire-by-wire and wire-by-constant approximations (line 16), which are then used to modify (approximate) the baseline DAG (line 17). Next, the error and delay of the approximate DAG are calculated (see Section IV-C). The obtained delay is compared with the delay target (lines 20-22). If the delay target is not met, the solution is discarded and considered invalid. Only valid solutions are reevaluated with a non-zero fitness score. The score aims to promote approximations with minimum error, that satisfy the delay target. To that end, the score is equal to the inverse of the error (line 25), since we treat the evolution as a maximization problem. As an error metric for quantifying the induced approximation error, we use the Normalized Mean Error Distance (NMED) [7]:

\[ NMED = \frac{1}{\max N} \sum_{i=0}^{N} \frac{|Y_{true} - Y_{appr}|}{Y_{true}} \]  (4)

where \( Y_{true} \) is the accurate output, \( Y_{appr} \) the approximate output, \( N \) the number of inputs and \( \max \) is the maximum possible value.

The output of the Genetic Algorithm is the solution with the highest fitness score, and therefore lowest possible error. Note that only solutions that meet the delay constraint are considered. The obtained solution is used to approximate the baseline DAG. Finally, the latter is synthesized using the EDA tool to exploit all the supported optimizations and remove any floating gates. Note that, our framework targets combinational circuits. Sequential circuits are out of the scope of this work. Nevertheless, sequential circuits can be seamlessly supported by using the EDA tools to perform the error and delay evaluations in Algorithm [1].

V. RESULTS AND EVALUATION

In this section, we evaluate the efficiency of our framework in eliminating the timing errors due to aging and thus to completely remove the aging-induced timing guardbands, boosting the performance of circuits. Our framework is evaluated over varying arithmetic circuits as well as more complex dataflows. Arithmetic circuits such as adders and multipliers are the basic building block of DSP and machine learning circuits [10]. In our analysis, we consider an 8-bit and a 16-bit speed optimized adder from the industry-strength Synopsys DesignWare library [48]. Similarly, an 8-
bit and a 16-bit speed optimized multiplier from DesignWare are examined. Moreover, our framework is evaluated over five image processing benchmarks: Sobel Edge Detector, Gaussian Blur filter, and a generic 3 x 3 Convolution kernel. Targeting maximum performance, all the examined circuits (arithmetic units and image processing circuits) are synthesized with zero-slack. Synopsys Design Compiler is used for circuit synthesis and the compile_ultra command is specified to obtain fully optimized netlists. Synopsys PrimeTime is used to perform static time analysis (STA). Mentor QuestaSim is used to run post-synthesis timing simulations and obtain the circuit’s outputs. The output of the timing simulation is used to calculate the circuit’s error in the case of approximated and/or aged circuits. For each circuit examined, two randomly generated input datasets are utilized. The first consists of 10^5 inputs and is used in the optimization phase of our framework. The second one comprises 10^6 inputs and it is used for the performed accuracy evaluation. Input stimuli of both datasets are generated by randomly sampling a uniform distribution in the segment defined by the input bit-width of each circuit.

### A. Timing Guardbands Elimination

First, we analyze the efficacy of our framework in eliminating the aging-induced timing guardbands. Fig. 3 presents the delay evaluation of the baseline circuits and the approximate ones generated by our framework. For each case, the delay of both the fresh as well as the aged (ΔVth=50mV) circuit is examined. All the delay values, reported in Fig. 3, are normalized w.r.t. the delay of the respective baseline fresh circuit. As shown in Fig. 3 for all the examined circuits, the delay of the aged approximate circuit (generated by our framework) is less or equal to the delay of the corresponding baseline fresh circuit. Hence, from day zero to the end of the projected lifetime (about 10 years or ΔVth=50mV), our approximate circuits can be operated at the maximum frequency of the respective baseline fresh circuit with error guarantees, i.e., they will not exhibit timing errors due to aging. As a result, by applying directed, aging-aware approximation, our framework enables operation at the maximum frequency for both the arithmetic circuits (Fig. 3a) as well as the more complex image processing benchmarks (Fig. 3b), eliminating thus the aging-induced timing guardbands of the baseline circuit.

