Emergent Design Challenges for Embedded Systems and Paths Forward: Mixed-criticality, Energy, Reliability and Security Perspectives

Special Session Paper

Siva Satyendra Sahoo[1], Akash Kumar[1], Martin Decky[2], Samuel C.B. Wong[3], Geoff V. Merrett[3], Yinyuan Zhao[4], Jiachen Wang[4], Xiaohang Wang[4], Amit Kumar Singh[5]
Technische Universität Dresden[1], Dresden Research Center of Huawei[2], University of Southampton[3], South China University of Technology[4], University of Essex[5]
{siva_satyendra.sahoo,akash.kumar}@tu-dresden.de,martin.decky@huawei.com,{scbw1g19,gvm}@ecs.soton.ac.uk
201630666455@mail.scut.edu.cn,jiachenwang3@gmail.com,xiaohangwang@scut.edu.cn,a.k.singh@essex.ac.uk

ABSTRACT

Modern embedded systems need to cater for several needs depending upon the application domain in which they are deployed. For example, mixed-critically needs to be considered for real-time and safety-critical systems and energy for battery-operated systems. At the same time, many of these systems demand for their reliability and security as well. With electronic systems being used for increasingly varying type of applications, novel challenges have emerged. For example, with the use of embedded systems in increasingly complex applications that execute tasks with varying priorities, mixed-criticality systems present unique challenges to designing reliable systems. The large design space involved in implementing cross-layer reliability in heterogeneous systems, particularly for mixed-critical systems, poses new research problems. Further, malicious security attacks on these systems pose additional extraordinary challenges in the system design. In this paper, we cover both the industry and academia perspectives of the challenges posed by these emergent aspects of system design towards designing high-performance, energy-efficient, reliable and/or secure embedded systems. We also provide our views on paths forward.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**.

KEYWORDS

Mixed-criticality, reliability, energy-efficiency, aging attack, covert channel attack, attack mitigation

ACM Reference Format:
Siva Satyendra Sahoo[1], Akash Kumar[1], Martin Decky[2], Samuel C.B. Wong[3], Geoff V. Merrett[3], Yinyuan Zhao[4], Jiachen Wang[4], Xiaohang Wang[4], Amit Kumar Singh[5]. 2021. Emergent Design Challenges for Embedded Systems and Paths Forward: Mixed-criticality, Energy, Reliability and Security Perspectives: Special Session Paper. In *2021 International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS '21), October 10–13, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3478684.3479246

1 INTRODUCTION

It is well known that there are several emergent design challenges for embedded systems. The reason being their increased usage in many application domains, e.g. automotive and smart homes, and their increased complexity to support a variety of applications or features in these application domains. Performance that relates to timing properties has always been of paramount importance in time-critical systems. It has also been focused extensively of other kinds of systems as well for better user experience, for example, frame rendering speed while watching a video [45].

The reliance of embedded systems on battery has demanded for energy-aware system design along with performance considerations [2, 46]. Optimisation of both energy and performance imposes challenges as the number of design points to be explored increases and various design points typically represent a trade-off between these two metrics. An embedded system consuming high energy typically dissipates high power as well and thus results in high temperature due to increased power density within the chip. Therefore, join optimisation of energy and temperature management has been extensively explored [47].

The rise in operating temperature of an embedded system leads to bad user experience and reliability issues. Reliability consideration during system design is being extensively explored [30, 41]. Modern embedded systems are also facing hardware security attacks of various types, e.g. denial of service (DoS) attack [53], aging attack [54] and covert channel attack [18]. In DoS, the needed service is denied by various means like blocking packets reaching a destination, and aging and covert channel attacks accelerate system aging and leak sensitive information, respectively.

This paper discusses emergent design challenges faced by the academia and industry while considering one or multiple metrics based on the design requirement, which typically depends on the application domain. Section 2 provides a industry perspective of the challenges and paths forward in ensuring reliability in embedded systems, particularly for mixed-criticality. Section 3 and Section 4 focus on the design space considerations and paths forward for energy-efficiency and cross-layer reliability, respectively. Section

5 focuses on the aging and covert channel attacks, their novel mitigation methods and paths forward. Finally, Section 6 concludes the paper.

