
Article:
McWilliam, R., Khan, S., Farnsworth, M. et al. (1 more author) (2018) Zero-maintenance of electronic systems: Perspectives, challenges, and opportunities. Microelectronics Reliability, 85. pp. 122-139. ISSN 0026-2714

https://doi.org/10.1016/j.microrel.2018.04.001

Zero-Maintenance of Electronic Systems: Perspectives, Challenges, and Opportunities

Richard McWilliam, *Member, IEEE,* Samir Khan, *Member, IEEE,* Michael Farnsworth, Colin Bell

***Abstract*—Self-engineering systems that are capable of repairing themselves in-situ without the need for human decision (or intervention) could be used to achieve *zero-maintenance*. This philosophy is synonymous to the way in which the human body heals and repairs itself up to a point. This article synthesises issues related to an emerging area of self-healing technologies that links software and hardware mitigations strategies. Efforts are concentrated on built-in detection, masking and active mitigation that comprises self-recovery or self-repair capability, and has a focus on system resilience and recovering from fault events. Design techniques are critically reviewed to clarify the role of fault coverage, resource allocation and fault awareness, set in the context of existing and emerging printable/nanoscale manufacturing processes. The analysis presents new opportunities to form a view on the research required for a successful integration of zero-maintenance. Finally, the potential cost benefits and future trends are enumerated.**

***Index Terms*—Fault-tolerance, self-repair, zero-maintenance, built-in fault detection, self-healing systems**

I. Introduction

The rising maintenance costs facing today's high value manufacturing industry is fuelling a new appetite for design strategies that reduce maintenance, repair and overhaul costs of complex high value systems [1]. The availability and dependability of electronic components and sub-components within complex, high-value systems is a critical driver for reducing the net cost per hour of operation. Fault events and associated system error states incur punitive costs due to; fault location and diagnosis; invasive inspection and test; provision for frequent maintenance intervals even if fault events have not occurred. Electronic systems and sub-systems have therefore become a pivotal element in fault-sensitive, service-driven sectors.

This article surveys several journal articles, conference papers, books and literature reviews on hardware approaches that are anticipated to pave the way towards zero-maintenance capabilities. Such capabilities are difficult (or even impossible) to implement exclusively within the software, mechanical or materials domains: instead the majority of strategies are partially (or fully) coupled with electronic systems and sub-systems since this permits a wide range of fault mitigation approaches. Hence cross-domain strategies feature heavily in the methods considered. The aim of this work is therefore to develop a core understanding of zero-maintenance within electronic systems and related design strategies for its implementation. This is achieved through a number of objectives outlined below:

- Analyse current trends in the evolution of self-recovery and self-repair towards achieving zero-maintenance;
- Present a quantitative (and where possible qualitative) comprehension of the design trade-off factors and metrics;
- Develop a cohesive understanding of zero-maintenance as a design approach and technology in its own right;
- Develop an appreciation of the core merits of zero-maintenance through a sector-wise view of the technology.

The title of this article has been chosen carefully, because the use of the word *perspectives* implies a personal analysis and presentation on behalf of the authors. The authors' expertise range in the areas of diagnostic design, signal processing, maintenance, self-healing and machine learning; with significant focus on practical implementation rather than theoretical. Nevertheless, the article's contents will be of general interest to electrical scientists and engineers, because some of the more practical issues of implementing self-recovery and self-repair capabilities are often not appreciated, let alone the costs attached to them. The focus is therefore towards detailing the philosophy of having *zero-maintenance*. The principal concepts of self-detection, fault masking/mitigation behaviour monitoring are analysed and categorised according to their design level implementations. Self-recovery in the presence of faults is an important step towards realising complex systems capable of maintaining their designed for function throughout their intended life-cycle. The other characteristic is that their common design metrics must be analysed in terms of fault coverage, resource allocation/cost and fault awareness set in the context of existing and emerging electronic manufacturing processes. To the best of the authors' knowledge, this is the first study which provides a detailed account on zero-maintenance systems and related system approaches. These accounts have been broken down into research questions, that are suggested together with their motivations in Table I. The questions are aimed to make some semantic distinctions which are important towards application. These can be divided in terms of approach (active or passive), methodologies, techniques, applications, implementation requirements and capabilities, cost implications and their impact.

R. McWilliam is with the Department of Mathematical Sciences, Durham University

S. Khan is with the Department of Aeronautics and Astronautics, University of Tokyo. E-mail khan@ailab.t.u-tokyo.ac.jp

M. Farnsworth is with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, South Yorkshire. E-mail m.j.farnsworth@sheffield.ac.uk

C. Bell is with the Department of Mechanical Engineering and Mathematical Sciences, Oxford Brookes University

Manuscript received xx xx, 20xx; revised August 26, 2017.

TABLE I
RESEARCH QUESTIONS

Research question	Table(s)	Motivation
What are the technological requirements for zero-maintenance?	VI, XI, XII	Identify the types of implementation tools that can be used to accomplish a task
What approaches can be considered to achieve these capabilities?	XIV, XV	Identify the types of active and passive approaches
What kind of design methodologies can be used for zero-maintenance?	XIII, XVI	Identify how to systematically solve the problem
What techniques can be used to provide built-in fault detection?	III	Identify the type of techniques or tests that can be used
What metrics can be used to estimate the cost implications?	VIII	Identify the most important key performance metrics
What are the challenges for a zero-maintenance philosophy on mission-critical and resilience systems?	XVI	Identify opportunities for self management systems
What kind of applications can benefit from its successful realisation?	XVII	Categorise the applications where it can be used

A. Contributions

The article focuses on an emerging and important topic across many complex engineering industries. It provides a broad overview of self healing and self repair techniques, which span multiple application domains. For each technique, the authors provide its basic form and then consider how it can be useful to achieve zero-maintenance. Literature developments are discussed accordingly. This template provides an easier and succinct understanding of these techniques, taking note of their applicability and limitations. Whilst most of the work covered focuses on the electronics industry, there is an increase interest from electro-mechanical domain. The authors have noted that zero-maintenance systems need to attain the characteristics of self-diagnosis, self-repair and self-immobilisation to some degree, to prevent more serious damage and catastrophic breakdowns. As research within this area is of practical importance, some older references have been included, e.g., early discussions about online self-test and repair, which never really took hold after electronics became repairable during manufacture and relatively robust in-service. This is not necessarily true of emerging non-CMOS technologies and so the online repair theme is *popular* once more. To accomplish the study aims and objectives, this research undertook a filtering process where the key main themes and subtopics (formalisms, design and strategies for zero-maintenance of electronic systems) were decided upon, and selected journals and conferences formed the bulk of material for review and analysis. These range from electronics, maintenance and repair, manufacture and more, all of which are directly related to self-recovery, self-repair and its maintenance application. To summarise the key contributions at the outset, the article provides:

- An organised and critical view of current and emerging trends towards achieving zero-maintenance;
- A state-of-the-art on common design techniques in terms of fault coverage, resource allocation/cost and fault awareness set in the context of existing and emerging electronic manufacturing processes;
- A discussion of the key performance metrics and their relevance within different application domains. This includes past methods that have seen renewed interest such as fine-grained device/interconnect redundancy;
- An overview of the potential impact upon mission-critical, high resilience systems whose useful lifespan depends on efficient, self-management of spare resources. There seems to be an over-emphasis on being able to deploy high-level models rapidly, without the need of an underlying technical expertise about the processing framework;
- Consideration is given to the relative effort needed for the successful integration of a zero-maintenance philosophy weighed against cost factors.

Also, the authors have written this article in a way, that can allow readers with a non-electronic background to gain an understanding of the zero-maintenance philosophy. The article first follows the trends towards zero-maintenance and discusses the challenges for its successful realisation in Section II. In Section III, a brief perspective on the quantification of zero-maintenance parameters is formed. Expanding upon this we delve into more detail on how this approach can be introduced into electronic systems, providing state of the art examples in Section IV. It also outlines recent approaches for quantifying the success and trade offs of these techniques using current metrics. The discussion then switches to focus upon design with an overview of the main active and passive methods for mitigating faults in electronic systems in Section V, before a look at how a number of design strategies can be undertaken to incorporate these techniques across a number of applications in Sections VI. Finally the paper looks forward, drawing conclusions about this emerging field, with an outlook on the opportunities and challenges that await.

II. TOWARDS ZERO-MAINTENANCE

Can a system's maintenance effort be reduced to zero? Perhaps a more realistic question is: can a system operate as originally intended, all the time?

If a system (or service or component) operates without failing, and delivers the exact business function without any manual intervention, then there is some semblance of moving towards zero-maintenance. Even though this may only be a theoretical possibility, it provides a perspective to explore questions for bridging a knowledge gap. An ideal self-engineering system should be capable of repairing itself in-situ without the need for human decision (or intervention). This will have a significant impact on reducing the overall cost of the maintenance process [2]. However, the application of this

philosophy within engineering is a challenge and the authors aim to draw attention to the need for maintenance systems to have self-diagnosis and self-repair capabilities, built-in logic for self-reconfiguration and a cost weighted solution.

The concept embodies the idea of enabling applications and systems to achieve and sustain near-zero spend, and transform the traditional maintenance practices from 'Fail-and-Fix' to 'Predict-and-Prevent' and ultimately to a 'Fail-Proof' state. A classification of literature is presented in Table II with a direct link between the topic of interest and the concepts discussed and analysed within. These articles highlight that concepts such as self-healing and self-repair are predominantly designed at the component and material science level. Within all these applications, there are generic concepts of zero-maintenance. However, for safety-critical applications, such as aerospace or nuclear that operate in inaccessible environments (e.g. space, offshore), there is a need to have a system level of self-diagnosis and self-repair, if not self-immobilisation, to prevent more serious damage and catastrophic breakdowns.

In the broadest sense, Reliability, Availability and Serviceability (RAS) performance metrics become highly dependent upon self-repair capability. Further downstream impact can be seen within integrated health monitoring, online self-test within which interrupt-free service is a critical profit margin driver [17], [18], [19]. As a result of this, growing interest has emerged for new design strategies with zero-maintenance properties. This article will focus on electronic components and sub-systems that are equipped with new fault detection and classification capabilities [20], [21]. Existing capabilities that can relate to these concepts are briefly introduced in Table III. Maintenance can be related to this emerging area of self-healing technologies as it links software and hardware mitigations strategies across multiple domains where many failure classes exist [22], [23]. Examples relating to electronic systems include mechatronics [24], control [25] and materials. In some cases an overlap exists between hardware and software domains occurs for example, in fault-tolerant GPU algorithms [26], VHDL methods for redundant layout in FPGAs [27] and FPGA bitstream manipulation [8] and therefore zero-maintenance is driven by a subset of failure classes. In addition to this, mitigation relies upon an awareness of the underlying hardware, especially multiple multi-processor, custom configurable architectures and reliability-driven compilation [7]. This survey therefore focuses on key emerging trends for hardware-driven mitigation methods that relate to Table III.

A. Technology platforms

The philosophy of zero-maintenance is relatable to several existing and emerging electronic technologies. State-of-the-art hardware fault mitigation techniques handle permanent faults using active detect-respond mitigation, whilst sometimes operating along side passive masking. FPGAs are frequently used for studies in this area [6] where online reconfiguration remains a highly challenging task [12], [28], [29]. More sophisticated reconfigurable platforms are emerging that support development of dynamic self-test and repair (STAR). Manufacturing yield enhancement has been a strong driver for significant investment for silicon electronics and the resulting yield-driven strategies continue to extract fully functional operation out of essentially error-prone fabrication processes [30]. These strategies are undergoing a transformation to include built in repair mechanisms that operate beyond the point of manufacture. Emerging non-silicone technologies of relevance to zero-maintenance are printable large area and nanoscale electronics that bring new challenges and opportunities for fault mitigation due to their differing fabrication processes and technology scaling.

In respect to hardware development platforms, a key goal of future zero-maintenance strategies is to proliferate fault detection and discrimination towards the lower design levels in order that detection occurs closer to the actual fault locale. This aspect is explored further in Section IV-A. It is further expected that fault mitigation operates most effectively when concentrated as far as possible to the same locale. To support development of such *fine-grained* fault mitigation, hardware in loop monitoring has advanced significantly in recent years to support industrial control and monitoring platforms for high-value sectors such as aerospace, mining, consumer transportation and exploration where even small down-time events incur considerable financial cost. In such cases the incorporation of built-in actions that inhibit further fault events become extremely valuable as does the ability to record the frequency and nature of abnormal events.

B. Implementation challenges

A precursor to eliminating effort required to maintain any function, is to monitor it. Under nominal operational conditions, dependability in electronic systems is secured when errors arising from faults can be diagnosed. Therefore fault detection and mitigation during normal operation becomes paramount. Faults stem from a number of sources, including production defects (infant mortality), ground level radiation effects, yield challenges for next generation fabrication processes, greatly increasing complexity of mission and safety critical electronic systems, testability of complex systems, highly integrated System in Package (SiP) ageing factors especially for high power devices and ultra low voltage ASICs. Within this survey, it is therefore assumed that the primary faults under consideration relate to transient upsets or permanent faults in each of these cases[1]. Therefore, several factors make achieving zero-maintenance a challenging endeavor:

1) Provisioning for various fault types: Commonly encountered faults and errors (potentially) arising within electronics are considered in Table IV. Such events may occur on a time-limited basis and maybe short-lived, repetitive (persistent) or periodic in nature. Upsets can cause temporary or permanent fault conditions, but errors do not necessarily result. Faults that do not cause immediate errors are termed sub-critical and may lie dormant between power cycles or remain indefinitely. Their influence upon error state is dependent upon the system state, and therefore sub-critical faults are

[1]A broad discussion of such matters is found in [31], in which various examples of fault and error manifestation are discussed.

