A Survey of Testing Techniques for Approximate Integrated Circuits

Marcello Traiola, *Member, IEEE,* Arnaud Virazel, *Member, IEEE,* Patrick Girard, *Fellow, IEEE,* Mario Barbareschi, and Alberto Bosio, *Member, IEEE,*

***Abstract*—Approximate Computing (AxC) is increasingly emerging as a new design paradigm to produce more efficient computation systems by judiciously reducing the computation quality. In particular, AxC has been successfully applied to Integrated Circuits (ICs), in the last years. Hence, concerning the test of such new class of ICs, namely Approximate Integrated Circuits (AxICs), new challenges – as well as new opportunities – have emerged. In this survey, we provide a thorough analysis of issues related to test procedures for AxICs and review the state-of-the-art techniques to deal with them. We resort to an illustrative example having the twofold aim of: (i) guiding the reader through the AxIC testing challenges and (ii) illustrating the existing solutions to correctly overcome them, while suitably taking advantage of opportunities coming from approximation. We analyze experimentally the most recent testing techniques for AxICs and highlight their mature aspects, as well as their shortcomings. Experimental outcomes show that the testing process for AxIC is not completely mature. Indeed, only under specific conditions existing testing procedures achieve good results.**

***Index Terms*—Testing, Approximate integrated circuits, Approximate computing, Hardware test, Robustness, Circuit faults, Automatic test pattern generation, ATPG, Approximation-aware test methodology, Test pattern, Production errors, Approximation-aware testing,**

I. Introduction

DESPITE significant energy efficiency improvements in the semiconductor industry, computer systems consume more and more energy [1]. In addition to that, a new kind of applications – usually referred to as Recognition, Mining and Synthesis (RMS) applications – is increasingly deployed in mobile devices and on Internet of Things (IoT) structures. Therefore, it is necessary to improve the next-generation silicon devices and architectures on which these applications will run. The *inherent resiliency property* of RMS applications has been thoroughly investigated over the last few years [1]–[4]. This interesting property leads RMS applications to be tolerant to errors – as long as their results remain close enough to the expected ones. Approximate Computing (AxC) [1], [2] is an emerging computing paradigm which takes advantage of the inherent application resiliency. AxC has gained increasing interest in the scientific community in the last years. It is based on the intuitive observation that selectively relaxing non-critical specifications may lead to improvements in power consumption, run time, and/or chip area. Therefore, several methodologies have been proposed to automatically *identify* and *characterize* the resilient parts of computer systems [4]–[12]. AxC has been applied to the whole digital system stack. Specifically, three main categories of *Approximate Kernels* (AxKs) have been identified, i.e., software-level AxKs, architectural-level AxKs, and circuit-level AxKs [1].

Software-level AxKs have been developed to provide programmers with the possibility to realize complex yet energy-efficient programs. This task is possible thanks to the abstraction of the *approximation* concept by means of *approximation-aware (ax-aware) programming languages* [11]–[13], *ax-aware correctness analysis engines* [14]–[19], and *ax-aware compilers* [20], [21].

At architectural-level, obtaining high-performance processing units at a low-energy cost, and memories and storage units with a good trade-off between performance and density, are ideal goals for designers. Architectural-level AxKs introduced the computation quality as a new parameter to push further next generation hardware components. Indeed, by sacrificing some quality, designers can further improve performance of computation units [22]–[25], density and energy efficiency of memories [26]–[32] and storage units [33].

Finally, circuit-level AxKs have been applied basically in two ways: (i) *over-scaling* and (ii) *functional approximation*. Over-scaling consists in lowering the Integrated Circuit (IC) supply voltage to reduce its energy consumption. If the circuit is systematically designed to benefit from over-scaling [34], [35], the timing errors are negligible compared to the energy gain. Nevertheless, the energy gain of over-scaling techniques turns out to be small [1]. Therefore, a considerable amount of work has been presented on circuit *functional approximation*: the circuit functionality is systematically changed – thus, some controlled errors are introduced – to achieve energy-efficient circuits. Circuit error can be measured according to different error metrics [36]. Three main approaches have been used to design Approximate Integrated Circuits (AxICs):

© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
DOI: 10.1109/JPROC.2020.2999613

M. Traiola, A. Virazel and P. Girard are with the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM), University of Montpellier/CNRS, Montpellier 34000 France, E-mail: {firstname.lastname}@lirmm.fr

M. Barbareschi is with the Department of Electrical Engineering and Information Technology (DIETI), University of Naples, Naples 80100 Italy, E-mail: mario.barbareschi@unina.it

A. Bosio is with the Lyon Institute of Nanotechnology (INL), Ecole Centrale de Lyon, Lyon 69000 France, E-mail: alberto.bosio@ec-lyon.fr

Manuscript received Jan 15, 2020; revised Mar 16, 2020; accepted May 28, 2020.

1) Ad-hoc approximate circuits, such as adders [37]–[51], multipliers [49]–[71], dividers [72]–[85], square root [85], and Multiply-and-ACcumulate (MAC) [86], [87]. A recent review of these studies can be found in the work by Jiang et al. [88].
2) Automatic approximate circuit synthesis methodologies [89]–[110].
3) Hardware neural accelerators to implement approximate functions [24], [111], [112]

This work focuses on digital AxICs, regardless of the approach employed to obtain them. Since approximation changes the IC behavior, it is important to revisit test and verification techniques to adapt them to AxICs. AxIC verification has been thoroughly investigated in previous studies. Specifically, formal verification methods have been applied to combinational AxICs [101], [113], [114] and sequential AxICs [114], [115].

We focus more specifically on testing aspects of AxICs. Previous research works [116]–[118] have shown that circuit approximation raises challenges for testing procedures, but also opportunities. On the one hand, the occurrence of a defect in the circuit can lead it to produce unexpected catastrophic errors. On the other hand, some defects can be tolerated, when they do not induce errors over a certain level. If properly investigated and managed, this phenomenon could lead to increase the number of circuits passing the test phase. This is usually referred to as *production yield increase*. Indeed, selling acceptably-functioning circuits – that still respect the user requirements, despite the defects – would increase the profit of semiconductor companies. This is especially critical due to the effect of *process variability* on CMOS technologies. Indeed, CMOS technologies at nano-scale have increasingly negative performance in terms of circuit yield and reliability [119]. To take advantage of the opportunity offered by AxICs, conventional test flow should be revisited.

Therefore, *Approximation-Aware testing (AxA testing)* comes into play. We identify three main AxA testing phases:

1) AxA fault classification,
2) AxA test pattern generation,
3) AxA test set application.

Briefly, *fault classification* has to divide faults into *catastrophic* (to test) and *acceptable* (not to test), according to a metric; *test pattern generation* has to produce tests able to cover all the catastrophic faults and, at the same time, to leave acceptable faults undetected; finally, the *test set application* role is to analyze the test outcomes and classify AxICs accordingly, into *catastrophically faulty*, *acceptably faulty*, and *fault-free*. Only AxICs falling into the first group will be rejected. Ultimately, this leads to a yield increase compared to the conventional test flow.

In this work, we thoroughly discuss the three AxA testing phases. Moreover, we review and evaluate the existing AxA testing techniques. We propose new metrics to accomplish the evaluation, and we perform extensive experiments to measure the effectiveness of state-of-the-art techniques. Obtained results show good maturity of fault classification and test pattern generation approaches, while some problems emerge concerning test set application techniques. In fact, only under specific conditions, existing test set application techniques achieve satisfactory results. The remainder of this paper is organized as follows.

Paper organization
- **§II Background**
  - §II-A Conventional IC test process
  - §II-B Error metrics for AxICs
- **§III Approximation-aware (AxA) testing**
  - §III-A AxA testing principles
  - §III-B Illustrative example
- **§IV AxA fault classification**
  - §IV-A Problem statement
  - §IV-B Single-condition-test-metric-aware fault classification
  - §IV-C Mean-error-metric-aware fault classification
- **§V AxA test pattern generation**
  - §V-A Problem statement
  - §V-B State-of-the-art AxA test pattern generation techniques
- **§VI AxA test set application**
  - §VI-A Problem statement
  - §VI-B A state-of-the-art solution
- **§VII Discussion and future perspectives**
- **§VIII Conclusion**

II. Background

In this section, we report some basic concepts, useful to fully understand the context and the motivation of this work. Firstly, we briefly describe the test process of conventional ICs. Secondly, we show how inherent properties of AxICs have led the scientific community to reconsider the procedures to test them.

A. Conventional IC test process

In IC manufacturing process or in the IC lifetime, physical defects may impact the circuit functionality. Therefore, tests are applied to ICs after their manufacture and during their lifetime, to detect corrupted ones. Being a fault defined as the logical manifestation of a defect, several fault models have been proposed in the literature to abstract the defect concept [120]. The most popular fault model used in practice is the Stuck-At Fault (SaF) model. In this abstraction, a circuit net is considered to be permanently set at a constant logic value, either '0' or '1'. Furthermore, the Transition fault (TF) model is used for representing delay defects, which prevent the correct data from reaching outputs at the right time. In many parts of this work, we refer to such fault models.

In the context of conventional IC test, any difference between the IC nominal behavior and the manufactured IC behavior leads to reject the circuit. The conventional testing flow is briefly reported below. Once a fault model has been chosen, an Automatic Test Pattern Generator (ATPG) takes into account all the possible faults of the Unit Under Test (UUT). Then, the ATPG begins a deterministic procedure aiming at generating input sequences – called test patterns – to sensitize the highest number of fault locations and produce detectable differences at outputs. The Fault Coverage (FC)

metric is used to measure the test quality. FC is defined as the ratio of the number of faults detected by a set of test patterns to the total number of faults in the fault list. Faults can be classified by the ATPG as *detectable*, *redundant*, and *undetectable*. At least one test pattern exists to test faults labeled as detectable. Redundant (or untestable) faults – which do not alter the circuit function – have no possible test patterns able to detect them. Finally, faults are called undetectable when the ATPG is not able to classify them, due to algorithmic limitation or due to constraints imposed by the ATPG itself or by the test engineer. Once the test set is ready, it is actually applied to manufactured ICs and the outcomes are analyzed: if there is any difference between IC test responses and expected ones, the IC is rejected; otherwise, the test passes and the IC is considered as good.

B. Error metrics for AxICs

As far as it concerns AxICs, the concept of *faulty* circuit changes and needs thorough investigation. As described in Section I, functional approximation aims to achieve gains in efficiency (time/area/energy) by relaxing some accuracy requirements. In order to still obtain satisfying results for the application, designers carefully modify the circuit structure to introduce an *acceptable error*. In order to measure the error, designers resort to **error metrics**. Hence, they define **error thresholds** to specify the maximum allowed (i.e., acceptable) error. In the testing context, the impact of detectable faults can be measured and expressed as error by using such metrics. If the obtained measure turns out to be higher than the acceptable threshold, then the circuit must be rejected. However, it may happen that the measured error stays below the acceptable threshold, then the AxIC must not be rejected. Moreover, depending on the metric, the error entailed by a fault changes. Indeed, by stimulating a faulty AxIC with an input vector i, we can measure the error e_{s_i} – caused by the fault f_s – by using a metric M. By considering the same input vector i but another metric $\widehat{M}$, the error due to the same fault f_s is measured as $\widehat{e_{s_i}}$. Usually, e_{s_i} and $\widehat{e_{s_i}}$ have different values. Therefore, the fault f_s can be considered as *acceptable* or as *catastrophic* depending on the metric(s) considered for the final application. As a result of this consideration, test procedures have to be carefully redesigned in order to address the challenges introduced by the approximation and to profitably take advantage of the opportunities.