2 MIXED-CRITICALITY SYSTEMS: THEORY VS. PRACTICE

The goal of mixed-criticality systems is to accommodate two types of workloads with extremely diverse requirement sets side-by-side. On the one hand, we have the high criticality workloads with strict requirements on safety, security, real-time behavior, etc. On the other hand, we have the low criticality workloads that prioritize raw performance and vendor/user customization. This section presents an overview of the industry best practices of designing and building hardware platforms and operating systems that form the foundation for mixed-criticality systems. We also discuss gaps between research and practice and how these gaps could be bridged.

2.1 Problem Statement

In an ideal world, a system that meets the maximal possible set of requirements in the domains of safety, security, integrity, reliability and dependability (high criticality for short) would be also considered a suitable system for the use cases where a lesser degree of adherence to the mentioned (and related) requirements is necessary (low criticality for short). Unfortunately, achieving the high-criticality requirements is not only costly at the design and implementation time, but a huge price needs to be also paid repeatedly during the deployment time.

The costs of achieving high-criticality requirements are both direct (e.g. the need to redo a labor-intensive and time-consuming validation, verification and certification of the whole system with each major change) and indirect (e.g. the increased and sometimes even prohibitive cost of certified hardware that would meet the performance requirements).

Moreover, high-criticality and low-criticality use cases differ also in many practical aspects. While high criticality is defined by a strict set of requirements that are well known at design time and rarely change, low criticality could be readily defined by the "good enough" criterion. This means a lack of a priori strict requirements which are replaced by an ever-changing set of business-oriented demands (time-to-market, user experience, etc.).

A system that is a mix of the high-criticality and low-criticality workloads calls for a combination of quite incompatible development processes and methodologies. High criticality typically requires the Waterfall model, V-Model or other strict requirement-driven approach. Low criticality typically requires a more iterative and prototype-based, if not even agile approach. A straightforward approach to reconcile these conflicting approaches is a complete physical separation – designing, implementing and maintaining the high-criticality and low-criticality parts as completely independent systems. The drawback of this approach is obviously in increased costs caused by the potential underutilization of resources and literally in the increased physical mass of the underlying hardware (which is costly in industries like automotive/aerospace).

Current efficient mixed-criticality systems try to reconcile the conflicting approaches in a different way: By providing a single platform that does not compromise the goals of the high-criticality and of the low-criticality parts and, at the same time, leverages the benefits of the consolidation. But similarly to everything else in life, there is no free lunch and the price that needs to be paid here is the increased complexity of the mixed-criticality system. Where the physically separated systems provide no mutual influence by default and any kind of desired interaction between high criticality and low criticality needs to be established explicitly, with mixed-criticality systems we need to explicitly seal all potential (covert) channels, leaving only those desirable channels open.

2.2 Constructing Mixed-criticality Systems in Academia

Academia and industry have historically taken different approaches towards constructing mixed-criticality systems. The pinnacle of the academic approach are methods such as worst-case execution time analysis, schedulability analysis and formal verification of correctness. These methods stand clearly on the high-criticality side of the mixed system and typically isolate the low-criticality parts into virtual machines that are only allowed to consume the spare resources of the system.

The benefits of the academic approach is the strength of the guarantees it can provide and their exhaustiveness regarding the individual properties they guarantee. The drawbacks are the practical limits of the scale and complexity of the systems (or, to be more precise, of the individual components of the system that can be analyzed in isolation) where these methods can provide positive results. The analyses typically cause some kind of state space explosion where the time and storage space required to examine all interesting cases grows exponentially. In case of applying abstractions for limiting the state space explosion, the process often involves a substantial amount of human labor that cannot be universally automated and in the worst case even problems with decidability.