As demonstrated in Fig. 3, the delay of the aged baseline circuits is on average 12.15% higher than the delay of the respective fresh baseline circuits. Therefore, either a timing guardband will be used that will degrade the performance of the examined circuits by 12.15% on average, or aging-induced timing errors will occur. In Fig. 4 we evaluate the error value (NMED) of the baseline circuits when operated at maximum frequency, i.e., without any timing guardbands. In addition, Fig. 4 presents the error of our approximate circuits. Note that, since the approximate circuits generated by our framework can operate at the maximum frequency of the baseline fresh circuit without any timing errors, their error value is constant for their entire projected lifetime (i.e., fresh and aged feature the same error). As shown in Fig. 4, the approximate circuits generated by our framework achieve significantly higher accuracy than the aged baseline circuits. For the case of arithmetic circuits (Fig. 4a), our approximate ones achieve on average 2340x lower NMED than the respective aged baseline circuits. As shown in Fig. 4b, the baseline circuits are affected by aging in a quite similar manner, i.e., they feature similar NMED values. This is explained by the fact that timing errors (due to aging) appear in the slowest paths that mainly affect the most significant bits (MSBs) in arithmetic circuits. This is not the case for the approximate circuits, generated by our framework. For the 16-bit circuits, NMED is significantly...
smaller compared to the 8-bit ones. This is explained by the fact that the examined 8-bit circuits are quite small, featuring only a small number of wires. Hence, approximating a wire in the 8-bit arithmetic circuits has higher impact on the final output compared to approximating a wire in the 16-bit arithmetic circuits. Nevertheless, our framework achieves significantly lower error than the baseline for both the 8-bit and 16-bit circuits. This error gain is also retained in the case of the examined image processing benchmarks (Fig. 4b). In Fig. 4b, our approximate circuits achieve on average 74% lower NMED than the respective aged baseline ones. The NMED reduction is 280x, 8x, 74x, 3x, and 7x for the Sobel, FIR, SAD, Blur and Conv benchmarks, respectively. It is noteworthy that, in Fig. 4b for all the approximate circuits generated by our framework, the maximum NMED is 3.25 × 10^{-2} and the average NMED is 5 × 10^{-3}. Hence, for a negligible error of 5 × 10^{-3} our framework is able to completely eliminate the aging-induced timing guardbands. In addition, note that our framework induces a functional error, known at design time, that is constant from the beginning to the end of the projected lifetime. On the other hand, errors due to aging are timing errors and depend on the input sequence and the state of the system, being thus practically infeasible to predict.

Finally, we analyze our approximate circuits with respect to the applied approximation technique. As mentioned in Section IV-B, our framework considers both wire-by-wire and wire-by-constant approximation to generate aging-aware approximate circuits. For each wire in the baseline netlist, an approximation candidate is selected that replaces the wire by either ‘0’ or ‘1’ (wire-by-constant) or by another wire in the netlist (wire-by-wire). For each circuit examined, for the respective set of approximation candidates, Fig. 5 presents the percentage of candidates that employ wire-by-0, wire-by-1, and wire-by-wire replacement. Similarly, for the final approximate netlist, Fig. 5a presents the percentage of selected approximation techniques w.r.t. the approximation candidates as well as the final approximate solution for (a) the arithmetic circuits and (b) the image processing benchmarks. Patterns distinguish between approximation candidates and final approximate solution, while bar colors indicate the approximation technique.

In this section we compare our framework against state-of-the-art works. In [17, 18], aging-aware precision scaling (APS) is used to generate an approximate multiplier and adder respectively. To compare against APS, we employ the methodology presented in [17, 18] and perform an exhaustive design space exploration to identify the optimal APS solution that satisfies the delay constraint. Hence, using our exhaustive exploration, we apply APS to all the examined circuits. Moreover, we compare our work against GLP [9]. GLP is applied to arithmetic circuits and employs a systematic gate-level pruning. GLP uses wire-by-constant replacement and in each iteration approximates the wires that minimize the significance-activity product (SAP). Finally, we compare our work against DSEwam [34]. DSEwam targets more complex dataflows and thus it is used with the examined image processing benchmarks. DSEwam uses analytical models to replace the exact functional units in a dataflow with faster approximate ones from an approximation library. For APS, GLP, and DSEwam we use the same delay constraint as in our work, i.e., the delay of the aged (ΔVth=50mV) approximate circuit is less than or equal to the delay of the fresh baseline.