Below we report the metrics to which we refer in this paper. Among commonly used metrics for AxICs, we can mention *Error Magnitude* (EM), *Bit-Flip Error* (BFE), *Worst Case Error* (WCE), *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), *Error Probability* (EP), and *Worst Case Bit-Flip Error* (WCBFE) [36], [50], defined as follows:

$$\text{EM}_i = \left| O_i^{approx} - O_i^{precise} \right|,\ i \in \mathcal{I} \tag{1}$$

$$\text{BFE}_i = \sum_{j=0}^{n-1} (O_{i,j}^{approx}) \oplus (O_{i,j}^{precise}),\ i \in \mathcal{I} \tag{2}$$

$$\text{WCE} = \max_{\forall i \in \mathcal{I}} \left| O_i^{approx} - O_i^{precise} \right| \tag{3}$$

$$\text{MAE} = \frac{\sum_{\forall i \in \mathcal{I}} \left| O_i^{\text{approx}} - O_i^{\text{precise}} \right|}{2^n} \tag{4}$$

$$\text{MSE} = \frac{\sum_{\forall i \in \mathcal{I}} \left| O_i^{\text{approx}} - O_i^{\text{precise}} \right|^2}{2^n} \tag{5}$$

$$\text{EP} = \sum_{\forall i \in \mathcal{I}:\ O_i^{\text{approx}} \neq O_i^{\text{precise}}} \frac{1}{2^n}. \tag{6}$$

$$\text{WCBFE} = \max_{\forall i \in \mathcal{I}} \sum_{j=0}^{n-1} (O_{i,j}^{approx}) \oplus (O_{i,j}^{precise}) \tag{7}$$

where:

$i \in \mathcal{I}$	an input value within the set of all possible inputs $\mathcal{I}$
$O_i^{precise}$	precise output integer representation, for input i
O_i^{approx}	approximate output integer representation, for input i
n	number of input signals to the circuit
$O_{i,j}$	j-th bit of the O_i output (precise or approx)

The goal of the next sections is to thoroughly describe AxA testing phases and to discuss in detail all the issues.

III. Approximation-Aware testing

In the context of approximate circuits (AxICs), the role of testing has to be reconsidered. Indeed, in presence of a fault, the actual error value at circuit's output becomes significant.

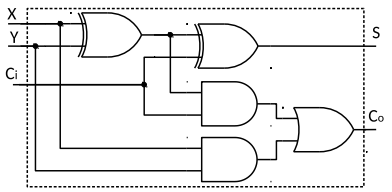
A. AxA testing principles

According to Chandrasekharan et al. [121], we classify AxIC faults into two groups, i.e., *non-redundant* faults and *approximation-redundant* (ax-redundant) faults. Non-redundant faults lead to error values higher than the acceptable threshold (i.e., catastrophic faults). Those faults must be detected in the testing phase. Conversely, ax-redundant faults cause error values lower than the threshold (i.e., acceptable faults). Those faults must not lead to AxIC rejection. Therefore, in this context, the test objective is twofold:

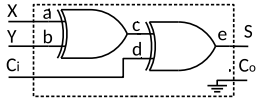
1) avoid that AxICs affected by non-redundant faults are shipped to customers;
2) ensure that AxICs affected by ax-redundant faults are not rejected.

The general and fundamental underlying assumption is the single fault condition, widely used in test techniques [120]. The AxA testing key advantage is the **yield increase**. Indeed, avoiding the detection of ax-redundant faults leads to reject fewer circuits, while guaranteeing that the AxICs shipped to the customer still respect error constraints.

Some studies in the literature proposed a similar idea. The *threshold testing* [122] was proposed in order to increase the production yield of conventional circuits (i.e., non-approximate circuits). A similar approach was adopted by Sindia et al. [123] to functionally classify conventional ICs



(a) Precise circuit



(b) Approximate circuit

	Input ($\mathcal{I}$)			Precise output (O_i^{precise})			Approximate output (O_i^{approx})		
i	C_i	X	Y	C_o	S	Int	C_o	S	Int
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	0	1	1
2	0	1	0	0	1	1	0	1	1
3	0	1	1	1	0	2	0	0	0
4	1	0	0	0	1	1	0	1	1
5	1	0	1	1	0	2	0	0	0
6	1	1	0	1	0	2	0	0	0
7	1	1	1	1	1	3	0	1	1

(c) Truth tables

Error Metric	Value
WCE	2
MAE	1
MSE	2
EP	0.5
WCBFE	1

(d) Error metric table

Fig. 1: Schematics of the full adder (a) and of its approximation (b) obtained by re-synthesizing the circuit with $C_o = 0$; (c): truth tables of both precise (i.e., non-approximate) and approximate versions. Output's integer representation for both circuits are also reported ("Int" column); (d): approximate circuit's error metric values.

and increase the yield. Although these approaches were not applied in the AxIC context, they are examples of *non-conventional testing*. In these techniques, the criterion to identify acceptable faults is defining a threshold based on the numerical error magnitude (see Equation 1) observed at circuit outputs. By imposing test generation constraints, authors were able to produce test patterns targeting non-acceptable faults. Specifically, a given input sequence could generate either an error higher than the threshold or lower, in presence of a detectable fault. In the first case, authors classified such fault as non-acceptable. Thus, they included the input sequence in the test-set. Conversely, if no input sequence could sensitize above-threshold errors for the given fault, they classified it as acceptable. In this way, they were capable to classify faults and generate test patterns only for non-acceptable faults at the same time. Nevertheless, a test sequence detecting a non-acceptable fault could still detect an acceptable one. Therefore, the authors of *threshold testing* proposed an enhanced test set application phase to verify whether test response errors were under the threshold or not. These techniques were applied only to non-approximate ICs and by considering only error magnitude metric. Thus, they can be considered as a special case of AxA testing [116].

In the next subsection, we introduce an illustrative example to suitably present the AxA test phases along the manuscript.

B. Illustrative example

Let us introduce a simple example, shown in Figure 1. We will refer to it all along the manuscript to discuss the different aspects of the AxA testing. In the figure, we report a 1-bit full adder (1a) and an approximate version of it (1b). We obtained the approximate version by simply setting the output $C_o = 0$ in the full adder and re-synthesizing the circuit. This functional approximation led to a more efficient circuit, i.e. with reduced area (2 logic gates instead of 5) and lower delay (2 logic levels instead of 3), but with some errors at outputs. Figure 1c reports the truth tables of both circuits. For the reader convenience, we also report the integer representation of both the circuit outputs ("Int" column). As reported in Figure 1d, by considering all the possible circuit inputs $i \in \mathcal{I}$, we can calculate the error values according to metrics described by Equations 3, 4, 5, 6, and 7. Values reported in Figure 1d are a direct consequence of the approximation. They constitute the error threshold values of the AxIC, fixed by specification and known at design time. Depending on the application context within which the AxIC will be utilized, considering a specific error threshold can be more appropriate than another. Erroneous values produced by the AxIC are supposed to be never higher than the error threshold considered for the final application. However, in the manufacturing phase or during AxIC lifetime, some defects can occur. As a result, the output's error value can unexpectedly be higher than the threshold. Therefore, the *fault classification* has to divide ax-redundant from non-redundant faults, according to the desired error metric and threshold. Then, the *test pattern generation* aims at producing test sets to detect only the non-redundant ones. Finally, in the *test set application* test responses are analyzed to reject only catastrophically faulty circuits, ultimately increasing the final yield.

IV. AxA fault classification

In AxA testing, the fault classification needs to be extended to take into account error metrics. In this perspective, the class of *detectable faults* is extended by including the two aforementioned ax-redundant and non-redundant sub-classes of faults. To measure the part of detectable faults classified as ax-redundant, we introduce the *expected Yield Increase* (eYI), expressed as follows:

$$\text{eYI} = \frac{\text{ax-redundant faults}}{\text{total faults}} \tag{8}$$

The purpose of such a metric is to establish an upper bound to the achievable yield increase.

In the next subsection, we discuss how different classes of error metrics impact the fault classification complexity.

A. Problem statement

In Table I, we report the error threshold value alterations caused by all possible Stuck-at faults in the approximate full adder (Figure 1). We highlight in red solid-bordered boxes the non-acceptable error values, i.e. higher than the respective thresholds t (Table 1d). Hereinafter, we use the notation

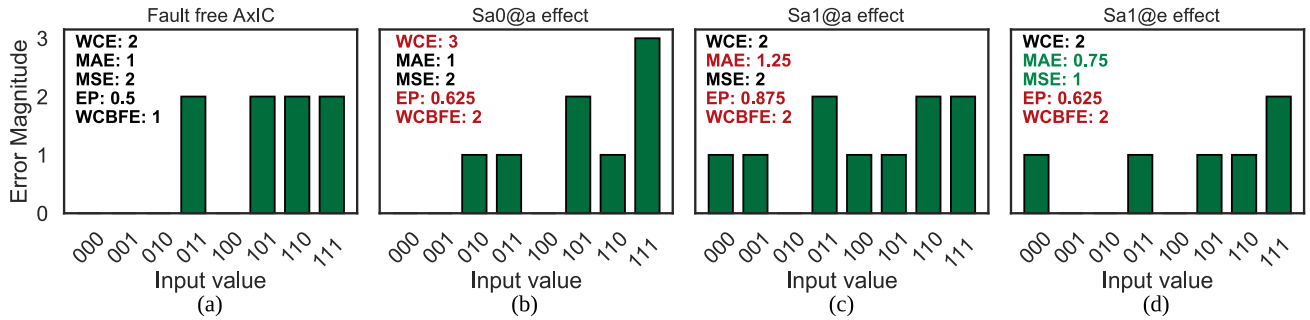


Fig. 2: (a) Error profile of the fault-free approximate circuit; (b) approximate circuit error profile in presence of the Sa0@a fault; (c) approximate circuit error profile in presence of the Sa1@a fault; (d) approximate circuit error profile in presence of the Sa1@e fault.

TABLE I: Approximate full adder error metric values for all possible Stuck-at faults, under single-fault assumption.

Fault	WCE *t=2*	MAE *t=1*	MSE *t=2*	EP *t=0.5*	WCBFE *t=1*
Sa0@a	3	1	2	0.625	2
Sa1@a	2	1.25	2	0.875	2
Sa0@b	3	1	2	0.625	2
Sa1@b	2	1.25	2	0.875	2
Sa0@c	2	1	1.5	0.75	2
Sa1@c	3	1.25	2.5	0.75	2
Sa0@d	3	1	2	0.625	2
Sa1@d	2	1.25	2	0.875	2
Sa0@e	3	1.5	3	0.875	2
Sa1@e	2	0.75	1	0.625	2

SaX@N to indicate a "stuck-at-X affecting the net N", where X can be either the value 1 or 0 and N is the label of the net. The reader can refer to Figure 1b for the net labels. By observing Table I, we can firstly remark that not all the metrics are impacted by the same faults. While all the faults impact EP and WCBFE, some faults affect the WCE and not MAE and MSE; some others have an effect on the MAE and not on WCE and MSE. Furthermore, in some particular cases, faults even reduce the observed error (green dash-bordered boxes in Table I).