TABLE II
CATEGORISATION OF REVIEW LITERATURE RELATING TO ZERO-MAINTENANCE PHILOSOPHY

Topic	Article	*Offline*	*Online*	*Chip Level*	*Package/board level*	*Yield enhancement*	*Fault detection*	*Fault masking*	*Modular redundancy*	*Cellular architecture*	*Reconfiguration*	*EDC*	*Fine-grained*
Microsystems	[3]		x	x				x	x				x
	[4]		x	x	x							x	
	[5]		x	x		x		x	x				x
FPGA-based	[6]		x	x				x	x			x	
	[7]	x	x	x		x				x	x		
	[8]		x	x			x			x	x		
	[9]	x	x	x			x	x	x	x	x		
	[10]		x	x				x	x				
	[11]		x	x			x					x	
Computer architecture	[12]		x	x	x		x	x	x		x		x
	[13]		x		x			x	x		x		
	[14]		x		x		x	x	x		x		
DRAM	[15]	x				x		x		x	x		
Evolutionary	[16]	x	x	x			x	x		x	x		x

TABLE III
CURRENT METHODS FOR RESILIENT OPERATION

Technique	**Context**	**Example of state-of-the-art**
Built-in self-test	Perform off line integrity checks before commencing normal operation	Power-On Self Test (POST) within computer BIOS
Online status reporting	Real-time fault checking	General dashboard warning light
Fault discrimination	System-level diagnosis via sub-module level BIT	Specific dashboard warning light
Fault monitoring	Detect and log fault occurrences	SMART hard disk monitoring
Fault masking	Typically majority voting within modular sub-system	TMR controller for safety critical plant systems
Active fault mitigation	Active response to eliminate faults within logic	Data scrubbing
Self-preservation	Prediction and mitigation against faults	Mostly found in electromechanical systems e.g., hard drive free-fall protection
Error mitigation	Correct errors that cannot be eliminated by fault mitigation	EDC for memory modules

assumed to compromise system dependability. Critical faults cause immediate, persistent and potentially cumulative errors. Even so, there is no guarantee that errors will be immediately observable. The susceptibility to fault-induced upsets increases with various factors, such as increasing die area, shrinking transistor gate dimension and reduced switching voltage. These are all common drivers in microelectronics and future nanoscale and printable electronics hence significant research effort has been directed towards faults occurring within future ASICs, interconnects and memory devices, especially those faults induced by radiation particle strikes [32]. In many cases physics of failure (PoF) models are used to help predict the likely system response. This evidence is then used to build a case for provision of redundant resources to be allocated at design-time and weighing up additional cost.

2) Detecting faults: Considering board and sub-system levels, physical breakdown of printed circuit boards (PCB) is a major concern in high performance systems, especially for high-voltage applications. An example of a simple fault analysis of PCBs is summarised in Table V, where various symptoms and related processes of elimination that involve costly and time consuming design steps are characterised (see also [33]). This classification exemplifies the complexity and effort associated with maintenance where, at the sub-system level, procedures for monitoring and assessing potential failures and mitigation strategies become increasingly complex. In critical applications, failure modes and effects analysis (FMEA) procedures may be employed at the system level to form predictive models for maintenance planning. This includes potential disruption caused by 'No Fault Found (NFF) scenarios and strategic provision of built-in test (BIT) logic [34]. Indeed, BIT logic is viewed as beneficial at board and system levels provided the additional complexity is feasible. An example of this is shown in Table VII, where the relative cost/benefit of BIT is estimated [35]. These factors contribute to the overall maintainability and availability of the system [36]. Potential causes of failures are typically assessed by system experts and preventative or corrective courses of action are determined. FMEA is less commonly applied to low-level design due to the complexity of analysing all sub-parts and hence is mostly confined to high-level integrity analysis. Degradation is also to be considered when designing zero-maintenance strategies; the onset of ageing may become accelerated in the presence of persistent faults hence mitigation

strategies offer the potential to slow this process by repair. Detection and monitoring is also extremely useful in demand-driven maintenance during the onset of ageing.

3) Responding according to fault severity: Systemic and device-level faults give rise to critical errors [37], while soft errors arise from a number of different fault conditions [38]. In particular, dormant fault and sub-critical faults can remain unnoticed for some time before causing errors. Further upset mechanisms include electromagnetic interference (EMI), thermal cycling and mechanical degradation of packaging. Faults may also manifest as incorrect logic levels appearing at gate inputs/outputs or else bit upsets within memory contents (or indeed the manifestation of incorrect voltages/current components within analogue circuitry). Different error behaviours are possible depending upon their location and duration [39]. Examples include: errors that are overwritten and do not cause failures; latent errors that persist but which do not cause output failure (but may affect internal states); and errors that are detected and corrected. A more complete model for the relationship between faults, errors and failures has been proposed in [31].

4) Integrated self-maintenance: Many faults manifest as flaws during manufacture and, while these must be removed during test and repair before the product is worthy of selling, there is potential to continue the process of detection and repair into the useful lifetime via runtime and POST maintenance. As a result of yield issues, many high-density ICs contain large pools of redundant elements that are partly consumed during production BIST, but the remaining redundant elements and associated BIT logic remain inactive thereafter [40]. Even after built-in test and repair (BISTAR), it is conceivable that non-critical faults may have been inadvertently triggered during resource reallocation and may compromise normal operation. BISTAR will continue to feature in future FPGAs [41], configurable ASICs [42] and nanoscale electronics [43] and it has been suggested that BISTAR logic could be made available for runtime test or repair of logic [44] and interconnects [45].

Due to the above challenges, achieving zero-maintenance, in its most generic form, is not an easy problem to solve. In fact, most current maintenance design techniques solve only specific formulations of the problem. These efforts are further influenced by rapidly changing technology requirements and availability as well as investment costs for development and test, factors that are often determined by the industry domain in which maintenance is required.

C. Formalisms of zero-maintenance

In the context of this paper, zero-maintenance can perhaps be best viewed as **a collection of capabilities that ensure error-free operation in the presence of faults occurring within an integrated sub-system or component, with minimal external intervention**. This is also related to the area of autonomous maintenance wherein autonomous systems take on similar capabilities, for example the use of external robotic systems to perform specific maintenance tasks [46]. A number of related maintenance requirements can be identified within the literature, though most come from related areas. A synopsis is given in Table VI including an indication of state of the art. At their most basic level, faults (and errors potentially arising as a result) are masked and/or removed such that their influence is no longer critical to error-free operation. Thus a minimum condition of zero-maintenance is that all faults are made sub-critical. By the same measure, it is also desirable to address sub-critical faults by masking strategies. Design for zero-maintenance therefore comprises fault-tolerant design augmented by active detect and response capabilities such that operational life is extended. Ideally fault-free operation is secured.

III. Quantifying zero-maintenance

Several metrics have been considered for quantifying the performance of self-maintenance strategies. Their design-time prioritisation is application-dependent and each must be evaluated in terms of their reliance upon redundant and coordination resources. Moreover, such resources must be allocated at design-time. Existing metrics are summarised in Table VIII, including fault capacity and performance impact. There are few real-world examples where evaluation of these parameters has been reported in the open literature although some detailed FPGA studies have been summarised in [8].

An important step in the design process is test and evaluation of the detection and mitigation strategy. Ideally this would be done within the actual hardware under test where emerging fault detection and mitigation strategies will aid with the test and verification of complex electronic systems. An example of this is the Slackprobe design for ARM processors that places embedded logic deep within strategic locations of the chip for critical timing monitoring. This logic is included within synthesis/layout steps and provides new insights into ageing effects as the timing becomes degraded. In may cases however the overhead associated with the test logic is high and data collected is not directly related to fault events. Besides embedded hardware monitoring, an alternative approach is to implement fault injection engine during the design evaluation phase. Fault injection is a more aggressive approach that emulates direct fault conditions at the hardware level under the control of a fault injection engine [52]. For low pin count ICs, hardware electronic faults may be injected directly. High-density devices such as ASICs and FPGAs require dedicated internal logic [53], [54]. A comprehensive description of fault injection techniques can be found in [55]. Fault injection has also been discussed at the transistor, gate, device and system level with in ASICs [56] but interconnect-related faults are not as well understood and further work is needed especially for self-repairing strategies [8].

Despite benefiting from mature software tools FPGAs are not optimised for on line reconfiguration. Fine-grained Triple Modular Redundancy (TMR) requires special considerations as discussed in [27]. Off-the-shelf solutions do exist that exploit either partial-reconfiguration facilities or else direct manipulation of the configuration bitstream. Several methods also exist for testing the resilience of FPGA strategies including stuck-at-faults, bitstream analysis, radiation testing and fault injection characterisation [10].

TABLE IV
FAULTS OCCURRING WITHIN INTEGRATED CIRCUITS

	Abbr.	Definition	Description
Transient	SEU	Single event upset	Single transient fault event
	SET	Single event transient	Transient pulse affecting gates and latches
	SBU	Single bit upset	Logical bit inversion within register/memory
	MBU	Multiple bit upset	Multiple register bits inversions
	MCU	Multiple cell upset	Faults manifesting/propagating within logic cells
	SEFI	Single event functional interrupt	Produces observable failure at cell/block output
Permanent	SHE	Single hard error	Caused by single fault, commonly stuck-at
	SEL	Single event latch-up	Rail to rail short circuit in pnpn circuits
	SESB	Single event induced snap-back	Rail to rail short circuit in nMOS circuit
	SEB	Single event burnout	Thermal runaway in power transistors
	SEGR	Single event gate rupture	Breakdown of gate dielectric

TABLE V
ESTABLISHED METHODS FOR PCB LEVEL TESTING AND REPAIR WITHOUT SELF-REPAIR STRATEGIES (ADAPTED FROM [33]).

Fault type	Cause of failure	Eliminated by	Failure model
Layout	Crosstalk, grounding, power rail noise, fan-in or fan-out violations	Correct application of layout rules	Stuck-at, intermittent
Construction	Inappropriate interconnect design or packaging, solder splash, bridging, dry joints	Careful construction and inspection	Stuck-at, bridging
IC internal failures	Fabrication defect, yield issue, packaging defect	Careful construction, screening	Stuck-at, metal-metal shorts
Environment	Accelerated component degradation	Use components qualified for environment conditions	Stuck-at, intermittent
Degradation (time-dependent)	Component ageing, modifications	Preventative maintenance	Stuck-at
Design and implementation	Critical races, static/dynamic errors, hazards	Correct design and validation	Stuck-at, intermittent

TABLE VI
RELATED MAINTENANCE REQUIREMENTS REPORTED IN THE ELECTRONIC DOMAIN

Method	Aim of strategy	State of the art
Built-in self-test	Detection of fault caused by upset at power-on and/or run-time	Automated SMART disk reporting [47]
Fine-grained fault masking	Fault masking at device level, possibly including fault identification	PaNDA chip [42]; interleaved logic; Gaisler Processors [48]
Built-in self-reconfiguration	Capability for online or offline design reconfiguration	Many lab demonstrations, but unclear whether used commercially
Built-in self-reallocation	Self-initiated reorganisation of logic fabric	Fundamental research
Robust state machines	State machine encoding for fault resilience	Adopted in commercial designs
Self-maintenance	Correction of faults in-service (active operation or in standby). May initiate partial repair until next scheduled maintenance.	Some commercial examples but mostly fundamental research
Survivability	Continuously recover from faults, consuming resources as necessary (possible at expense of performance)	Systems for long-term space exploration
Self-diagnosis	Ability to determine most effective course of action; reporting of remaining repair capacity.	Some software-based techniques [49]; MEMS integrity test [50]
Self-preservation	Able to reduce the impact of fault condition before major action is required	Primarily electromechanical systems e.g., disk drive shock protection [51].

TABLE VII
COST/BENEFIT FOR BIT, VIEWED FROM VLSI PERSPECTIVE ADAPTED FROM [35]. + COST INCREASE, - COST REDUCTION, +/- COST INCREASE LEADING TO SAVING.

	Design, test & dev.	Fabrication	Production testing	Maintenance testing	Diagnosis, repair	Service interruption
Chips	+/-	+	-			
Boards	+/-	+	-		-	
Systems	+/-	+	-	-	-	-

TABLE VIII
KEY PERFORMANCE METRICS FOR ASSESSING ZERO-MAINTENANCE STRATEGIES

	Metric	Description
Response	Fault coverage	Refers to fractional area of overall circuit that protected and diversity of faults that can be handled
	Fault granularity	Minimum design layer at which faults can be detected/addressed
	Fault capacity	Number of remaining faults that can be sustained by mitigation strategy
	Performance reduction	Loss of application performance due to zero-maintenance operations
	Latency	Time required for recovery
Resources	Resource overhead	Number of additional components needed over and above basic design
	Resource re-use	Achieving efficient consumption of redundant resources (active methods)
	Energy usage	Energy consumed during recovery and consumed by additional resources overall
Diagnostic	Reporting	Discrimination & reporting of fault events at multiple system levels
	Remaining lifetime	Indication of remaining operational hours after which faults will no longer be handled
	Logging	Capacity to store a log of fault history and classification

IV. DESIGN MODELS FOR MITIGATION

Mitigation strategies take the form of either distributed passive masking or active detect and respond. As noted earlier, some are already utilised in manufacturing test and repair and, to a lesser extent, degradation management. In the former case detection followed by repair is sometimes referred to as built-in test and repair (BISTAR). There is a large class of platforms based on reconfigurable FPGAs and PLDs that are explored later in this section. An important question is how can the mitigation strategy determine the most appropriate level of response? A sub-component that is affected by transient upset will not benefit from active response capability since unless permanent effects result; repairing temporary faults by reallocating valuable redundant resources is extremely inefficient and would not address further occurrences of transient upset. Fault masking would be the primary design strategy in this case. In other cases however, the decision is less straightforward because both transient and permanent fault handling may be expected to occur and hence resources must be traded against design overall cost/benefit factors. In [57] general considerations of early fault-tolerant computers were explored and a number of questions raised: when should fault tolerance be considered? How do errors manifest, human error, software fault or hardware fault? How is the benefit quantified? As stated earlier, the motivation for fault-tolerance has shifted from manufacturing consistency toward random and cumulative faults caused by environmental and ageing factors.

