Test and Reliability of Approximate Hardware

Marcello Traiola, Bastien Deveautour, Alberto Bosio, Patrick Girard, Arnaud Virazel

Test and Reliability of Approximate Hardware

Marcello Traiola, Bastien Deveautour, Arnaud Virazel, Patrick Girard, Alberto Bosio

Abstract The undeniable need of energy efficiency in today's devices is leading to the adoption of innovative computing paradigms - such as Approximate Computing. As this paradigm is gaining increasing interest, important challenges, as well as opportunities, arise concerning the dependability of those systems. This chapter will focus on test and reliability issues related to approximate hardware systems. It will cover problems and solutions concerning the impact of the approximation on hardware defect classification, test generation, and test application. Moreover, the impact of the approximation on the fault tolerance will be discussed, along with related design solutions to mitigate it.

1 Introduction and background

Since the early 1970s, the demand in electronic components and the necessity to push the limits of manufactured circuits for increased performance and transistor density has never stopped. Consequently, each new technology node suffers from reliability issues due to manufacturing defects, variability, interference, and wear-out. These well-known drawbacks lead to the occurrence of faults that can finally cause system failures in Integrated Circuits (ICs). Several approaches, such as test and fault tolerance, allow effectively improving the reliability of ICs, making them to work as intended.

Marcello Traiola, Bastien Deveautour, Alberto Bosio
University of Lyon, Ecole Centrale Lyon, INSA Lyon, CNRS, UCBL, CPE Lyon, Lyon Institute of Nanotechnology (INL), UMR5270, Ecully, France – e-mail: marcello.traiola@ec-lyon.fr; bastien.deveautour@cpe.fr; alberto.bosio@ec-lyon.fr

Arnaud Virazel, Patrick Girard
Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM), University of Montpellier, French National Centre for Scientific Research (CNRS), 34000 Montpellier, France – e-mail: arnaud.virazel@lirmm.fr; patrick.girard@lirmm.fr

Approximate Computing (AxC) is a paradigm that is extensively used to improve the circuit efficiency, by intentionally degrading the computation result quality. From the hardware standpoint, AxC enables the creation of circuits whose output values may differ from the original circuit for a certain set of input values [45]. The degradation introduced through the approximation must be judicious, in order to preserve quality loss below a certain threshold. Depending on the application scenario and the related reliability figure, such quality threshold may be more or less stringent.

In this chapter, we review the issues related to the test and the fault-tolerance of approximate hardware, along with the existing solutions.

1.1 Reliability Issues in Nanometer Technology

The reliability of digital circuits and systems is kept high owing to several methods. These methods ensure that the designs perform their function under defined conditions and during their estimated lifespan. They cover different aspects related to the manufacturing and the in-field functioning of electronics. For instance, clean-rooms control impurities, industrial control systems achieve production consistency; burn-in and testing, before and after packaging, ensure the detection of design weaknesses and manufacturing defects after stressing the circuits. All these methods are necessary before introducing the semiconductors to the market, but they are not foolproof. Even though miniaturization offers many advantages, each new CMOS node faces reliability issues, as the trend is rapidly approaching the physical limits of operation and manufacturing [1].

Digital systems can experience failures during the three phases of their lifespan depicted in the bathtub curve in Figure 1 [39]. Early failures are labeled as infant mortality; random failures occur during the working life and wear-out failures happen

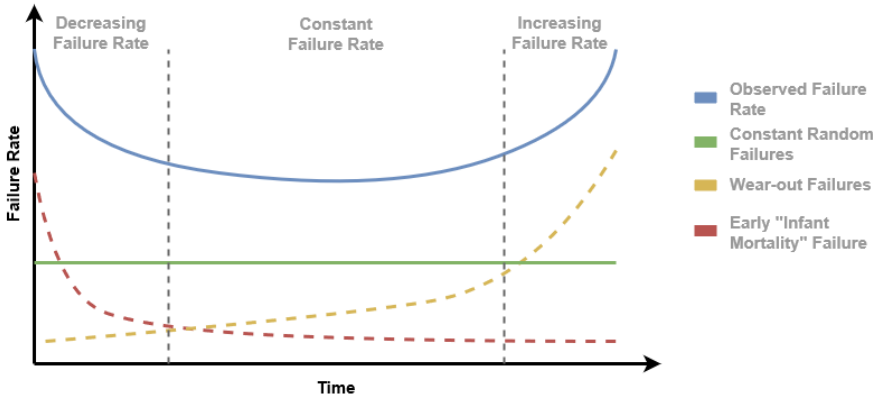


Fig. 1 The bathtub curve describing the failure rate of electronic circuits over their lifespan [39]

at the end of the circuit's lifespan. Different kinds of faults may occur to ICs, due to different phenomena. Faults can be classified according to their duration in three categories: *transient*, *intermittent* and *permanent*.

Transient faults randomly affect the correct functioning of the IC for a short time window. After this period, the device returns to a normal behavior. Variability and interferences are the main causes of transient faults.

Intermittent faults occur randomly like transient faults, but they never really disappear. In fact, their occurrence often precedes the occurrence of a permanent fault. Aging is the primary cause of intermittent faults.

Permanent faults are irreversible and are mostly due to manufacturing defects. They can also appear at the end of the circuit's lifetime due to extreme wear-out effects.

As mentioned above, faults may have different causes: manufacturing defects, variability, interferences and aging (wear-out).

Manufacturing defects and variability – Early failures during the infant mortality phase are mainly due to manufacturing imperfections, possibly leading to permanent defects in a chip. Moreover, the variability of transistor characteristics has always been an issue in IC design [83]. This phenomenon prevents the circuit from functioning correctly even though each individual transistor behaves correctly [51]. Furthermore, while the number of transistors doubles every 2-3 years, according to past microprocessor data, the die size remains relatively constant [33]. This scaling inevitably leads to a chip power density increase leading to inadequate heat sinking, thus to hot spots. In turn, this alters the timing characteristics of circuits [36]. Inevitably, the increasingly-complex logic of ICs entails the emergence of defects more and more hard to detect [59, 60]. Therefore, thoroughly testing the manufactured chips is crucial before releasing them to the market.

Interferences – Variability generated by manufacturing imperfections may generate unexpected circuit behaviors during operation. Furthermore, as transistors become smaller, their supply voltage (Vdd) decreases. These conditions are favorable for the occurrence of temporary effects, such as transient or intermittent faults. These temporary effects can be the result of electromagnetic influences, alpha-particle emission or cosmic radiations. They are responsible for the greatest part of digital malfunctions and more than 90% of the total maintenance costs are credited to them [57]. Internal interferences can also be a cause of temporary malfunctioning. With the scaling of components, the scaling of the interconnect line thickness (width and separation) must also follow. In these conditions, a high crosstalk noise is becoming a major issue due to larger capacitive couplings between interconnects in a polluted environment. Additionally, supply voltage scaling lowers the noise sensitivity threshold and increases the transient fault sensibility of new technology nodes due to high energy particle from environment or within the packaging.

Wear-Out – Although area scaling followed an exponential trend, supply voltage had a way slower scaling pace. There are two main reasons behind that: (i) the need to keep up with the competitive frequency growth; (ii) the need to retain the basic noise immunity and cell stability [66]. As a result, the discrepancy between area and voltage scaling leads to high power density and elevated temperatures. This, in turn, firstly causes modifications in the timing characteristics of the design and, eventually, wears out the chip's metal lines. Wear-out failures appear in-field after a certain period of use and limit both performances and lifetime of modern electronics [65]. This is especially critical for applications which demand high throughput (e.g. data centers) or for which technical support is expensive (e.g. space equipment).

When a fault propagates through the logic, it can be captured by a flip-flop (or memory cell) and stored as faulty value.
Depending on their nature, faults can become hard or soft errors. If the error reaches the service interface, this may cause a subsequent failure and alters the service [5].

Soft errors occur when particles, such as high-energy neutrons from cosmic rays or alpha particles generated from impurities in the packaging, strike a sensitive zone of the microelectronic device. This causes voltage glitches which propagate through combinational logic parts of the IC. Those events are referred to as *Single-Event Transients (SETs)* [6, 22, 15]. We refer to *Single-Event Upset (SEU)* as the phenomenon for which an SET propagates to some memory element of the circuit and its value is captured.

Hard errors are consequence of permanent silicon defects (due to manufacturing imperfections or to the wear-out). In the last decades, as transistor density increased, the likelihood of getting more hard errors in a given core kept increasing as well. In addition, the high frequencies increase the switching activity rate that accelerate material aging due to temperature and voltage stress [12].

1.2 Reliability Improvement Approaches

To improve the reliability, a thorough testing is crucial after the IC manufacturing and during the IC lifetime. Testing of digital circuits is performed thanks to binary test patterns applied to circuit's inputs [18]. If the test result and the expected one (or *golden*) do not match, the circuit is declared as *faulty*. Testing can be classified depending on the goal it is intended to serve:

Production testing – After chip manufacturing, the production testing determines whether the actual manufacturing process produced correct devices or not. This process is performed by the device manufacturer that owns full details about the internal structure of the manufactured system and usually exploits Automated Test Equipments (ATEs) for performing the tests.

In-field testing – Conversely, when the device is already in the field and under certain operative conditions, it requires to be tested during its normal operational life. Thus, a periodic *in-field* testing strategy is implemented. In this case, the test is carried out through the test mechanisms embedded in the device itself. Today, industrial standards – such as ISO26262, for automotive, and DO254, for avionics – provide the necessary guidelines to implement these test strategies in the context of different safety-critical applications.

Two types of tests are usually performed on VLSI chips:

Functional test – The test is performed through the device functional inputs and observing only the functional outputs. This leads to test the device functionality rather than the faults.

Structural test – The test is designed by taking into account the device's structural information, thanks to its netlist (i.e., the topological distribution of its logic gates). Very sophisticated algorithms, implemented within *Automatic Test Pattern Generators (ATPGs)*, exist to generate efficient structural tests.

The ATPG produces input sequences (referred to as *test set*) to efficiently detect possible faults in digital circuits. It is based on very advanced algorithms, such as the FAN [24]. Once the test set is available, by simulating it with the fault-free circuit, the fault-free output (or expected output) is obtained. In the test phase, if a fault occurs, the circuit outputs will be different from expected, when the test set is applied. In this way, we are able to detect the faults affecting the circuit.
The above reported concepts are not intended to be exhaustive; for an extensive dissertation about them, readers may refer to [37].

Furthermore, other reliability improvement practices can be adopted. They can be basically characterized into three categories, namely *fault avoidance*, *fault removal* and *fault tolerance* [32].

Fault avoidance – It aims at minimizing the sensibility of ICs to faults. To do so, specific tools and techniques assist the designers to specify, design and manufacture systems by addressing the source of the mechanisms causing the failures [62, 56, 88, 31].

Fault removal – It includes approaches whose function is to detect and eliminate existing faults during specification and design. Fault removal also refers to removing faulty components during production and operational phases [62].

Fault tolerance – It aims at guaranteeing the service provided by the product despite the presence or appearance of faults [32]. This chapter focuses specifically on fault tolerance.