To avoid the drawbacks and leverage the benefits, the mixed-criticality systems are designed within a carefully constrained design space. A textbook example is the seL4 microkernel [20] which is not only designed as a typical microkernel, but whose design and implementation has been completely subjected to the goals of formal verification and the support for mixed criticality [12]: The kernel is single-threaded, completely passive, reactive and static. While the general notion of minimality is shared with most other microkernels, the extreme simplification of the inner mechanisms of the kernel leads to increased complexity of the user space up to the point where the kernel-provided abstractions cause abstraction inversion anti-patterns where even trivial functionality needs to be implemented by means of non-trivial interaction of many underlying resources.

To rephrase, many of the academic approaches towards mixed-criticality trade strong guarantees within the realm of their proof assumptions for increased complexity outside of the realm of their proof assumptions. It is arguable whether this leads to practically more reliable and dependable systems overall unless compensated by different means.

2.3 Constructing Mixed-criticality Systems in Industry

The typical industrial approach towards mixed-criticality systems is based on formalizing the best practices in software engineering and carefully certifying the whole design/development process. This is a rather holistic approach that mimics similar approaches in other engineering fields: There is the fundamental theory on the one hand, but on the other hand there are also countless examples of good and bad designs in the history (or, to be more precise, designs that stood and designs that did not stand the test of time) and countless examples of past mistakes that can be avoided. A common observation is that many mistakes happen either due to negligence and poor craftsmanship. Failures also tend to happen due to unforeseen combinations of otherwise benign circumstances.

These methods can be demonstrated, for example, on the safety certification processes such as the ISO 26262 [14] in the automotive industry and the DO-178C [13] in the aerospace industry. These certification processes focus very strongly on the organizational aspects and on due diligence (clear definition of responsibilities of organizational roles, well-defined and documented organizational processes, guaranteed escalation paths that make sure the safety goals are never compromised due to business goals, use of qualified tools in qualified configurations, etc.), on proper safety requirements and hazards analyses, tracking and traceability (every design and implementation artifact needs to have a paper trail to a safety goal, every safety-relevant aspect must have a complete test coverage, etc.) and on proper development guidelines (following a strict coding standard such as MISRA, etc.).

The industrial certification approaches provide a relatively large degree of flexibility for integrating low-criticality components into the mixed-criticality system because they allow to individually evaluate to what degree a component is critical (or which aspects of the component are critical) and apply different certification means accordingly. The essential requirement is that each such decision is explicit, documented and confirmed by the organizational roles that have proper authority. Given the practical and organizational nature of the industrial approaches, they naturally incorporate aspects such as the interaction with other systems and the interaction with the users into the whole framework.

While many of the industrial certification processes allow and encourage exhaustive formal verification methods, such formal methods are mostly optional and non-essential. This is also the source of the main drawback of the industrial approaches: They mostly provide only statistical and probabilistic guarantees and they are unable to go beyond that. This is especially problematic with respect to the mixed-criticality systems because there are no strong guarantees that a low-criticality component (which has been subject to much less scrutiny than the high-criticality components) cannot affect the system in a way that would jeopardize the goals of the high-criticality components.

2.4 Paths Forward

The author believes strongly in the importance of incremental progress, in cross-pollination of ideas, in the transfer of knowledge and experience from industry to academia and vice versa. Thanks to this progress, the academic approaches have grown over the years from being useful on toy textbook examples to being useful on real-world systems (or at least on a useful subset of them). Thanks to this progress, the industrial approaches have matured over the years from a set of well-meant but naïve guidelines to rigorous standards with a solid track record.

However, the author also believes that our greatest enemies are not bugs, failures and vulnerabilities. Our greatest enemy is our own desire to always build new things that are just slightly out of the reach of the current methods that should assure their correctness. While inventing without a constraint is clearly a great motivator for humans, it also means that we are always building mixed-criticality systems that are potentially slightly unreliable. An alternative would be to use our current methods and knowledge on systems that are a few years old. Such systems are still operational and both individuals and our entire society rely on them.

This voluntary "taming of complexity" would be obviously not easy to defend from the short-term economic point of view, but given the fact that our society relies heavily on high-criticality and mixed-criticality systems, the long-term economic benefits of decreased costs of failures, outages and other issues should prevail.