Tables I and II present the comparison of our work against state-of-the-art ones for the examined arithmetic and image processing benchmarks.

**Table I**: Comparative error (NMED) evaluation of our framework against state-of-the-art works for arithmetic circuits

|                | Add8 | Add16 | Mult8 | Mult16 |
|----------------|------|-------|-------|--------|
| APS [17, 18]   | 2.9 × 10^{-9} | 3.0 × 10^{-9} | 3.9 × 10^{-3} | 6.9 × 10^{-5} |
| GLP [9]        | 3.8 × 10^{-4} | 4.5 × 10^{-4} | 1.0 × 10^{-1} | 2.4 × 10^{-2} |
| Ours           | 2.2 × 10^{-2} | 1.1 × 10^{-5} | 6.0 × 10^{-3} | 4.1 × 10^{-6} |

**Table II**: Comparative error (NMED) evaluation of our framework against state-of-the-art works for image processing benchmarks

|                | Sobel | FIR   | SAD   | Blur  | Conv  |
|----------------|-------|-------|-------|-------|-------|
| APS [17, 18]   | 2.5 × 10^{-2} | 2.3 × 10^{-2} | 1.8 × 10^{-3} | 1.2 × 10^{-1} | 2.3 × 10^{-2} |
| DSEwam [34]    | 8.8 × 10^{-4} | 1.4 × 10^{-4} | -     | 1.6 × 10^{-4} | -     |
| Ours           | 1.7 × 10^{-4} | 1.4 × 10^{-5} | 5.0 × 10^{-4} | 3.3 × 10^{-2} | 1.3 × 10^{-2} |

Finally, we analyze our approximate circuits with respect to the applied approximation technique. As mentioned in Section IV-B, our framework considers both wire-by-wire and wire-by-constant approximation to generate aging-aware approximate circuits. For each wire in the baseline netlist, an approximation candidate is selected that replaces the wire by either ‘0’ or ‘1’ (wire-by-constant) or by another wire in the netlist (wire-by-wire). For each circuit examined, for the respective set of approximation candidates, Fig. 5 presents the percentage of candidates that employ wire-by-0, wire-by-1, and wire-by-wire replacement. Similarly, for the final approximate netlist,
processing circuits, respectively. As shown in Tables I and II, our framework achieves always the highest accuracy. In the case of arithmetic circuits (Table II), APS employs an aging-aware approximation, achieving 472x lower NMED than the aging-unaware GLP. By applying directed fine-grained aging-aware approximation, our framework significantly outperforms both APS and GLP. The approximate circuits generated by our framework feature on average 12.6x and 11000x times lower NMED than the APS and GLP approximate circuits, respectively. Similar results are obtained for the image processing circuits in Table I. The aging-unaware DSEwam features significantly higher NMED than the approximate circuits obtained with APS. It is noteworthy that DSEwam wasn’t able to generate a solution for the SAD and Conv circuits. As shown in Table II, for all the examined circuits, our framework achieves on average 3-4x lower NMED than APS. The NMED reduction ranges from 1.3x up to 145x. Compared to DSEwam, for the Sobel, FIR and Blur circuits, our framework delivers 1756x lower NMED on average.

C. Execution Time Evaluation

Our framework implements a Genetic algorithm to address the optimization problem of aging-driven approximation as defined in (1). The time complexity of a Genetic algorithm is mainly defined by the number of epochs, the population size, the time complexity of one solution (i.e., the time required to evaluate one offspring) as well as the number of available threads. Note that the optimization phase of our framework (see Fig. I and Algorithm I) is implemented purely in Python and C. Hence, our framework i) can fully leverage the inherent parallel nature of the genetic algorithm allowing the optimization process to run concurrently on multiple threads exploiting all the available computing resources and ii) is not constrained by license limitations in commercial EDA tools that would restrict the scalability of our approach. In Table III, we report the execution time required to evaluate one offspring for the examined arithmetic circuits (up) and image processing benchmarks (down). An offspring evaluation comprises both computing its error value as well as estimating its delay. All experiments are conducted on a desktop computer featuring an AMD Ryzen 7 2700X processor at 3.2GHz and 32GB of RAM. The values reported in Table III refer to the average value over 50 trials. Overall, the execution time of a single offspring evaluation is very low for all the circuits, being at maximum 1.6s for the largest examined circuit (i.e., SAD). Still, as observed, the execution time is proportional to the circuit complexity (they are linearly correlated with an $R^2$ value of 0.972). The low execution times reported in Table III along with the scalability of our approach demonstrate the time-efficiency of our framework. The number of epochs and the population size can be set as required by the user to control the overall execution time of our framework. Nevertheless, note that running the algorithm for a few epochs, even though it will accelerate the optimization phase, it may potentially lead to solutions of lower quality. In our paper, we have empirically set the population size and the number of epochs with respect to the circuit size.