While improving the manufacturing process and thoroughly testing the manufactured circuits certainly help to cope with permanent faults, they do not solve random

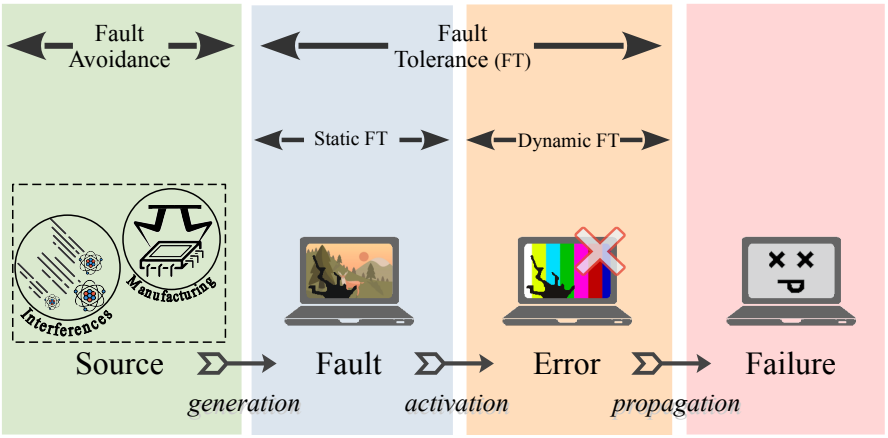


Fig. 2 Reliability improvement approaches across the fault-to-failure life cycle [10]

failures. Even the best efforts and investments to avoid or remove faults cannot prevent them from appearing in any operational system. However, as depicted in Figure 2, it is possible to prevent those faults by using hardware fault tolerance techniques. Some of them, such as *fault masking*, are classified as static, whereas some others, such as *Error Correcting codes*, as dynamic [10]. Some fault-tolerant designs are destined to resilient application, which are intrinsically able to provide and maintain an acceptable level of service despite faults occurring in the process. In these cases, the fault-tolerance mechanism aims at keeping the faults under an established level of impact. All the fault tolerance techniques are usually referred to as *redundancy*, a principle introduced by John von Neumann in 1950s [50]. The basic idea is to improve the system reliability by adding redundant information. Redundancy can be classified as *spatial*, *temporal* or *of data* according to Mathew et. al. [43].

Spatial Redundancy – It refers to techniques employing extra hardware to process the same information multiple times in parallel. Even if some hardware parts are affected by a transient or permanent fault, a logic voting scheme can produce a single correct output thanks to the redundant computations. On the other hand, the extra hardware resources used to improve reliability lead to area and power overheads [41].

Temporal Redundancy – It is based on repeating a computation or a transmission to detect (and sometimes correct) possible temporary errors [16]. In some cases, spending extra computation/transmission time (thus sacrificing performance) to tolerate faults is preferred to spatial redundancy, which implies additional area and power costs.

Data Redundancy – It is particularly employed in memory devices and it is based on detection and correction codes stored along with the data [64]. This extra information is generated from the original data to effectively identify the presence of one or more transient or permanent faults and possibly correct them.

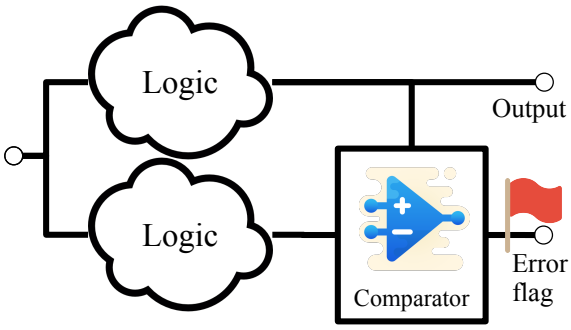


Fig. 3 Generic structure of the *duplication with comparison* architecture.

The detection of errors during the lifetime functional operation of a system is called *on-line detection* [29]. Examples of commonly used fault detection technique are *duplication with comparison* and *error detecting codes*. Error detecting codes are based on using redundant information to detect possible errors in stored/transmitted data. The data are guaranteed to be correct if some characteristics are respected, e.g. parity, checksum, Cyclic Redundancy Check (CRC). Those are generally computed from the data and compared to the expected ones, stored/transmitted along with the data. Error detecting codes can provide protection against SEUs and permanent faults [53] and are mainly used in memories, owing to their regular structure [17]. Error detection codes can be used in logic circuits to detect malfunctioning, however their efficiency relies on ad-hoc designs [29]. Duplication with comparison, sketched in Figure 3, employs two identical copies (at least functionally) of a system whose outputs are compared. A possible error affecting one of the copies is detected by comparing the two results. Its implementation simplicity and its ability to detect a wide variety of faults – permanent, transient and timing – are the main reasons of its popularity.

Once a fault has been detected, a proper method to recover from its effect is needed. The *recovery* consists in restoring the last error-free state of the system. This approach is referred to as *rollback*. The rollback consists in repeating the last operation(s) before the fault detection. Usually, architectures based on rollback detect faults through spatial redundancy and correct them by applying temporal redundancy. Periodical or occasional checkpoints save the state of the system; when an error is detected, these checkpoints are restored and the next operations are recomputed. The rollbacks can restore the system to several thousand previous states or simply to the previous cycle [44, 77]. Fault tolerance achieved through fault detection and rollback implies lower area and power costs than using spatial redundancy to prevent a fault from propagating and eventually causing an error. However, in case of a recovery, power and timing costs are proportional to the quantity of computing cycles that are recovered.

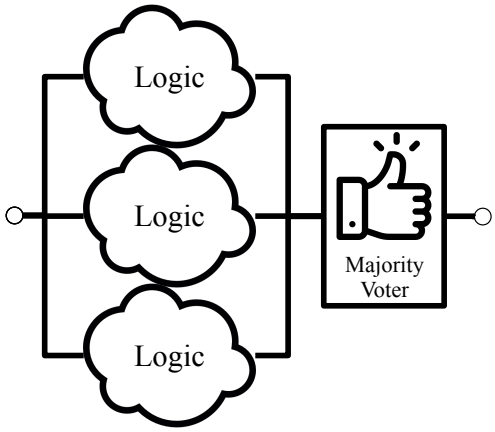


Fig. 4 Generic structure of the Triple Modular Redundancy architecture.

1.2.1 Fault-Tolerant Architectures

To deal with reliability issues in ICs, in the last decades, several fault-tolerant architectures have been proposed, such as Pair-and-A-Spare [48], Razor [19], Soft and Timing Error Mitigation (STEM) [4], Conjoined Pipeline (CPipe) [67], Triple Modular Redundancy (TMR) [40], and Dynamic Adaptive Redundant Architecture TMR [85]. Since, in this chapter, we focus on the generic TMR, in the following we briefly summarize its structure and functioning.

Triple Modular Redundancy (TMR) is one of the most popular fault-tolerant architecture, based on spatial redundancy. Its first application to computing systems dates back to 1962 [40]. Its generic structure is sketched in Figure 4. The TMR exists in different versions; the simplest is the one where three exact copies of the Combinational Logic (CL) of a circuit are connected to a voting circuit. The latter *masks* a possible faulty output that differs from the other two, by performing a majority vote. This configuration can mask single faults occurring in the CL. However, any fault located in the input or output register causes a system failure. In general, a fault occurring in any *unprotected* (i.e., non-triplicated) part of the circuit would result in a common-mode failure, thus undetectable for the approach. TMR variants triplicating the entire circuit – flip-flops included – and also the voter exist and mask any single fault in the circuit. On the other hand, the costs associated to the TMR structure are considerably high. Since the resources are triplicated, area and power overhead costs are at least 200%. The majority voter adds extra cost, also in terms of additional timing.

1.2.2 Selective Hardening Approaches

As discussed, all hardening methods rely on some variants of redundancy. Along the error recovery mechanisms, redundancy demands considerable resources to tol-

erate faults [35]. Moreover, often fault-tolerant solutions have limited application scenarios (e.g. ECC are primarily adapted for memory circuits). On the other hand, approaches most effectively dealing with a wide spectrum of failures often lead to massive hardware redundancy (e.g., TMR entails more than 200% area and power overheads [21]).

Researchers addressed this concern by proposing the *selective hardening* approach. The underlying idea is straightforward: if hardening the whole circuit is too resource-demanding, then only some chosen parts of the circuit are hardened. The choice of the parts needing hardening depends on two main factors, i.e. their particular exposure to failure and the criticality of their role in the system functioning. While the selective hardening greatly reduces overall error rate and the redundancy costs, the fault coverage decreases. Thus, the approach trades off some reliability to gain resources and tries to find "sweet spots" where the costs is greatly reduced and the system reliability is not too much degraded. Selective hardening techniques firstly perform a *vulnerability assessment* and then use fault-tolerance approaches to protect the most vulnerable parts. Examples of work in this direction can be found in [20, 46, 9, 52, 54, 42, 90].

1.3 Towards Approximate-Computing-Based Reliable Hardware

As reported in [26], the continuous scaling of CMOS technologies into the nanometer range has increased the effect of variability and degradation mechanisms on the yield and reliability of CMOS circuits and systems. In fact, the normal lifetime of miniaturized ICs is more and more reduced [30]. Approximate computing aims at transforming this problem into an opportunity [3]. The basic idea is to "*embrace*" errors as an intrinsic property of ICs and systematically design optimized approximate circuits functioning regardless of errors.

To achieve such a goal, on the one hand, test procedures have to be re-designed to be aware of the introduced approximation. On the other hand, the relaxation of non-critical computational constraints typical of the AxC [45] must be judicious – similarly to Selective Hardening – in order to keep a satisfactory reliability level. Therefore, we need to consider how AxC impacts on the role of testing and fault tolerance.

From a test perspective, the concept of *fault* changes. Indeed, the circuit is allowed to produce an *acceptable error*, by design. The maximum allowed error is defined by error metrics [38]. In the same way, during the test, the impact of a fault needs to be measured with such metrics. If the obtained measure is higher than the acceptable threshold, then the circuit has to be rejected, otherwise it has to pass the test. Thus, test procedures have a twofold role: (i) reject circuits whose error is greater than the threshold, and (ii) avoid rejecting *acceptably faulty* circuits. This ultimately leads to yield increase and possibly to the test cost reduction (i.e., the fewer the faults to test, the fewer the required test vectors). As a result of this consideration, test procedures have to be carefully redesigned.

Concerning the fault tolerance, the application of AxC techniques has to satisfy different constraints, depending on the system criticality. In detail, if the target system is composed of modules having different criticality, the approximation cannot be applied indiscriminately, rather it has to be adapted. The parts of the system that are less critical have to be firstly identified; then, selective – possibly approximation-based – hardening techniques can be applied. Previous works [78] proposed a very fast and low computationally-intensive method that helps to select the most sensitive parts of a logic design. This allows identifying the necessary hardening degree to fulfill the soft error reliability constraints and reduce the design cost, in terms of area and power [81]. Based on this analysis, called *structural susceptibility analysis*, a selective hardening technique using the *Hybrid Transient Fault-Tolerant (HyTFT)* architecture is proposed in [79]: by reducing the number of output nodes of the CL and comparing it with a full version of the circuit, this selective hardening approach not only reduces the size of the comparator but also significantly reduces the size of the duplicated CL copy in a vulnerability-aware manner. The use of the structural susceptibility analysis employed in the HyTFT architecture has proven to be more efficient in terms of area and power consumption with respect to a full duplication scheme. However, this analysis does not consider any error metrics as usually done in AxC (e.g. Worst-Case Error for error magnitude). In this context, fault-tolerant architectures bringing an AxC-based partial protection have been emerging. It is important to make sure that they are a good alternative to the classical selective hardening architectures and to be aware of the challenges that they raise. Finally, all of this begs the question: is it possible to apply AxC techniques also to the most critical parts of the system, despite the imprecise nature of AxC-based circuits?

While Section 3 addresses the approximation-based fault tolerance, next section discusses the aspects related to the test of approximate circuits.

2 Testing approximate hardware

The application of AxC in hardware leads to systems widely referred to as *Approximate Integrated Circuits (AxICs)*. An extensively used method to design those circuits is *functional approximation* [55]. This section specifically focuses on the test of functionally-approximate circuits. Unlike *approximate test* [80], where the constraints of the test process itself are relaxed, in this context we focus on how effectively test approximate circuits. As already mentioned, test techniques must be revisited to be applied to AxICs, since the approximation changes their functional behavior. As a matter of fact, extending to AxICs the conventional testing concepts is not trivial. In particular, in the context of AxICs, even if a fault leads to exhibit a different behavior than expected, it may still be acceptable, thus the AxIC should not be discarded. Mastering these mechanisms may lead to increase the production process yield and/or reduce the test cost.

This section reviews the *approximation-aware* test flow, that deals with such aspects. The flow is composed of three main steps: (i) fault classification, (ii) test

pattern generation, and (iii) test set application. Faults producing critical effects on the circuit behavior are divided from those producing acceptable effects, in the fault classification phase. Test stimuli covering all the critical faults and simultaneously leaving undetected as much acceptable faults as possible are produced in the *test pattern generation*. Finally, in the *test set application*, AxICs under test are classified as critically faulty – thus discarded – or as acceptably faulty, or fault-free – thus they pass the test. In turn, this reduces the number of AxICs discarded due to acceptable faults, thus it increases the manufacturing yield.