Moreover, we report in Figure 2a the *error magnitude (EM) profile* of the *fault-free* approximate circuit (i.e. the circuit produces errors due to the approximation and not due to defects); also, three EM profiles in presence of a fault are reported. Specifically, we show the EM profile for the following faults: Sa0@a (Figure 2b), Sa1@a (Figure 2c), and Sa1@e (Figure 2d). The figures show how the EM profile changes differently, depending on the fault. As a result, the errors measured by the different metrics change, too. The fault impact on MAE and MSE depends on the variation of each bar in the graph. In other words, it depends on the EM of the faulty AxIC for each possible input. Similarly, the impact of the fault on the EP depends on the variation of the total number of input vectors generating an error. Conversely, WCE and WCBFE values change only if the maximum possible error changes, as a result of the fault.

To classify the fault as non-redundant according to the WCE metric, it is sufficient to prove the existence of an input vector leading the error to exceed the WCE threshold. This is the case for Sa0@a (Figure 2b). The input vector '111' leads to $EM = 3$. Conversely, if we can prove that such input vector does not exist (as for some other faults, in the example), we can classify the fault as ax-redundant w.r.t. the WCE metric. As we will show in the next subsection, existing techniques deal with this task fairly easily.

On the other hand, when performing the classification according to metrics such as MAE, MSE and EP, the task becomes more complex. Since each input vector contributes to the final error measure, finding a single input vector i for which the fault effect increases the EM is not enough to classify the fault as non-redundant w.r.t. MAE, MSE and EP. In fact, we could find another vector j "balancing" the effect of i. In our example, in the case in Figure 2b, we can see how the vectors '010', '011', '110', and '111' "balance" each other effects: some vectors increase the error value, some others decrease it, ultimately leading to a null effect on MAE and MSE metrics. Therefore, we need to measure the fault impact on the final error for all possible circuit input vectors. When the complexity of the measure becomes unmanageable, a workload-dependent subset of input vectors can be used.

In conclusion, it turns out to be less complex to evaluate the impact of a fault when using metrics for which only a single condition has to be verified, as for the WCE. Henceforth, we refer to those metrics as *single-condition-test (SCT) metrics*. Conversely, classifying faults according to metrics which involve the calculation of a mean is a $O(2^n)$ complexity problem, where n is the number of input bits. We refer to such metrics as *Mean-Error metrics*.

In the literature, some works have been performed regarding fault classification. Each one focuses on particular metrics. The two next subsections describe the state of the art of AxA fault classification techniques.

B. SCT-metric-aware fault classification

Concerning fault classification according to SCT metrics, all the proposed works are based on a common idea. To present it, let us refer to the aforementioned full adder example (Figure 1), using the WCE as SCT metric. The maximal amount of allowed error (treshold t) according to the WCE is

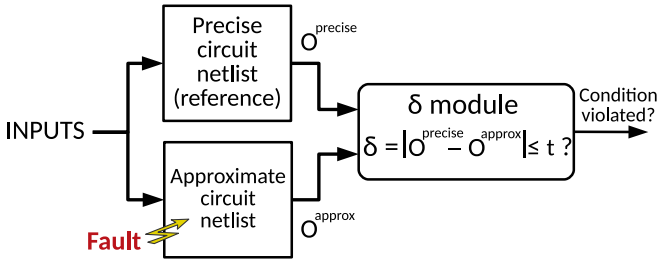


Fig. 3: SCT-metric-aware classification scheme

2. The AxIC is considered faulty if it produces any deviation δ from the precise value which is greater than 2. Any fault f modifying the AxIC output can either lead to reject the circuit ($\delta > 2$, non-redundant fault) or not ($\delta \leq 2$, ax-redundant fault). Therefore, to classify a fault as non-redundant, the existence of an input vector i leading the faulty AxIC to exhibit $\delta > 2$ has to be demonstrated. If such vector does not exist, then the fault is classified as ax-redundant. To do so, a *delta module* calculating the deviation caused by f can easily be embedded in the scheme represented in Figure 3. A digital circuit can easily implement the mentioned scheme. By using an Automatic Test Pattern Generation (ATPG), one can find the aforementioned input vector i, if it exists. Likewise, one can also resort to a Boolean satisfiability problem (SAT) formulation to represent the scheme. By solving the SAT problem, one can prove whether the input vector i exists or not. Both the mentioned techniques formally prove whether the vector i exists or not [116]. Concerning the full adder example, the vector i exists for five out of ten faults, classified as non-redundant according to WCE metric.

Chandrasekharan et al. [121] proposed a SAT-based solution for classifying faults in the AxIC context. For each fault in the fault list, they created a SAT problem instance and resolved it to classify the fault. They used WCE and WCBFE as SCT metrics. Once the non-redundant fault list is obtained, they used conventional ATPG to generate the final test set. Another SAT-based solutions was proposed by Gebregiorgis et al. [124] to classify faults according to the WCE metric and obtain test patterns to detect non-redundant faults (see Section V). Finally, we proposed [125] an ATPG-based fault classification for AxICs. We utilized the efficient ATPG structural algorithms for classifying faults according to the WCE metric and, at the same time, obtaining test patterns detecting non-redundant faults (see Section V). All the three techniques in [121], [124], [125] achieved good results in terms of fault classification.

Chandrasekharan et al. [121] performed experiments on a large set of AxICs. Specifically, they used 16-bit adders [37], [40], [43], [46], some arithmetic designs proposed by Aoki Laboratory [126], along with EPFL [127] and ISCAS-85 [128] benchmarks. They targeted the Stuck-at-Fault model. Table II reports experimental results from their work [121]. Results show significant expected Yield Increase (eYI) (Equation 8) values, especially when using the WCE metric. Indeed, on average, the eYI is between 33% and 58% with a maximum of 81%. For the WCBFE metric, results show an average between 29% and 45% with again a maximum of 81%.

TABLE II: SAT-based fault classification results [121], in terms of expected Yield Increase (eYI)

	WCE				WCBFE			
	eYI*			Avg.	eYI*			Avg.
	Avg.	Max.	Min.	Time(s)	Avg.	Max.	Min.	Time(s)
Ax 16-bit adders[1]	58%	71%	42%	16	29%	68%	10%	5
Arith designs[2]	47%	77%	18%	3355	35%	81%	3%	89
EPFL bench[3]	33%	62%	7%	5833	44%	62%	11%	10291
ISCAS-85 bench[4]	40%	81%	9%	302	45%	78%	3%	1517

*eYI: expected Yield Increase
[1]from [37], [40], [43], [46] [2]from [126] [3]from [127] [4]from [128]

TABLE III: SAT-based fault classification results [124], in terms of expected Yield Increase (eYI)

Floating-point circuits

	Error margin	5%	10%	15%	20%	25%	30%
eYI*	Avg.	13%	32%	52%	64%	73%	78%
	Max.	19%	39%	58%	71%	80%	84%
	Min.	7%	29%	47%	58%	65%	68%

Fixed-point multiplier

	Error margin	5%	10%	15%	20%	25%	30%
eYI*	Avg.	20%	43%	55%	65%	71%	76%
	Max.	30%	55%	66%	73%	77%	80%
	Min.	7%	20%	35%	44%	53%	62%

Fixed-point divider

	Error margin	5%	10%	15%	20%	25%	30%
eYI*	Avg.	15%	37%	52%	63%	69%	73%
	Max.	23%	50%	64%	72%	76%	80%
	Min.	3%	14%	29%	39%	47%	55%

*eYI: expected Yield Increase

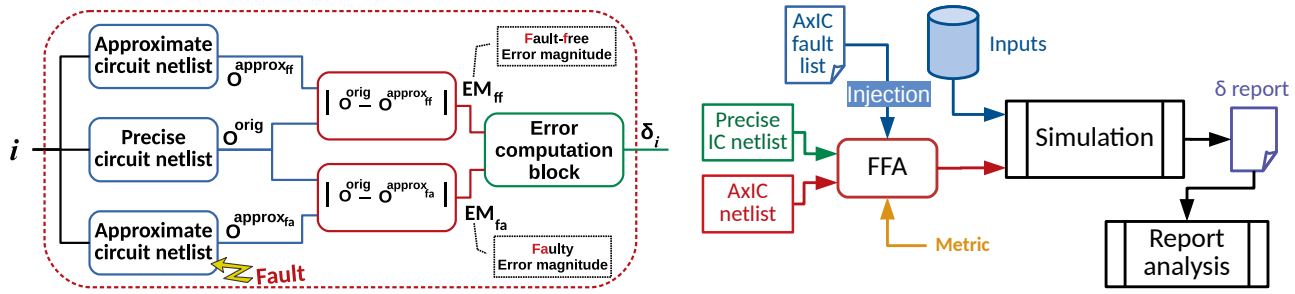
SAT-flow runtime average (s): 4376

Gebregiorgis et al. [124] performed experiments on some circuits of the AxBench suite [49]. Specifically, they used some floating-point circuits (Adder, Comparator, Multiplier, Divider, and Sqrt) and two fixed-point circuits (Multiplier and Divider) with variable fraction parts (5, 8 and 11 bits). They targeted the Stuck-at-Fault model and investigated different accepted error margins ranging from 5% to 30%. Gebregiorgis et al. [124] expressed their results in terms of Fault Reduction (FR). For example, for a non-redundant faults reduction from 100 to 80 (thus leaving 20 ax-redundant faults), they expressed the result as FR $= \frac{100}{80} = 1.25$. To be consistent with other results reported, we converted their results in terms of eYI by applying the subsequent formula:

$$\text{eYI} = 1 - \frac{1}{\text{FR}}. \tag{9}$$

In the above example, eYI $= 1 - \frac{1}{1.25} = 0.2$. By resorting to Equation 8, we find the correct number of ax-redundant faults, i.e. 20. In Table III, we report the eYI results. Also in this case, results show a significant eYI. On average, results range between 13% and 78%.

We applied the ATPG-based classification technique [125] on a large set of approximate state-of-the-art approximate arithmetic circuits [37], [40], [43], [46], [50]. Specifically, we carried out experiments on more than 1100 different AxICs, namely 8-bit and 16-bit adders (Add8, Add16), 8-bit, 16-bit



(a) Fault Filtering Architecture (FFA) (b) Overall flow

Fig. 4: Mean-Error-metric-aware fault classification approach that we proposed [129]

TABLE IV: ATPG-based fault classification results [125], in terms of expected Yield Increase (eYI)

	SaF				TF			
	eYI*			Avg.	eYI*			Avg.
	Avg.	Max.	Min.	Time(s)	Avg.	Max.	Min.	Time(s)
Add8	19%	99%	0%	0.64	25%	99%	0%	0.64
Add16	80%	94%	60%	0.71	81%	96%	61%	0.77
Mul8	55%	85%	1%	0.91	55%	84%	1%	1.01
Mul16	62%	94%	28%	0.96	64%	97%	23%	1.04
Mul32	85%	99%	41%	2.60	87%	99%	42%	2.78

*eYI: expected Yield Increase

and 32-bit multipliers (Mul8, Mul16, Mul32). We resorted to Stuck-at-Fault (SaF) and Transition Fault (TF) models and a commercial ATPG [130], instrumented using the conventional options (static and dynamic compaction). Table IV reports the experimental results that we obtained [125]. Significant eYI values were achieved also with this technique. Indeed, for the majority of the circuits, eYI was above 50%, on average. Only for 8-bit adders it was on average around 19%, when using the SaF model, and 25%, when using the TF model.