D. Power Discussion

As described in Section IV, the aim of our framework is to inject the bare minimum approximation so that aging-induced timing errors are eliminated and the error imposed by the applied logic approximation is minimized. Hence, although power reduction may be achieved due to the applied approximation, it is out of the scope of the respective optimization decision (1) of our work. For the sake of completeness, this section presents a power analysis over the examined circuits. Table IV presents the power consumption of the baseline and our approximate circuits under both fresh and aged conditions. To measure the power consumption, for each circuit, the switching activity obtained from the gate-level timing simulation is fed to Synopsys PrimeTime to run the power analysis. Note that, in Table IV, the power of the aged circuits is lower than the power of the respective fresh ones. This is explained by the fact that aging increases the threshold voltage of underlying transistors and thus both leakage and dynamic power become less as it has been demonstrated in [51]. As shown in Table IV, the circuits generated by our framework feature mainly lower power consumption than the corresponding baseline circuits. However, three out of the nine circuits generated by our framework (i.e., the 16-bit adder, the 16-bit multiplier, and the Sobel) exhibit higher power consumption than the baseline. For these circuits our framework achieved the highest error reduction (as shown in Fig. I) while still satisfying the delay constraint of (1). As a result of inducing minimal approximation but also significantly decreasing the CPD of the circuit, a higher power consumption is attained. Hence, to satisfy both the delay constraint of (1)
TABLE V: Dominated solutions generated by our framework for the 16-bit adder, 16-bit multiplier, and Sobel that feature both lower critical path delay as well as lower power than the corresponding baseline.

| Circuit | Norm. Delay | Error (NMED) | Power [W] |
|---------|-------------|--------------|-----------|
|         | Fresh | Aged |         | Fresh | Aged |
| Add16   | 0.888  | 0.992 | 6.44 × 10^{-5} | 1.3 × 10^{-4} | 1.32 × 10^{-3} |
| Multi16 | 0.871  | 0.987 | 4.28 × 10^{-5} | 1.04 × 10^{-2} | 1.00 × 10^{-2} |
| Sobel   | 0.886  | 0.996 | 1.38 × 10^{-2} | 5.58 × 10^{-5} | 5.39 × 10^{-3} |

as well as the power consumption decreases, we need to reformulate our optimization problem as follows:

\[
\text{given } fresh\ BL\ netlist \text{ find } aged\ AX\ netlist
\]
\[
s.t. CPD(aged\ AX\ netlist) \leq CPD(fresh\ BL\ netlist),
\]
\[
\text{Power}(fresh\ AX\ netlist) \leq \text{Power}(fresh\ BL\ netlist),
\]
\[
\text{and } \min(\text{Error}(aged\ AX\ netlist)).
\]

Nevertheless, this is a different optimization problem and doesn’t ensure minimum error due to logic approximation, i.e., our optimization goal in (1). To address (5), a multi-objective heuristic solver as in [8] may be employed, along with our delay and error modeling, and a high-level power estimation similar to [12].

In Table V we present some dominated solutions generated by our framework during the optimization phase for the 16-bit adder, 16-bit multiplier, and Sobel. The approximate circuits presented in Table V satisfy the delay constraint of our optimization problem (1) and also achieve lower power consumption than the respective baseline. These offsprings were generated and evaluated by our Genetic algorithm but they were eventually discarded later since they constitute dominated solutions as a better solution (i.e., with lower error) was found in the next iterations. Note that the error of the aged baseline is 1.08 × 10^{-2}, 3.47 × 10^{-2} and 4.69 × 10^{-2} for the 16-bit adder, 16-bit multiplier and Sobel, respectively. Therefore, the circuits of Table V achieve lower power as well as 168x, 810x and 3x lower error, respectively, than the corresponding baseline. Nevertheless, the circuits of Table V feature 5x, 10x and 81x higher error than the final non-dominated solutions generated by our framework (see Fig. 6 and Table IV). Hence, although Tables V and IV demonstrate that there exists at least one solution for (3), the circuits of Table V are not solutions of our optimization problem as defined by (1).