2.1 Approximation-Aware Fault Classification

Fault classification is the first step of the approximation-aware testing. It divides acceptable faults from critical ones [72, 71, 11, 25], under the single-fault assumption [37]. Moreover, the fault classification process produces the *expected yield increase* value of the approximation-aware test w.r.t. conventional test [76]. It is defined as follows:

$$\text{Expected yield increase} = \frac{\text{Acceptable faults}}{\text{Total faults}} \tag{1}$$

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

Measuring the output deviations of AxICs is a crucial task for a successful classification. Different error metrics have been proposed in the literature to measure arithmetic AxIC output deviations [38]. For instance, a widely accepted metric is the Worst Case Error (WCE), defined as follows:

$$\text{WCE} = \max_{\forall i \in \mathcal{I}} \left| O_i^{\text{approx}} - O_i^{\text{precise}} \right| \tag{2}$$

where:
$i \in \mathcal{I}$ is the input value within the set of all possible inputs $\mathcal{I}$,
$O_i^{precise}$ is the precise output integer representation, for input i,
O_i^{approx} is the approximate output integer representation, for input i.

The complexity of evaluating the impact of a fault depends on the considered error metric [70]. Metrics based on the calculation of a mean entail a higher computational effort to carry out the fault classification compared to other metrics. Indeed, calculating the mean error of an arithmetic circuit requires the error measure for all the possible circuit input values, which is a $O(2^n)$ complexity task (where n is the number of input bits).

Existing techniques to deal with fault classification are based on the idea of masking the effects of acceptable faults by using a filter [72, 70, 11, 25], as shown in Figure 5. In details, at design time, both the original precise netlist (i.e., not approximate) and the AxIC netlist are put together with an *evaluation module* to form

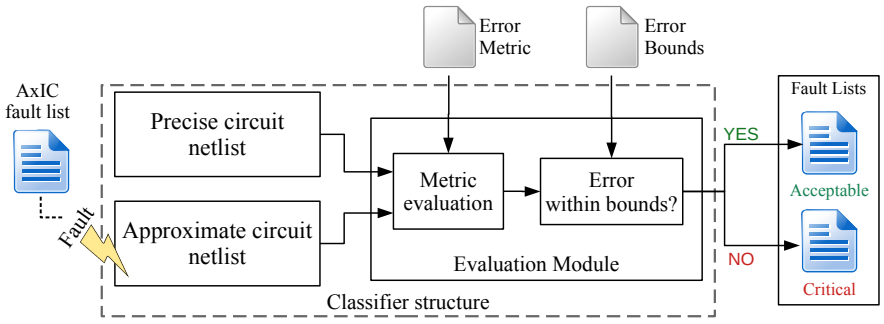


Fig. 5 Approximation-aware fault classification

a new netlist (*classifier structure*, in the figure). The evaluation module compares the two circuit outputs with respect to the chosen error metric(s). It produces an error only if the AxIC produces an output outside the defined error bounds. The netlist of the so-built classifier is elaborated with conventional test approaches, such as conventional ATPG approaches or a SAT-based ones, to classify the AxIC possible faults. The underlying idea is using the evaluation module to filter acceptable fault effects. Therefore, only critical faults generate an error condition. For instance, when using the ATPG, if the generation procedure finds a test for a given fault, this means that the fault is critical; conversely, no tests are produced for acceptable faults, since their effect is masked.

Figure 6 reports the results of the approximation-aware fault classification applied to approximate arithmetic circuits [34, 89, 87, 61, 47, 86], expressed in terms of average *expected yield increase* (Equation (1)). Results are extracted from [25, 11, 72] and the analyzed circuits are 8- and 16-bit adders [34, 89, 87, 61, 47], 8-, 16-, and 32-bit multipliers [47], single precision IEEE-754 standard floating point circuits [86], and fixed point multipliers and dividers [86]. We report the results by organizing the circuits in groups based on the allowed percentage error. The reference metric is the WCE, defined in Equation (2).

Adders and multipliers, on average, show an expected yield increase above 50%, except for 8-bit adders showing an average around 19% for circuits allowing WCE greater than 0% and lower than 10%. Fixed- and floating-point units show results ranging between 13% and 78%, on average. As foreseen, the expected yield increase gets higher when a larger error is allowed; this happens as the effect of more faults is masked, thus they are considered as acceptable. More details on the approaches are reported in [72, 70, 11, 25, 76].

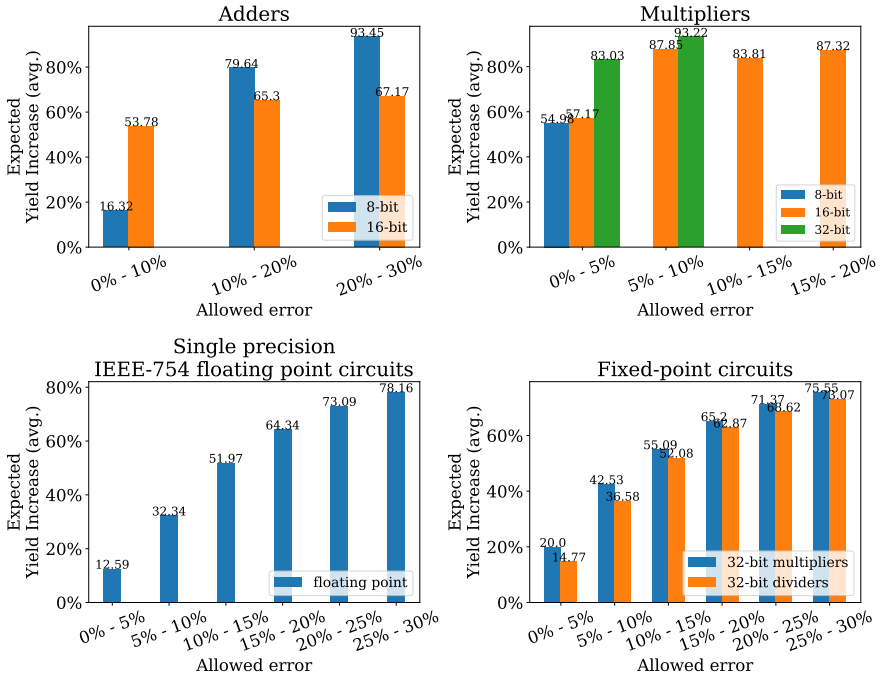


Fig. 6 Approximation-aware fault classification experimental results from [72, 11, 25]

2.2 Approximation-Aware Test Pattern Generation

Test pattern generation is the second step of the approximation-aware testing. In conventional test, the higher the *fault coverage* a test generation methodology is able to achieve, the more efficient it is considered to be. This slightly changes, when it comes to AxICs. Indeed, on the one hand, test patterns must detect all critical faults; on the other hand, they should detect as few as possible acceptable faults. In turns, this allows identifying and rejecting all AxICs affected by critical defects and, simultaneously, preventing the elimination of those affected by acceptable defects.

Thus, we have to revisit the *Fault Coverage (FC)* definition, by extending it into two subclasses: *acceptable FC* and *critical FC*, defined below:

$$\text{Acceptable FC} = \frac{\text{Acceptable faults detected}}{\text{Total acceptable faults}} \tag{3}$$

$$\text{Critical FC} = \frac{\text{Critical faults detected}}{\text{Total critical faults}} \tag{4}$$

In the AxIC context, an ideal test set should lead to 100% critical FC and 0% acceptable FC [74, 76].

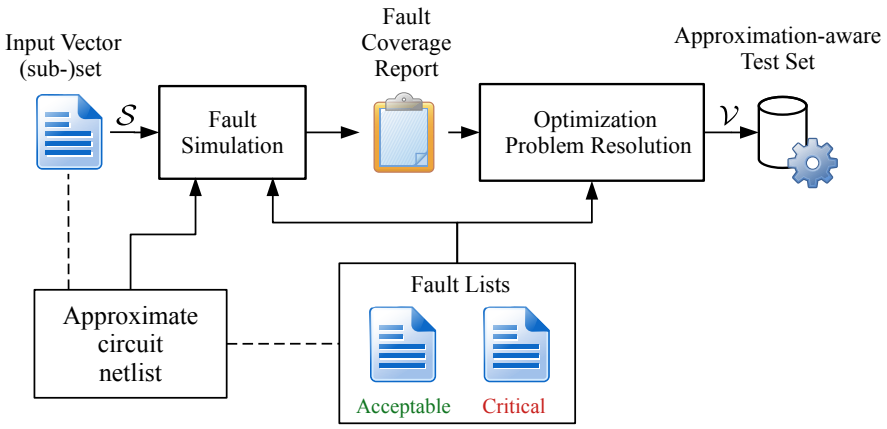


Fig. 7 Approximation-aware test generation

Efforts towards achieving such test set have been spent. In [74], a methodology searching for the smallest subset of tests detecting all the critical faults and minimizing the detected acceptable faults has been developed. It is sketched in Figure 7: (i) The input vector (sub-)set $\mathcal{S}$ is generated for the AxIC; (ii) the coverage of both AxIC's critical and acceptable faults is measured by using fault simulation (i.e., circuit simulation by applying the test set and with the fault introduced in the netlist); (iii) an *Integer Linear Programming (ILP)* optimization problem to find the smallest test subset detecting 100% critical faults FC and minimizing the detected acceptable faults is formulated; (iv) the optimization problem solution delivers the sought test set $\mathcal{V}$. The described technique guarantees to find the best possible vector combination among the input vectors in the set $\mathcal{S}$. Thus, if $\mathcal{S}$ is the exhaustive AxIC input set, the final test set will be the *global optimum*. The described approach is independent of the specific fault classification technique and of the chosen error metrics. The reader can refer to [74] for further details.

Figure 8 reports the results of the approximation-aware test pattern generation applied to approximate arithmetic circuits from [34, 89, 87, 61, 47], compared to the conventional generation (i.e., obtained with conventional ATPG tools). As shown, the conventional generation technique exhibits an average acceptable fault coverage between 65% and 92%. Significant lower (thus better) acceptable fault coverage values were achieved by using the approximation-aware technique from [74]. Indeed, acceptable fault coverage values between 33% and 76% were achieved. Furthermore, the average number of necessary tests is reduced when using the approximation-aware generation, thus reducing the necessary time to test the circuit. However, concerning the execution time, the approximation-aware test pattern generation technique entails a overhead due to the ILP problem intrinsic complexity.

Finally, due to the internal structure of the circuits, the ideal outcomes (i.e., 100% covered critical faults and 0% covered acceptable faults) are far from being achieved. Thus, in the last phase of the test, a methodology to distinguish acceptable

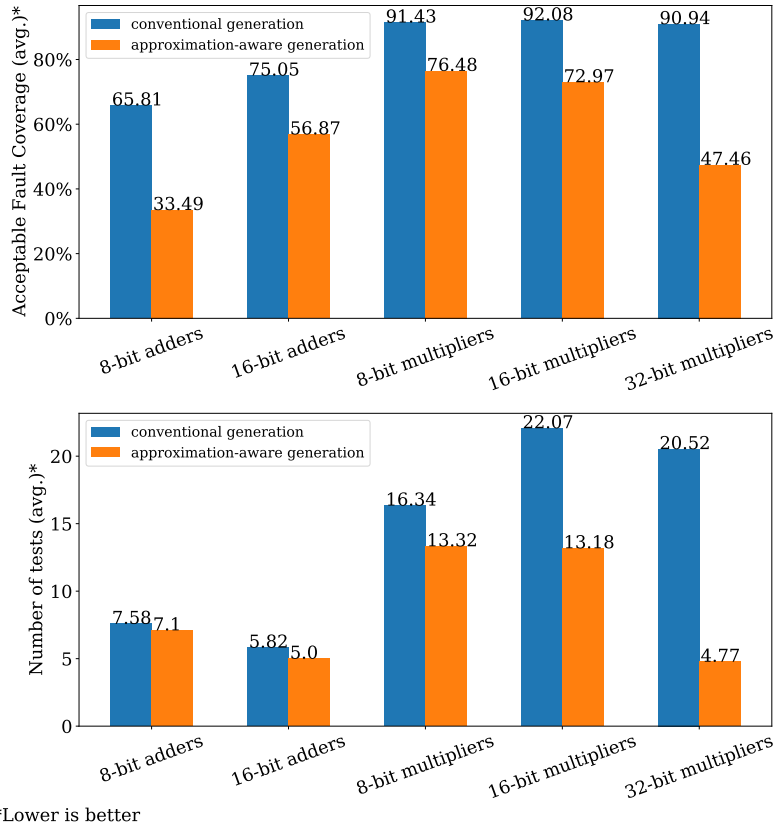


Fig. 8 Approximation-aware test pattern generation experimental results from [74, 76]

from critical fault, after the test application, is needed. We show this in the next subsection.