In conclusion, when considering SCT-metrics, the state of the art concerning the AxIC fault classification is quite mature. Indeed, existing works extensively contribute to determine the expected Yield Increse (eYI), which is the desired target for the final yield increase.

C. Mean-Error-metric-aware fault classification

As highlighted in Subsection IV-A, classifying faults according to Mean-Error metrics is more complex compared to SCT metrics. We addressed the fault classification problem by considering Mean-Error metrics [129]. We proposed the *Fault Filtering Architecture* (FFA) shown in Figure 4a. Given a fault, an input vector i, and a Mean-Error metric, the FFA is able to determine whether such fault changes or not the metric value for the given vector (i.e., a single bar in Figure 2a) and also to compute the magnitude of the error variation (δ_i in the figure). Moreover, since we measured the error variation δ_i, we did not need to know the actual error threshold value. Finally, by using the exhaustive set of input vectors $\mathcal{I}$ (or an application-workload-related subset $\mathcal{J} \subset \mathcal{I}$), we performed the classification. Simulating vectors belonging to $\mathcal{I}$ (or $\mathcal{J}$), while injecting – one by one – all the faults, allowed us to know all the δ_i values. If the sum of all the δ_i is greater than 0, the fault is considered non-redundant, otherwise it is considered ax-redundant. Figure 4b sketches the overall flow. We applied the FFA-based technique [129] on small circuits (i.e., 8-bit adders and multipliers from EvoApprox8b library [50]) by using the exhaustive set of input vectors $\mathcal{I}$. The simulation produced a detailed δ *report* about the fault impact on the EM profile. In Table V, we report the results. We performed fault

TABLE V: Mean-Error-metric-aware fault classification results of [129], in terms of expected Yield Increase (eYI)

	MAE and MSE				EP			
	eYI*			Avg.	eYI*			Avg.
	Avg.	Max.	Min.	Time(s)	Avg.	Max.	Min.	Time(s)
Add8	2%	12%	0%	448	1%	9%	0%	107
Mul8	7%	21%	0%	72165	3%	10%	0%	924

*eYI: expected Yield Increase

classification by using MAE, MSE and EP metrics and the Stuck-at-fault model. It was possible to perform the analysis of both MAE and MSE metrics with the same experiments. Therefore, the eYI obtained was the same. In the case of multipliers, up to 21% eYI was obtained when using the MAE and MSE metrics and up to 10% when evaluating EP metric. For 8-bit adders, we achieved up to 12% eYI when considering MAE and MSE metrics and up to 9% in the case of EP. When looking at the average results, for 8-bit multipliers, 7% eYI was achieved for MAE and MSE and 3% for the EP. For 8-bit adders, only 2% eYI for MAE and MSE and 1% for EP were attained. Concerning the average execution time, results showed that it is quite long. This is due to the intrinsic complexity of the problem.

To complete the evaluation of the technique, we extend the experimental results by adding those obtained with the rest of the AxICs, specifically the 16-bit approximate adders and multipliers and the 32-bit approximate multipliers. The whole set of possible inputs for 16-bit AxICs is composed of 2^{32} vectors (i.e., all the combinations of two 16-bit operands). For 32-bit multipliers, we reach 2^{64} vectors. Therefore, exhaustive analysis is quite time and energy consuming. A workload-dependent analysis helps to cope with such a high complexity. Thus, we performed experiments by using an input vector subset $\mathcal{J} \subset \mathcal{I}$ generated randomly. In Table VI, we report results obtained with a random input vector set composed of

2^{12} vectors. The table reports only results for MAE and MSE,

TABLE VI: Mean-Error-metric-aware fault classification results for random workload experiments, in terms of expected Yield Increase (eYI).

	MAE and MSE			
	eYI*			Avg.
	Avg.	Max.	Min.	Time(s)
Add16	2%	10%	0%	10
Mul16	12%	61%	1%	181
Mul32	21%	82%	1%	1765

*eYI: expected Yield Increase

since EP values were already 100% by design (i.e., due to the approximation), for the examined AxICs. Consequently, all the faults are ax-redundant by design. As reported in Table VI, an average of 2% eYI was achieved for 16-bit adders, 12% for 16-bit multipliers and 21% for 32-bit multipliers. Examined circuits have intrinsic quite high Mean-Error-metrics thresholds, due to aggressive approximation (see details in the related work [50]). This contributes to the higher eYI values. As expected, execution time of workload-related experiments is reduced, compared to exhaustive ones.

In conclusion, also when considering Mean-Error-metrics the state of the art on AxIC fault classification is quite mature. The task itself is complex if addressed exhaustively. If workload-dependent analysis are carried out, complexity becomes manageable.

V. AxA test pattern generation

To turn the expected Yield Increase (eYI) into actual gain, we have to go through the other two phases of the AxA test. In this section, we discuss the AxA test pattern generation issue and show the state-of-the-art solutions.

In conventional testing, we generate input sequences to test all the faults classified as detectable. In AxA testing, test patterns should target all non-redundant faults, in order to prevent catastrophic errors at circuit outputs. However, test patterns testing non-redundant faults could also detect an ax-redundant ones. This, in turn, would lead to consider the AxIC as faulty, although it is still acceptable. This phenomenon is also known as *over-testing*, i.e., a good product is considered as faulty by the test process. Ultimately, this leads the final yield increment to deviate from the expected one (eYI). Therefore, the obtained test sets should detect as few ax-redundant faults as possible. Therefore, we revisit the concept of *test quality* by dividing the fault coverage (FC) into *ax-redundant FC* (axR FC) and *non-redundant FC* (nR FC), as defined below:

$$\text{axR FC} = \frac{\text{detected ax-redundant faults}}{\text{ax-redundant faults}} \tag{10}$$

$$\text{nR FC} = \frac{\text{detected non-redundant faults}}{\text{non-redundant faults}} \tag{11}$$

To achieve a high quality test set, the ax-redundant FC has to be kept as low as possible, the non-redundant FC has to be maximized. We introduce another metric to evaluate the effect of the AxA testing procedures on yield, the *Yield Increase Loss* (YIL), defined below:

$$\text{YIL} = \frac{\text{detected ax-redundant faults}}{\text{total faults}} \tag{12}$$

It represents the **lost** yield increase, due to the detection of ax-redundant faults. The YIL is in the range $[0, \text{eYI}]$. We can observe that the YIL can be expressed also as follows:

$$\text{YIL} = \text{axR FC} \cdot \text{eYI} \tag{13}$$

This means that the ax-redundant (axR) FC metric represents the part of eYI that is not actually achieved, after the whole test procedure application.

Two fundamental problems can jeopardize the test pattern generation quality, as far as it concerns AxICs:

1) In order to achieve 100% non-redundant FC, it is not always possible to avoid testing some ax-redundant faults (i.e., ax-redundant FC > 0%).
2) conventional *approximation-unaware* test generation procedures might not be able to achieve a qualitatively good test set.

The first problem is intrinsic to the structure of AxICs, the second one is relative to conventional test generation algorithms. In this section, we describe and discuss existing solutions related to the second problem. Section VI addresses the first one.

A. Problem statement

Let us refer to the full adder example in Figure 1 to illustrate the issue. In Table VII, we report again the error metric value alterations caused by all possible Stuck-at faults in the approximate full adder. Furthermore, we report all the input vectors detecting each fault.

TABLE VII: Approximate full adder test vectors for all possible Stuck-at faults, under single-fault assumption.

Fault	WCE	MAE	MSE	EP	WCBFE	Test vectors*							
	t=2	*t=1*	*t=2*	*t=0.5*	*t=1*	0	1	2	3	4	5	6	7
Sa0@a	3	1	2	0.625	2			x	x			x	x
Sa1@a	2	1.25	2	0.875	2	x	x			x	x		
Sa0@b	3	1	2	0.625	2		x		x		x		x
Sa1@b	2	1.25	2	0.875	2	x		x		x		x	
Sa0@c	2	1	1.5	0.75	2		x	x			x	x	
Sa1@c	3	1.25	2.5	0.75	2	x			x	x			x
Sa0@d	3	1	2	0.625	2					x	x	x	x
Sa1@d	2	1.25	2	0.875	2	x	x	x	x				
Sa0@e	3	1.5	3	0.875	2		x	x		x			x
Sa1@e	2	0.75	1	0.625	2	x			x		x	x	

*0="000", 1="001",..., 7="111"

Firstly, let us assume that the fault classification is performed by using the MSE metric (threshold $t = 2$). Table VII shows that two faults entail a catastrophic error, Sa1@c ($MSE = 2.5$) and Sa0@e ($MSE = 3$). Vector 4 detects the two faults, as well as vector 7. However, both vectors detect also three ax-redundant faults (37.5% ax-redundant FC).

Moreover, there is no vector detecting all the non-redundant faults and achieving 0% ax-redundant FC. This highlights the first aforementioned problem: to achieve 100% non-redundant FC, it is not always possible to have also 0% ax-redundant FC.

The same phenomenon occurs when considering the MAE metric (threshold $t = 1$). In this case, five non-redundant faults are detected (Sa1@a, Sa1@b, Sa1@c, Sa1@d, Sa0@e). The best test vector subset turns out to be {0, 4}, having 100% non-redundant FC and still 40% ax-redundant FC. We can find other combinations, such as {0, 1}, {0, 2}, and {1, 4}, which achieve 100% non-redundant FC but also 60% ax-redundant FC. Thus, they have a *lower quality*.

Hence, we begin to notice the second mentioned problem, well illustrated by resorting to the WCE metric (threshold $t = 2$). Five non-redundant faults emerge from the classification (Sa0@a, Sa0@b, Sa1@c, Sa0@d, Sa0@e). Among all the vector combinations testing the five faults, some have a higher quality than others. For example, the combination {1, 3, 6} attains 100% non-redundant FC but also 100% ax-redundant FC. The combination {3, 4} achieves 100% non-redundant FC and 80% ax-redundant FC. The best solution is to use only the vector {7}, which indeed covers 100% of non-redundant faults, while having 0% ax-redundant FC. An ideal *AxA test pattern generation* technique should produce the qualitatively best test set for the relative metric.

Conventional ATPG algorithms do not give any guarantee of high-quality test pattern generation, when it comes to AxICs. To illustrate the phenomenon, we used a commercial ATPG [130] to create test sets for the approximate full adder of our example (Figure 1). We instrumented the ATPG using the conventional options (static and dynamic compaction) and used the Stuck-at-Fault model. For each metric, we used the corresponding non-redundant fault list and we executed the ATPG to generate test patterns. This is the test flow adopted in studies from the literature [121], [129]. In the left part of Table VIII, we report, for each metric, the obtained conventional ATPG solutions in term of test sets, ax-redundant FC and YIL. In the right part of the table, we report the corresponding best possible solutions (i.e., solutions that an ax-aware generation should find). Firstly, as expected, 100% non-redundant FC coverage was achieved for all experiments. For the MAE metric, the conventional ATPG test set was {1,7,0} and led to an ax-redundant FC of 100% (50% YIL). As already mentioned, the best test set {0,4} achieves 40% ax-redundant FC (20% YIL). Concerning the WCE metric, results were even worse: conventional ATPG generated {1,6,0}, leading to 100% ax-redundant FC (50% YIL), while the single vector {7} achieves 0% ax-redundant FC (0% YIL). Only for MSE metric results were optimal with both approaches. When considering EP or WCBFE, all the faults are non-redundant and need to be tested. In this case, no approximation-aware test technique is needed. Therefore, we did not include EP and WCBFE metrics in the experiment.