All methods for constructing sound mixed-criticality systems have huge upfront costs, while the costs of failures, outages and other issues are spread over time and in many cases are externalities not paid directly by the original stakeholders. It is important to change the default mindset of many users which says that failures of computer systems are "normal" and essentially unavoidable. Prof. Andrew Tanenbaum once asked the following question: "If one in million car tires randomly exploding is not acceptable, why is this still acceptable in software?" Finding the right answers to this question will help us bridge the gaps in our knowledge.

3 ENERGY-DRIVEN DESIGN

The rapid expansion of embedded/IoT brings with it the additional challenge of how to power the billions of anticipated devices. While a majority is still likely to be tethered to mains power, a significant proportion of devices will be locally powered – and this puts a significant burden on both the sustainability of battery technology, as well as the workforce required to replace and maintain them. As such, the opportunity to power embedded devices from locally harvested environmental energy is ever more attractive. This is, however, not without its challenges including very low and highly dynamic available power. This variability, coupled with other design constraints, can lead to the system being depleted of energy, causing reliability and dependability issues and ultimately system failure. As a result, the system needs to be designed around the properties of the energy source, e.g. availability, timeliness, conversion efficiency, storage capability etc. To this end, in this section we introduce an *energy-driven* paradigm to effectively design such systems.

3.1 System Architecture and Computing Landscape of Energy-Driven Systems

Energy-driven system design applies primarily to cyber-physical systems (CPS) that are powered directly from ambient sources of energy, such as solar, air flow, vibration and RF [25]. When an energy-driven approach is taken to the design of these systems, they differ from traditional CPS by focusing on power supply and demand characteristics early in the design cycle, and directly alongside application criteria. This may involve, for example, questioning whether or not periodic sensing/reporting is necessary, or whether instead the device functionality can be synchronised with energy available. This may not be possible for many applications, and often requires reconsidering the fundamental *purpose* of the application.

A typical energy harvesting IoT device is illustrated in Figure 1. It consists of a compute/control core, sensing elements, a radio, power management circuitry, the energy harvester, and some energy storage. The compute capability of these systems is highly variable

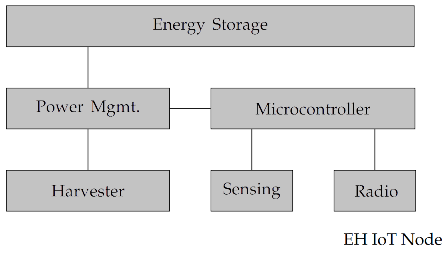


Figure 1: A typical energy harvesting IoT device

Table 1: IETF classes of constrained devices

Name	data size (e.g. RAM)	code size (e.g. Flash)
Class 0	<< 10 KiB	<< 100 KiB
Class 1	~ 10 KiB	~ 100 KiB
Class 2	~ 50 KiB	~ 250 KiB

to accommodate different uses cases. For example, the Internet Engineering Task Force (IETF) define three classes of networked constrained devices as summarised in Table 1 [5].

Class 2 systems are less resource constrained and are fully capable of performing IP-style networking. They are likely to be mains-tethered, and hence not the target of energy-driven design. Class 0 systems are severely constrained sensor-like nodes, which acts as a peripheral with simple communication links to Class 1 or Class 2 devices that act as gateways to the Internet. Numerous examples of the application of energy-driven paradigms to Class 0 systems have been reported, for example battery-less speedometer/odometer [4], footstep-powered pedometer [31], solar-powered wild-life monitoring cameras [26], etc. Class 1 systems are constrained nodes that are capable of hosting a fairly robust networking stack, but fall short of being able to support the traditional Internet Protocol (IP). This class of systems are the main target for IoT-style IP such as 6LowPAN and CoAP and the applicability of energy-driven frameworks to this class of systems is an area of active research.