2.3 Approximation-Aware Test Set Application

Test pattern application is the last step of approximation-aware testing. When AxICs are involved, observing a different response than expected not always means that the circuit has to be rejected. Indeed, if a fault causing the issue was acceptable (i.e., its effect are within error bounds) the AxIC has to pass the test. Unfortunately, as shown in the previous subsection, often tests built to detect critical faults also detect acceptable ones. Thus, a methodology to determine whether the detected fault is acceptable or not is needed.

The well-know *signature analysis* concept [23] can help in this context. Briefly, it compacts test responses of a Circuit Under Test (CUT) into a *signature* and compares

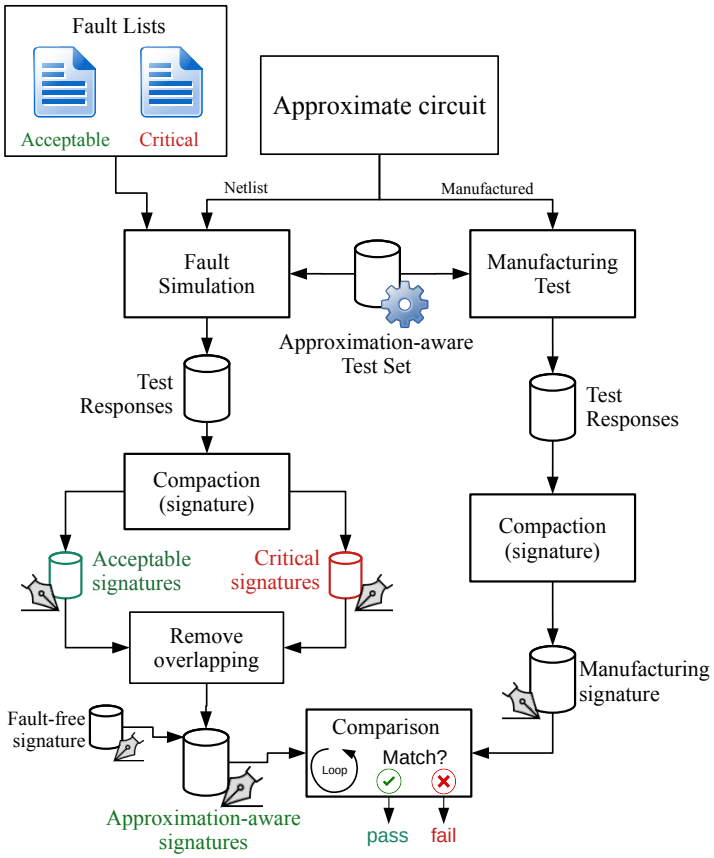


Fig. 9 Approximation-aware test application

it with the expected one (produced by the fault-free circuit). If they match, the CUT passes the test. In [75], this concept is adapted to approximation aware test, as depicted in Figure 9. It is divided in two phases:

1. At design time (left side, in the figure), test patterns are *fault-simulated* with the AxIC's fault lists. For each fault, a signature is generated. If any, acceptable signatures overlapping with critical ones are removed. The golden signature (i.e., fault-free) is added to the acceptable ones to obtain the *approximation-aware signatures*.
2. At test time (right side in the figure), the test responses of the manufactured AxIC generate a signature; all the approximation-aware signatures are compared with it; if at least one match is obtained, then the AxIC passes the test.

Results obtained with the discussed technique are reported in Figure 10, in terms of acceptable fault coverage. In general, the technique delivered 100% critical fault

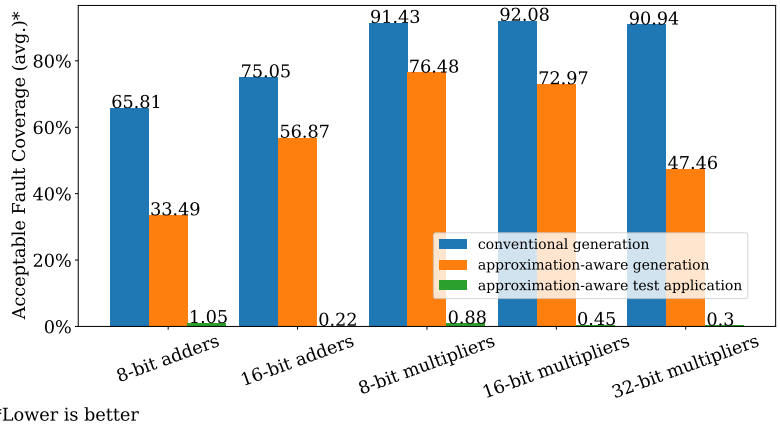


Fig. 10 Approximation-aware test set application experimental results from [75, 69]

coverage and 0.58% acceptable fault coverage on average, which are very close to the ideal ones. For further details, the reader can refer to [75].

3 Fault-Tolerant Architectures Based on Approximate Computing

AxC has already been used in the literature in the context of fault-tolerant architectures. In [27] and [58], the authors presented the *Approximate TMR (ATMR)* and its extension, the *Full ATMR (FATMR)*. Just as the TMR, the ATMR scheme employs three CL copies, two AxICs and a precise one, and the FATMR uses three AxICs. In these architectures, only one AxIC delivers an erroneous response for a given input vector. The idea is that each approximate module has its own unique *domain of approximation*. Since the three modules together always deliver at least two correct outputs, the (F)ATMR can mask any approximate responses coming from one of the AxICs. However, in case of a fault, the so-designed architecture can only protect the circuit for a set of input vectors defined by the designer. Other proposals of a low cost TMR based on approximate computing were presented in [63] and more recently in [68] and [28, 27]. Finally, authors in [2] show the interest of AxC for fault tolerance in arithmetic circuits. They proposed a configurable-accuracy approximated adder embedding a correction technique. While effective, this solution is also workload-dependent.

3.1 Selective Hardening Based on Approximate Duplication

The selective hardening philosophy, briefly discussed in Section 1.2.2, aims at minimizing the cost entailed by fault-tolerant architectures while trying to minimize the related reliability loss. As previously mentioned, AxC goes in a similar direction, aiming at minimizing the logic area cost at the expense of precision in the computation. This makes interesting to employ an AxIC as redundant module in a *duplication and comparison* scheme (see Figure 3) and assess its contribution to the trade-off between reliability and cost. Therefore, in this section, we study the trade-off between reliability and cost of the selective hardening technique proposed in [79], and briefly discussed in section 1.3, by comparing different duplication approaches. In details, we explore four duplication scenarios :

1. a full duplication scheme,
2. a reduced duplication scheme based on the structural susceptibility analysis presented in [81],
3. a reduced duplication scheme based on the logical weights of the arithmetic circuit outputs, and
4. a reduced duplication scheme based on an approximate circuits from a public benchmark suite [47], composed of arithmetic AxICs.

Most of the conceived AxICs are arithmetic circuits, as their precision loss is easily measurable [38]. In light of this, we can assess the precision reduction of the duplication schemes with metrics commonly used for arithmetic circuits. Among the common ones, we resort to the Worst-Case Error (WCE) metric (Equation (2)). Finally, the four considered scenarios are workload-independent.

Next subsection introduces the *Structural Susceptibility Analysis*, then we describe the different considered duplication scenarios.

3.1.1 Structural Susceptibility Analysis

The structural susceptibility analysis methodology proposed in [81] is based on the fact that not all outputs of a CL block have the same susceptibility to Single Event Transient (SET) effects and assumes that their susceptibility is a function of the number of nodes in their fan-in logic cone. It exploits the structural properties of the output fan-in cone to get their relative susceptibility estimates. The outputs are ranked on the basis of their relative susceptibility and the most susceptible ones are selected for error detection. The susceptibility of each fan-in cone is calculated as the sum of all the logic cell weights in the corresponding fan-in cone. The cell weight is calculated as the number of inputs and outputs of that cell. Figure 11 sketches the structural susceptibility analysis application to a simple example circuit. The shaded regions mark the boundaries of the two output fan-in cones. For each gate, W_i indicates the respective weight. The sum of all the weights in the cone gives the preliminary fan-in cone susceptibility value (S_j). In this example, $S_1 = 14$ and $S_2 = 12$ are the preliminary fan-in cone susceptibility values for O_1 and O_2,

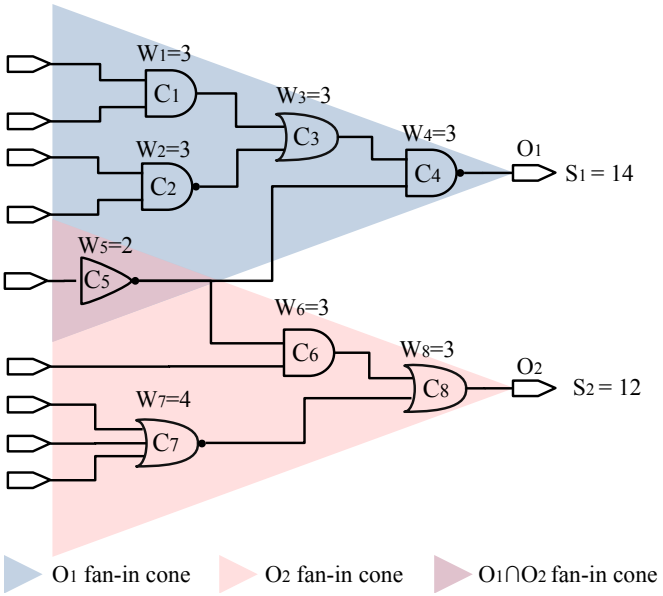


Fig. 11 Example of structural susceptibility analysis application to a simple circuit.

respectively. Thus, O_1 is more susceptible to propagate an error than O_2. In other words, providing an error detection mechanism on the output O_1 can better improve the reliability of the circuit, compared to having it placed on O_2. Finally, to produce the final susceptibility values, the technique assigns the weight of the shared cells (i.e., belonging to multiple output fan-in cones) only to the most susceptible output.

In order to provide an example of its effectiveness, we report in Figure 12 (taken from [14]) the comparison of the structural susceptibility analysis results and a fault injection experiment, for the b03 circuit from the ITC'99 benchmark suite. The red line represents the normalized distribution of the structural susceptibility (S_j) for

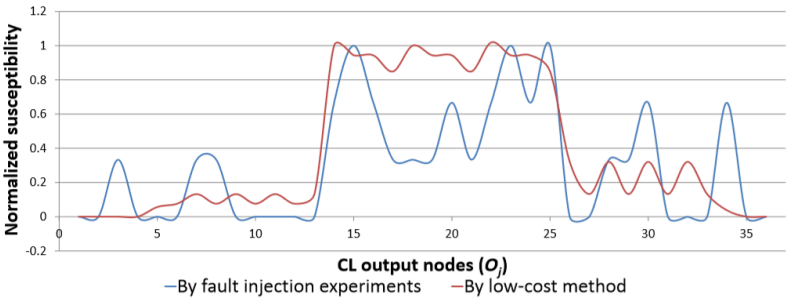


Fig. 12 Comparison of the structural susceptibility analysis and a fault injection experiment, for the b03 circuit from the ITC'99 benchmark suite. Taken from [14]

each circuit output; the blue line represents the distribution of the average number of soft error failures observed at the circuit outputs after a fault injection campaign. Further analyses and validations of the structural susceptibility analysis can be found in [81].