To help set the context for design for zero-maintenance a suggested relationship between fault severity and design mitigation as given in Fig. 1 for a non-specific sub-component. A similar view was suggested by Noura et al in the context of resilient control systems [25]. The view presents several typical design considerations: the absolute of limit of fault tolerance, the regions of applicability for each class of fault handling and the performance degradation for more sophisticated

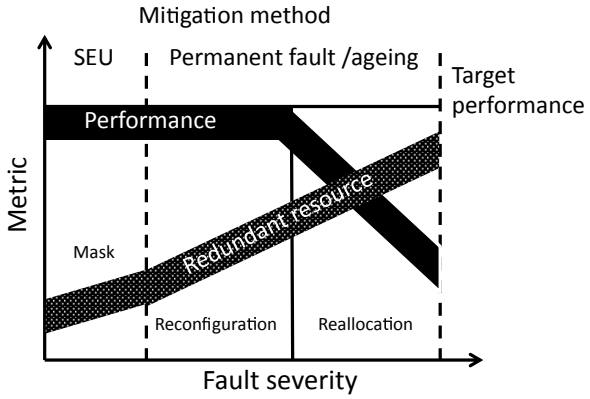

Fig. 1. Progressive strategy for fault-tolerant mitigation with performance impact (adapted from [25]).

fault handling procedures. The anticipated impact on sub-component performance somewhat qualitative, but is indicative of the benefits of employing more complex maintenance strategies. Resource restructuring is a simple concept, but in practice involves a complex hardware reorganisation process that requires coordination and accurate fault diagnostics. This tends to become especially complex for current reconfigurable FPGA designs such as STAR (self-test and repair) [58] that is capable of detecting and isolating faulty logic by activating spare logic, but which requires significant external processing resources.

A. Passive methods

Significant advances in electronics fabrication and packaging methods have taken place over the past decades and the perceived reliability and dependability of resulting ASICs is high. In recent years a re-emergence of passive mitigation has occurred in part to a response to new nanoscale electronics but also the onset of SEUs within current-generation ASICs. The most prevalent passive fault-tolerant strategy is n-modular

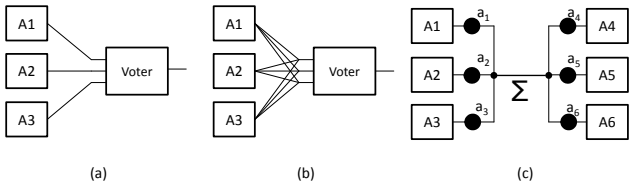

Fig. 2. Majority strategies for fine-grained redundancy. (a) basic majority using logical voter (b) full majority logical voter (c) weighted analogue majority signal formed by summation of weighted contributions. In this example output of each module $A_1, A_2, ..., A_6$ is weighted according to $a_1, a_2, ...a_6$.

redundancy (nMR) involving majority signalling. An early review of nMR design considerations for computer hardware is provided by Carter [13] while fundamental nMR concepts may be traced back to the much-cited work of Von Neumann [59] that was inspired by solutions observed in biological systems that construct resilient systems out of many spare components. Each component is assumed prone to failure and hence error-free operation is secured in the presence of faults through massively redundant node, interconnections and majority signal blocks. Various majority signalling methods have been proposed including partial and full majority (Fig. 2a-b) and summing (Fig. 2c). More recently, fault masking has again arisen as a key approach in nanoscale design but for now utilising small n [60]. Numerous variations on this theme exist, each bringing their own performance / resource trade-off at the fine-grained transistor or logic gate levels.

The potential impact on availability and cost of ownership brought about by fault masking is considered further by Maxion [14]. Proven fault handling measures include self-purging [61], where the classical nMR with voter structure is modified to provide explicit fault detection and isolation of each module. The required switching mechanism constitutes a departure from purely fault masking to detect-isolate actions. In this case the overall reliability is highly dependent upon the individual switching reliability. Further augmentations to the TMR approach were suggested in [39] to include explicit detection and handling of transient and permanent errors. In this scheme, fault detection and self-reassertion of the correct logic state is possible within individual modules. In addition detect-isolate is performed when a permanent fault occurs in a single module, in which case the design reverts to a master/checker scheme. Although this implementation does not include a full self-reconfiguration implementation, it is attractive because there is minimal logic overhead in comparison to the conventional TMR implementation. Of course, the high resource overhead of the TMR scheme itself is still present.

Graceful degradation is difficult to achieve by passive methods due to the inherently limited fault capacity. Although detection is not directly involved, it is possible to extract and utilise majority signals for basic diagnostics. This is done at the modular level to generate enumerated states such as '*ok*', '*fault has occurred but still ok*', '*cannot tolerate further faults*' (critical condition) and '*unavailable*'. Although feasible at the modular sub-system levels, their implementation requires considerable design effort at lower levels (i.e. closer to the origin of fault) and hence new architectures are needed to support this level of granularity. The benefits are clearly significant for integrity monitoring in the presence of transient upsets.

1) Bottom-up design methods: Fine-grained strategies have seen specific interest recently owing to concerns over the vulnerabilities of emerging nanoscale and to some extent state-of-the-art printable electronics. At the lowest design levels, provisioning of redundant transistors has been discussed for compensating yield tolerance occurring during manufacture [62]. Such strategies inevitably involve compromises, namely redundancy overhead, complexity of fault detection and degradation of performance through the use of non-optimal transistors. However, within the zero-maintenance paradigm the benefits of graceful and predictable degradation and self management of faults are extremely attractive. At the sub-system level, online fault discrimination and monitoring operates at the modular level and within maintenance-heavy products, such as land, air and space vehicles, it becomes possible to monitor component aging via key response factors that that are expected to degrade more progressively over time [2]. These methods focus on self-correction of stuck-at faults within nanoscale logic units, for which there are two reasons to consider fine-grained redundancy: firstly, fabrication processes are more prone to defect and variability [63] and the high density of nanoscale manufacturing exacerbates the challenge of high volume production. Secondly, the reduced device dimensions will result in increased susceptibility to in-service faults. With the advent of Large Scale Integrated (LSI) ICs, fine-grained redundancy was employed in mission critical electronic systems such the flight computer of the Apollo lunar mission. Its use was ultimately restricted due to improvements in manufacturing tolerances and the unfavourable view of maintenance at the time [64]. Increasing component density brings a number of issues relating to upset vulnerability and manufacturing defect that must be reduced so that these technologies are able to complete with current silicon technologies. Further considerations include reduced operating and threshold voltage that, together with higher feature density, will lead to lower SEU immunity. Defect mitigation commands the highest effort in the first instance [31] but a number of approaches nonetheless been reported in recent years as summarised in Table IX.

nMR has also been incorporated within FPGA configurations at the block level [65], extended to fine-grained gate redundancy by a method closely related to quadded logic[60] and ultimately appearing at the transistor level via N-modulo redundancy (nMR) [66] and most commonly TMR implementations [67]. A further variation involves *scrubbing* in which the configuration is periodically refreshed from a golden bitstream [68]. This method is very popular due to its simplicity although service interruption does occur. In [43], multiplexed redundancy was considered at the lowest design levels to improve the reliability of logic gates. Future space exploration will further leverage fine-grained strategies wherever possible, principally because of the additional complexity of incorporating design for active repair [66]. Essentially, once a fault masking strategy has been determined it may be

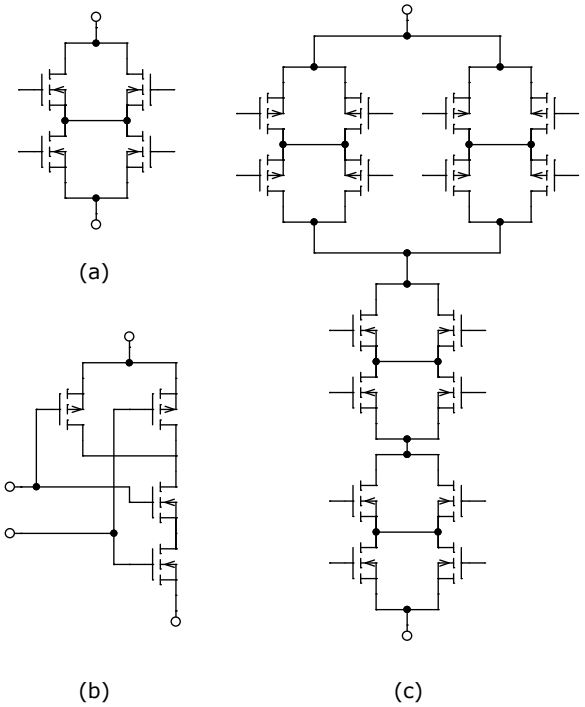

Fig. 3. Fine-grained passive redundancy principal for electronics. (a) quad transistor structure (b) standard CMOS NAND logic gate (c) equivalent quad redundant NAND design.

applied fairly readily to hierarchical logic designs, whether fine-grained, cell or block level although evaluation of its resulting impact fault rate behaviour and associated reliability still needs to be performed. In [43], Han and Jonker noted that different strategies are likely to be required to address transient and permanent (and defect) related failures since the underlying binomial distribution applied to examine transient upset behaviour is unsuitable for modelling permanent faults, where statistical independence can no longer be assumed. Majority voting may occur as a function of the topology of the functional gate [69], the principle being to eliminate the classical voter logic by a summation of electrical currents at a common node. In this instance RoRa (reliability-oriented place and route) algorithms may be employed to instantiate TMR-like structures and evaluate them using V-Place, a model-based tool that uses a topology heuristic to assess and recover performance metrics.

Information redundancy is a further long-established approach distinguished by the fact that errors are allowed to occur and must be corrected. Although mainly as active mitigation there exists hardware designs that exploit this approach when errors can be tolerated. For example, computing in the presence of noise caused by SEUs can be achieved when the numerical error created by an upset does not impinge upon the accuracy of the output [77].

B. Active methods

In contrast to fault-masking redundancy methods, fault detection and reconfiguration aim to achieve fault tolerance via BISR capability. Sometimes this is referred to as built-in self-test and repair (BISTAR), the result of combining BIT and BIST. The contrasting behaviour of active and passive strategies is exemplified by revising Fig.1 where, for active strategies, redundant resources are called upon (and consumed) by rising fault severity. By contrast, passive strategies possess much more limited fault capacity in the form of redundant resource but offer fast recovery. Active methods are armed with sufficient resources to maintain a consistently higher fault capacity but with longer recovery times after each fault event. In the ideal case a combined strategy would only address transient faults by the masking method and permanent faults by the active method.

BISTAR initiates a direct response to an upset event in the form of active alteration of the functional logic (and possibly additional dormant logic). BISTAR is fundamentally different to passive mitigation in that a measured response is taken by reorganising internal resources. The variety of active mitigation methods is summarised in Table X. Reorganisation usually involves a process of fault detection and localisation (possibly automated but most likely manual) followed by replacement of faulty logic with logic that is assumed to be fully functional. An example partial reconfiguration of an FPGA by user-initiated loading of a new bitstream after detecting a problem is suspected. This type of maintenance assistance can be highly sophisticated however it does not constitute self-repair. Self-diagnosis should also be qualified here: many circuits contain BIT hardware that is designed to enhance the fault detection process, however these mostly require external test hardware to be connected in order to achieve diagnosis capability [78].

Built-in test (BIT) logic was originally created for production test and repair but lacks the detection capability proposed for online mitigation [79]. This approach has been refined over the years to improve the efficiency of production test, including the introduction of self-repair logic in memory chips. For example, redundant row and column cells for more efficient repair [80] is essentially a more straightforward reallocation process whereby resources are reorganised to circumvent defects. Once processed by the production tester, the configuration remains locked for the remaining lifetime of the component. The envisaged diagnostic, resource and response performance/cost trade-offs were considered in Table VIII and clearly the incorporation of self-repairing capabilities needs additional resources beyond those envisaged for the production test and repair. An example of this is seen in adaptive cache design where a variable trade-off is implemented in hardware [81].

1) Detection and classification: Detection in electronic sub-systems can occur online, during the active operation with minimal service interruption or offline, performed when the system is placed in a special diagnostic mode or else during power on self-test. Present-day detection is typically implemented using some form of boundary scan logic that is made available during offline test. In complex systems this is usually performed during regular or emergency maintenance and inspection. Offline detection is commonplace in production test and repair during which dedicated logic is activated to facilitate the test and repair process [35]. Notably however,

TABLE IX
BOTTOM-LEVEL DESIGN METHODOLOGIES FOR ELECTRONICS

Method	Example	Mechanism(s)
Quad-structures	[70], [71]	Quad transistors arranged in serial/parallel fault tolerance. Output is formed from majority of pull up/down resistor current.
N2-transistor structures	[72]	Massively-redundant micro-architectures; variant of quad structures.
Triple transistor structures for CMOS logic	[73]	Triple transistor redundancy scheme for minimal area overhead.
Yield enhancement for next-gen electronics	[74], [75]	Redundant transistors designed to be allocated only during production test
Interleaved logic and interconnect	[60]	Send multiple copies of logic levels to redundant gates so that output is calculated several times. This provides majority output and can be arranged to inherently mask certain faults.
Multiplexed redundancy	[43]	Redundancy via additional interconnections
FPGA configuration	[76]	Provisioning additional resources ar programmable block level

TABLE X
ACTIVE MITIGATION STRATEGIES

Method	State of the art	Example
Offline test and repair (BISTAR)	Resources organised during test mode. Valid for chip, package and SoC level test & diagnostics.	[15], [44], [35]
Fine-grained reallocation	Transistor-level reorganisation using switch-over.	[82]
Online test and repair	Fault & mitigation for critical sub-systems that must not be interrupted; integrated self-test & repair.	Autonomous fault management; ECSS F3 on-board fault management [83]
Error mitigation	Errors permitted to occur then corrected at data level	[48], [84]
Self-healing	Autonomic reorganisation, possibly without explicit detection;	FLASH memory recovery, self-healing materials; interconnect possible but very low TRL. [85], [86]

complexity is retained within production test units that are only available during production. Extensions have been proposed that increase the on-chip complexity in lieu of faster overall test procedures [87]. Beyond production BIT and BIST, the principles of detection lies within the wider research area of anomaly detection [88]. For example, at higher system levels cluster analysis is a powerful data-driven approach for system-level health monitoring and diagnostic [89]. However, in the context of this paper it is assumed that detection logic is needed much closer to the point of origin of faults, requiring a step-change in design and integration practices. The goal has become to ensure that errors are not permitted to manifest within electronic logic, interconnections and memory. Fault discrimination is a critical augmentation of detection and opportunities exist within microelectronics design to integrate detection and discrimination close to the point of fault, including assessment of the seriousness of fault. This is made all the more important given detection is the first step towards all active mitigation strategies. In contrast, masking approaches do not depend upon online fault detection, but instead majority signals are produced that, in turn, generate signals that reflect the current integrity in the presence of threats. Logging and reporting of fault events is clearly a further desirable feature within the general maintenance model, however detection and counting of transient upsets is not trivial. Active mitigation is however very well-suited to status monitoring even before active recovery is considered.