Energy harvesting was initially introduced as a means of increasing operational life-time. If energy harvested is greater than energy consumed within the target deployment life-cycle, *energy neutrality* is said to have been achieved. Energy neutrality is a typical constraint and design target for self-powered devices, expressed as:

$$\int_{(n-1)T}^{nT} P_h(t)dt = \int_{(n-1)T}^{nT} P_c(t)dt \tag{1}$$

where P_h is the power harvested, P_c is the power consumed, and T is a period of interest. Traditional systems also require that the following inequality holds:

$$E_0 + \int_0^T P_h(t) - P_c(t)dt > E_{min}, \forall T \tag{2}$$

where $E_{initial}$ is the initial state of charge at deployment time, e.g. a fully charged battery, and E_{on} is the minimum energy that needs to be in the storage element (e.g. a supercapacitor) to power on the system. This means that the storage never depletes past the turn-off threshold at any point during the deployment life-cycle. One approach to energy-driven design is to relax the constraint in Equation 2, such that the system sustains operation through intermittent power supply failures.

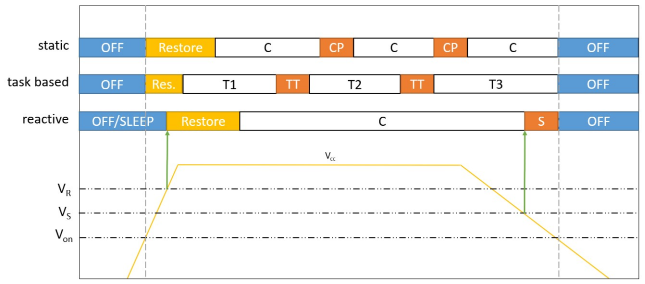


Figure 2: Contrasting the behaviour of the three IC classes

3.2 Intermittent Computing

Intermittent computing (IC) is a key enabler of energy-driven systems, because it caters for the case when Equation 2 does not hold. It addresses the problem of sustaining forward progress in computation through intermittent power cycles. At one extreme, IC serves as a fail-safe backup-restore mechanism during rare power failure events. However, contemporary research addresses the other extreme where frequent but often unpredictable power failure is the norm. It is thus important to ensure that the mechanism used for IC is efficient, i.e. minimises overhead, and can ensure correct operation [23][22][17]. IC schemes can be classified into the following classes [48], and are illustrated graphically in Figure 2:

(1) *Static Checkpointing*: Snapshots of the system state are statically inserted into the code at design-time, e.g. at the beginning of every function call. Execution rolls-back to the previous checkpoint after recovering from a power failure.
(2) *Task-based*: A runtime, perhaps as part of the operating system, manages the task graphs of atomic tasks that constitute the application. A power interruption causes a reissue of the uncompleted task.
(3) *Reactive*: Snapshots are saved when an imminent power failure is detected, e.g. by monitoring changes in the supply voltage. On recovery, execution continues from the instruction that was interrupted.

Challenges in IC include ensuring their correctness, maximising performance/efficiency, and compatibility/scalability considerations. Example of ensuring correctness include mitigating idempotency violations, i.e. errors due to re-execution of code, and handling live-locks due to Sisyphean tasks that require more energy in an atomic fashion to complete than can be harvested [29]. Maximising performance/efficiency means achieving the greatest amount of forward progress in terms of computation and use the least amount of system resources such as memory and non-volatile memory, given the power and energy constrains/properties of the system. Finally, it is desirable to contain/hide the complexity due to the IC strategy and the underlying intermittent physical properties exposed from the application programmer.

Another concept that is often discussed in the context of IC is maintaining *peripheral consistency*. An external peripheral, for example a radio module, would have its own state machine, and the peripheral may be at an unknown state after an intermittent power failure. A mechanism is therefore required to keep track of the peripherals' states and recover them to the state prior to the power failure [1] [6].

3.3 Closed-loop Modelling of Power and Device Functionality

As there is inherent cross-coupling between device power supply and consumption behaviour, modelling such systems with existing simulation frameworks is unfeasible. Functional simulators, such as QEMU, do not capture the performance and power consumption behaviour of the compute block and so are ill-suited for energy-driven design. Furthermore, cycle-level performance simulators, such as gem5 and MSPSim, do not capture power consumption characteristics and are thus also unsuitable.