3.1.2 Selective Error Detection Architectures for Arithmetic Circuits

An error detection architecture must be capable of detecting transient, permanent and timing faults that may occur in an arithmetic circuit. The error detection scheme that we evaluate employs duplication and comparison to detect the occurrence of faults (see Figure 3). Since the architecture relies on duplication of the arithmetic logic and on a comparator, its conventional implementation incurs an overhead of more than 100% in terms of area and power. Cleverly selecting the functions to be duplicated is a practicable way to allow the designer to control the trade-off between the area/power overhead and the reliability improvement. Below, we briefly describe the different duplication scenarios that we compare.

Full duplication scheme – In this scenario, the architecture is able to detect all faults (transient, permanent and timing faults) that may occur in the arithmetic circuit. A full comparator circuit is used in this case.

Reduced duplication scheme based on structural susceptibility analysis – To obtain multiple circuits having different structural susceptibilities and overheads, we use the structural susceptibility analysis, described in Section 3.1.1. As illustrated in Figure 13, each copy is created by selecting a set of outputs ranked according to the analysis. In this scenario, the comparator is reduced since the obtained copies have fewer outputs. This duplication scheme is able to detect only faults affecting the common area between the original circuit and the reduced circuit. Consequently,

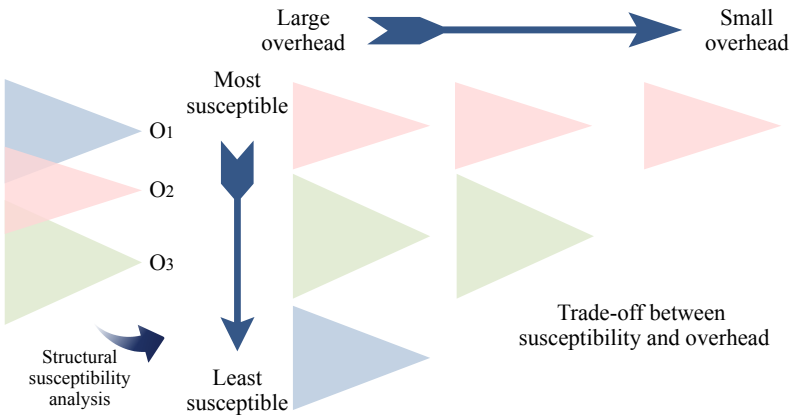


Fig. 13 Reduced duplication scheme based on structural susceptibility analysis

we have to assess the impact of the undetected faults by using the aforementioned error metrics. In this scenario, the WCE is defined as follows:

$$WCE = \sum_{i \in B} 2^i \tag{5}$$

where B indicates the set containing the positions in the original circuit of the outputs that are truncated in the reduced circuit. For example, if the removed outputs are in position 0 and 2 in the original circuit, then the WCE will be $2^0 + 2^2 = 5$.

Reduced duplication scheme based on output logic weights – In this scenario, we consider the possibility to duplicate the arithmetic circuit by using a functional metric. Indeed, we consider that the outputs of the arithmetic circuit can be ranked from Least Significant Bit (LSB) to Most Significant Bit (MSB). This partial duplication scheme may be considered as an Unequal Error Protection (UEP) scheme [7]. As shown in Figure 14, the reduced arithmetic circuits are obtained by eliminating fan-in cones from the one driving the LSB up to the one driving the MSB. In this case, the reduced circuit entailing the smallest overhead corresponds to the logic cone driving the MSB output. Also in this case, we assess the impact of the undetected faults by calculating the error metric (in this case, the WCE in Equation (5)).

Reduced duplication scheme based on approximate circuits – This scenario consists in using as reduced circuit an approximate arithmetic circuit from the public benchmark suite in [47]. The approximate version is selected based on its reduced area and timing properties, compared to the original circuit. In this scenario, the comparator provides an error signal when the precise arithmetic circuit produces a response having a difference w.r.t. the AxC version larger than a selected WCE value. It is worth noting that, in this scenario the reduced circuit has the same number of outputs as the original circuit. More details on the design of such a comparator can be found in Section 2.1 (referred to as *evaluation module*) and in [73]. For this scenario, WCE is calculated by using Equation (2).

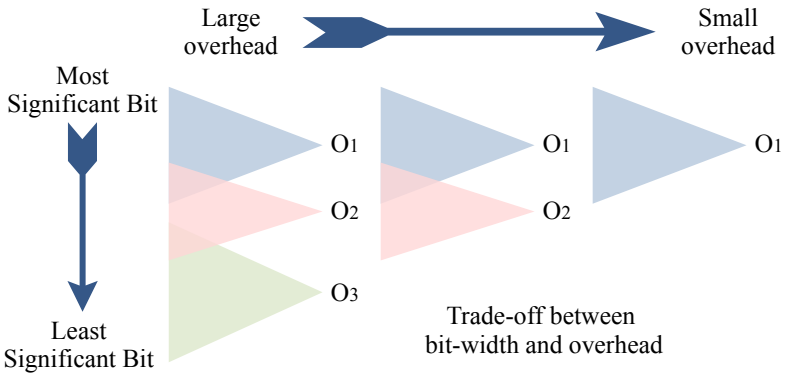


Fig. 14 Reduced duplication scheme based on output logic weights

3.1.3 Comparisons

The four duplication scenarios are compared by using four arithmetic circuits: an 8-bits carry look-ahead adder, an 8-bits carry look-ahead multiplier, a 12-bits array multiplier and a 16-bits array multiplier. For these four arithmetic circuits more than 1100 AxC versions from the public benchmark suite in [47] have been considered. The AxC versions were selected by considering equal or lower area and equal or shorter critical path w.r.t. the precise version. To compare the different scenarios, we present the results in terms of area and power consumption overhead (obtained with the NanGate 45nm Open Cell Library [49]) with respect to the full duplication scenario, as well as WCE metric values. The results are reported in Figure 15.

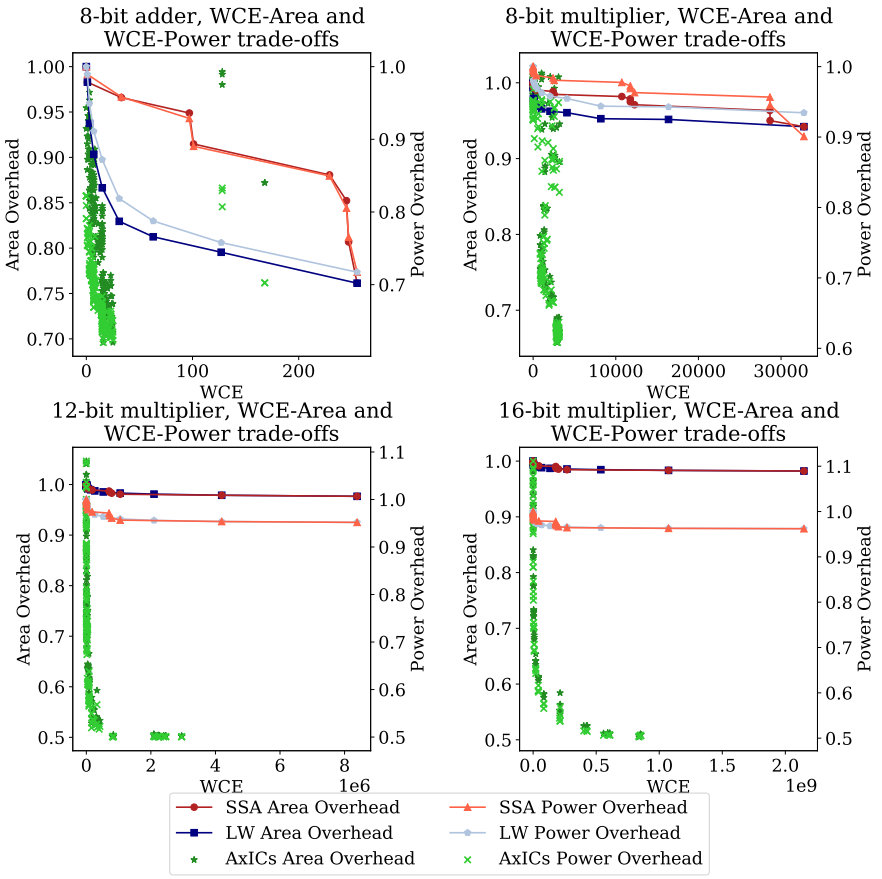


Fig. 15 Experimental results comparing the four scenarios: reduced duplication based on the structural susceptibility analysis (SSA); reduced duplication based on the logical weights of the arithmetic circuit outputs (LW); reduced duplication scheme based on an approximate arithmetic circuits from [47] (AxICs)

Firstly, they show that, in terms of trade-off between the WCE and the area/power overhead, using the reduced duplication based on the logical weights of the arithmetic circuit outputs (LW in the graphs) leads to better results (i.e., lower area for a given WCE) compared to the structural susceptibility analysis (SSA in the graphs). More importantly, when the reduced duplication based on approximate arithmetic circuits is used, an even better trade-off is achieved.

Experimental results demonstrate the interest of using approximate structures as duplication scheme since both area overhead and power consumption are reduced compared to a full duplication scheme, while maintaining good levels on error metrics. The reader can refer to [14] for a more extensive discussion and additional experimental results.

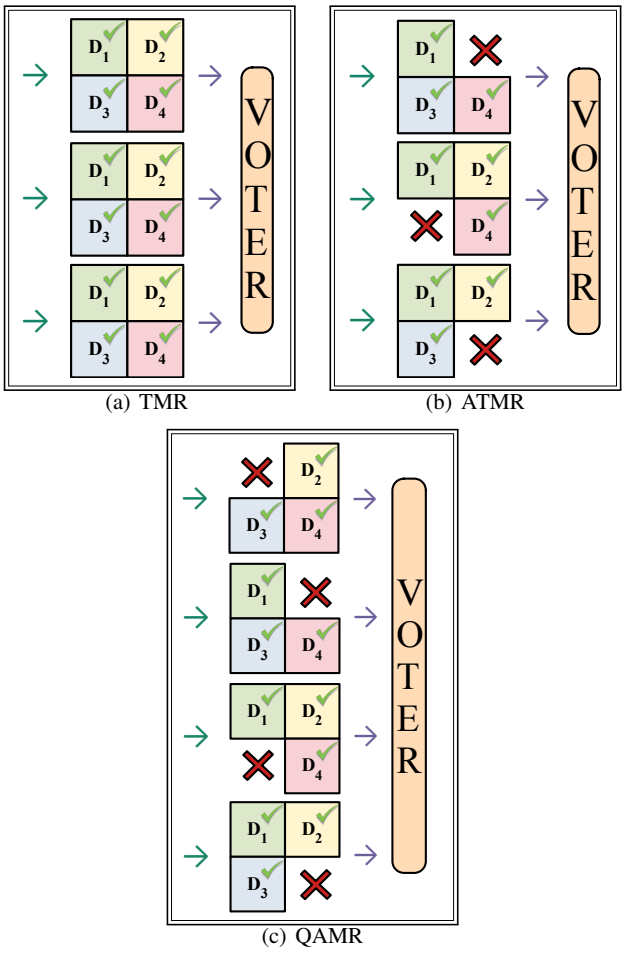
3.2 Ensuring Fault Tolerance Through Approximate Redundancy

During the lifespan of a system used in harsh (e.g. radiative) environment, its hardware is subject to various physical phenomena that may alter its performance or provoke errors [82]. Moreover, some systems demand a high level of reliability since failures would imply catastrophic outcomes. Aerospace systems, submarine telecom or even medical instruments cannot risk particle strikes, wear-out or aging. However, high levels of reliability usually require heavy fault tolerant designs to reach such high requirements. Several structures have been designed to maintain the accuracy of these safety-critical applications. A well-known existing structure capable of tolerating soft and hard errors is the Triple Modular Redundancy (TMR), briefly introduced in Section 1.2.1. A triplication of the circuit with a majority voter ensures logic error masking at a cost of a 200% area and power overhead. Indeed, a TMR masks (i.e., tolerates) permanent or transient faults occurring in one module – or in several modules, provided that they do not impact the same outputs if several modules are faulty – for any vector applied to its inputs. AxC philosophy may not seem compatible with safety-critical applications. AxC has been applied to applications where an approximate result is sufficient for their purpose [68]. While AxC was applied to TMR [28, 63, 68, 27] and led to reduced overheads, it entailed also a reduced error-masking capability. Unfortunately, this makes approximate TMR not suitable in safety-critical scenarios. To overcome the above issue, in [13] the Quadruple Approximate Modular Redundancy (QAMR) was introduced. QAMR is a novel scheme using approximate computing to ensure full logic masking (tolerance) of transient and permanent faults. As the TMR, the QAMR masks all single faults occurring in the logic replicas, delivering to the voter a majority of correct responses. It achieves the same fault tolerance as the TMR while still benefiting from the advantages of approximate computing. The QAMR uses four approximate circuit replicas. The essential condition to be respected is that the four approximated replicas are complementary, i.e., at any given time, they must produce at least three precise responses (i.e., non-approximated).