There are examples when fault detection is not used but instead the masking strategy is primed to initiate a default response should particular error occur. Examples include switch-over [36], RAID data storage and error detection and correction (EDC) codes. EDC operates by exploring redundancy applied at the information level by data coding. This differs from hardware mitigation since errors are allowed to manifest and are then corrected. In contrast, hardware methods seek to prevent errors from arising when faults occur. EDC is however extremely popular in memory ICs due to regular architecture of memory cells that allows direct application of data codes. An abstraction between hardware and EDC strategies is sometimes apparent for example, in the protection

of configuration bitstreams for FPGAs. In this case EDC is used to protect data patterns that are in turn responsible for hardware configuration [9]. However the respective detection strategies are still distinct. A further example is when EDC applied to protect look up tables (LUT) that in turn implement finite state machines (FSM). Whereas the FSM is traditionally implemented as a combination of LUT data and execution logic, it is possible in some cases to implement the FSM almost entirely out of memory and therefore EDC becomes the predominant detection mechanism. Alternatively, fault tolerance may be encoded at the state level [90]. Once again, errors are allowed to occur and exist until repaired by the EDC strategy. A further variation of this case is when the FSM is provisioned with both hardware fault mitigation and EDC, thus protecting against different fault mechanisms [84]. Detection also appears in the context of self-healing technologies [22] where autonomic responses are triggered upon certain events. In Table XI a summary of fault detection strategies related mostly to modular designs is presented.

Fine-grained built-in test (BIT) strategies represent an important step towards zero-maintenance capability in electronic sub-systems because they require that faulty conditions are identified and localised before the response is formulated. Another case is that of remotely activated BIST using the System JTAG (SJTAG) method [91] wherein the test itself is offline but is activated by a remote data centre. By comparison, online detection is still in its relative infancy but will bring the facility to identify faults during normal operation with minimal impact on overall performance. Zero-maintenance requires considerably more complex design effort to gather useful information to inform the recovery process. One future strategy is to make extensive use of hardware fault detection at the lowest design levels in conjunction with fine-grained redundant design with new BIT logic [92]. This marks a departure from the reliance upon traditional external test units and a step towards online BIST.

Chip- and board-level system considerations are also important, especially the integrity of interconnects. This is an especially challenging area that has been considered as part of the boundary scan approach [45]. There has been limited progress in this area although IEEE standard 1149.1 has evolved to cater for multi-chip modules (MCMs) [93], on-chip test and even hardware emulation and debug. Analogue time-domain refractometry (TDR) approaches have also been proposed to improve fault coverage and retain low MCM complexity [94]. In most cases online operation is not considered and hence the addition of field support phase capability has been discussed with respect to modified boundary scan hardware [44] and could potentially be applied at the board or system levels. This would essentially constitute an automated approach to circumvention of localised board issues. Beyond this however, there is a great need for further work in interconnect-level fault detection they represent critical points of failure and methods are confined to production test [95].

2) Self-reconfiguration: Off-the-shelf FPGA devices contain reconfiguration logic coordinated by user-provided configuration bitstream that has been generated by software design tools and is typically fixed at design-time. These devices have become highly popular and have been identified as critical to system dependability in many applications [96]. At the same time, SoC platforms such as the Zync and Cyclone SoC include integrated processors for runtime reconfiguration of programmable FPGA resources including logic, LUTs, memory and interconnects. The principle of reconfiguring resources within an FPGA in response to faults is discussed in [97], noting that the configuration fabric itself remains fixed. Resource utilisation in such cases rarely approaches 100% and therefore unused memory blocks, look up tables, logic resources are available even within complex designs. The challenge associated reconfiguration however lies in effective detection, coordination and resource coverage within the detection strategy. Since access to available resources tend to be clustered with limited granularity, coverage must be carefully managed otherwise available resources may not be accessible.

An external governing process is usually needed that operates online or offline depending on the implementation. This process may take one of two forms: alternative pre-verified bitstreams stored externally and loaded into the device [98] or direct manipulation of the live bitstream by an internal or external governing process. Embedded or software-defined processors have been proposed as part of this but are considered analogous to external processors in terms of resource usage. Sophisticated algorithms such as the STAR approach [99] attempt to self-manage dynamic resource allocation by quarantining active logic that is under test, with the intention that areas of memory and logic affected by faults are less likely to generate errors.

Besides FPGA chips, reconfiguration is also performed within multi-processor hardware where, again, the underlying logic fabric remains fixed but the resource utilisation is dynamically managed by software techniques depending on fault conditions. Typical examples include load balancing and thermal management as well as modular standby sub-components. Finally, the fault-tolerant properties of neural networks have been investigated by incorporating TMR principles into the network [100]. A summary of key design features expected of reconfigurable computer systems can be found in Table XII.

The most flexible of all active response strategies involves dynamic resource reallocation using a generic pool of resources. Here, the underlying logic fabric itself may be altered and manipulated in response to persistent fault events. Dynamic allocation involves substitution of faulty logic and interconnect with healthy hardware in response to a variety of fault conditions. Successful strategies include heterogeneous architectures arranged as a pool of available resources with multiple fixed logic designs that achieve the same task, but each of which have different fault behaviours, thus providing resilience against systemic failures [101]. Kothe (2006) discusses a method for direct reconfiguration at the fine-grained transistor level [82] where a switching fabric is included at the transistor level. Self-coordination at the logic cell level uses simple trigger-based signals to activate or deactivate each cell. This method is related to bio-inspired cellular arrays, and in particular cellular automata, whose functionality is defined by DNA-like instructions stored by cells organised as a regular array structure. This approach is conceptually attractive due to

TABLE XI
FUNDAMENTAL DETECTION CAPABILITIES FOR ZERO-MAINTENANCE

Method	Mechanism
Checkpointing	Save known good state for potential future roll-back. Usually software implemented.
Spare module	A duplicate module exists in either online or offline state. This is activated when a problem is detected. Requires fault detection.
Duplicate & match	Primitive of TMR: perform same operation twice and compare results. Often implemented using design diversity i.e., two different implementations are used to help avoid systemic errors.
Hardware error detection and correction (EDC)	Generally performed on data to be stored and transmitted. Redundancy is introduced into the data set itself such that certain errors caused by fault are detected and corrected.
Temporal redundancy	Time-sharing of resources in order to generate majority vote.
State machine encoding	Protection by inserting of redundant states that indicate error states.
Virtual TMR	Uses reconfigurable logic blocks to implement TMR when configured correctly. Implementations can be dynamic.
System-level voting	Abstract levels of majority voting to determine system correctness at high level e.g. software level in computing. Somewhat independent of underlying hardware and does not provide indication of cause of error.

TABLE XII
FEATURES FOUND IN RECONFIGURABLE SYSTEMS

Feature	Description
Reprogrammable	Configuration by bitstream or machine code, which can be updated and reloaded.
Configuration means	Access to configuration bitstream (FPGAs) or instruction memory (processors)
Regular architecture	Multiple identical arrays of logic and memory fabric arranged in a regular fashion.
Embedded processors	Specific to FPGAs: software-defined processor allow complex task management with hardware resource abstraction.
Partial reconfiguration	Pre-verified configurations (FPGA) or machine code (processors) primed for fast reconfiguration of select resources without halting active task.

the relatively simple rule sets that govern global functionality.

Other instances of self-organisation include dynamic re-allocation of microprocessors workloads in the event of a failed worker [102] and self-assembly of patterns by convergent cellular automata that are used to coordinate functional logic cells [103]. The latter achieves self-reassembly of the correct configuration even in the event that every cells state is randomised. The Plastic Cellular Architecture (PCA) has also been suggested [104] taking the form of a CA coordination layer together with a reconfigurable functional logic layer called a ”reconfigurable plane”. Taken together, they form a self-reconfiguring logic mechanism even though self-repair was not the original focus of this work. Evolutionary algorithms may also be applied to drive self-reconfiguration [105] in which the rule sets are dynamically evolved over time. In [106] a self-recovery mechanism based on a diffusion model is demonstrated using a reconfigurable hardware platform. A further approach uses the redundant genetic information observed in prokaryote organisms to create an artificial prokaryote that controls circuit configuration [107].

V. TOWARDS A DESIGN STRATEGY FOR ZERO-MAINTENANCE

From the above overview, it is clear that a multitude of strategies relating to maintenance exists that should be carefully matched to the application. This section attempts to set out a forward strategy for designing zero-maintenance into electronic sub-systems and components with reference to the literature classified according to the components of zero-maintenance. A summary of applicable methods is set out in Table XIII in terms of passive and active mechanisms.

For the case of transient upset mitigation the resource requirement is likely to be fixed and the degree of overhead is directly linked to expected fault frequency and corresponding fault capacity. Active mitigation strategies are capable of dynamically issuing redundant elements during the course of operation but resource allocation is considerably more complex and must be determined at design-time. Therefore, redundant resources depicted in Fig. 1 will consumed as cumulative permanent faults occur over time.

An example design for mitigation is seen in for control systems and signal processing applications, where it was noted that system performance degradation is traded for enhanced

fault-tolerance [25], [108], [109]. This compromise was generalised in Fig. 1 to include redundant resource requirement. As previously discussed, passive strategies mitigate through masking faults without explicit detection and seek to continue in the presence of those faults without error. Active strategies rely instead upon explicit fault detection before taking action to remove faulty logic. Clearly it is important in the latter case to avoid false alarms triggered by transient faults that cannot be effectively handled by an active response. Even when considering the advancements made key questions remain: how is the integrity of redundant resources ensured, should they be trusted? This may merit a self-test of standby resources before their allocation in response to fault events. A related issue is integrity of redundant resource pool monitoring and communication of remaining resources to the upper design levels. Last but not least–how can integrity of checking logic be ensured?

A. Design Perspectives

High-value industrial systems are synonymous with high recurring maintenance, repair and overhaul effort that increases the overall cost of through-life support [1]. Failures associated with electronic sub-components lead to costly repair activity and down-time, a symptom that is on the increase within electronic components. OEM and system integrators therefore seek new design methods driven within the electronics domain by increasing IC density, shrinking transistor critical dimension, aggressive voltage and frequency scaling and increasingly complex interconnect and packaging.

Evidence of existing design aspects in that relate zero-maintenance in electronics has been gathered in Table VI, while a deeper analysis of passive mitigation strategies is presented in Table XIV. Although several instances relate to production test and repair, EDC has appeared within low-level hardware including FSM logic. Fine-grained transistor level strategies have matured in production yield enhancement but further opportunities exist to extend this to through-life operation. An alternative yield method relates to direct mitigation against signal delays caused variability in sequential logic components such as flip flops [110]. nMR methods have seem most use within masking strategies, including quad-redundancy, but are generally limited to TMR within FPGAs. In the latter case, FPGA reconfiguration is not necessary since the strategy allocates fixed resources for fault masking. By contrast, active methods are classified in Table XV including custom ASIC, COTS FPGA and mixed signal strategies. There is a good representation of both online and offline approaches–though again modular methods are limited to $n \leq 4$.

A significant challenge to zero-maintenance is that significant design effort must be invested at the outset on minimising susceptibility to error events that are prevalent to the application. In some cases it becomes economically infeasible or technically implausible to eliminate fault events. An example is radiation hardening, which can be achieved using special shielding materials, but adds design complexity in order to meet thermal and weight specifications. An alternative is to use rad-hard chips that are considerably more expensive and are typically not available in current-generation designs unless full custom design is undertaken. Indeed the current trend is to use state of the art commercial and off-the-shelf (COTS) chips due to their higher performance and lower power consumption. This is true even for high-value sectors such as transport and space exploration that need high performance and reliability. The automotive industry is now using high performance SoC wherein critical sub-systems (steering, braking) run concurrently with non-critical services (entertainment and navigation interfaces) all with low power consumption.

Integration of both passive and active strategies is likely since each brings their specific benefits and by the fact that various fault scenarios might be encountered during through-life operation. Combined strategies have been considered in robust microprocessors for some time–an early example seen in [126], where modular redundancy was used. Modern strategies tend to be more concentrated towards the fine grained levels while retaining some degree of modular redundancy at higher design levels. Though by no means complete, the Tables XIV & XV illustrate the breadth of techniques in use or demonstrated in principle. Yet integrated design methodologies are comparatively under-developed. Recognising the need for an integrated view, Henkel (2014) proposes a map of 'Multi-layer dependability modelling and optimization' for electronics micro-architectures in [12] that demonstrates the wide range of available design options. In addition, advancements in hardware and open-source software have been accelerated for implementing concepts related to deep learning and support libraries. These include improvements in Graphical Processing Units (GPUs) as well as work on other technologies such as FPGAs, Tensor Processing Units (TPUs), and other chip systems and architectures that match specific artificial intelligence (AI) and machine learning requirements [89]. With an emphasis on bridging these gaps in AI, novel chips are being built to support different computing models. E.g., Intel is introducing an AI-oriented processor the Intel Nervana Neural Network Processor which is a purpose-built architecture for deep learning, and Huawei's Kirin 970 mobile AI computing platform. Such technology platforms not only influences performance but also cut down costs for organisations who will be making use of these systems in the future.

Generalising to higher design layers or cross-layer techniques, allow us to bring benefits to more than one design level, and are essential in order to leverage the investment of zero-maintenance resources. As an example, re-use of fault information generated at device levels for higher levels, e.g., using fault information generated during fault masking operations at the application layer during maintenance assessment would achieve greater design benefits. The qualitative relationship between performance and resource metrics was illustrated in Fig. 1. There is, however, a lack of detailed trade-off analysis for different methods. A top-level view of the possible operations involved within a complex zero-maintenance design is depicted in Fig. 5. Note that, although masking is included in this scheme, there is no active response.