E. Process Variation Discussion

In this section we evaluate the impact that process variations might have on the approximation efficiency and accuracy of our framework. To achieve this, we perform a Monte Carlo analysis with 1000 samples to capture the impact of process variations on the output error of the examined circuits. It is noteworthy that, because in our work we target complex industrial circuits, generated and optimized using mature EDA tool flows (i.e., Synopsys Design Compiler and DesignWare library), Monte Carlo analysis using SPICE simulations is infeasible due to the vast time that would be required. To overcome this challenge, a Monte Carlo analysis is conducted at the circuit level employing the detailed delay profile of the circuit (i.e., SDF file) as extracted from Static Timing Analysis using mature tool flows (i.e., Synopsys PrimeTime). Note that for all examined circuits, both the baseline aged and the approximate aged are analyzed under variability effects for fair comparisons. Each circuit is subjected to 1000 different variations (i.e., 1000 different SDF files are generated per circuit), by modifying the timing characteristics of its composing gates. To generate a variation (modified) SDF file, we parse the original SDF file of the circuit as generated by PrimeTime using the aged library and for each gate in the netlist we replace its delay \(\delta\) with a new value \(\delta'\). The considered normal distribution has \((\sigma/\mu = 10\%)\), which has been demonstrated in [52] to be sufficient to capture the effects of different variability sources such as the gate work-function, channel length, geometrical variation, and oxide thickness. Finally, for each circuit and each modified SDF file, we run a timing simulation using Mentor Questasim and capture the output error. All circuits are simulated at the CPD of the respective fresh baseline at nominal process parameters. Overall, to perform the Monte Carlo analysis for all the examined circuits, 18000 gate-level timing simulations are conducted.

Fig. 6 summarizes the outcome of the Monte Carlo analysis. For each circuit, the output error (NMED) is presented as a box plot. Note that in the aged baseline circuits the output error is only due to the aging-induced timing errors, while in our aged approximate circuits the output error is subject to both timing errors and functional approximation. As shown in Fig. 6 the error of our aged circuit is always lower than the error of the baseline aged circuit. Specifically, the IQ2 error of our approximate aged circuits is 40x lower, on average, than that of the aged baseline. Overall, the error reduction in our approximate aged circuits ranges from 1.31x up to 6446x.

In other words, despite process variation effects, by applying functional approximation in a directed manner, our framework is still able to significantly decrease the aging-induced error. In addition, it is noteworthy that four out of the nine (44%)
circuits generated by our framework feature zero error variance and thus do not exhibit any timing errors. Hence, by setting tighter delay constraints in Fig. 4, we can completely eliminate the timing errors for all the circuits examined. However, in the latter case our framework would have to apply higher approximation and thus induce a larger static functional error (compared to Fig. 4). Though, studying and mitigating process variation effects is out of the scope of our paper.

VI. CONCLUSION

In this work, we propose and implement the first automated framework for aging-aware circuit approximation. Our framework employs a genetic algorithm to apply wire-by-wire and wire-by-constant approximation in a systematic manner and suppress aging circuit degradation. By inducing a small and known a-priori functional error, our framework manages to eliminate aging-induced timing errors and thus remove aging-induced timing guardbands, boosting the circuit’s performance. Finally, we demonstrate that, targeting to eliminate aging-induced timing guardbands, our framework significantly outperforms state-of-the-art approaches by delivering orders of magnitude lower error.