The next subsection presents the motivation behind the QAMR approach, along with its fundamentals, and a circuit approximation method adapted to its purpose.

3.2.1 The QAMR scheme

As already mentioned, several proposals (known as *ATMR*) have been made to reduce the TMR area overhead by using AxC [28, 63, 68, 27]. Unfortunately, the



(a) TMR (b) ATMR

(c) QAMR

Fig. 16 (a): three identical copies are used in TMR: all produce the correct results for all the function's domains (D1-D4) when no errors occur. — (b): in ATMR approach the three approximate modules produce exact results only for three out of four function's domains; some domains are not covered three times as in the TMR scheme. — (c) QAMR approach : four approximate modules cover each function's domain three times, thus obtaining the same coverage as in the TMR scheme.

ATMR suffers from severe limitations in term of reliability. Let us resort to Figure 16 to illustrate the above mentioned issue. Let f be a generic multi-output function, whose input domain D can be split into four sub-domains $D1, D2, D3, D4$. The conventional TMR approach uses three identical modules implementing f and a voting scheme to ensure fault tolerance in the case a module incurs some defective conditions. In such case, the defective module will produce incorrect or incomplete response, for some inputs. Thanks to the other two fault-free modules, the correct response can be produced. Figure 16(a) sketches the three system copies used in TMR approach. All three copies produce correct results for all the function's domains (D1-D4), when no errors occur. The ATMR approach, on the other hand, proposes to use three reduced (i.e., approximate) modules, as sketched in Figure 16(b). The three modules produce correct results only for some of the function's domains, when no errors occur. In the figure, two modules out of three produce the correct results for domains $D2, D3$ and $D4$. Using three approximate modules instead of the three original copies surely enables the opportunity to achieve efficiency gains, but also exposes the computation to some errors in case of faults. Defects impacting $D2, D3$ and $D4$ will not be tolerated. For instance, if a defect impacts the domain $D2$ of the second module, only the third module produces the right output and a correct vote might not be possible. To use the ATMR as fault-tolerant solution for safety-critical applications, only the protected parts of the system (i.e., $D1$ in the example is correct for all the modules) can be critical. Such design constraint can be challenging and not always achievable, even for resilient applications. Therefore, the ATMR is not suitable as fault-tolerant solution for safety-critical applications.

In the light of this, the goal of QAMR is to achieve the TMR reliability level while still profiting from approximate computing advantages. To show the principle of the QAMR scheme, we resort to Figure 16(c). The QAMR offers a complete coverage since the four approximate modules are realized so that, overall, the function's domains are covered three times, as in the TMR scheme. At the same time, the four AxICs enable the opportunity to achieve efficiency gains. The underlying insight is that a good AxC technique achieves more gains than it reduces the system's accuracy.

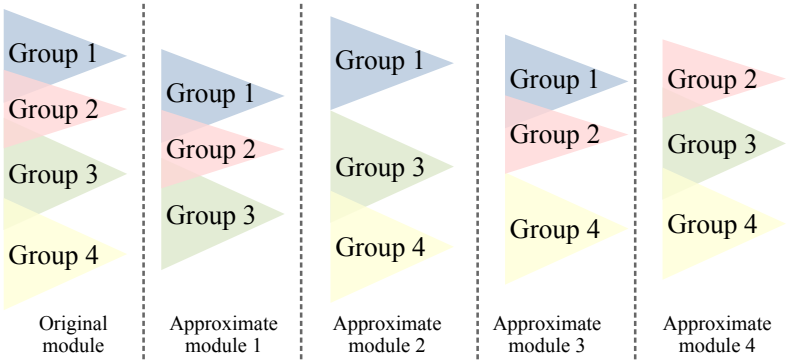


Fig. 17 QAMR approximation approach [13]

Figure 17 sketches the approximation approach proposed in [13] to realize the modules required by the QAMR architecture. For a given circuit, one group of outputs, along with their fan-in logic cone, is removed from the original circuit to obtain a first approximate module; this process is performed four times with different output groups to form four different AxICs. Removing a given output from *only one* of the four approximate replicas is fundamental for the QAMR to work as intended. By realizing the QAMR in this way, using the same voter as the TMR is possible. Indeed, for a given output, the voter still receives three replicas as input. For better clarity, Figure 18 shows an example of a 4-bits output circuit in the QAMR scheme. Since each AxIC has only one missing output, the voter is able to execute the majority vote just as in a classic TMR scheme.

With circuit having lots of outputs, exploring all possible ways of grouping their to-be-removed outputs is not feasible. For example, for a circuit having 245 outputs, the number of possible combinations is $\approx 10^{142}$. Therefore, in [13], the complementary groups of outputs to be removed were generated pseudo-randomly, in an iterative experimental campaign. Experiments have been carried out on combinational circuits from the public LGSynth'91 benchmark suite [84]. For each circuit, the conventional TMR (i.e., three precise replicas and the voter) was compared with the QAMR versions obtained in the experimental campaign (i.e., four approximate modules and the same voter). For further details on the experimental setup, the reader can refer to [13]. Figure 19 summarizes the results of the pseudo-random exploration. In the graphs, relative area gain (y-axis) and relative timing gain (x-axis) w.r.t. the original TMR are reported. The TMR is reported in the origin, symbol ★. Both area and timing gains are calculated as follows:

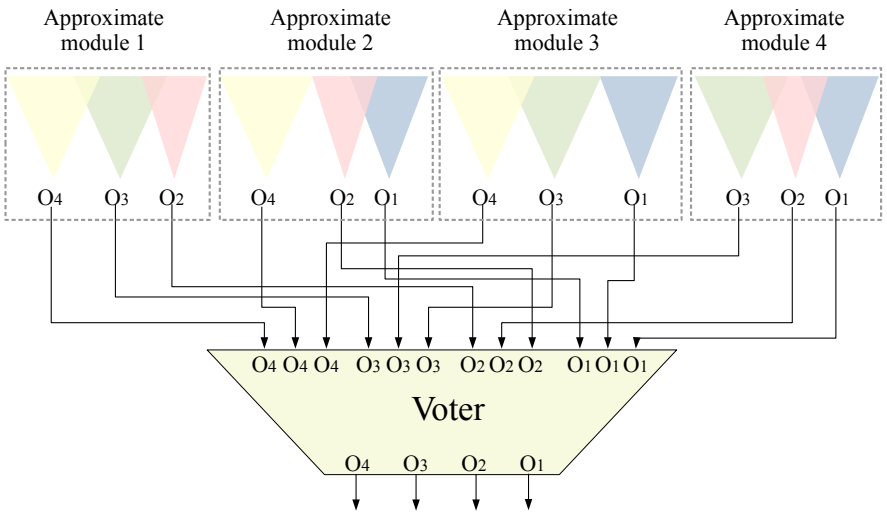


Fig. 18 QAMR example with conventional voting scheme

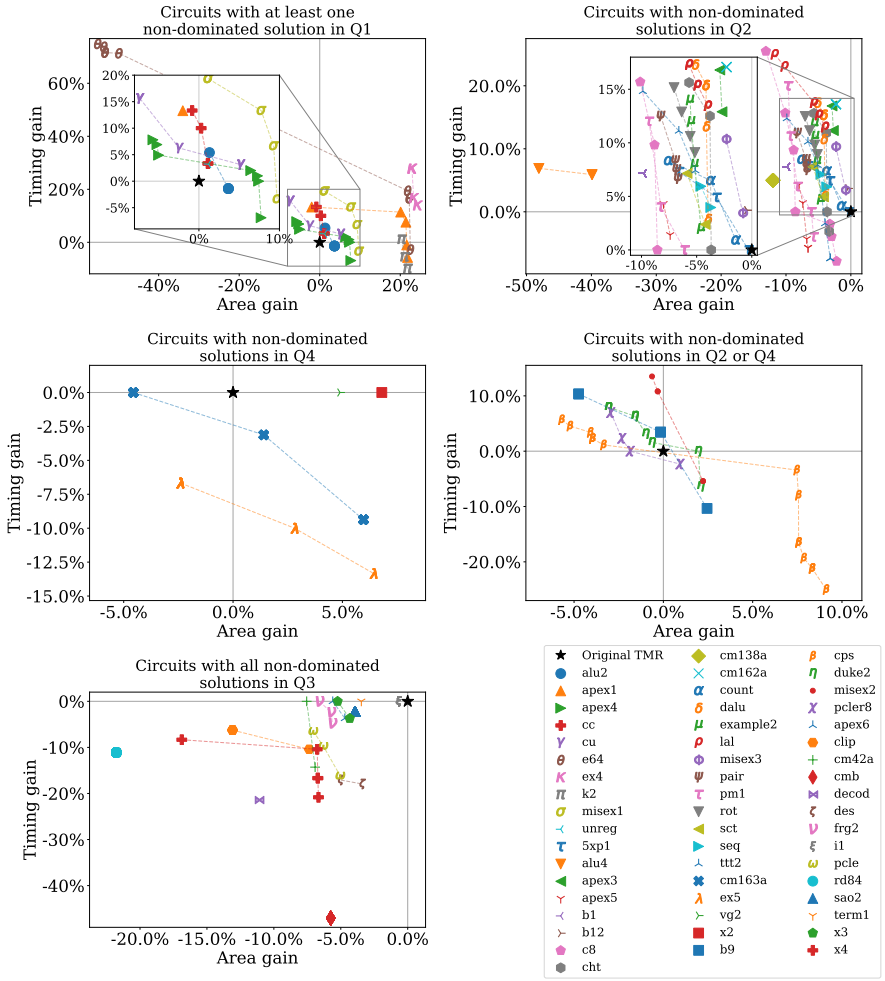


Fig. 19 Non-dominated solutions, in terms of relative area and timing gain, obtained with a pseudo-random exploration [13].

$$\alpha_{\text{gain}}\% = \frac{\alpha_{\text{TMR}} - \alpha_{\text{QAMR}}}{\alpha_{\text{TMR}}} \cdot 100, \tag{6}$$

where α represents either the area or the timing of the two architectures.

For each circuit, the graphs show the non-dominated solutions found with the pseudo-random exploration, i.e., the solutions having either better or at least equal area or delay w.r.t. each other solution in the set. They are organized as follows:

1. circuits presenting at least one non-dominated solution achieving gains in terms of **both area and delay** (*Q1* quadrant, $x > 0 \wedge y > 0$) are reported in the first graph;

2. circuits – that are not in Q1 – presenting at least one non-dominated solution achieving gains only in terms of delay (*Q2* quadrant, $x \leq 0 \wedge y > 0$) are reported in the second graph;
3. circuits – that are not in Q1 – presenting at least one non-dominated solution achieving gains only in terms of area (*Q4* quadrant, $x > 0 \wedge y \leq 0$) are reported in the third graph;
4. circuits – that are not in Q1 – presenting non-dominated solutions either in terms of delay or area are reported in the fourth graph;
5. circuits presenting all non-dominated solutions not achieving any gains (*Q3* quadrant, $x \leq 0 \wedge y \leq 0$) are reported in the last graph.

To help the reader, in the first and second graphs we added a $\approx$ 2X zoom of the densest parts. The graphs highlight that, while for some circuits the exploration did not find any solutions achieving gains (e.g. the *decod* circuit, symbol ⋈, quadrant Q3), in general it is possible to obtain superior QAMR implementations w.r.t. the original TMR, in terms of timing (e.g. *alu4*, symbol ▼, quadrant Q2), area (e.g. *x2*, symbol ■, quadrant Q4), or both (e.g. *ex4*, symbol *K*, quadrant Q1).