As discussed in Section IV-B1, fault event monitoring and reporting is particular to the zero-maintenance paradigm and should be carried out in all cases i.e., passive or active fault

TABLE XIII
FUNDAMENTAL ZERO-MAINTENANCE DESIGN STRATEGIES

Strategy	**Mechanism**	**Active or passive**	**Example recovery mechanism (if applicable)**
Fault quarantine	Limit spread of fault	Passive	None, but may assist future repair
Fault detection	Built-in checking for presence of faults	Active	Masking voters; next-gen boundary scan; error-detection
Fault masking	Continue in presence of faults	Passive	nMR; EDC; reduced precision; interleaving; FSM-coding
Retry/scrub	Simple fault eradication	Active	Not a direct repair operation
Fault classify	Detect and report on fault events	Active	None; but important for status and active mitigation
Self-reconfiguration	Adapt to eradicate permanent faults	Active	Autonomous FPGA reconfiguration
Self-healing	Reallocate resources and combine multiple methods.	Both	Self-healing materials; cellular automata

TABLE XIV
CLASSIFICATION OF PROMINENT PASSIVE MASKING STRATEGIES (FIXED CONFIGURATION)

Method	Article	*TMR*	*QMR*	*>QMR*	*Redundant transistors*	*Redundant gates*	*Redundant blocks/modules*	*Redundant interconnects*	*EDC codes*	*State machine coding*	*Defect tolerance*
Fault masking	[60]		x			x		x			x
	[73]	x			x						x
	[72]		x	x	x						x
	[59]			x			x	x			
	[43]			x		x	x				x
	[42]			x	x						x
Error Correction	[4]								x	x	
	[48]	x					x				
	[84]			x			x				
	[111]	x					x		x		
	[112]	x					x		x		
	[113]			x			x		x		
FPGA implemented	[6]	x					x		x		
	[27]	x					x				
	[114]	x					x		x		
	[115]			x				x			x

and so that trends such as varying SEU rates may be analysed. Fault management via active mitigation is a desirable concept in zero-maintenance that has not yet seen adoption acceptance within critical applications such as transportation (aviation, rail, automotive) and space. Further complications face active strategies that display complex behaviour since their complex behaviour can be difficult to certify. A further factor is a lack of standards though progress is being made in autonomous vehicles and space exploration [83].

B. Hierarchical perspective

Design for zero-maintenance must lead to higher intrinsic systems robustness and dependability while at the same time reducing the overall complexity and frequency of maintenance operations. Lessons have been learned in some high-volume electronic products, such as electromechanical hard disk drives [127], where the traditional assumptions about hazards rates for early and useful life periods do not apply in the same way as for previous electronic ICs. This gave rise to initiatives for introducing built-in test and self-diagnosis capabilities [128]. In addition, health monitoring utilises external sensor data for accelerated testing to better understand degradation effects [129]. It is notable, however, that these methods are hardware/software codependent and thus are coordinated at the higher system levels.

A useful perspective is to regard zero-maintenance as a multi-level design problem in similar fashion to the highly integrated nature of electronic systems design. An early analysis

TABLE XV
CLASSIFICATION OF PROMINENT ACTIVE MITIGATION STRATEGIES BASED ON RECONFIGURATION

Method	Article	*Online*	*Offline*	*Re-routing*	*DMR*	*TMR*	*QMR or higher*	*Cell reconfiguration*	*Temporal redundancy*	*Interconnect redundancy*	*Intrinsic redundancy*	*Self-tuning*	*Delay lines*
Time-shared redundancy	[3]	x				x			x				
	[112]	x				x			x				
	[116]	x					x		x				
ASIC (fine-grained)	[82]	x		x	x								
	[117]		x	x	x								
ASIC, (heterogeneous)	[106]	x						x					
	[118]	x						x					
	[119]		x	x			x						
	[120]		x	x			x						
FPGA implementations	[99]		x	x				x					
	[121]	x		x				x					
	[122]		x	x				x					
	[123]	x	x	x					x				
Analogue & mixed-signal	[77]	x			x				x				x
	[86]	x								x			
	[124]		x									x	
	[125]		x									x	

of the cost/benefit expected from traditional BIT techniques for VLSI was provided by Agrawal in 1993 [35]. More recently, Henkel provided a categorization of strategies specific to on-chip systems [130], where the priority areas of manufacturing variability, ageing, soft-errors and hardware mitigation were identified as key to design for resilience. This viewpoint can be broadened slightly to include key system levels of interest to zero maintenance:

- *Application level*: fault reporting, fault capacity, assessment and scheduling of swap-out or refurbishment
- *Software level*: resource awareness and adaptability e.g., multiple core management; software fault flagging
- *Integrated hardware level*: board and interconnect failures between sub-module interconnections and wiring looms; interfaces for diagnostic information, power-on self-test
- *Integrated chip level*: System-on-Chip and micro-architectures; cellular reallocation; reconfiguration; adaptive analogue methods
- *Cell, gate and transistor levels*: component redundancy; individual fault masking and detection; variability and yield compensation

From this analysis, we suggest the design hierarchy model in Figure 4 that considers the technological categories relevant to zero-maintenance and requirement for fault prognostics and diagnostics.

Further inspiration is found in technologies for extreme harsh environments such as unmanned space exploration mission will require ultra-reliable craft and vehicles for long-term colonisation [131]. For these cases, future developments will need to address enhanced resilience against component ageing, thermal stressing and soft/hard-errors. A variety of

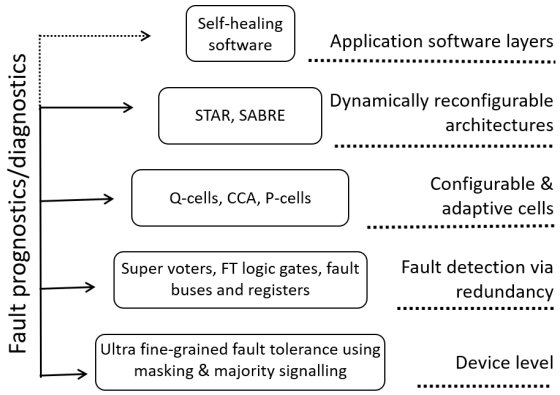

Fig. 4. Proposed hierarchy of design for zero maintenance in electronics. In respect to fault prognostics/diagnostics, a solid arrow suggests the flow of fault awareness and action taken while a dashed arrow suggests information flow only.

self-healing technologies are likely to contribute, albeit each having realistic limitations. An example of this is recovery of flash memory technology via built-in energy pulsing [132]. Annealing of the wear characteristic was demonstrated but under specific conditions. Intense radiation is a major concern not only for exploration but also avionics and satellites and is the subject of many studies [133]. Fine-grained redundancy for autonomous applications has been discussed for situations in which human intervention is hazardous or impossible with applications in deep sea exploration, nuclear inspection, deep drilling and mining [134], [135]. Fault events cause serious consequences for safety critical systems and often require costly fault-finding and maintenance. When subjected to one

or more of these influences the resulting sub-system behaviour can be somewhat complex and difficult to predict. One possible model considers faults occurring at different system levels in electronic components wafer (transistor, interconnect, metal layers on wafer), packaging (ASIC, FPGA, package wires, chip to chip connections, SoC interconnects) and sub-component (PCB, packaging, cables and connectors). However it is not clear from this how errors proliferate into malfunctions across different systems levels. This is compounded by the challenge of fault detection and location. FMEA originated from the design of flight electronic systems in the 1950s [136] and specifically aims to identify corrective actions needed to prevent failures by an exhaustive analysis of system components and identification of failure modes and probabilities and possible detection and recovery action steps. PSPICE modeling can be helpful in determining component ratings (thermal, voltage/current and noise) and to recommend de-rating where necessary.

From the above discussion, it is clear that a structured design methodology is required to address zero-maintenance within electronics. With this in mind a breakdown of key challenges with this model is proposed in Table XVI, in which key mechanisms are described according to known existing strategies together with potential benefits and challenges in the context of zero-maintenance. There is evidence of similar hierarchical considerations at the lower design levels in [137], where a BISR architecture is defined up to the cellular hardware level.

Since the available passive and active methods span all hardware layers from fine-grained logic to reconfigurable logic fabric, it necessary to integrate design strategies across most hardware design levels. Within this structure, it is important to consider the specific benefits to real-world maintenance aspects and in particular effective detection and integration with software methods. By way of example, GPUs provide an interesting case in point because they already utilise a variety of error-handling strategies. Depending upon the architecture used a combination of low-level hardware EDC to protect vulnerable high-density memory areas [138] and a high-level error tolerance due to the visual latitude of the application is exploited [26]. There is however some debate over the benefit of low-level hardware EDC protection since software redundant methods have also been shown to be effective [139]. Furthermore, this model generally allows errors to manifest rather than mitigating against the underlying faults. A fault-tolerant design approach faces similar design decisions.

Evidence of package-level mitigation (Table II) and interconnect fault handling (Table XIV) exists. However there is a great need for enhanced test and verification for state of the art system on chip (SoC) and system in package (SiP) technologies for supporting design validation as well as fault-handling. This need straddles multiple design levels, with fault detection and status monitoring placed as close as possible to the point of origin of faults. The hierarchical model extends further to electro-mechanical interfaces, as seen in MEMS integrated design [140][141][142], and ultimately into the higher software layers.

C. Application-specific considerations

Fault mitigation has in-part driven the evolution of some COTS devices such as FPGA and microcontrollers. Radation-hardened FGPAs have been developed but functional resilience capabilities have also been introduced for fault tolerance. The potential impact of fault-tolerance within FPGAs has been considered at the electromechanical system level of a robot [143]. Specific examples discussed earlier in Section IV included bitstream EDC, managed partial reconfiguration and runtime manipulation of the configuration, each of which may be exploited in mission critical applications and online [144]. COTS ICs have also acquired EDC protection for internal registers and on-chip redundancy although such measures do not address the problem of eliminating faults at the outset [48], [4]. Furthermore, state of the art reconfigurable platforms now contain their own hard-wired microprocessor cores (and even larger pools of SRAM) fully integrated on-chip and therefore on line detect and thus self-repair has become an important area of research. Once again this indicates a merging between passive and active fault mitigation for microprocessors and reconfigurable platforms.

A clear assessment of technology readiness level (TRL) for zero-maintenance methods is difficult due to the emergent nature of most methods. Examples of state-of-the-art are listed in Table XVII based on the known examples. Factors that influence technology readiness include co-software/hardware demonstration level, proven reliability of the design and assessment of potential impact on fabrication cost. This is evident in examples of microprocessor design [19], programmable logic chips [42] and RF microelectronics [86]. Future applications include security (attack and defend), FDIR (Fault detection, isolation, recovery) and ISHM (Integrated System Health Management) [131]. An important indicator in this context will be integration readiness level (IRL) proposed for evaluating the complexity of integrating related strategies into existing space exploration systems [145].

VI. Conclusions and outlook

Over the past few years, a number of maintenance requirements have emerged within engineering systems and have gained considerable focus in the areas of detection, classification and self-recovery from fault conditions. The increasing demand for robust products, composed of complex sub-systems that must maintain the longest possible operational life-span, has brought self-repairing capability firmly into the design and manufacturing fray. At the same time, electronic sub-systems have become particularly vulnerable to environmentally-induced random and cumulative fault conditions and thus a significant body of literature has appeared that seeks to exploit the unique flexibility available within electronics for fault mitigation. As such, the authors have observed a number of recent trends paving the way towards zero-maintenance designs for electronics with the common goal of regaining: robust, error-resilient operation with major impacts expected in product availability and cost of maintenance. From fault detection and classification to passive and active fault mitigation, integrated self-recovery is becoming

TABLE XVI
DETAILED HIERARCHY FOR ZERO-MAINTENANCE STRATEGIES

Method	Benefits	Challenges
Status reporting	Logging key metrics (remaining capacity, faults classification etc.) to assist maintenance planning	Fault signals must traverse multiple design levels; effective filtering and interpreting of signals at application layer
Self-maintenance	Correction faults in-service (and online where possible) using a variety of actions to prolong operational window	Will most likely require several strategies across multiple integrated levels: masking, detection/reporting and active reallocation
Built-in self-test and repair (BISTAR)	Combined detection and restoration on chip	Requires detect and active logic for reallocation of resources. Difficult to validate all possible states
Built-in reconfiguration	Capability to restore correct operation in the event of fault	Relies upon a reconfigurable chip. Usually requires external initiation
Supervised repair (production yield enhancement)	Highly efficient reallocation of resources for reduced failure rate	Requires complex production tester and precisely controlled environment
Built-in self-test (BIT)	Relatively simple logic overhead; detection of errors at specific intervals	Often limited to detecting persistent faults power on time
Self-preservation / fault avoidance	Put measures in place to reduce impact of further faults	Requires accurate monitoring, possibly anticipation of impending fault events
Fault masking	Fixed structure; validation is relatively easy	Of limited fault capacity, can become potentially high resource overhead for fine-grained strategies

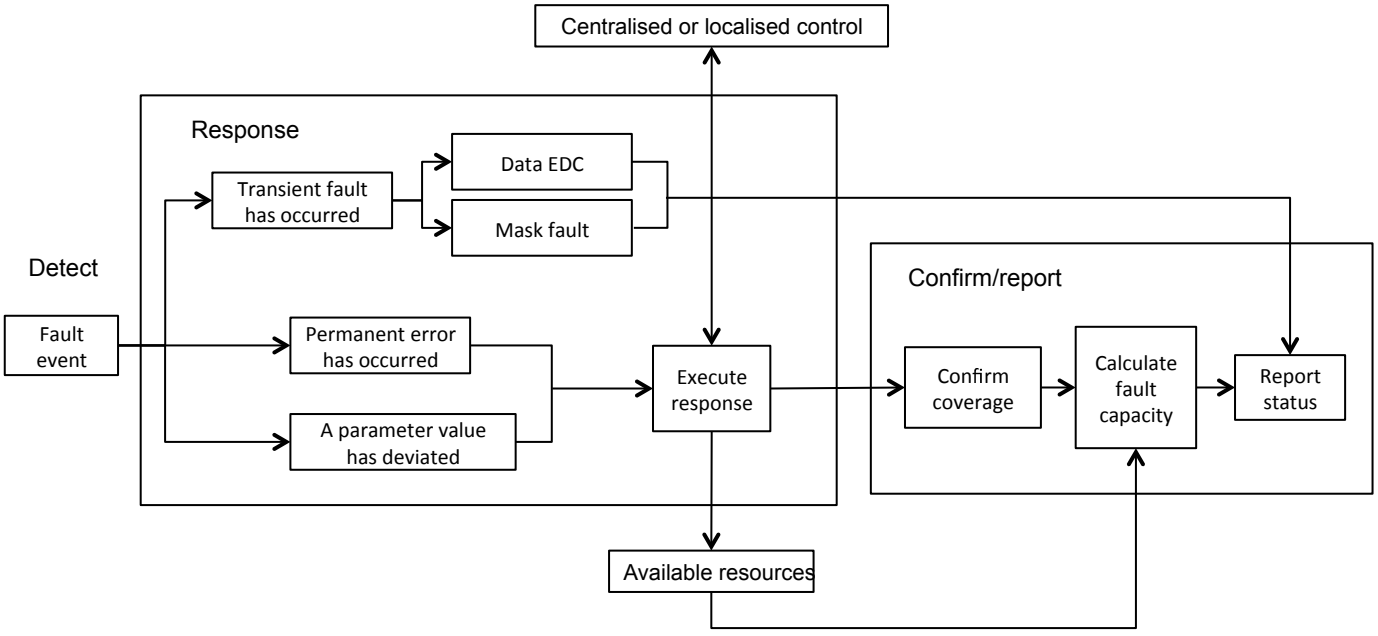

Fig. 5. Depiction of a possible top-level design of zero-maintenance systems with various options. Several implementations are possible: the example shown here includes masking and active response for permanent error or analogue parameter. Collection of status including evaluation of available resources are also shown.