TABLE VIII: Test vector generation results when using an ideal ax-aware test vector generation and a conventional ATPG tool [130] for the AxIC in Figure 1.

	Conventional ATPG solution[1]			Ideal solution[1]		
Metric	Test sets	axR FC[2]	YIL[2]	Test sets	axR FC[2]	YIL[2]
MSE	{4}	37.50%	30%	{4} or {7}	37.50%	30%
WCE	{1, 6, 0}	100.00%	50%	{7}	0.00%	0%
MAE	{1, 7, 0}	100.00%	50%	{0, 4}	40.00%	20%

[1]100% non-redundant FC always achieved [2]Lower is better

axR FC = Ax-redundant FC
YIL = Yield Increase Loss

In conclusion, conventional ATPG techniques do not guarantee qualitatively the best solutions. This is due to the fact that state-of-the-art ATPG algorithms have not been designed to avoid testing some faults while generating test patterns. An ideal AxA test pattern generation technique generates qualitatively the best possible test set. Nevertheless, the AxIC structure may still lead to obtain ax-redundant FC > 0%, as shown in the example. The latter problem is addressed in Section VI. The next section reviews state-of-the-art AxA test pattern generation techniques.

B. State-of-the-art AxA test pattern generation techniques

In the literature, some studies [121], [129] proposed to use the conventional ATPG to generate test sets: the non-redundant fault list is generated in the fault classification phase and then it is used as target for the ATPG. We refer to this technique as *conventional generation* technique.

Other researches [124], [125] proposed techniques to simultaneously identify non-redundant faults and generate test patterns covering them. These techniques are based on the common idea to use the scheme shown in Figure 3 to obtain input vectors producing output errors greater than the threshold t, in presence of a fault f. This simultaneously marks f as non-redundant fault and produces a test pattern detecting it. Hereafter, we refer to these techniques as *ax-aware generation* techniques. Finally, we proposed a new AxA pattern generation technique to produce test patterns minimizing the ax-redundant FC, compared to conventional ATPG, while not impacting non-redundant FC [131]. Specifically, the proposed technique generates an input vector set $\mathcal{S}$ and searches within it for the optimal subset $\mathcal{V}$ which attains the required coverage. Generally, the set $\mathcal{S}$ is itself a sub-set of the exhaustive input vector set. Figure 5 depicts the proposed

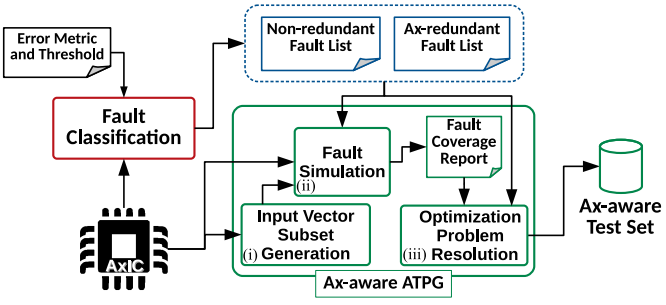


Fig. 5: AxA pattern generation technique [131]

AxA pattern generation flow [131]. The technique is composed of three main phases: (i) input vector subset ($\mathcal{S}$) generation, (ii) fault simulation and (iii) optimization problem formulation and resolution. The first phase takes as input the AxIC and

generates a user-configurable number of input vectors ($\mathcal{S}$). In this phase, various algorithms for input vector generation can be used. The fault simulation phase takes as input the generated $\mathcal{S}$, the AxIC and the two fault lists (ax-redundant and non-redundant). The fault lists are previously obtained by using any of the AxA fault classification techniques described in Section IV with any error metrics. The output of the fault simulation phase is a fault coverage report which records, for each fault, all the input vectors in $\mathcal{S}$ covering it. Finally, the goal of the third phase is to select the smallest test pattern subset $\mathcal{V} \subset \mathcal{S}$ which minimizes the ax-redundant FC and achieves total non-redundant FC. To accomplish this task, an Integer Linear Problem (ILP) is formulated, by using the fault coverage report, the vector set $\mathcal{S}$ and the fault lists. The problem solution constitutes the final ax-aware test set $\mathcal{V}$. In the remainder of the paper, we refer to this technique as *pattern-selection-based generation*.

To evaluate the test set quality for the mentioned approaches, we performed experiments on a large set of approximate state-of-the-art approximate arithmetic circuits [37], [40], [43], [46], [50], introduced in Subsection IV-B. We considered the WCE as error metric, since the previously discussed techniques [124], [125] are only applicable in this case. After fault classification, we obtained test patterns by using conventional, ax-aware, and pattern-selection-based generations. For the latter, to obtain the input vector set $\mathcal{S}$, we performed multiple test generations with the ax-aware technique to have a wider research space for the ILP problem.

Then, we performed fault-simulation by using the generated patterns and the two fault lists (i.e., non-redundant and ax-redundant faults) in order to measure non-redundant and ax-redundant FCs.

The achieved non-redundant FC was always 100%, which confirms that all the techniques achieve the first objective of AxA testing, i.e. detecting all the non-redundant faults (see Section III). Then, in Table IX, we report results in terms of ax-redundant FC, as well as in terms of Yield Increase Loss (YIL).

As shown, conventional generation technique exhibits an average ax-redundant FC between 65% and 92%, with peaks at 100%, corresponding to a YIL between 14% and 50%. Significant lower (thus better) ax-redundant FC and YIL values were achieved by using ax-aware and pattern-selection-based techniques. Ax-aware generation technique showed average ax-redundant FC between 43% and 83%, corresponding to a YIL between 9% and 39% (lower is better). Pattern-selection-based generation further improved results, by obtaining ax-redundant FC between 33% and 76%, corresponding to a YIL between 7% and 36%. On the other hand, concerning execution time, the pattern-selection-based generation technique entails a much higher overhead. This is due to the intrinsic complexity of the ILP problem. Thus, the technique may suffer from scalability issues.

Even though obtained results are quite good, they are still far from ideal. Indeed, while these techniques improve the test quality, it turns out that some ax-redundant faults are still detected. Ultimately, this leads to a yield increase lower than expected (i.e., YIL > 0%). This is due to the intrinsic

TABLE IX: Ax-redundant FC (axR FC) and Yield Increase Loss (YIL) results. YIL and axR FC indicate the absolute and the relative loss of yield increase, respectively (see Section V).

Conventional generation[1]

	Add8		Add16	
	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]
Min.	0.00%	0.00%	65.52%	52.68%
Max.	100.00%	77.45%	89.90%	71.15%
Avg.	**65.81%**	**14.04%**	**75.05%**	**59.83%**
Time[3]	0.61		0.75	

	Mul8		Mul16		Mul32	
	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]
Min.	73.55%	5.20%	64.00%	16.53%	72.06%	33.00%
Max.	99.37%	73.92%	100.00%	50.00%	97.73%	75.79%
Avg.	**91.43%**	**43.20%**	**92.08%**	**31.63%**	**90.94%**	**50.55%**
Time[3]	0.87		0.91		2.3	

Ax-aware generation[1]

	Add8		Add16	
	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]
Min.	0.00%	0.00%	58.91%	38.99%
Max.	100.00%	71.57%	90.84%	69.66%
Avg.	**43.17%**	**9.66%**	**72.64%**	**58.16%**
Time[3]	0.64		0.8	

	Mul8		Mul16		Mul32	
	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]
Min.	17.81%	1.02%	51.90%	16.62%	11.03%	6.47%
Max.	95.43%	74.18%	96.88%	40.00%	85.86%	39.87%
Avg.	**83.03%**	**39.62%**	**77.11%**	**26.02%**	**47.61%**	**24.98%**
Time[3]	0.91		0.96		2.6	

Pattern-selection-based generation[1]

	Add8		Add16	
	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]
Min.	0.00%	0.00%	48.79%	38.08%
Max.	84.85%	54.95%	67.64%	57.01%
Avg.	**33.49%**	**7.61%**	**56.87%**	**45.43%**
Time[3]	5		15339	

	Mul8		Mul16		Mul32	
	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]	axR FC[2]	YIL[2]
Min.	17.81%	1.02%	48.00%	14.33%	11.03%	6.47%
Max.	91.16%	69.51%	91.98%	36.36%	85.65%	39.87%
Avg.	**76.48%**	**36.39%**	**72.97%**	**24.78%**	**47.46%**	**24.90%**
Time[3]	2275		21230		2343	

[1]100% non-redundant FC always achieved [2]Lower is better
[3]Average time in seconds

axR FC = ax-Redundant Fault Coverage YIL = Yield Increase Loss

structure of the AxICs, as discussed at the beginning of the section. Therefore, it is necessary to rely on the *test set application* phase to correctly divide AxICs affected by ax-redundant faults from AxICs affected by non-redundant ones. Thus, it turns out that enhancing the test set application phase is **crucial** to obtain a high-quality approximation-aware test. The next section addresses such aspects.

VI. AxA TEST SET APPLICATION

As shown in the previous section, test pattern generation techniques can generate qualitatively different test sets. To

TABLE X: Output (in integer format) of the example circuit (see Figure 1) for different cases: precise (Fig 1a), fault-free approximate (see Fig 1b), and faulty approximate with different Stuck-at faults.

Input $\mathcal{I}$				$^{*}O^{precise}$	Fault-free	Faulty $^{\dagger}O^{approx}$									
i	C_i	X	Y		$^{\dagger}O^{approx}$	Sa0@a	Sa1@a	Sa0@b	Sa1@b	Sa0@c	Sa1@c	Sa0@d	Sa1@d	Sa0@e	Sa1@e
0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1
1	0	0	1	1	1	1	0	0	1	0	1	1	0	0	1
2	0	1	0	1	1	0	1	1	0	0	1	1	0	0	1
3	0	1	1	2	0	1	0	1	0	0	1	0	1	0	1
4	1	0	0	1	1	1	0	1	0	1	0	0	1	0	1
5	1	0	1	2	0	0	1	1	0	1	0	1	0	0	1
6	1	1	0	2	0	1	0	0	1	1	0	1	0	0	1
7	1	1	1	3	1	0	1	0	1	1	0	0	1	0	1
Fault classification (MAE‡):						ax-red.	non-red.	ax-red.	non-red.	ax-red.	non-red.	ax-red.	non-red.	non-red.	ax-red.

*Precise output;
†Approximate output;
‡Mean Average Error (MAE) = 1

$\boxed{Value}$: vector i detects the fault $\rightarrow Value$ is different from fault-free O_i^{approx}
$\boxed{\boxed{Value}}$: approximate circuit output. $\rightarrow Value$ is different from $O_i^{precise}$.

improve the final test quality, *test set application* plays an important role. In this phase, we need techniques able to distinguish between the detection of an ax-redundant fault and a non-redundant one, when the actual test is performed on the manufactured AxIC. The goal is to classify AxICs into *catastrophically faulty* (i.e., affected by a non-redundant fault), *acceptably faulty* (i.e., affected by an ax-redundant fault), and *fault-free*. Only catastrophically faulty AxICs are rejected. The classification should be performed according to circuit responses.