REFERENCES

[1] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in 2013 18th IEEE European Test Symposium (ETS), 2013, pp. 1–6.
[2] M. Pashaeifar, M. Kamal, A. Afzali-Kusha, and M. Pedram, “A theoretical framework for quality estimation and optimization of dsp applications using low-power approximate adders,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 1, pp. 327–340, 2018.
[3] L. B. Soares, M. M. A. da Rosa, C. M. Diniz, E. A. C. da Costa, and S. Bampi, “Design methodology to explore hybrid approximate adders for energy-efficient image and video processing accelerators,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 6, pp. 2137–2150, 2019.
[4] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, “A low latency generic approximate multiplier,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 5, pp. 1694–1702, 2017.
[5] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. Di Meo, “Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3021–3034, 2020.
[6] K. Bhardwaj, P. S. Mane, and J. Henkel, “Power- and area-efficient approximate Wallace tree multiplier for error-resilient systems,” in International Symposium on Quality Electronic Design, 2014, pp. 263–269.
[7] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pemestzi, “Design-efficient approximate multiplication circuits through partial product perforation,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 10, pp. 3105–3117, 2016.
[8] V. Mraze, R. Hrbacek, Z. Vasic, and L. Sekanina, “Evaprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods,” in Design, Automation Test in Europe Conference Exhibition (DATE), March 2017, pp. 258–261.
[9] J. Schlachter, T. Camus, K. V. Palem, and C. Enz, “Design and applications of approximate circuits by gate-level pruning,” IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 25, no. 5, pp. 1694–1702, 2017.
[10] G. Zervakis, K. Kollogergi, D. Anagnostoios, N. Zompaakis, and K. Siosios, “Vader: Voltage-driven netlist pruning for cross-layer approximate arithmetic circuits,” IEEE Trans. on Very Large Scale Integration Systems, vol. 27, no. 6, pp. 1460–1464, 2019.
[11] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, “Synthesizing pinnions monolithic inductors circuits through probabilistic design techniques,” ACM Trans. on Embedded Computing Systems, vol. 12, no. 26, pp. 1–26, 2013.
[12] G. Zervakis, H. Amrouch, and J. Henkel, “Design automation of approximate circuits with runtime reconfigurable accuracy,” IEEE Access, vol. 8, pp. 53522–53538, 2020.

[13] S. Jain, S. Venkataraman, and A. Raghunathan, “Approximation through logic isolation for the design of quality configurable circuits,” in Design, Automation & Test in Europe Conf. & Exhibition, 2016, pp. 612–617.
[14] K. Nepa, Y. Li, R. I. Bahar, and S. Reda, “Abacus: A technique for automated behavioral synthesis of approximate computing circuits,” in Design, Automation & Test in Europe Conference & Exhibition, 2014, pp. 1–6.
[15] S. Venkataraman, V. I. Kozhikkottu, A. Sabne, K. Roy, and A. Raghunathan, “Logic synthesis of approximate circuits,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 39, no. 10, pp. 2503–2515, 2020.
[16] S. De, J. Huisken, and H. Corporaal, “An automated approximation methodology for arithmetic circuits,” in International Symposium on Low Power Electronics and Design (ISLPED), 2019, pp. 202–209.
[17] H. Kim et al., “Aging compensation with dynamic computation approximation,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 4, pp. 1319–1332, 2020.
[18] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, “Towards aging-induced approximations,” in Design Automation Conference (DAC), 2017, pp. 1–6.
[19] S. Salamin, G. Zervakis, O. Spantidi, I. Anagnostopoulos, J. Henkel, and H. Amrouch, “Reliability-Aware Quantization for Anti-Aging NPUs,” in Design, Automation and Test in Europe Conference, 2021.
[20] B. Boroujerdian, H. Amrouch, J. Henkel, and A. Gerstlauer, “Trading off temperature guardbands via adaptive approximations,” in International Conference on Computer Design (ICCD), 2018, pp. 202–209.
[21] L. Shen, J. Han, W. Liu, and F. Lombardi, “Algorithm and design of a fully parallel approximate coordinate rotation digital computer (cordic),” IEEE Trans. Multi-Scale Comput. Syst., vol. 3, no. 3, pp. 139–151, 2017.
[22] G. Paim, L. M. G. Rocha, H. Amrouch, E. A. C. da Costa, S. Bampi, and J. Henkel, “A cross-layer gate-level-to-application co-simulation for design space exploration of approximate circuits in hevc video encoders,” IEEE Trans. Circuits Syst. Video Technol., pp. 1–1, 2019.
[23] G. Paim et al., “On the Resiliency of NCVFET Circuits Against Voltage Over-Scaling,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 4, pp. 1481–1492, 2021.
[24] Z.-G. Tsooulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, “Weight-oriented approximation for energy-efficient neural network inference accelerators,” IEEE Transactions on Circuits and Systems I: Regular Papers, 2020.
[25] V. Mraze, Z. Vasic, L. Sekanina, M. A. Hanif, and M. Shafique, “Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining,” in International Conference on Computer-Aided Design (ICCAD), 2018, pp. 1–6.
[26] S. Mahapatra et al., “A Comparative Study of Different Physics-Based NBTI Models,” IEEE Trans. Electron Devices, vol. 60, no. 3, 2013.
[27] C. Hu, Modern semiconductor devices for integrated circuits. Prentice Hall, 2010.
[28] S. Roy, D. Liu, J. Singh, J. Um, and D. Z. Pan, “OSFA: A New Paradigm of Aging Aware Gate-Sizing for Power/Performance Optimizations Under Multiple Operating Conditions,” IEEE Trans. Comput-Aided Design Integr. Circuits Syst., vol. 35, no. 10, pp. 1618–1629, 2016.
[29] J. Keane and C. H. Kim, “Transistor aging,” in IEEE Spectr, vol. 48, April 2011, p. 28–33.
[30] S. Venkataramani, K. Roy, and A. Raghunathan, “Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits,” in Design, Automation & Test in Europe Conf. & Exhibition, 2017, pp. 1367–1372.
[31] M. Češka, J. Matyáš, V. Mracek, L. Sekanina, Z. Vasic, and T. Vojnar, “Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished,” in Int. Conf. Computer-Aided Design, Nov 2017, pp. 416–423.
[32] M. T. Leipnitz and G. L. Nazar, “High-level synthesis of approximate designs under real-time constraints,” ACM Trans. Embed. Comput. Syst., vol. 18, no. 5s, pp. 59–1–59:21, Oct. 2019.
[33] J. Castro-Godinez, J. Mateus-Vargas, M. Shafique, and J. Henkel, “A HXLS: Design Space Exploration and High-Level Synthesis of Approximate Accelerators using Approximate Functional Units and Analytical Models,” in Int. Conf. Computer-Aided Design, 2020, pp. 1–9.
[34] M. Ebrahimi, F. Oboril, S. Kiani, and M. B. Tahoori, “Aging-aware Approximate Accelerators,” in Int. Conf. Computer-Aided Design, 2013, pp. 61–68.
[35] S. Kiani, F. Firouzi, M. Ebrahimi, and M. B. Tahoori, “Aging-aware standard cell library design,” in Design, Automation & Test in Europe Conf. & Exhibition, 2014, pp. 1–4.
[37] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti, “Combating aging with the colt duty cycle equalizer,” in Annual International Symposium on Microarchitecture, 2010, pp. 103–114.