TABLE XVII
APPLICATION EXAMPLES LINKED TO ZERO-MAINTENANCE FOR HIGH-VALUE SECTORS

	Application	Faults	Status	Opportunities for zero-maintenance
Aviation	Civil and military aircraft; autonomous flight	Radiation effects from ground to 60,000ft	nMR; mutual-checking/voting systems; resilient sensors for test beds [146]	Predictable recovery; assistive fault monitoring and discrimination
Energy	Wind turbine and nuclear decommissioning; inspection; disaster assessment	Heavy radiation; remote servicing/inspection; detection & monitoring for harsh environments	Fault-detection and diagnosis [147]; robotic inspection [148];	Self-maintaining control systems; sustained operation; functional hardening [149]
Space	Exploration and orbital communications systems	SEU/MBU mitigation	Modularized electronics, interconnect [150]; resilient SoC [151], transceivers [152] & redundancy [66], for spacecraft altitude [153]	Unmanned systems that self-maintain (e.g. Skylon); modular spares; sophisticated fault diagnosis [131]
Vehicles	Dependability of safety-critical subsystems	Vibration; temperature cycling; ground level SEU	Reconfigurable MCM platforms; Secure Soc; electromechanical redundancy, for electric ship power systems [154]	The self-healing vehicle [155]; mitigation for systemic failures; fail-save; rapid recovery
Rail	On-board power and control systems; electrification	Ageing; fault prevention and discrimination	Power-on self-test routines for power systems. Guided repair [156]	Enhanced fault logging & localization; maintaining predictable operation
Mining	Autonomous exploration; pipeline inspection	Extreme environment (drilling, mining, deep sea)	Autonomous vehicles; extreme environment sensing and predictive failure monitoring [157]	Prolonged unmanned operation with limited communications; fault logging collaborative systems

adopted within the electrical and electronic discipline; coupled systems with significant benefits to be seen across transport, health care, mining/exploration, nuclear energy and space exploration engineering systems. The aim was to set out the potential impacts with regard to intermittent fault scenarios and the overall maintenance cost overheads, that plague modern complex systems. This is principally due to the fact that:

- Electronics support the necessary flexibility for provision of resources and *built-in intelligence*;
- Upset mechanisms are projected to become more prevalent within next-generation nanoscale design;
- Aggressive scaling of COTS devices towards the necessary complexity, especially with regard to online detection and resource allocation.

In the light of the above mentioned, the authors had noted significant recent research activity reported in future nanoscale analogue and digital design, contemporary CMOS, current and future configurable integrated circuits, electromechanical sub-systems and assemblies, complex mechatronic and cyber physical systems. As a consequence, this paper considered the current and future technology trends that will make possible a zero-maintenance design; that is based on masking and active mitigation techniques driven by emerging nano- and printable-electronics technologies. This includes built-in fault detection and logging, fine-grained redundancy (with new possibilities for masking), self-reconfiguration and self-reallocation. An alternative perspective is that zero-maintenance is composed of methods for self-monitoring and self-management of internal redundant resources. It is also clear that, rather than being outdated, passive masking strategies have an important role to play in some environments though they become ineffectual as ageing-related faults begin to manifest.

A. Future opportunities

Several opportunities exist for fundamental advancement in this area: novel detection mechanisms need to be identified to underpin masking and reconfiguration strategies; The handling of interconnect-related failures is sorely neglected [8] and needs further research work to understand its impact upon design for zero-maintenance. Finally, Combining multiple fine-grained sensing and repair strategies for both manufacturing yield and through-life maintenance will lead to more efficient resource reuse. The associated design challenges are equally significant: i) justifying cost of resources and design complexity (including test and evaluation); ii) gaining confidence in new strategies; iii) overcoming barriers for certification in certain application areas; iv) understanding the options for integrating zero-maintenance at various design levels (and potentially cross-layered approaches). In most instances, there is a lack of EDA tool support for effective design exploration and evaluation. From the multitude of strategies considered, it seems clear that no single approach to zero-maintenance in electronics will win and that a multi-objective design approach is necessary

Design for zero-maintenance incurs many competing design goals across several areas in electronics sub-systems. These systems are composed of many sub-systems that must operate error-free otherwise downtime leads to a direct loss of service high MRO costs. At the chip-level, speed, efficiency and cost per unit area are paramount though test and verification complexity are becoming limiting factors. Board-level and higher levels bring in new factors including weight, absolute

cost, integration density and accessibility for inspection. At the top design level, inter-modular issues are prevalent including cabling and connectors, density issues (e.g., thermal management) and efficiency of space usage (e.g., in compact avionics bay). Once again, the test and verification complexity is ever-increasing. Traditional approaches such as design for reliability, component screening and modular redundancy have been enlisted with mission critical systems but their cost is often viewed as being unjustifiable. New low-level capabilities promise to bring underlying fault detection and discrimination but these capabilities must be integrated.

Where possible, zero-maintenance techniques should be tightly integrated with maintenance-related technologies. This is possible owing to the already mature hierarchical architecture of microelectronics design, though perhaps less well-developed at the higher board, sub-module levels. The potential impact brought about by zero-maintenance should not be underestimated, especially for industrial applications that are heavily penalised by downtime. At the same time, autonomous maintenance strategies may lead to difficulties in certification. For example, if the trustworthiness of a sub-component is dependent on knowing its exact configuration at any time, then self-configuring approaches may be challenging, and fault masking may be the only route. Information arising from fault registration, discrimination and localisation will provide new opportunities for assessing system status in terms of fault capacity, environmental harshness and onset of ageing. This could potentially revolutionise predictive and scheduled maintenance as well as better-informing major MRO and major failure events as well as offering self-preservation capabilities when faults can be predicted. Fault localisation offers direct benefits for fault tracing in complex systems and to avoid no-fault found scenarios. However, it is less obvious how zero-maintenance strategies will directly solve this particular problem and they may instead reside alongside bespoke fault locating strategies.

Active mitigation methods are undoubtedly closest to the vision of self-healing systems, but are still reliant upon effective (re-)organisation of their internal resources. In this context, self-healing may not become autonomic (i.e., fully self-initiated) due to the need for fault registration and reporting of events where resources have been re-organised, re-generated or drawn down from external sources [22]. An example of a process that is part self-healing is seen in [158], in which a data storage device contains embedded heating elements able to restore neighbouring non-responsive storage cells. This has the effect of re-establishing the correct material response and so the recovery of degraded memory cells is accelerated. The precise relationships between self-maintenance and self-healing technologies have yet not been established.

It could be argued that a better balance must be achieved between preventing faults from ever occurring in the first place and mitigation, e.g., super conducting quantum locked fields for space travel that may provide both simultaneous propulsion and strong shielding against radiation particles for both passengers and electronic systems. Such developments could help readdress the balance between fault prevention/mitigation coupled hardware/software design. In the event of disruptive future technologies, one might still ask what would be the consequences of failures of the clever idea itself i.e., in the above example resulting in loss of shielding? This still validates the wider arguments in this article even if there were a paradigm shift in hardware/software capabilities.

Despite recent advancements, the building blocks of design for zero-maintenance need to be better understood. The traditional paradigm of multi-level design is well-suited, but a holistic approach is needed to combine strategies potentially across multiple design levels. An evolution from tolerance towards active mitigation has been observed in electronics, due in part to projected capabilities of (and manufacturing challenges associated with) nanoscale electronics. However, although emerging strategies promise much for zero-maintenance capabilities within future electronic systems, their success is dependent upon a more precise and predictable evaluation of the associated resource overhead, performance impact and fault capacity metrics. The future of this field looks promising and equally pervasive to a host of applications and domains. We ourselves are looking towards development of practical demonstrations of whole systems that exhibit this philosophy of zero-maintenance, beyond just electronics but also mechanical and robotic systems, particularly within the manufacturing and through-life services industries, where we will likewise also outline our perspectives, challenges and opportunities.

REFERENCES

[1] R. Roy, J. A. Erkoyuncu, and A. Shaw, "The Future of Maintenance for Industrial Product-Service Systems," in *Product-Service Integration for Sustainable Solutions* (H. Meier, ed.), Lecture Notes in Production Engineering, pp. 1–15, Springer Berlin Heidelberg, Jan. 2013.

[2] M. Farnsworth, R. McWilliam, S. Khan, C. Bell, and A. Tiwari, *Design for Zero-Maintenance*. In: Redding L., Roy R., Shaw A. (eds) Advances in Through-life Engineering Services. Decision Engineering. Springer, 2017.

[3] X. She and K. McElvain, "Time Multiplexed Triple Modular Redundancy for Single Event Upset Mitigation," *IEEE Transactions on Nuclear Science*, vol. 56, pp. 2443–2448, Aug. 2009.

[4] K. Reick, P. Sanda, S. Swaney, J. Kellington, M. Mack, M. Floyd, and D. Henderson, "Fault-Tolerant Design of the IBM Power6 Microprocessor," *IEEE Micro*, vol. 28, no. 2, pp. 30–38, 2008.

[5] M. Breuer, S. Gupta, and T. M. Mak, "Defect and error tolerance in the presence of massive numbers of defects," *IEEE Design Test of Computers*, vol. 21, no. 3, pp. 216–227, 2004.

[6] K. Morgan, D. McMurtrey, B. Pratt, and M. Wirthlin, "A Comparison of TMR With Alternative Fault-Tolerant Design Techniques for FPGAs," *Nuclear Science, IEEE Transactions on*, vol. 54, pp. 2065 –2072, Dec. 2007.

[7] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for FPGAs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, pp. 501–533, Apr. 2006.

[8] M. G. Parris, C. A. Sharma, and R. F. Demara, "Progress in Autonomous Fault Recovery of Field Programmable Gate Arrays," *ACM Comput. Surv.*, vol. 43, pp. 31:1–31:30, Oct. 2011.

[9] E. Stott, P. Sedcole, and P. Cheung, "Fault tolerant methods for reliability in FPGAs," in *International Conference on Field Programmable Logic and Applications, 2008. FPL 2008*, pp. 415 –420, Sept. 2008.

[10] L. Sterpone, *Electronics System Design Techniques for Safety Critical Applications*. Springer, 1st ed., 2008.

[11] L. Bauer, C. Braun, M. Imhof, M. Kochte, H. Zhang, H. Wunderlich, and J. Henkel, "OTERA: Online test strategies for reliable reconfigurable architectures #x2014; Invited paper for the AHS-2012 special session #x201c;Dependability by reconfigurable hardware #x201d;," in *2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 38–45, June 2012.

[12] J. Henkel, L. Bauer, H. Zhang, S. Rehman, and M. Shafique, "Multi-Layer Dependability: From Microarchitecture to Application Level," in *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, (New York, NY, USA), pp. 47:1–47:6, ACM, 2014.

[13] W. Carter and W. Bouricius, "A Survey of Fault Tolerant Computer Architecture and its Evaluation," *Computer*, vol. 4, no. 1, pp. 9–16, 1971.

[14] R. A. Maxion, D. P. Siewiorek, and S. A. Elkind, "Techniques and Architectures for Fault-Tolerant Computing," *Annual Review of Computer Science*, vol. 2, no. 1, pp. 469–520, 1987.

[15] S. Hamdioui, G. Gaydadjiev, and A. J. Van de Goor, "The state-of-art and future trends in testing embedded memories," in *Records of the 2004 International Workshop on Memory Technology, Design and Testing, 2004*, pp. 54–59, 2004.

[16] M. A. Trefzer and A. M. Tyrrell, *Evolvable Hardware: From Practice to Application*. Springer, Sept. 2015.

[17] S. Michalak, A. Dubois, C. Storlie, H. Quinn, W. Rust, D. DuBois, D. Modl, A. Manuzzato, and S. Blanchard, "Neutron Beam Testing of High Performance Computing Hardware," in *2011 IEEE Radiation Effects Data Workshop (REDW)*, pp. 1–8, 2011.

[18] A. Patel and K. Prakash, "Fault-tolerant features of modern processors - A case study," 2010.

[19] J. Rivers, M. Gupta, J. Shin, P. Kudva, and P. Bose, "Error Tolerance in Server Class Processors," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, pp. 945 –959, July 2011.

[20] C. . D. S. Gao, Zhiwei; Cecati, "A survey of fault diagnosis and fault-tolerant techniques-part ii: fault diagnosis with knowledge-based and hybrid / active approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3768–3774, 2015.