In Subsection VI-A, we show and discuss the issue in details. So far, no techniques have been presented to deal with such aspect, in the context of AxA testing. Therefore, we applied to AxICs the technique described in Subsection III-A, the *threshold testing*, originally presented for conventional ICs [122]. In Subsection VI-B, we discuss its suitability.

A. Problem statement

As mentioned in the previous section, the proper structure of an AxIC usually makes impossible for a test set to avoid the detection of some ax-redundant faults.

To show the issue in detail, we resort to the full adder example introduced in Section III-B. In the left part of Table X, we report the outputs of the precise IC ($O^{precise}$) and of the fault-free AxIC (O^{approx}), for each input vector $i \in [0, 7]$. Output values are reported as integer (e.g., 00 = "0", 01 = "1", etc.). To measure the error, we used the MAE metric (Equation 4). In the example, the MAE threshold is 1. A fault f, affecting the AxIC, is considered non-redundant only if it makes the MAE value become greater than 1. As already shown in Section IV, based on the difference between the obtained faulty outputs (faulty O_i^{approx}) and the precise output ($O_i^{precise}$), faults are classified. We report the class of each fault in the last row of the table.

In order to illustrate the problem, we report in the right part of Table X the impact of each stuck-at fault on the AxIC output. We report in red solid-bordered boxes the *faulty* O_i^{approx} values that differ from the *fault-free* O_i^{approx} ones. Thanks to this output difference, in the test application phase we can detect whether a fault affected the AxIC or not. While in conventional test each difference between actual and expected outputs leads to reject the circuit, when it comes to AxICs we have to reconsider this mechanism. Indeed, even if we dispose of the best possible test patterns – i.e., achieving maximum non-redundant FC and low ax-redundant FC –, a test vector intended to detect a non-redundant fault can still detect an ax-redundant one, ultimately rejecting a still-acceptable circuit. In the example, the best possible test set is the couple {0, 4} (see Table VIII). In Table X, we can remark that the vector 4 detects four non-redundant faults (Sa1@a, Sa1@b, Sa1@c, Sa0@e), but also one ax-redundant fault (Sa0@d). Similarly, vector 0 detects four non-redundant faults (Sa1@a, Sa1@b, Sa1@c, Sa1@d), but also one ax-redundant fault (Sa1@e). Therefore, the two vectors detect all the non-redundant faults (i.e., five faults) but also 40% of the ax-redundant faults (two out of five). Therefore, while the *expected Yield Increase* (eYI) is 50% (five ax-redundant faults avoided, out of ten total faults), by using the classic test application we drop to 30% actual increase (three ax-redundant faults avoided, out of ten total faults). To avoid this over-testing phenomenon, we need to perform a further analysis in the test application phase (i.e., after the application of the test patterns to the manufactured AxIC). In the next subsection we show how a state-of-the-art technique partially deals with this issue.

B. A state-of-the-art solution

As discussed in Subsection III-A, *threshold testing* [122] can be considered as a special case of AxA testing. In their work [122], the authors slightly modified the test set application phase, by adding a further verification: once a test vector is applied to the IC, test responses are compared with the expected precise ones (i.e., those produced by the non-approximate fault-free circuit). If the difference (i.e., the EM) is lower than a given threshold, the circuit is considered still acceptable. Otherwise, the circuit is rejected.

To preliminary study the suitability of threshold testing technique for AxICs we applied it to our full adder example. We used the WCE and the MAE as error metrics and the corre-

TABLE XI: Test set application technique (introduced in *threshold testing* [122]) applied on the full adder example (Figure 1).

		Non-redundant faults (result must be > threshold)					Ax-redundant faults (result must be $\leq$ threshold)				
Metric threshold	**WCE = 2**	Sa0@a	Sa0@b	Sa1@c	Sa0@d	Sa0@e	Sa1@a	Sa1@b	Sa0@c	Sa1@d	Sa1@e
Ax-aware vectors	vector 7 EM:	3 ✓	3 ✓	3 ✓	3 ✓	3 ✓	2 ✓	2 ✓	2 ✓	2 ✓	2 ✓
Conventional ATPG vectors	vector 1 EM:	0 ✗	1 ✗	0 ✗	0 ✗	1 ✗	1 ✓	0 ✓	1 ✓	1 ✓	0 ✓
	vector 6 EM:	1 ✗	2 ✗	2 ✗	1 ✗	2 ✗	2 ✓	1 ✓	1 ✓	2 ✓	1 ✓
	vector 0 EM:	0 ✗	0 ✗	1 ✗	0 ✗	0 ✗	1 ✓	1 ✓	0 ✓	1 ✓	1 ✓
		✓$EM > 2$		✗$EM \leq 2$			✓$EM \leq 2$		✗$EM > 2$		

Metric threshold	**MAE = 1**	Sa1@a	Sa1@b	Sa1@c	Sa1@d	Sa0@e	Sa0@a	Sa0@b	Sa0@c	Sa0@d	Sa1@e
Ax-aware vectors	vector 0 EM:	1 ✗	1 ✗	1 ✗	1 ✗	0 ✗	0 ✓	0 ✓	0 ✓	0 ✓	1 ✓
	vector 4 EM:	2 ✓	2 ✓	1 ✗	1 ✗	2 ✓	1 ✓	1 ✓	2 ✗	2 ✗	1 ✓
Conventional ATPG vectors	vector 1 EM:	1 ✗	0 ✗	0 ✗	1 ✗	1 ✗	0 ✓	1 ✓	1 ✓	0 ✓	0 ✓
	vector 7 EM:	2 ✓	2 ✓	3 ✓	2 ✓	3 ✓	3 ✗	3 ✗	2 ✗	3 ✗	2 ✗
	vector 0 EM:	1 ✗	1 ✗	1 ✗	1 ✗	0 ✗	0 ✓	0 ✓	0 ✓	0 ✓	1 ✓
		✓$EM > 1$		✗$EM \leq 1$			✓$EM \leq 1$		✗$EM > 1$		

✓= right decision ✗ = wrong decision

sponding approximation-aware and conventional test sets (see Table VIII). We simulated each test vector after injecting, one by one, all ax-redundant and non-redundant faults. Hence, we measured the corresponding Error Magnitude (EM). For each metric, if the measured EM was greater than the corresponding threshold t, the circuit was considered faulty and rejected, otherwise the test passed. We report results in Table XI. Results highlight that the technique provided correct results only under both the following conditions:

1) The WCE metric was used
2) Approximation-aware test vectors were used.

Indeed, as shown at the top of the table, faults classified according to WCE metric (threshold $t = 2$) and tested by ax-aware test vectors were correctly identified: all non-redundant faults caused an EM value higher than the threshold (i.e., EM > 2); all the ax-redundant faults produced an EM value under or equal to the threshold (i.e., EM $\leq$ 2). Conversely, conventional ATPG vectors were not able to attain the same result when WCE was considered. Furthermore, as shown at the bottom of the table, for MAE metric ($t = 1$) the technique did not deliver correct outcomes. In fact, some non-redundant faults were masked (i.e., EM $\leq$ 1) and some ax-redundant ones were detected (i.e., EM > 1) when using both ax-aware and conventional vectors.

In conclusion, to properly apply the technique, three constraints need to be satisfied:

1) precise circuit test responses must be known;
2) the considered metric must be an SCT metric (e.g., WCE, WCBFE);
3) test patterns must be produced with an ax-aware generation technique.

To corroborate the statement, we applied the technique to experiments shown in Section V-B. The considered metric was the WCE. In Tables XII and XIII, we report experimental results. By comparing results with those in Table IX, we can clearly notice that the technique gives optimal results (i.e., 100% non-redundant FC and 0% ax-redundant FC) when the above listed conditions are respected. Conversely, outcomes achieved when using conventional test patterns were not acceptable (i.e., the third condition was violated). Indeed, although a better ax-redundant FC (Table XIII) was achieved w.r.t. Table IX, the non-redundant FC (Table XII) not always reached 100%. This leads to undermine the first key aspect of AxA testing, i.e. detecting all the non-redundant faults, which cause catastrophic errors. Concerning the overhead in terms of execution time, the test set application technique introduced in *threshold testing* [122] entails only an extra comparison between the test outcomes and the error threshold value.

TABLE XII: Non-redundant FC (nR FC) results when using the test set application technique introduced in *threshold testing* [122].

		Conventional generation				
		Add8	Add16	Mul8	Mul16	Mul32
nR FC[1]	Min.	27.78%	34.94%	62.98%	68.63%	77.18%
	Max.	100.00%	100.00%	100.00%	100.00%	100.00%
	Avg.	**97.23%**	**56.46%**	**91.60%**	**93.60%**	**94.60%**

[1]non-redundand FC should be always 100%

N.B. For *ax-aware* and *pattern-selection-based* generations, the non-redundant FC was always 100%.

TABLE XIII: Ax-redundant FC (axR FC) and Yield Increase Loss (YIL) results when using the test set application technique introduced in *threshold testing* [122].

	Conventional generation			
	Add8		Add16	
	axR FC[1]	YIL[1]	axR FC[1]	YIL[1]
Min.	0.00%	0.00%	0.23%	0.22%
Max.	100.00%	10.71%	4.17%	2.52%
Avg.	**26.85%**	**3.28%**	**1.45%**	**1.09%**

	Mul8		Mul16		Mul32	
	axR FC[1]	YIL[1]	axR FC[1]	YIL[1]	axR FC[1]	YIL[1]
Min.	0.84%	0.50%	2.38%	0.57%	2.92%	1.49%
Max.	70.00%	15.29%	45.24%	17.92%	41.41%	18.28%
Avg.	**14.69%**	**5.47%**	**15.31%**	**5.40%**	**17.85%**	**8.18%**

[1]Lower is better

N.B. For *ax-aware* and *pattern-selection-based* generations, the ax-redundant FC and the YIL were always 0%.

In conclusion, existing AxA test set application techniques guarantee a high-quality test process for AxICs only under the aforementioned conditions.

VII. Discussion and future perspectives

All AxA testing phases bring important contributions to the final test quality. *AxA fault classification* separates catastrophic faults from the acceptable ones. Results of this phase determine the expected Yield Increase (eYI). The actual yield increase is the result of the synergy between *AxA test pattern generation* and *AxA test set application*. Indeed, when acceptable faults are detected despite a high-quality test set, a further analysis has to be performed after the test application to state whether the faulty behavior of the AxIC is tolerable or not.

As discussed in previous sections, state-of-the-art *fault classification* techniques are quite mature. Concerning AxA *test pattern generation* techniques, in Section V we showed that they provide test sets either non-optimum in a short time or optimum with a significant time overhead. Therefore, those techniques should be optimized to provide high quality test sets without incurring a big overhead. Furthermore, we showed that AxA *test application techniques* allow achieving optimal results only under some conditions. As a result, it is necessary to develop new AxA *test set application* techniques to improve final AxA test quality under any conditions.