For energy-driven design, a *closed-loop* functional, performance, and energy simulation tool is required, e.g. *Fused* [49]. As IC is an integral component of energy-driven design, the tool was built specifically to simulate power-supply dependent compute behaviour as illustrated in Figure 3. Performance events and states, e.g. memory accesses and cache hit/miss, are generated in a cycle-accurate manner and these statistics are used to estimate the the system's power consumption. The power consumption expressed as a current draw affects the properties of the power delivery circuitry and would for example result in a decaying V_{cc} if demand is greater than supply thus resulting in intermittent operation.

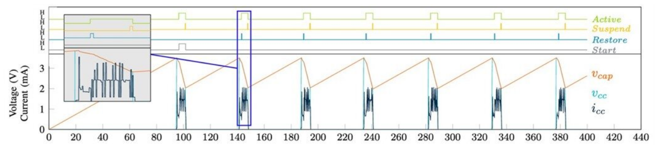


Figure 3: Intermittent computing as simulated on *Fused*.

Extensions to *Fused* were introduced in [52] to enable virtual prototyping of complete energy harvesting devices, including both on-chip and off-chip peripherals, the power management circuitry, the harvesters, and the ambient condition. An example of such a system modelled is shown in Figure 4.

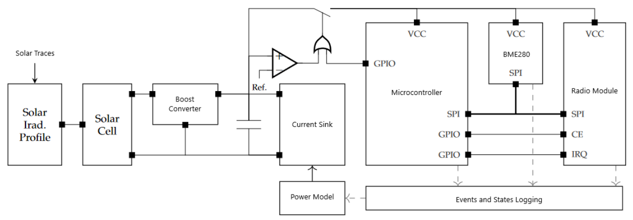


Figure 4: Virtual prototype of an IC system in *Fused*

Fused can form an integral part of an energy-driven design flow, as illustrated in Figure 5. Power supply and consumption characteristics are made explicit as a major component in the design repository, on par with the application specifications, hardware components, and software components. In this framework, Fused acts as a virtual prototyping platform and an emulator, allowing the exploration of different architectural, hardware, and software solutions or optimization points. Multiple iterations of application mapping, optimization, evaluation, validation, and tests can then be carried out against the virtual prototype before implementation.

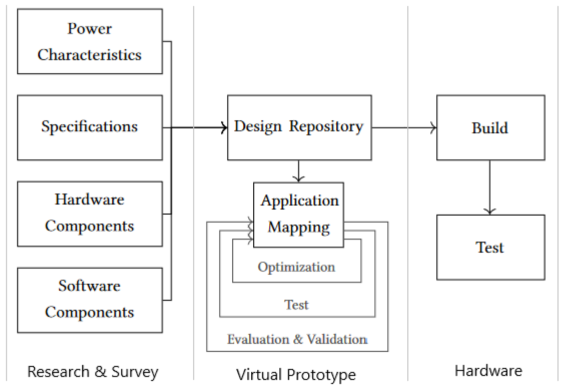


Figure 5: Design flow of energy-driven systems featuring *Fused* as an integral virtual prototyping tool

3.4 Paths Forward

Emerging uses cases driven by IoT introduces new demands on energy-driven design. The explosion of data generated by the IoT is increasing the motivation for performing greater computation at the edge, meaning that energy-driven systems will need to cope with an ever-increasing compute load (e.g. inference and perhaps even distributed learning). IC approaches will also need to adapt to the properties of emerging nonvolatile memory technologies beyond flash and FRAM, such as STT-MRAM, ReRAM, and phase change memory, ensuring correctness, reliability and efficiency [9]. IoT devices inherently need to communicate as part of a larger network, and additional challenges are posed as the nodes in a network of energy harvesting devices frequently, and often unpredictably, go offline due to scarcity in harvested energy. From the perspective of maintaining peripheral consistency, communications and networking present an extremely challenging case, due to a combination of local state (the radio module), volatile data currently being transmitted/received, and the state of other nodes in the network. An intermittent network implies the following:

(1) Many links are intermittent; special treatment is required to maintain the always-on abstraction that the data link layer is supposed to provide to upper layers.
(2) The power budget of the nodes are asymmetric and time-varying, i.e. some nodes can harvest more energy that other nodes, but there is a temporal and spatial variation.