[21] C. . D. S. Gao, Zhiwei; Cecati, "A survey of fault diagnosis and fault-tolerant techniques part i: Fault diagnosis with model-based and signal-based approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[22] R. Frei, R. McWilliam, B. Derrick, A. Purvis, A. Tiwari, and G. D. M. Serugendo, "Self-healing and self-repairing technologies," *The International Journal of Advanced Manufacturing Technology*, vol. 69, pp. 1033–1061, Nov. 2013.

[23] H. Psaier and S. Dustdar, "A survey on self-healing systems: approaches and systems," *Computing*, vol. 91, pp. 43–73, Jan. 2011.

[24] S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, and S. Kokaji, "Self-Repairing Mechanical Systems," *Autonomous Robots*, vol. 10, pp. 7–21, Jan. 2001.

[25] H. Noura, D. Theilliol, J.-C. Ponsart, and A. Chamseddine, *Fault-tolerant Control Systems: Design and Practical Applications*. Springer Science & Business Media, July 2009.

[26] J. W. Sheaffer, D. P. Luebke, and K. Skadron, "The visual vulnerability spectrum: characterizing architectural vulnerability for graphics hardware," tech. rep., DTIC Document, 2006.

[27] Sandi Habnic, "Functional triple modular redundancy (FTMR)," Dec. 2002.

[28] M. Reorda, L. Sterpone, and A. Ullah, "An error-detection and self-repairing method for dynamically and partially reconfigurable systems," *IEEE Transactions on Computers*, vol. 66, no. 6, pp. 1022–1033, 2017.

[29] O. Eldash, K. Khalil, and M. Bayoumi, "On on-chip intelligence paradigms," *Electrical and Computer Engineering (CCECE) IEEE 30th Canadian Conference*, vol. 1, no. 1, pp. 1–6, 2017.

[30] A. K.-K. Wong, *Resolution Enhancement Techniques in Optical Lithography*. SPIE Press, 2001.

[31] B. Parhami, "Defect, Fault, Error,..., or Failure?," *IEEE Transactions on Reliability*, vol. 46, pp. 450–451, Dec. 1997.

[32] P. Hazucha, T. Karnik, J. Maiz, S. Walstra, B. Bloechel, J. Tschanz, G. Dermer, S. Hareland, P. Armstrong, and S. Borkar, "Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-/spl mu/m to 90-nm generation," in *Electron Devices Meeting, 2003. IEDM '03 Technical Digest. IEEE International*, pp. 21.5.1–21.5.4, 2003.

[33] R. G. Bennetts, *Introduction to Digital Board Testing*. Crane, Russak, 1982.

[34] S. Khan, P. Phillips, C. Hockley, and I. Jennions, "No fault found events in maintenance engineering part 2: Root causes, technical developments and future research," *Reliability Engineering and System Safety*, vol. 123, no. 1, pp. 196–208, 2014.

[35] V. Agrawal, C. Kime, and K. Saluja, "A tutorial on built-in self-test. I. Principles," *IEEE Design Test of Computers*, vol. 10, no. 1, pp. 73–82, 1993.

[36] P. O'Connor and A. Kleyner, *Practical Reliability Engineering*. John Wiley & Sons, Nov. 2011.

[37] F. Sexton, "Destructive single-event effects in semiconductor devices and ICs," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 603–621, 2003.

[38] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, Sept. 2010.

[39] M. Ebrahimi, S. Miremadi, H. Asadi, and M. Fazeli, "Low-Cost Scan-Chain-Based Technique to Recover Multiple Errors in TMR Systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, pp. 1454–1468, Aug. 2013.

[40] I. Koren and A. Singh, "Fault tolerance in VLSI circuits," *Computer*, vol. 23, pp. 73–83, July 1990.

[41] A. Agarwal, J. Cong, and B. Tagiku, "The Survivability of Design-specific Spare Placement in FPGA Architectures with High Defect Rates," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, pp. 33:1–33:22, Apr. 2013.

[42] J. Walker, M. Trefzer, S. Bale, and A. Tyrrell, "PAnDA: A Reconfigurable Architecture that Adapts to Physical Substrate Variations," *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1584–1596, 2013.

[43] J. Han and P. Jonker, "A defect- and fault-tolerant architecture for nanocomputers," *Nanotechnology*, vol. 14, pp. 224–230, Feb. 2003.

[44] R. Sedmark, "Boundary-scan: beyond production test," in *, 12th IEEE VLSI Test Symposium, 1994. Proceedings*, pp. 415–420, 1994.

[45] A. Hassan, V. Agarwal, B. Nadeau-Dostie, and J. Rajski, "BIST of PCB interconnects using boundary-scan architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 10, pp. 1278–1288, 1992.

[46] M. Farnsworth, C. Bell, S. Khan, and T. Tomiyama, "Autonomous Maintenance for Through-Life Engineering," in *Through-life Engineering Services* (L. Redding and R. Roy, eds.), Decision Engineering, pp. 395–419, Springer International Publishing, 2015.

[47] B. Allen, "Monitoring hard disks with smart," *Linux J.*, no. 117, p. 9, 2004.

[48] J. Andersson, J. Gaisler, and R. Weigand, "Next Generation MultiPurpose Microprocessor," vol. 682, p. 7, Aug. 2010.

[49] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky, "Are Disks the Dominant Contributor for Storage Failures?: A Comprehensive Study of Storage Subsystem Failure Characteristics," *Trans. Storage*, vol. 4, pp. 7:1–7:25, Nov. 2008.

[50] H. V. Allen, S. C. Terry, and D. W. De Bruin, "Accelerometer systems with self-testable features," *Sensors and Actuators*, vol. 20, no. 1, pp. 153–161, 1989.

[51] Q. Jia, "Write Fault Protection Against Shock Disturbance in Hard Disk Drives Without a Shock Sensor," *IEEE Transactions on Magnetics*, vol. 43, pp. 3689–3693, Sept. 2007.

[52] M.-C. Hsueh, T. Tsai, and R. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, pp. 75–82, Apr. 1997.

[53] S. Chau, "Fault injection boundary scan design for verification of fault tolerant systems," in *Test Conference, 1994. Proceedings., International*, pp. 677–682, 1994.

[54] T. Chakraborty and C.-H. Chiang, "A novel fault injection method for system verification based on FPGA boundary scan architecture," in *Test Conference, 2002. Proceedings. International*, pp. 923–929, 2002.

[55] H. Quinn, D. Black, W. Robinson, and S. Buchner, "Fault Simulation and Emulation Tools to Augment Radiation-Hardness Assurance Testing," *IEEE Transactions on Nuclear Science*, vol. 60, pp. 2119–2142, June 2013.

[56] ESA, "Techniques for Radiation Effects Mitigation in ASICs and FPGAs," Dec. 2011.

[57] A. Avizienis, "Fault-tolerance: The survival attribute of digital systems," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1109–1125, 1978.

[58] J. Emmert, C. Stroud, and M. Abramovici, "Online Fault Tolerance for FPGA Logic Blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, pp. 216–226, Feb. 2007.

[59] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata studies*, vol. 34, pp. 43–98, 1956.

[60] P. A. Jensen, "Quadded NOR Logic," *IEEE Transactions on Reliability*, vol. R-12, pp. 22 –31, Sept. 1963.

[61] J. Losq, "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," *IEEE Transactions on Computers*, vol. C-25, pp. 569–578, June 1976.

[62] A. M. Tyrrell, "Fault Tolerant Applications," in *Evolvable Hardware*, Natural Computing Series, pp. 191–207, Springer Berlin Heidelberg, 2015. DOI: 10.1007/978-3-662-44616-4_7.

[63] M. A. Trefzer, J. A. Walker, S. J. Bale, and A. M. Tyrrell, "Fighting stochastic variability in a D-type flip-flop with transistor-level reconfiguration," *IET Computers Digital Techniques*, vol. 9, no. 4, pp. 190–196, 2015.

[64] Eldon Hall, "General Design Characteristics of the Apollo Guidance Computer," June 1996.

[65] M. Straka, J. Kastil, and Z. Kotasek, "Fault Tolerant Structure for SRAM-Based FPGA via Partial Dynamic Reconfiguration," in *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, pp. 365–372, 2010.

[66] M. Niknahad, O. Sander, and J. Becker, "Fine grain fault tolerance- A key to high reliability for FPGAs in space," in *2012 IEEE Aerospace Conference*, pp. 1 –10, Mar. 2012.

[67] V. Petrovic and M. Krstic, "Design Flow for Radhard TMR Flip-Flops," in *2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pp. 203–208, Apr. 2015.

[68] K. Chapman, *SEU Strategies for Virtex-5 Devices*, vol. XAPP864. 2010.

[69] A. Namazi and M. Nourani, "Gate-Level Redundancy: A New Design-for-Reliability Paradigm for Nanotechnologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 775 –786, May 2010.

[70] J. Han, E. Leung, L. Liu, and F. Lombardi, "A Fault-Tolerant Technique Using Quadded Logic and Quadded Transistors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–1, 2014.

[71] M. Niknahad, O. Sander, and J. Becker, "QFDR-an integration of Quadded Logic for modern FPGAs to tolerate high radiation effect rates," in *2011 12th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pp. 119–122, Sept. 2011.

[72] A. El-Maleh, B. Al-Hashimi, A. Melouki, and F. Khan, "Defect-tolerant n2-transistor structure for reliable nanoelectronic designs," *IET Computers Digital Techniques*, vol. 3, pp. 570 –580, Nov. 2009.

[73] A. Mukherjee and A. S. Dhar, "New triple-transistor based defect-tolerant systems for reliable digital architectures," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1917–1920, May 2015.

[74] J. Han, J. Gao, P. Jonker, Y. Qi, and J. Fortes, "Toward hardware-redundant, fault-tolerant logic for nanoelectronics," *IEEE Design Test of Computers*, vol. 22, pp. 328 – 339, Aug. 2005.

[75] R. Kumawat, V. Sahula, and M. S. Gaur, "Reliable circuit analysis and design using nanoscale devices," *Proc . of SPIE*, vol. 8760, pp. 87602C–87602C, Jan. 2013.

[76] M. Straka, J. Kastil, Z. Kotasek, and L. Miculka, "Fault tolerant system design and SEU injection based testing," *Microprocessors and Microsystems*, no. 0.

[77] B. Pratt, M. Fuller, M. Rice, and M. Wirthlin, "Reduced-Precision Redundancy for Reliable FPGA Communications Systems in High-Radiation Environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 1, pp. 369–380, 2013.

[78] M. Sankaranarayanan and A. Vaidyanathan, "Black box model based self healing solution for stuck at faults in digital circuits," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 5, pp. 2451–2458, 2017.

[79] K. Chakraborty and P. Mazumder, *Fault Tolerance and Reliability Techniques for High Density Random Access Memories*. Prentice Hall PTR, 2002.

[80] S. Kimura, M. Takahashi, T. Okuyama, S. Tsuchiya, and Y. Suzuki, "A fault-tolerant control algorithm having a decentralized autonomous architecture for space hyper-redundant manipulators," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, pp. 521 –527, July 1998.

[81] A. Alameldeen, Z. Chishti, C. Wilkerson, W. Wu, and S.-L. Lu, "Adaptive Cache Design to Enable Reliable Low-Voltage Operation," *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 50–63, 2011.

[82] R. Kothe, H. Vierhaus, T. Coym, W. Vermeiren, and B. Straube, "Embedded Self Repair by Transistor and Gate Level Reconfiguration," in *Design and Diagnostics of Electronic Circuits and systems, 2006 IEEE*, pp. 208 –213, 2006.

[83] F. de Novaes Kucinskis and M. G. V. Ferreira, "Taking the ECSS autonomy concepts one step further," in *SpaceOps 2010 Conference Delivering on the Dream Hosted by NASA Mars*, pp. 25–30, 2010.

[84] X. Wendling, R. Rochet, and R. Leveugle, "ROM-based synthesis of fault-tolerant controllers," in *, 1996 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 1996. Proceedings*, pp. 304–308, 1996.

[85] Q. Wu, G. Dong, and T. Zhang, "A First Study on Self-Healing Solid-State Drives," in *Memory Workshop (IMW), 2011 3rd IEEE International*, pp. 1 –4, May 2011.

[86] A. Goyal, M. Swaminathan, A. Chatterjee, D. Howard, and J. Cressler, "A New Self-Healing Methodology for RF Amplifier Circuits Based on Oscillation Principles," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, pp. 1835 –1848, Oct. 2012.

[87] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. Higgins, and J. Lewandowski, "Built in self repair for embedded high density SRAM," in *Test Conference, 1998. Proceedings., International*, pp. 1112–1119, 1998.

[88] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[89] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing*, vol. 107, no. 1, pp. 241–265, 2018.

[90] A. H. El-Maleh, "A sequential circuit fault tolerance technique with enhanced area and power," in *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 301–304, Dec. 2015.

[91] I. Reis, P. Collins, and M. van Houcke, "On-line Boundary-Scan Testing in Service of Extended Products," in *Test Conference, 2006. ITC '06. IEEE International*, pp. 1–10, Oct. 2006.

[92] L. Whetsel, "Built-in self-test (BIST) using boundary scan," 1996.

[93] Y. Zorian, "A structured testability approach for multi-chip modules based on BIST and boundary-scan," *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, vol. 17, no. 3, pp. 283–290, 1994.

[94] B. Kim, A. Chatterjee, M. Swaminathan, and D. Schimmel, "A novel low-cost approach to MCM interconnect test," in *Test Conference, 1995. Proceedings., International*, pp. 184–192, 1995.

[95] Y. Fkih, P. Vivet, B. Rouzeyre, M.-L. Flottes, and G. Di Natale, "A 3d IC BIST for pre-bond test of TSVs using ring oscillators," in *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International*, pp. 1–4, 2013.

[96] I. Villalta, U. Bidarte, J. Gmez-Cornejo, J. Lzaro, and C. Cuadrado, "Dependability in FPGAs, a Review," in *2015 Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 1–6, Nov. 2015.

[97] S. Habermann, R. Kothe, and H. T. Vierhaus, "Built-in self repair by reconfiguration of FPGAs," in *On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International*, pp. 2–pp, 2006.