Let us now express some further considerations. From our research, it results that the approximation of integrated circuits induces the need of an extra effort to achieve a high quality test. As an analogy, in the past, the VLSI advent led the complexity of microelectronic systems to grew considerably. As a consequence, performing IC testing became more and more difficult. Then, the well-known Design-for-Testability (DfT) concept was introduced. DfT was introduced basically to keep test development and test time reasonably low while assuring the detection of all faults within an IC [120]. With DfT, the testability finally became part of the common design requirements. In the same way, in our opinion, taking into account the final circuit testability in the approximation process is of great interest. In particular, since the main goal of approximate computing is to reduce the cost, we think that simply applying conventional DfT techniques to AxICs is not suitable. On the contrary, dedicated *Approximation-for-Testability (AfT)* techniques need to be developed to ensure a high testability for the AxICs. In fact, there is no technique in the literature dealing with such aspect. We aim at studying this aspect in future and developing methodologies to address it.

VIII. Conclusions

In this paper, we presented a survey of the state-of-the-art testing techniques for approximate integrated circuits (AxICs). In this context, the core problem is to ensure that defects occurring to AxICs do not introduce *catastrophic* errors, i.e. greater than the acceptable error threshold. At the same time, test procedures must not reject AxICs affected by *acceptable* defects, i.e., introducing only errors smaller than the error thresholds.

We identified and illustrated in detail three main phases: approximation-aware *fault classification*, approximation-aware *test pattern generation*, and approximation-aware *test set application*. We proposed new metrics to evaluate state-of-the-art techniques, we measured their contributions to all the test phases, and we highlighted their mature and weak aspects. In particular, experiments showed that fault classification and test pattern generation techniques are quite mature, although the latter need to be optimized. On the contrary, test set application techniques need major improvements, even though – under some conditions – good test quality can be achieved. Furthermore, we deem interesting and valuable for the scientific community to move towards the study of new Approximation-for-Testability principles.

References

[1] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb 2016.

[2] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016.

[3] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, May 2013, pp. 1–6.

[4] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–9.

[5] V. K. Chippa, S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing: An integrated hardware approach," in *2013 Asilomar Conference on Signals, Systems and Computers*, Nov 2013, pp. 111–117.

[6] Qian Zhang, F. Yuan, R. Ye, and Q. Xu, "Approxit: An approximate computing framework for iterative methods," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2014, pp. 1–6.

[7] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "Sage: Self-tuning approximation for graphics engines," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2013, pp. 13–24.

[8] W. Baek and T. Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation." ACM SIGPLAN, June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/green-framework-supporting-energy-conscious-programming-using-controlled-approximation/

[9] M. Ringenburg, A. Sampson, I. Ackerman, L. Ceze, and D. Grossman, "Monitoring and debugging the quality of results in approximate programs," *SIGARCH Comput. Archit. News*, vol. 43, no. 1, pp. 399–411, Mar. 2015.

[10] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, "Rumba: An online quality management system for approximate computing," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, June 2015, pp. 554–566.

[11] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," in *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '11. New York, NY, USA: ACM, 2011, pp. 164–174.

[12] J. Bornholt, T. Mytkowicz, and K. S. McKinley, "Uncertain< t >: A first-order type for uncertain data," Tech. Rep. MSR-TR-2013-46, April 2013. [Online]. Available: https://www.microsoft.com/en-us/research/publication/uncertaint-a-first-order-type-for-uncertain-data/

[13] M. Carbin, S. Misailovic, and M. C. Rinard, "Verifying quantitative reliability for programs that execute on unreliable hardware," *SIGPLAN Not.*, vol. 48, no. 10, pp. 33–52, Oct. 2013.

[14] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, "The ins and outs of the probabilistic model checker mrmc," *Performance Evaluation*, vol. 68, no. 2, pp. 90 – 104, 2011, advances in Quantitative Evaluation of Systems.

[15] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: Probabilistic symbolic model checker," in *Computer Performance Evaluation: Modelling Techniques and Tools*, T. Field, P. G. Harrison, J. Bradley, and U. Harder, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 200–204.

[16] S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. Navidpour, "Proving programs robust," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ser. ESEC/FSE '11. New York, NY, USA: ACM, 2011, pp. 102–112.

[17] S. Misailovic, M. Carbin, S. Achour, Z. Qi, and M. C. Rinard, "Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels," *SIGPLAN Not.*, vol. 49, no. 10, pp. 309–328, Oct. 2014.

[18] M. Rinard, "Probabilistic accuracy bounds for fault-tolerant computations that discard tasks," in *Proceedings of the 20th Annual International Conference on Supercomputing*, ser. ICS '06. New York, NY, USA: ACM, 2006, pp. 324–334.

[19] A. Sampson, P. Panchekha, T. Mytkowicz, K. S. McKinley, D. Grossman, and L. Ceze, "Expressing and verifying probabilistic assertions," *SIGPLAN Not.*, vol. 49, no. 6, pp. 112–122, Jun. 2014.

[20] S. Misailovic, S. Sidiroglou, H. Hoffmann, and M. Rinard, "Quality of service profiling," in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 25–34.

[21] E. Schkufza, R. Sharma, and A. Aiken, "Stochastic optimization of floating-point programs with tunable precision," *SIGPLAN Not.*, vol. 49, no. 6, pp. 53–64, Jun. 2014.

[22] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," *SIGARCH Comput. Archit. News*, vol. 40, no. 1, pp. 301–312, Mar. 2012.

[23] U. R. Karpuzcu, I. Akturk, and N. S. Kim, "Accordion: Toward soft near-threshold voltage computing," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014, pp. 72–83.

[24] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2012, pp. 449–460.

[25] A. Chandrasekharan, D. Große, and R. Drechsler, "Proact: A processor for high performance on-demand approximate computing," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 463–466. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060415

[26] S. Z. Gilani, N. S. Kim, and M. Schulte, "Scratchpad memory optimizations for digital signal processing applications," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[27] D. J. Palframan, N. S. Kim, and M. H. Lipasti, "Precision-aware soft error protection for gpus," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014, pp. 49–59.

[28] M. Shoushtari, A. BanaiyanMofrad, and N. Dutt, "Exploiting partially-forgetful memories for approximate computing," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 19–22, March 2015.

[29] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: Saving dram refresh-power through critical data partitioning," *SIGARCH Comput. Archit. News*, vol. 39, no. 1, pp. 213–224, Mar. 2011.

[30] A. Raha, S. Sutar, H. Jayakumar, and V. Raghunathan, "Quality configurable approximate dram," *IEEE Transactions on Computers*, vol. 66, no. 7, pp. 1172–1187, July 2017.

[31] M. Jung, D. M. Mathew, C. Weis, and N. Wehn, "Invited: Approximate computing with partially unreliable dynamic random access memory — approximate dram," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–4.

[32] Y. Chen, X. Yang, F. Qiao, J. Han, Q. Wei, and H. Yang, "A multi-accuracy-level approximate memory architecture based on data significance analysis," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 385–390.

[33] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate storage in solid-state memories," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2013, pp. 25–36.

[34] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[35] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys, "Pushing the limits of voltage over-scaling for error-resilient applications," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 476–481.

[36] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, Sept 2013.

[37] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proceedings of the 2009 12th International Symposium on Integrated Circuits*, Dec 2009, pp. 69–72.

[38] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 8, pp. 1225–1229, Aug 2010.

[39] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: Imprecise adders for low-power approximate computing," in *IEEE/ACM International Symposium on Low Power Electronics and Design*, Aug 2011, pp. 409–414.

[40] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *DAC Design Automation Conference 2012*, June 2012, pp. 820–825.

[41] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2012, pp. 728–735.

[42] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in *2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*, Aug 2013, pp. 690–693.

[43] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 48–54.

[44] W. Liu, L. Chen, C. Wang, M. O'Neill, and F. Lombardi, "Inexact floating-point adder for dynamic image processing," in *14th IEEE International Conference on Nanotechnology*, Aug 2014, pp. 239–243.

[45] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '15. New York, NY, USA: ACM, 2015, pp. 343–348.

[46] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[47] T. Ban, B. Wang, and L. Naviner, "Design, synthesis and application of a novel approximate adder," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2018, pp. 488–491.

[48] T. Zhang, W. Liu, E. McLarnon, M. O'Neill, and F. Lombardi, "Design of majority logic (ml) based approximate full adders," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[49] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran, "Axbench: A multiplatform benchmark suite for approximate computing," *IEEE Design Test*, vol. 34, no. 2, pp. 60–68, April 2017.

[50] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approx 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.

[51] B. Barrois, O. Sentieys, and D. Menard, "The hidden cost of functional approximation against careful data sizing — a case study," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 181–186.

[52] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *2011 24th Internatioal Conference on VLSI Design*, Jan 2011, pp. 346–351.

[53] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–4.

[54] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2015, pp. 418–425.

[55] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, April 2015.

[56] L. Qian, C. Wang, W. Liu, F. Lombardi, and J. Han, "Design and evaluation of an approximate wallace-booth multiplier," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 1974–1977.

[57] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, "Architectural-space exploration of approximate multipliers," in *Proceedings of the 35th International Conference on Computer-Aided Design*, ser. ICCAD '16. New York, NY, USA: ACM, 2016, pp. 80:1–80:8.

[58] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2638–2644, Aug 2016.

[59] M. Češka, J. Matyaš, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, "Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 416–423.

[60] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1435–1441, Aug 2017.

[61] M. Imani, D. Peroni, and T. Rosing, "Cfpu: Configurable floating point multiplier for energy-efficient computing," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2017, pp. 1–6.

[62] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han, "Scalable construction of approximate multipliers with formally guaranteed worst case error," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 11, pp. 2572–2576, Nov 2018.

[63] V. Mrazek, Z. Vasicek, and L. Sekanina, "Design of quality-configurable approximate multipliers suitable for dynamic environment," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Aug 2018, pp. 264–271.

[64] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, Sep. 2018.

[65] W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. Swartzlander, and F. Lombardi, "Design and analysis of approximate redundant binary multipliers," *IEEE Transactions on Computers*, vol. 68, no. 6, pp. 804–819, June 2019.

[66] H. Jiang, C. Liu, F. Lombardi, and J. Han, "Low-power approximate unsigned multipliers with configurable error recovery," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 189–202, Jan 2019.

[67] S. Rehman, B. S. Prabakaran, W. El-Harouni, M. Shafique, and J. Henkel, *Heterogeneous Approximate Multipliers: Architectures and Design Methodologies*. Springer International Publishing, 2019, pp. 45–66.

[68] S. Venkatachalam, E. Adams, H. J. Lee, and S. Ko, "Design and analysis of area and power efficient approximate booth multipliers," *IEEE Transactions on Computers*, pp. 1–1, 2019.

[69] C. Tung and S. Huang, "Low-power high-accuracy approximate multiplier using approximate high-order compressors," in *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*, April 2019, pp. 163–167.

[70] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–9, 2019.

[71] R. R. Osorio and G. Rodríguez, "Truncated simd multiplier architecture for approximate computing in low-power programmable processors," *IEEE Access*, vol. 7, pp. 56 353–56 366, 2019.

[72] L. Chen, J. Han, W. Liu, and F. Lombardi, "Design of approximate unsigned integer non-restoring divider for inexact computing," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '15. New York, NY, USA: ACM, 2015, pp. 51–56. [Online]. Available: http://doi.acm.org/10.1145/2742060.2742063

[73] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the design of approximate restoring dividers for error-tolerant applications," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2522–2533, Aug 2016.

[74] R. Zendegani, M. Kamal, A. Fayyazi, A. Afzali-Kusha, S. Safari, and M. Pedram, "Seerad: A high speed yet energy-efficient rounding-based approximate divider," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1481–1484.