The presented data clearly indicate that QAMR offers a cheaper alternative to the standard TMR scheme for safety-critical applications. The results show that QAMR is feasible and demonstrate that there is a real interest in using AxC to realize more efficient fault-tolerant architectures for safety-critical applications. Since the exploration results are far from being optimal, as also shown in [8], further studies are needed to provide enhanced approximation techniques fully exploiting AxC opportunities in safety-critical scenarios.

4 Conclusion

Regardless of the field of application, i.e. trading, health-care, satellite telecommunications, civilian transports, military equipment, data centers, etc., there is an increasing demand of electronics able to perform complex and resource-intensive operations. Most of these operations demand a high degree of reliability, availability and safety. However, the increasing vulnerability of transistors and interconnects require electronic system designs to overcome reliability issues, which intensify for every new emergent CMOS technology generation. Furthermore, the complexity of modern electronic systems makes difficult to realize error detection, recovery, masking, etc. Moreover, also area and power limitations, as well as high performance demands, are compelling requirements to satisfy. These requirements force the industry to limit the overheads in reliability enhancements or come up with more adaptive designs that respond to the reliability problematic of the targeted field of application. Among the work of the last two decades, approximate computing brought multiple opportunities to different extents. The fundamental goal is to improve the system efficiency (time/area/energy) by relaxing result's accuracy requirements. This also brought along new challenges, concerning the reliability of electronic chips. In particular, in this chapter, we focused on test and fault tolerance issues related to

approximate hardware. We firstly showed how approximate computing changes the conventional concepts of digital circuit testing and reviewed the approximate-aware test flow, able to deal with such scenario. Then, we discussed how approximate computing can be used to reduce the cost of fault detection and fault tolerance mechanisms and showed how the state-of-the-art methodologies achieve this goal.

Finally, we deem interesting and valuable for the scientific community to move toward the study of new *Approximation-for-Reliability* principles, allowing the development of enhanced approximation methodologies that take into account also reliability aspects.

References

1. International Roadmap for Devices and Systems (IRDS™) 2020 Edition - IEEE IRDS™. URL https://irds.ieee.org/editions/2020
2. Al-Maaitah, K., Qiqieh, I., Soltan, A., Yakovlev, A.: Configurable-accuracy approximate adder design with light-weight fast convergence error recovery circuit. In: 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), pp. 1–6 (2017). DOI 10.1109/AEECT.2017.8257753
3. Anghel, L., Benabdenbi, M., Bosio, A., Traiola, M., Vatajelu, E.I.: Test and reliability in approximate computing. Journal of Electronic Testing **34**(4), 375–387 (2018). DOI 10.1007/s10836-018-5734-9
4. Avirneni, N.D.P., Somani, A.: Low overhead soft error mitigation techniques for high-performance and aggressive designs. IEEE Transactions on Computers **61**(4), 488–501 (2012). DOI 10.1109/TC.2011.31
5. Avizienis, A., U, V., Laprie, J.c., Randell, B.: Fundamental concepts of dependability (2001). URL http://www.cs.ncl.ac.uk/publications/trs/papers/739.pdf
6. Benedetto, J., Eaton, P., Avery, K., Mavis, D., Gadlage, M., Turflinger, T., Dodd, P., Vizkelethyd, G.: Heavy ion-induced digital single-event transients in deep submicron processes. IEEE Transactions on Nuclear Science **51**(6), 3480–3485 (2004). DOI 10.1109/TNS.2004.839173
7. Borade, S., Nakiboğlu, B., Zheng, L.: Unequal error protection: An information-theoretic perspective. IEEE Transactions on Information Theory **55**(12), 5511–5539 (2009). DOI 10.1109/TIT.2009.2032819
8. Bosio, A., O'Connor, I., Traiola, M., Echavarria, J., Teich, J., Hanif, M.A., Shafique, M., Hamdioui, S., Deveautour, B., Girard, P., Virazel, A., Bertels, K.: Emerging computing devices: Challenges and opportunities for test and reliability*. In: 2021 IEEE European Test Symposium (ETS), pp. 1–10 (2021). DOI 10.1109/ETS50041.2021.9465409
9. Bottoni, C., Coeffic, B., Daveau, J.M., Naviner, L., Roche, P.: Partial triplication of a SPARC-V8 microprocessor using fault injection. In: 2015 IEEE 6th Latin American Symposium on Circuits Systems (LASCAS), pp. 1–4 (2015). DOI 10.1109/LASCAS.2015.7250415
10. Castano, V., Schagaev, I.: Resilient Computer System Design. Springer International Publishing, Cham (2015). URL https://link.springer.com/book/10.1007/978-3-319-15069-7. OCLC: 1194524751
11. Chandrasekharan, A., Eggersglüß, S., Große, D., Drechsler, R.: Approximation-aware testing for approximate circuits. In: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 239–244 (2018). DOI 10.1109/ASPDAC.2018.8297312
12. Chen, C.C., Milor, L.: Microprocessor aging analysis and reliability modeling due to back-end wearout mechanisms. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **23**(10), 2065–2076 (2015). DOI 10.1109/TVLSI.2014.2357756

13. Deveautour, B., Traiola, M., Virazel, A., Girard, P.: Qamr: an approximation-based fully reliable tmr alternative for area overhead reduction. In: 2020 IEEE European Test Symposium (ETS), pp. 1–6 (2020). DOI 10.1109/ETS48528.2020.9131574
14. Deveautour, B., Virazel, A., Girard, P., Gherman, V.: On Using Approximate Computing to Build an Error Detection Scheme for Arithmetic Circuits. Journal of Electronic Testing **36**(1), 33–46 (2020). DOI 10.1007/s10836-020-05858-5. URL http://link.springer.com/10.1007/s10836-020-05858-5
15. Dodd, P., Shaneyfelt, M., Felix, J., Schwank, J.: Production and propagation of single-event transients in high-speed digital logic ics. IEEE Transactions on Nuclear Science **51**(6), 3278–3284 (2004). DOI 10.1109/TNS.2004.839172
16. Dubrova, E.: Fault-Tolerant Design. Springer New York, New York, NY (2013). DOI 10.1007/978-1-4614-2113-9. URL http://link.springer.com/10.1007/978-1-4614-2113-9
17. Dutta, A., Jas, A.: Combinational Logic Circuit Protection Using Customized Error Detecting and Correcting Codes. In: 9th International Symposium on Quality Electronic Design (isqed 2008), pp. 68–73 (2008). DOI 10.1109/ISQED.2008.4479700. ISSN: 1948-3295
18. Eldred, R.D.: Test routines based on symbolic logical statements. J. ACM **6**(1), 33–37 (1959). DOI 10.1145/320954.320957
19. Ernst, D., Kim, N.S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K., Mudge, T.: Razor: a low-power pipeline based on circuit-level timing speculation. In: Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36., pp. 7–18 (2003). DOI 10.1109/MICRO.2003.1253179
20. Fazeli, M., Ahmadian, S.N., Miremadi, S.G., Asadi, H., Tahoori, M.B.: Soft error rate estimation of digital circuits in the presence of Multiple Event Transients (METs). In: 2011 Design, Automation Test in Europe, pp. 1–6 (2011). DOI 10.1109/DATE.2011.5763020. ISSN: 1558-1101
21. Fazeli, M., Miremadi, S., Ejlali, A., Patooghy, A.: Low energy single event upset/single event transient-tolerant latch for deep subMicron technologies. IET Computers & Digital Techniques **3**(3), 289 (2009). DOI 10.1049/iet-cdt.2008.0099. URL https://digital-library.theiet.org/content/journals/10.1049/iet-cdt.2008.0099
22. Ferlet-Cavrois, V., Massengill, L.W., Gouker, P.: Single event transients in digital cmos—a review. IEEE Transactions on Nuclear Science **60**(3), 1767–1790 (2013). DOI 10.1109/TNS.2013.2255624
23. Frohwerk, R.A.: Signature analysis: a new digital field service method (1977)
24. Fujiwara, H.: FAN: A fanout-oriented test pattern generation algorithm. In: The IEEE International Symposium on Circuits and Systems (ISCAS) (1985). URL https://www.researchgate.net/publication/234044505_FAN_A_fanout-oriented_test_pattern_generation_algorithm
25. Gebregiorgis, A., Tahoori, M.B.: Test pattern generation for approximate circuits based on boolean satisfiability. In: 2019 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1028–1033 (2019). DOI 10.23919/DATE.2019.8714898
26. Gielen, G., Wit, P.D., Maricau, E., Loeckx, J., Martin-Martinez, J., Kaczer, B., Groeseneken, G., Rodriguez, R., Nafria, M.: Emerging yield and reliability challenges in nanometer cmos technologies. In: Design, Automation and Test in Europe (DATE), pp. 1322–1327 (2008). DOI 10.1109/DATE.2008.4484862
27. Gomes, I.A., Martins, M.G., Reis, A.I., Kastensmidt, F.L.: Exploring the use of approximate tmr to mask transient faults in logic with low area overhead. Microelectronics Reliability **55**(9), 2072–2076 (2015). DOI https://doi.org/10.1016/j.microrel.2015.06.125. URL https://www.sciencedirect.com/science/article/pii/S0026271415300676. Proceedings of the 26th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis
28. Gomes, I.A.C., Martins, M., Reis, A., Kastensmidt, F.L.: Using only redundant modules with approximate logic to reduce drastically area overhead in tmr. In: 2015 16th Latin-American Test Symposium (LATS), pp. 1–6 (2015). DOI 10.1109/LATW.2015.7102522
29. Göessel, M., Ocheretny, V., Sogomonyan, E., Marienfeld, D.: New Methods of Concurrent Checking, *Frontiers In Electronic Testing*, vol. 42. Springer Netherlands, Dordrecht (2008). DOI 10.1007/978-1-4020-8420-1. URL http://link.springer.com/10.1007/978-1-4020-8420-1

30. Hamdioui, S.: Electronics and computing in nano-era: The good, the bad and the challenging. In: 2015 10th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS), pp. 1–1 (2015). DOI 10.1109/DTIS.2015.7127342
31. Hareland, S., Maiz, J., Alavi, M., Mistry, K., Walsta, S., Dai, C.: Impact of cmos process scaling and soi on the soft error rates of logic processes. In: 2001 Symposium on VLSI Technology. Digest of Technical Papers (IEEE Cat. No.01 CH37184), pp. 73–74 (2001). DOI 10.1109/VLSIT.2001.934953
32. Heimerdinger, W., Weinstock, C.: A conceptual framework for system fault tolerance. Tech. Rep. CMU/SEI-92-TR-033, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (1992). URL http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11747
33. Huang, W., Stan, M.R., Gurumurthi, S., Ribando, R.J., Skadron, K.: Interaction of scaling trends in processor architecture and cooling. In: 2010 26th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), pp. 198–204 (2010). DOI 10.1109/STHERM.2010.5444290
34. Kahng, A.B., Kang, S.: Accuracy-configurable adder for approximate arithmetic designs. In: DAC Design Automation Conference 2012, pp. 820–825 (2012). DOI 10.1145/2228360.2228509
35. Koren, I., Krishna, C.M.: Fault-Tolerant Systems. Morgan Kaufmann, San Francisco (CA) (2021). DOI 10.1016/C2018-0-02160-X. URL www.sciencedirect.com/book/9780128181058/fault-tolerant-systems
36. Kumar, R.: Temperature Adaptive and Variation Tolerant CMOS Circuits. University of Wisconsin–Madison (2008)
37. L. Bushnell, M., D. Agrawal, V.: Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits (2000). DOI 10.1007/b117406
38. Liang, J., Han, J., Lombardi, F.: New metrics for the reliability of approximate and probabilistic adders. IEEE Transactions on Computers **62**(9), 1760–1771 (2013). DOI 10.1109/TC.2012.146
39. Lienig, J., Bruemmer, H.: Reliability Analysis. In: Fundamentals of Electronic Systems Design, pp. 45–73. Springer International Publishing, Cham (2017). DOI 10.1007/978-3-319-55840-0_4. URL http://link.springer.com/10.1007/978-3-319-55840-0_4
40. Lyons, R.E., Vanderkulk, W.: The Use of Triple-Modular Redundancy to Improve Computer Reliability. IBM Journal of Research and Development **6**(2), 200–209 (1962). DOI 10.1147/rd.62.0200
41. Maheshwari, A., Burleson, W., Tessier, R.: Trading off transient fault tolerance and power consumption in deep submicron (DSM) VLSI circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **12**(3), 299–311 (2004). DOI 10.1109/TVLSI.2004.824302. URL http://ieeexplore.ieee.org/document/1281801/
42. Maniatakos, M., Makris, Y.: Workload-driven selective hardening of control state elements in modern microprocessors. In: 2010 28th VLSI Test Symposium (VTS), pp. 159–164 (2010). DOI 10.1109/VTS.2010.5469589. ISSN: 2375-1053
43. Mathew, J., Shafik, R.A., Pradhan, D.K. (eds.): Energy-Efficient Fault-Tolerant Systems. Springer New York, New York, NY (2014). DOI 10.1007/978-1-4614-4193-9. URL http://link.springer.com/10.1007/978-1-4614-4193-9
44. Mehrara, M., Attariyan, M., Shyam, S., Constantinides, K., Bertacco, V., Austin, T.: Low-cost protection for ser upsets and silicon defects. In: 2007 Design, Automation Test in Europe Conference Exhibition, pp. 1–6 (2007). DOI 10.1109/DATE.2007.364449
45. Mittal, S.: A survey of techniques for approximate computing. ACM Comput. Surv. **48**(4), 62:1–62:33 (2016). DOI 10.1145/2893356
46. Mohanram, K., Touba, N.: Cost-effective approach for reducing soft error failure rate in logic circuits. In: International Test Conference, 2003. Proceedings. ITC 2003., vol. 1, pp. 893–901 (2003). DOI 10.1109/TEST.2003.1271075. ISSN: 1089-3539
47. Mrazek, V., Hrbacek, R., Vasicek, Z., Sekanina, L.: Evoapprox8b: Library of approx adders and multipliers for circuit design and benchmarking of approximation methods. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 258–261 (2017). DOI 10.23919/DATE.2017.7926993