[98] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgeford, "Enhanced Architectures, Design Methodologies and Cad Tools for Dynamic Reconfiguration of Xilinx Fpgas," in *Proceedings of the 16th International Conference on Field Programmable Logic and Applications (FPL06)*, (Madrid, Spain), Aug. 2006.

[99] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici, "Dynamic fault tolerance in FPGAs via partial reconfiguration," in *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, pp. 165 –174, 2000.

[100] M. Krcma, Z. Kotasek, and J. Kastil, "Fault tolerant Field Programmable Neural Networks," in *Nordic Circuits and Systems Conference (NORCAS): NORCHIP International Symposium on System-on-Chip (SoC), 2015*, pp. 1–4, Oct. 2015.

[101] R. Kaushik, J. Byunghoo, P. Dimitrios, and R. Anand, "Integrated systems in the more-than-moore era: designing low-cost energy-efficient systems using heterogeneous components," *IEEE Design and Test*, vol. 33, no. 3, pp. 56–65, 2016.

[102] M. Kawanaka, M. Tsunoyama, and S. Naito, "A fault-tolerant parallel processor modeled by a two-dimensional linear cellular automaton," *Systems and Computers in Japan*, vol. 25, pp. 1–11, Mar. 2007.

[103] D. Jones, R. McWilliam, and A. Purvis, "Designing convergent cellular automata.," *Biosystems*, vol. 96, no. 1, pp. 80–85, 2008.

[104] K. Nagami, K. Oguri, T. Shiozawa, H. Ito, and R. Konishi, "Plastic cell architecture: towards reconfigurable computing for general-purpose," in *IEEE Symposium on FPGAs for Custom Computing Machines, 1998. Proceedings*, pp. 68–77, 1998.

[105] W. Barker, D. Halliday, Y. Thoma, E. Sanchez, G. Tempesti, and A. Tyrrell, "Fault Tolerance Using Dynamic Reconfiguration on the POEtic Tissue," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 666 –684, Oct. 2007.

[106] A. M. Tyrrell and A. J. Greensted, "Evolving dependability," *J. Emerg. Technol. Comput. Syst.*, vol. 3, July 2007.

[107] M. Samie, G. Dragffy, and T. Pipe, "Novel bio-inspired self-repair algorithm for evolvable fault tolerant hardware systems," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary*

Computation Conference: Late Breaking Papers, GECCO '09, (New York, NY, USA), pp. 2143–2148, ACM, 2009.

[108] D. Jose and R. Tamilselvan, "Fault tolerant and energy efficient signal processing on fpga using evolutionary techniques," *Computational Intelligence, Cyber Security and Computational Models. Springer*, vol. 1, no. 1, pp. 155–164, 2016.

[109] X. Yang, Y. Li, and Y. Tong, "Application of interactive evolutionary strategy in fault-tolerant system capable of online self-repairing," *International Journal of Computational Science and Engineering*, vol. 15, no. 1, pp. 57–65, 2016.

[110] T. Koal, S. Scharoba, and H. T. Vierhaus, "Combining Correction of Delay Faults and Transient Faults," in *2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pp. 99–102, Apr. 2015.

[111] J. Sklaroff, "Redundancy Management Technique for Space Shuttle Computers," *IBM Journal of Research and Development*, vol. 20, no. 1, pp. 20–28, 1976.

[112] T. Koal, M. Ulbricht, and H. Vierhaus, "Virtual TMR Schemes Combining Fault Tolerance and Self Repair," in *2013 Euromicro Conference on Digital System Design (DSD)*, pp. 235–242, 2013.

[113] R. McWilliam, P. Schiefer, and A. Purvis, "Demonstration of a Self-recovering ALU Using a Convergent Cellular Automata," *Procedia CIRP*, vol. 11, pp. 373–378, 2013.

[114] F. Kastensmidt and P. Rech, "Radiation Effects and Fault Tolerance Techniques for FPGAs and GPUs," in *FPGAs and Parallel Architectures for Aerospace Applications* (F. Kastensmidt and P. Rech, eds.), pp. 3–17, Springer International Publishing, 2016. DOI: 10.1007/978-3-319-14352-1_1.

[115] N. Campregher, P. Y. K. Cheung, G. Constantinides, and M. Vasilko, "Reconfiguration and Fine-Grained Redundancy for Fault Tolerance in FPGAs," in *International Conference on Field Programmable Logic and Applications, 2006. FPL '06*, pp. 1–6, 2006.

[116] P. Schiefer, R. McWilliam, and A. Purvis, "Fault Tolerant Quadded Logic Cell Structure with Built-in Adaptive Time Redundancy," *Procedia CIRP*, vol. 22, pp. 127–131, 2014.

[117] R. Moric, B. J. Phillips, and M. J. Liebelt, "Defect tolerant prefix adder design," vol. 7268, pp. 72680F–72680F–9, 2008.

[118] P. Bremner, Y. Liu, M. Samie, G. Dragffy, A. G. Pipe, G. Tempesti, J. Timmis, and A. M. Tyrrell, "SABRE: a bio-inspired fault-tolerant electronic architecture," *Bioinspiration & Biomimetics*, vol. 8, p. 016003, Mar. 2013.

[119] M. Kawanaka, M. Tsunoyama, and S. Naito, "A fault-tolerant parallel processor modeled by a two-dimensional linear cellular automaton," *Systems and Computers in Japan*, vol. 25, no. 6, pp. 1–11, 1994.

[120] N. Kamiura, Y. Hata, and K. Yamato, "A repairable and diagnosable cellular array on multiple-valued logic," in , *Proceedings of The Twenty-Third International Symposium on Multiple-Valued Logic, 1993*, pp. 92–97, 1993.

[121] S. Mitra, W.-J. Huang, N. Saxena, S.-Y. Yu, and E. McCluskey, "Reconfigurable architecture for autonomous self-repair," *Design Test of Computers, IEEE*, vol. 21, pp. 228 – 240, June 2004.

[122] S. Habermann, R. Kothe, and H. T. Vierhaus, "Built-in Self Repair by Reconfiguration of FPGAs," in *Proceedings of the 12th IEEE International Symposium on On-Line Testing*, IOLTS '06, (Washington, DC, USA), pp. 187–188, IEEE Computer Society, 2006.

[123] J. Huang, M. Tahoori, and F. Lombardi, "Fault tolerance of switch blocks and switch block arrays in FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 7, pp. 794–807, 2005.

[124] F. Smith, "A new methodology for single event transient suppression in flash FPGAs," *Microprocessors and Microsystems*, vol. 37, pp. 313–318, May 2013.

[125] W.-H. Chen and B. Jung, "Self-Healing Phase-Locked Loops in Deep-Scaled CMOS Technologies," *Design Test of Computers, IEEE*, vol. 27, pp. 18 –25, Dec. 2010.

[126] B. J. Flehinger, "Reliability Improvement through Redundancy at Various System Levels," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 148–158, 1958.

[127] F.-B. Sun and S. Zhang, "Does Hard Disk Drive Failure Rate Enter Steady-State Afer One Year?," in *Reliability and Maintainability Symposium, 2007. RAMS '07. Annual*, pp. 356–361, Jan. 2007.

[128] Y. Wang, Q. Miao, E. Ma, K.-L. Tsui, and M. Pecht, "Online Anomaly Detection for Hard Disk Drives Based on Mahalanobis Distance," *IEEE Transactions on Reliability*, vol. 62, pp. 136–145, Mar. 2013.

[129] S. Kamarthi, A. Zeid, and Y. Bagul, "Assessement of current health of hard disk drives," in *IEEE International Conference on Automation Science and Engineering, 2009. CASE 2009*, pp. 246–249, Aug. 2009.

[130] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, "Reliable On-chip Systems in the Nano-era: Lessons Learnt and Future Trends," in *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, (New York, NY, USA), pp. 99:1–99:10, ACM, 2013.

[131] J. Schumann, O. J. Mengshoel, and T. Mbaya, "Integrated Software and Sensor Health Management for Small Spacecraft," in *Proc. of IEEE International Conference on Space Mission Challenges for Information Technology*, (Palo Alto, CA), pp. 77–84, 2011.

[132] H.-T. Lue, P.-Y. Du, C.-P. Chen, W.-C. Chen, C.-C. Hsieh, Y.-H. Hsiao, Y.-H. Shih, and C.-Y. Lu, "Radically extending the cycling endurance of Flash memory (to #x003e; 100m Cycles) by using built-in thermal annealing to self-heal the stress-induced damage," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, pp. 9.1.1–9.1.4, Dec. 2012.

[133] R. Velazco, P. Fouillat, and R. Reis, eds., *Radiation Effects on Embedded Systems*. Springer, 1 ed., May 2007.

[134] D. D. Sworder and T. Kazangey, "Optimal Control, Repair, and Inventory Strategies for a Linear Stochastic System," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 2, pp. 342 –347, July 1972.

[135] S. Carthik, S. Alireza, A. Ahmad, and D. Ronald, "Self-healing reconfigurable logic using autonomous group testing," *Microprocessors and Microsystems*, vol. 37, pp. 174–184, Oct. 2013.

[136] B. S. Dhillon, "Failure modes and effects analysis Bibliography," *Microelectronics Reliability*, vol. 32, pp. 719–731, May 1992.

[137] M. Balaz and S. Kristofik, "Generic Self Repair Architecture with Multiple Fault Handling Capability," in *2015 Euromicro Conference on Digital System Design (DSD)*, pp. 197–204, Aug. 2015.

[138] I. S. Haque and V. S. Pande, "Hard data on soft errors: A large-scale assessment of real-world error rates in gpgpu," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 691–696, IEEE, 2010.

[139] M. Dimitrov, M. Mantor, and H. Zhou, "Understanding software approaches for GPGPU reliability," in *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pp. 94–104, ACM, 2009.

[140] G. K. Fedder, T. Mukherjee, and L. Pileggi, "Self-configuring CMOS Microsystems," in *Control Technologies for Emerging Micro and Nanoscale Systems* (E. Eleftheriou and S. O. R. Moheimani, eds.), no. 413 in Lecture Notes in Control and Information Sciences, pp. 181–200, Springer Berlin Heidelberg, Jan. 2011.

[141] M. Farnsworth, A. Tiwari, M. Zhu, and E. Benkhelifa, "A multi-objective and multidisciplinary optimisation algorithm for micro-electromechanical systems," *Studies in Computational Intelligence*, vol. 731, pp. 205–238, 2018.

[142] M. Farnsworth, A. Tiwari, and M. Zhu, "Multi-level and multi-objective design optimisation of a mems bandpass filter," *Applied Soft Computing*, vol. 52, pp. 642–656, 2017.

[143] J. Podivinsky, O. Cekan, M. Simkova, and Z. Kotasek, "The evaluation platform for testing fault-tolerance methodologies in electro-mechanical applications," *Microprocessors and Microsystems*, vol. 39, pp. 1215–1230, Nov. 2015.

[144] F. L. Kastensmidt, R. Reis, and L. Carro, *Fault-Tolerance Techniques for SRAM-Based FPGAs*. Springer, Feb. 2007.

[145] B. Sauser, R. Gove, E. Forbes, and J. E. Ramirez-Marquez, "Integration maturity metrics: Development of an integration readiness level," *Information, Knowledge, Systems Management*, vol. 9, pp. 17–46, Jan. 2010.

[146] T. J. Kaiser, B. J. LaMeres, T. Buerkle, J. A. Hogan, and R. J. Weber, "Experimental Conformation of Ionizing Sensing for Space Radiation Environmental Awareness," *IEEE Sensors Journal*, vol. 16, pp. 3482–3483, May 2016.

[147] J. Ma and J. Jiang, "Applications of fault detection and diagnosis methods in nuclear power plants: A review," *Progress in Nuclear Energy*, vol. 53, pp. 255–266, Apr. 2011.

[148] J. De Geeter, M. Decrton, and E. Colon, "The challenges of telerobotics in a nuclear environment," *Robotics and Autonomous Systems*, vol. 28, pp. 5–17, July 1999.

[149] H. F. J. Shipurkar, Udai; Polinder, "A review of methods to increase the availability of wind turbine generator systems," *CPSS Transactions on Power Electronics and Applications*, vol. 1, no. 1, pp. 66–82, 2016.

[150] X. Iturbe, K. Benkrid, T. Arslan, C. Hong, A. Erdogan, and I. Martinez, "Enabling FPGAs for future deep space exploration missions: Improving fault-tolerance and computation density with R3tos," in *2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 104–112, 2011.

[151] X. Iturbe, D. Keymeulen, E. Ozer, P. Yiu, D. Berisford, K. Hand, and R. Carlson, "An integrated SoC for science data processing in next-generation space flight instruments avionics," in *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 134–141, Oct. 2015.
[152] S. Taube, V. Petrovic, and M. Krstic, "Fault tolerant implementation of a SpaceWire interface," in *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 614–617, Dec. 2014.
[153] B. D. S. Z. D. Yin, Shen; Xiao, "A review on recent development of spacecraft attitude fault tolerant control system," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3311–3320, 2016.
[154] J. A. S. Babaei, Maziar; Shi, "A survey on fault detection, isolation, and reconfiguration methods in electric ship power systems," *IEEE Access*, vol. 6, no. 1, pp. 9430–9441, 2018.
[155] M. Amor-Segan, R. McMurran, G. Dhadyalla, and R. Jones, "Towards the Self Healing Vehicle," in *2007 3rd Institution of Engineering and Technology Conference on Automotive Electronics*, pp. 1 –7, June 2007.
[156] A. Abdel-Malek, B. Scallan, J. M. Bruno, B. A. Mathewson, J. E. Schlabach, G. J. Fera, and I. Gomez, "Diagnosis and repair system and method," Apr. 2007. US7209817 B2.
[157] J. Sarangapani and D. R. Schricker, "Method and apparatus for predicting a fault condition," Sept. 1999. U.S. Classification 702/179, 702/181, 702/182, 701/1, 701/32.1; International Classification G05B23/02, G05B19/406; Cooperative Classification G05B23/0232; European Classification G05B23/02S4H2B.
[158] Q. Wu, G. Dong, and T. Zhang, "A First Study on Self-Healing Solid-State Drives," in *Memory Workshop (IMW), 2011 3rd IEEE International*, pp. 1–4, May 2011.