[38] N. Karimi, A. K. Kanaparthi, X. Wang, O. Sinanoglu, and R. Karri, “Magic: Malicious aging in circuits/cores,” ACM Trans. on Architecture and Code Optimization (TACO), vol. 12, no. 1, pp. 1–25, 2015.

[39] D. Guad, M. Shafigue, F. Kriebel, S. Rehman, D. Sun, and J. Henkel, “Hayat: Harnessing dark silicon and variability for aging deceleration and balancing,” in Design Automation Conference, 2015, pp. 1–6.

[40] T. Sato and T. Ukezono, “Exploiting configurable approximations for tolerating aging-induced timing violations,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 103, no. 9, pp. 1028–1036, 2020.

[41] S. Mishra et al., “A simulation study of nbt1 impact on 14-nm node finet technology for logic applications: Device degradation to circuit-level interaction,” IEEE Trans. Electron Devices, vol. 66, no. 1, pp. 271–278, 2018.

[42] S. Salamin, V. M. Van Santen, H. Amrouch, N. Parivar, S. Mahapatra, and J. Henkel, “Modeling the interdependences between voltage fluctuation and bti aging,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 7, pp. 1652–1665, 2019.

[43] S. Mahapatra, Fundamentals of Bias Temperature Instability in MOS Transistors. Springer, 2016.

[44] S12. (2021). [Online]. Available: https://s12.org/open-cell-library/

[45] W. Zhao and Y. Cao, “New generation of predictive technology model for sub-45 nm early design exploration,” IEEE Trans. Electron Devices, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.

[46] “Predictive Technology Model,” http://ptm.asu.edu/, 2012.