48. Naeimi, H., DeHon, A.: Fault-tolerant sub-lithographic design with rollback recovery. Nanotechnology **19**(11), 115708 (2008). DOI 10.1088/0957-4484/19/11/115708. URL https://iopscience.iop.org/article/10.1088/0957-4484/19/11/115708
49. NanGate: Nangate 45nm open cell library. URL http://www.nangate.com/?pageid=2325
50. Neumann, J.v.: Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components. In: C.E. Shannon, J. McCarthy (eds.) Automata Studies. (AM-34), pp. 43–98. Princeton University Press (1956). DOI 10.1515/9781400882618-003. URL https://www.degruyter.com/document/doi/10.1515/9781400882618-003/html
51. Oda, S., Ferry, D.K. (eds.): Nanoscale Silicon Devices, 0 edn. CRC Press (2018). DOI 10.1201/b19251. URL https://www.taylorfrancis.com/books/9781482228687
52. Pagliarini, S.N., Naviner, L.A.d.B., Naviner, J.F.: Selective hardening methodology for combinational logic. In: 2012 13th Latin American Test Workshop (LATW), pp. 1–6 (2012). DOI 10.1109/LATW.2012.6261262. ISSN: 2373-0862
53. Peterson, W.W., Weldon, E.J.: Error-correcting codes, 2d ed edn. MIT Press, Cambridge (1972)
54. Polian, I., Reddy, S.M., Becker, B.: Scalable Calculation of Logical Masking Effects for Selective Hardening Against Soft Errors. In: 2008 IEEE Computer Society Annual Symposium on VLSI, pp. 257–262 (2008). DOI 10.1109/ISVLSI.2008.22. ISSN: 2159-3477
55. Rehman, S., Prabakaran, B.S., El-Harouni, W., Shafique, M., Henkel, J.: Heterogeneous Approximate Multipliers: Architectures and Design Methodologies, pp. 45–66. Springer (2019). DOI 10.1007/978-3-319-99322-5_3
56. Rushby, J.: Formal methods and their role in the certification of critical systems. In: R. Shaw (ed.) Safety and Reliability of Software Based Systems, pp. 1–42. Springer London, London (1997)
57. Sachdev, M.: Defect Oriented Testing for CMOS Analog and Digital Circuits, *Frontiers in Electronic Testing*, vol. 10. Springer US, Boston, MA (1999). DOI 10.1007/978-1-4757-4926-7. URL http://link.springer.com/10.1007/978-1-4757-4926-7
58. Sanchez-Clemente, A.J., Entrena, L., Hrbacek, R., Sekanina, L.: Error mitigation using approximate logic circuits: A comparison of probabilistic and evolutionary approaches. IEEE Transactions on Reliability **65**(4), 1871–1883 (2016). DOI 10.1109/TR.2016.2604918
59. Santoro, M.: New methodologies for eliminating No Trouble Found, No Fault Found and other non repeatable failures in depot settings. In: 2008 IEEE AUTOTESTCON, pp. 336–340 (2008). DOI 10.1109/AUTEST.2008.4662636. ISSN: 1558-4550
60. Segura, J., Hawkins, C.F.: CMOS electronics: how it works, how it fails. IEEE Press ; Wiley-Interscience, New York (2004). OCLC: ocm53192483
61. Shafique, M., Ahmad, W., Hafiz, R., Henkel, J.: A low latency generic accuracy configurable adder. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2015). DOI 10.1145/2744769.2744778
62. Shivakumar, P.: Techniques to improve the hard and soft error reliability of distributed architectures. Thesis (2007). URL https://repositories.lib.utexas.edu/handle/2152/3304
63. Sierawski, B.D., Bhuva, B.L., Massengill, L.W.: Reducing soft error rate in logic circuits through approximate logic functions. IEEE Transactions on Nuclear Science **53**(6), 3417–3421 (2006). DOI 10.1109/TNS.2006.884352
64. Sosnowski, J.: Transient fault tolerance in digital systems. IEEE Micro **14**(1), 24–35 (1994). DOI 10.1109/40.259897. URL http://ieeexplore.ieee.org/document/259897/
65. Srinivasan, J., Adve, S., Bose, P., Rivers, J.: The case for lifetime reliability-aware microprocessors. In: Proceedings. 31st Annual International Symposium on Computer Architecture, 2004., pp. 276–287 (2004). DOI 10.1109/ISCA.2004.1310781
66. Srinivasan, J., Adve, S., Bose, P., Rivers, J.: The impact of technology scaling on lifetime reliability. In: International Conference on Dependable Systems and Networks, 2004, pp. 177–186 (2004). DOI 10.1109/DSN.2004.1311888
67. Subramanian, V., Somani, A.K.: Conjoined pipeline: Enhancing hardware reliability and performance through organized pipeline redundancy. In: 2008 14th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 9–16 (2008). DOI 10.1109/PRDC.2008.54

68. Sánchez-Clemente, A., Entrena, L., García-Valderas, M., López-Ongil, C.: Logic masking for set mitigation using approximate logic circuits. In: 2012 IEEE 18th International On-Line Testing Symposium (IOLTS), pp. 176–181 (2012). DOI 10.1109/IOLTS.2012.6313868
69. Traiola, M.: Test techniques for approximate digital circuits. phd thesis, Université Montpellier (2019). URL https://tel.archives-ouvertes.fr/tel-02485781
70. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: 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), pp. 1–6 (2018). DOI 10.1109/DFT.2018.8602939
71. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: On the comparison of different atpg approaches for approximate integrated circuits. In: IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits Systems, pp. 85–90 (2018). DOI 10.1109/DDECS.2018.00022
72. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: Testing approximate digital circuits: Challenges and opportunities. In: 2018 IEEE 19th Latin-American Test Symposium (LATS), pp. 1–6 (2018). DOI 10.1109/LATW.2018.8349681
73. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: Testing approximate digital circuits: Challenges and opportunities. In: 2018 IEEE 19th Latin-American Test Symposium (LATS), pp. 1–6 (2018). DOI 10.1109/LATW.2018.8349681
74. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: A test pattern generation technique for approximate circuits based on an ilp-formulated pattern selection procedure. IEEE Transactions on Nanotechnology pp. 1–1 (2019). DOI 10.1109/TNANO.2019.2923040
75. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: Maximizing yield for approximate integrated circuits. In: 2020 Design, Automation Test in Europe Conference Exhibition (DATE) (2020)
76. Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: A survey of testing techniques for approximate integrated circuits. Proceedings of the IEEE **108**(12), 2178–2194 (2020). DOI 10.1109/JPROC.2020.2999613
77. Tran, D., Virazel, A., Bosio, A., Dilillo, L., Girard, P., Pravossoudovitch, S., Wunderlich, H.J.: A Hybrid Fault Tolerant Architecture for Robustness Improvement of Digital Circuits. In: 2011 Asian Test Symposium, pp. 136–141 (2011). DOI 10.1109/ATS.2011.89. ISSN: 2377-5386
78. Wali, I.: Circuit and system fault tolerance techniques. phdthesis, Université Montpellier (2016). URL https://tel.archives-ouvertes.fr/tel-01807927
79. Wali, I., Deveautour, B., Virazel, A., Bosio, A., Girard, P., Sonza Reorda, M.: A Low-Cost Reliability vs. Cost Trade-Off Methodology to Selectively Harden Logic Circuits. Journal of Electronic Testing **33**(1), 25–36 (2017). DOI 10.1007/s10836-017-5640-6. URL https://doi.org/10.1007/s10836-017-5640-6
80. Wali, I., Traiola, M., Virazel, A., Girard, P., Barbareschi, M., Bosio, A.: Towards approximation during test of integrated circuits. In: 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), pp. 28–33 (2017). DOI 10.1109/DDECS.2017.7934574
81. Wali, I., Virazel, A., Bosio, A., Dilillo, L., Girard, P.: An effective hybrid fault-tolerant architecture for pipelined cores. In: 2015 20th IEEE European Test Symposium (ETS), pp. 1–6 (2015). DOI 10.1109/ETS.2015.7138733
82. Weide-Zaage, K., Chrzanowska-Jeske, M.: Semiconductor Devices in Harsh Conditions. URL https://www.routledge.com/Semiconductor-Devices-in-Harsh-Conditions/Weide-Zaage-Chrzanowska-Jeske/p/book/9780367656362
83. Wirnshofer, M.: Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits. Springer Series in Advanced Microelectronics. Springer Netherlands (2013). DOI 10.1007/978-94-007-6196-4. URL https://www.springer.com/gp/book/9789400761957
84. Yang, S.: Logic synthesis and optimization benchmarks user guide version 3.0 (1991)
85. Yao, J., Okada, S., Masuda, M., Kobayashi, K., Nakashima, Y.: DARA: A Low-Cost Reliable Architecture Based on Unhardened Devices and Its Case Study of Radiation Stress Test. IEEE Transactions on Nuclear Science **59**(6), 2852–2858 (2012). DOI 10.1109/TNS.2012.2223715

86. Yazdanbakhsh, A., Mahajan, D., Esmaeilzadeh, H., Lotfi-Kamran, P.: Axbench: A multiplatform benchmark suite for approximate computing. IEEE Design Test **34**(2), 60–68 (2017). DOI 10.1109/MDAT.2016.2630270
87. Ye, R., Wang, T., Yuan, F., Kumar, R., Xu, Q.: On reconfiguration-oriented approximate adder design and its application. In: 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 48–54 (2013). DOI 10.1109/ICCAD.2013.6691096
88. Zhou, Q., Mohanram, K.: Gate sizing to radiation harden combinational logic. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**(1), 155–166 (2006). DOI 10.1109/TCAD.2005.853696
89. Zhu, N., Goh, W.L., Yeo, K.S.: An enhanced low-power high-speed adder for error-tolerant application. In: Proceedings of the 2009 12th International Symposium on Integrated Circuits, pp. 69–72 (2009)
90. Zoellin, C.G., Wunderlich, H.J., Polian, I., Becker, B.: Selective Hardening in Early Design Steps. In: 2008 13th European Test Symposium, pp. 185–190 (2008). DOI 10.1109/ETS.2008.30. ISSN: 1558-1780