[75] S. Hashemi, R. I. Bahar, and S. Reda, "A low-power dynamic divider for approximate applications," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6.

[76] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, and Z. Navabi, "Truncapp: A truncation-based approximate divider for energy efficient dsp applications," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 1635–1638.

[77] L. Chen, F. Lombardi, P. Montuschi, J. Han, and W. Liu, "Design of approximate high-radix dividers by inexact binary signed-digit addition," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 293–298. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060404

[78] L. Chen, J. Han, W. Liu, P. Montuschi, and F. Lombardi, "Design, evaluation and application of approximate high-radix dividers," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 299–312, July 2018.

[79] W. Liu, J. Li, T. Xu, C. Wang, P. Montuschi, and F. Lombardi, "Combining restoring array and logarithmic dividers into an approximate hybrid design," in *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*, June 2018, pp. 92–98.

[80] H. Jiang, L. Liu, F. Lombardi, and J. Han, "Adaptive approximation in arithmetic circuits: A low-power unsigned divider design," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 1411–1416.

[81] K. Manikantta Reddy, M. H. Vasantha, Y. B. Nithin Kumar, and D. Dwivedi, "Design of approximate dividers for error tolerant applications," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2018, pp. 496–499.

[82] S. Behroozi, J. Li, J. Melchert, and Y. Kim, "Saadi: A scalable accuracy approximate divider for dynamic energy-quality scaling," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19. New York, NY, USA: ACM, 2019, pp. 481–486. [Online]. Available: http://doi.acm.org/10.1145/3287624.3287668

[83] S. Venkatachalam, E. Adams, and S. Ko, "Design of approximate restoring dividers," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5.

[84] M. Imani, R. Garcia, A. Huang, and T. Rosing, "Cade: Configurable approximate divider for energy efficiency," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 586–589.

[85] H. Jiang, L. Liu, F. Lombardi, and J. Han, "Low-power unsigned divider and square root circuit designs using adaptive approximation," *IEEE Transactions on Computers*, pp. 1–1, 2019.

[86] K. Chen, L. Chen, P. Reviriego, and F. Lombardi, "Efficient implementations of reduced precision redundancy (rpr) multiply and accumulate (mac)," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 784–790, May 2019.

[87] G. A. Gillani, M. A. Hanif, B. Verstoep, S. H. Gerez, M. Shafique, and A. B. J. Kokkeler, "Macish: Designing approximate mac accelerators with internal-self-healing," *IEEE Access*, vol. 7, pp. 77 142–77 160, 2019.

[88] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, pp. 60:1–60:34, Aug. 2017. [Online]. Available: http://doi.acm.org/10.1145/3094124

[89] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2010, pp. 957–960.

[90] ——, "A new circuit simplification method for error tolerant applications," in *Design, Automation Test in Europe (DATE)*, March 2011, pp. 1–6.

[91] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: Systematic logic synthesis of approximate circuits," in *DAC Design Automation Conference 2012*, June 2012, pp. 796–801.

[92] 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 Conference Exhibition (DATE)*, March 2013, pp. 1367–1372.

[93] J. Miao, A. Gerstlauer, and M. Orshansky, "Approximate logic synthesis under general error magnitude and frequency constraints," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 779–786.

[94] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "Abacus: A technique for automated behavioral synthesis of approximate computing circuits,"

in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[95] K. Nepal, S. Hashemi, H. Tann, R. I. Bahar, and S. Reda, "Automated high-level generation of low-power approximate computing circuits," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 18–30, Jan 2019.

[96] J. Miao, A. Gerstlauer, and M. Orshansky, "Multi-level approximate logic synthesis under general error constraints," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2014, pp. 504–510.

[97] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan, "Aslan: Synthesis of approximate sequential circuits," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014.

[98] A. Yazdanbakhsh, D. Mahajan, B. Thwaites, J. Park, A. Nagendrakumar, S. Sethuraman, K. Ramkrishnan, N. Ravindran, R. Jariwala, A. Rahimi, H. Esmaeilzadeh, and K. Bazargan, "Axilog: Language support for approximate hardware design," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 812–817.

[99] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, June 2015.

[100] Chaofan Li, Wei Luo, S. S. Sapatnekar, and Jiang Hu, "Joint precision optimization and high level synthesis for approximate computing," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[101] M. Soeken, D. Große, A. Chandrasekharan, and R. Drechsler, "Bdd minimization for approximate computing," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2016, pp. 474–479.

[102] A. Chandrasekharan, M. Soeken, D. Große, and R. Drechsler, "Approximation-aware rewriting of aigs for error tolerant applications," in *Proceedings of the 35th International Conference on Computer-Aided Design*, ser. ICCAD '16. New York, NY, USA: ACM, 2016, pp. 83:1–83:8. [Online]. Available: http://doi.acm.org/10.1145/2966986.2967003

[103] Y. Wu and W. Qian, "An efficient method for multi-level approximate logic synthesis under error rate constraint," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6.

[104] L. Holík, O. Lengál, A. Rogalewicz, L. Sekanina, Z. Vašíček, and T. Vojnar, "Towards formal relaxed equivalence checking in approximate computing methodology," pp. 1–6, 2016.

[105] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, "Towards digital circuit approximation by exploiting fault simulation," in *2017 IEEE East-West Design Test Symposium (EWDTS)*, Sep. 2017, pp. 1–7.

[106] Z. Vasicek, V. Mrazek, and L. Sekanina, "Evolutionary functional approximation of circuits implemented into fpgas," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec 2016, pp. 1–8.

[107] ——, "Automated circuit approximation method driven by data distribution," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 96–101.

[108] S. Hashemi and S. Reda, "Generalized matrix factorization techniques for approximate logic synthesis," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 1289–1292.

[109] Y. Wu and W. Qian, "Alfans: Multi-level approximate logic synthesis framework by approximate node simplification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019.

[110] S. Su, C. Zou, W. Kong, J. Han, and W. Qian, "A novel heuristic search method for two-level approximate logic synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019.

[111] R. S. Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger, "General-purpose code acceleration with limited-precision analog computation," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, June 2014, pp. 505–516.

[112] B. Li, P. Gu, Y. Shan, Y. Wang, Y. Chen, and H. Yang, "Rram-based analog approximate computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 1905–1917, Dec 2015.

[113] S. Froehlich, D. Große, and R. Drechsler, "One method - all error-metrics: A three-stage approach for error-metric evaluation in approximate computing," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 284–287.

[114] A. Chandrasekharan, M. Soeken, D. Große, and R. Drechsler, "Precise error determination of approximated components in sequential circuits with model checking," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6.

[115] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2011, pp. 667–673.

[116] I. Polian, "Test and reliability challenges for approximate circuitry," *IEEE Embedded Systems Letters*, vol. 10, no. 1, pp. 26–29, March 2018.

[117] L. Anghel, M. Benabdenbi, A. Bosio, M. Traiola, and E. I. Vatajelu, "Test and reliability in approximate computing," *Journal of Electronic Testing*, vol. 34, no. 4, pp. 375–387, Aug 2018.

[118] A. Chandrasekharan, D. Große, and R. Drechsler, *Design Automation Techniques for Approximation Circuits: Verification, Synthesis and Test*. Springer, 2019.

[119] G. Gielen, P. D. Wit, E. Maricau, J. Loeckx, J. Martin-Martinez, B. Kaczer, G. Groeseneken, R. Rodriguez, and M. Nafria, "Emerging yield and reliability challenges in nanometer cmos technologies," in *Design, Automation and Test in Europe (DATE)*, March 2008, pp. 1322–1327.

[120] M. L. Bushnell and V. D. Agarwal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*, 01 2000.

[121] A. Chandrasekharan, S. Eggersglüß, D. Große, and R. Drechsler, "Approximation-aware testing for approximate circuits," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2018, pp. 239–244.

[122] Z. Jiang and S. K. Gupta, "An atpg for threshold testing: obtaining acceptable yield in future processes," in *Proceedings. International Test Conference*, 2002, pp. 824–833.

[123] S. Sindia and V. D. Agrawal, "Tailoring tests for functional binning of integrated circuits," in *2012 IEEE 21st Asian Test Symposium*, Nov 2012, pp. 95–100.

[124] A. Gebregiorgis and M. B. Tahoori, "Test pattern generation for approximate circuits based on boolean satisfiability," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 1028–1033.

[125] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, "Testing approximate digital circuits: Challenges and opportunities," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, March 2018, pp. 1–6.

[126] T. U. Aoki Laboratory. (2016). [Online]. Available: http://www.aoki.ecei.tohoku.ac.jp/arith

[127] L. Amarú, P.-E. Gaillardon, and G. D. Micheli. (2015) The epfl combinational benchmark suite. [Online]. Available: http://infoscience.epfl.ch/record/207551

[128] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the iscas-85 benchmarks: a case study in reverse engineering," *IEEE Design Test of Computers*, vol. 16, no. 3, pp. 72–80, July 1999.

[129] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, "Investigation of mean-error metrics for testing approximate integrated circuits," in *2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2018, pp. 1–6.

[130] Tetramax. [Online]. Available: https://www.synopsys.com/

[131] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, "A test pattern generation technique for approximate circuits based on an ilp-formulated pattern selection procedure," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 849–857, 2019.

Marcello Traiola received the Ph.D. degree in Computer Engineering from the University of Montpellier, France, in 2019. He received the Master's Degree in Computer Engineering cum laude in 2016, from the University of Naples Federico II, Italy. After three years at LIRMM (Laboratory of Computer Science, Robotics and Microelectronics of Montpellier), since February 2020 Marcello is a post-doctoral researcher at Lyon Institute of Nanotechnology (École centrale de Lyon), in France. His main research topics are emerging computing paradigms with special interest in testing and reliability. He is an IEEE member.

Arnaud Virazel received the Ph.D. degree in Microelectronics from the University of Montpellier, France, in 2001. He is currently Associate Professor at the University of Montpellier, and works in the Microelectronics Department of the LIRMM (Laboratory of Computer Science, Robotics and Microelectronics of Montpellier — France). His research interests span diverse disciplines, including DfT, BIST, diagnosis, reliability, delay testing, power-aware testing and memory testing. He is an IEEE member.

Patrick Girard received a Ph.D. degree in Microelectronics from the University of Montpellier, France, in 1992. He is currently Research Director at CNRS (French National Center for Scientific Research) and works in the Microelectronics Department of the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM) - France. His research interests include all aspects of digital testing and memory testing, with emphasis on critical constraints such as timing and power. Reliability and fault tolerance are also part of his research activities. Patrick Girard is a Fellow of the IEEE.

Mario Barbareschi received the Ph.D. in Computer and Automation Engineering in 2015 and the Master Degree in Computer Engineering cum laude in 2012, both from the University of Naples Federico II, Italy, where he is currently working a post-doctoral fellow. His research interests include Hardware Security and Trust, Cyber Physical Security, Approximate Computing and embedded systems based on the FPGA technology.

Alberto Bosio received the Ph.D. in Computer Engineering from Politecnico di Torino in Italy in 2006. Currently he is a full professor at the INL-École Centrale de Lyon in France. He published articles spanning diverse disciplines, including testing, verification, reliability, approximate computing and emerging technologies. He is an IEEE member.