[47] Synopsys. (2021) Synopsys EDA Tools. [Online]. Available: http://synopsys.com/

[48] F. Klemme and H. Amrouch, “Machine learning for on-the-fly reliability-aware cell library characterization,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 68, no. 6, pp. 2569–2579, 2021.

[49] S. Lee and A. Gerstlauer, Approximate High-Level Synthesis of Custom Hardware. Springer International Publishing, 2019, pp. 205–223.

[50] H. Amrouch, S. Mishra, V. van Santen, S. Mahapatra, and J. Henkel, “Impact of bti on dynamic and static power: From the physical to circuit level,” in Int. Reliability Physics Symp., 2017, pp. CR–3.1–CR–3.6.

[51] H. Amrouch et al., “Impact of variability on processor performance in negative capacitance finet technology,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3127–3137, 2020.

Konstantinos Balaskas received his Bachelor Degree in Physics and Master Degree in Electronic Physics from the Aristotle University of Thessaloniki in 2018 and 2020, respectively. Currently, he is a pursuing the PhD degree at the same institution. His research interests include approximate computing, machine learning and digital circuit design and optimization.

Georgios Zervakis is a Research Group Leader at the Chair for Embedded Systems (CES) at the Karlsruhe Institute of Technology (KIT), Germany. He received the Diploma and the Ph.D. degree from the Department of Electrical and Computer Engineering (ECE), National Technical University of Athens (NTUA), Greece, in 2012 and 2018, respectively. Before joining KIT, Georgios worked as a primary researcher in several EU-funded projects as member the Institute of Communication and Computer Systems (ICCS), Athens, Greece. His research interests include approximate computing, low power design, design automation, and integration of hardware acceleration in cloud.

Hussam Amrouch (S’11-M’15) is a Junior Professor for the Semiconductor Test and Reliability (STAR) chair within the Computer Science, Electrical Engineering Faculty at the University of Stuttgart as well as a Research Group Leader at the Karlsruhe Institute of Technology (KIT), Germany. He received his Ph.D. degree with distinction (Summa cum laude) from KIT in 2015. His main research interests are design for reliability and testing from device physics to systems, machine learning, security, approximate computing, and emerging technologies with a special focus on ferroelectric devices. He holds seven HIPEAC Paper Awards and three best paper nominations at top EDA conferences: DAC’16, DAC’17 and DATE’17 for his work on reliability. He currently serves as Associate Editor at Integration, the VLSI Journal. He has served in the technical program committees of many major EDA conferences such as DAC, ASP-DAC, ICCAD, etc. and as a reviewer in many top journals like T-ED, TCAS-I, TVLSI, TCAD, TC, etc. He has 120 publications in multidisciplinary research areas across the entire computing stack, starting from semiconductor physics to circuit design all the way up to computer-aided design and computer architecture. He is a member of the IEEE. ORCID 0000-0002-5649-3102.

Jörg Henkel (M’95-SM’01-F’15) is the Chair Professor for Embedded Systems at Karlsruhe Institute of Technology. Before that he was a research staff member at NEC Laboratories in Princeton, NJ. He received his diploma and Ph.D. (Summa cum laude) from the Technical University of Braunschweig. His research work 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, ICCAD, ESWEEK and DATE. For two consecutive terms he served as the Editor-in-Chief for 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 for major ACM and IEEE Journals. He has led several conferences as a General Chair incl. ICCAD, ESWEEK and serves as a Steering Committee chair/member for leading conferences and journals for embedded and cyber-physical systems. Prof. Henkel coordinates the DFG program SPP 1508 “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, and a Fellow of the IEEE.

Kostas Siozios received his Diploma, Master and Ph.D. Degree in Electrical and Computer Engineering from the Democritus University of Thrace, Greece, in 2001, 2003 and 2009, respectively. Currently, he is Assistant Professor at Department of Physics, Aristotle University of Thessaloniki. His research interests include Low-Power Hardware Accelerators, Resource Allocation, Machine Learning, Decision-Making Algorithms, Cyber-Physical Systems (CPS) and IoT for Smart-Grid. He has published more than 130 papers in peer-reviewed journals and conferences. Also, he has contributed in 5 books of Kluwer and Springer. The last years he works as Project Coordinator, Technical Manager or Principal Investigator in numerous research projects funded from the European Commission (EC), European Space Agency (ESA), as well as the Greek Government and Industry.