Current research to address the first issue are exploring approaches such as the use of wake-up receivers (WuRx); low-power radio hardware that can listen on a physical channel constantly [24]. The second issue is a network layer problem as it affects routing, latency, throughput, and congestion, and approaches from energy-aware networking may be tailored to specific challenges [3] [15]. The applicability of an energy-driven paradigm to IP-style networking, as envisioned for IoT, is yet to be demonstrated.

4 CROSS-LAYER RELIABILITY-AWARE DESIGN

Widely used methods for low-power/energy design such as Dynamic Voltage and Frequncy Scaling (DVFS) can adversely affect the functional reliability of systems, especially in harsh operating environment. The traditional phenomenon-based approach to improving reliability by redundancy results in power and timing overheads—both critical factors in resource constrained embedded real-time

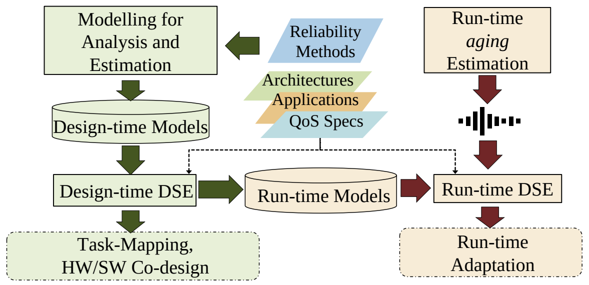


Figure 6: CLR-integrated system-level design methodology

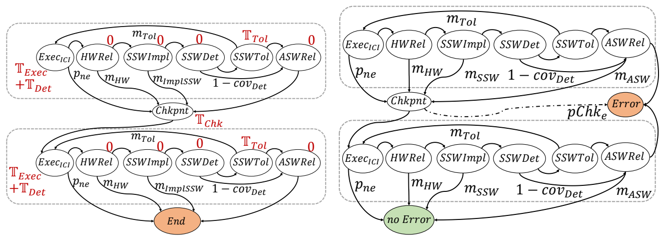


Figure 7: Markov Chain-based reliability modeling for implementing checkpointing along with *HWRel* and *ASWRel* [39]

systems. Over the past decade, Cross-Layer Reliability (CLR) has opened up new opportunities for designing application-specific reliability in resource-constrained embedded systems [34]. This approach allows the designer to leverage the tolerances of an application to degradation of some form of reliability—timing, functional, lifetime—while improving the other(s). However it also results in a tremendous increase in the Design Space Exploration (DSE) complexity by introducing design decisions of selecting and tuning the fault-tolerance methods to be implemented at each abstraction layer. This section presents some of the results from the state-of-the-art research into Cross-layer Reliability (CLR)-integrated system-level design methodology for heterogeneous embedded systems. The constituent topics of the methodology are shown in Figure 6. The section briefly presents the results related to each of the topics followed by a discussion on the path forward for building cost-efficient and reliable embedded systems.

4.1 Modeling and Analysis

The research into building reliable electronic systems has resulted in the development of various fault-mitigation methods employing different types of redundancy and resulting in varying levels of efficacy and resource overheads. For instance, improving functional reliability with temporal redundancy based methods such checkpointing with rollback/recovery can result in lower power dissipation overheads than some spatial redundancy based method such as Triple Modular Redundancy (TMR), albeit at the cost of higher timing overheads. Similarly, a combination of both such methods can be deployed at separate layers to improve the functional reliability even further. In order to analyze the effect of using such combination of fault-mitigation approaches, appropriate models need to be designed. Such models should be able to integrate the effect of the overheads and the varying efficacy of tunable fault-tolerance methods.

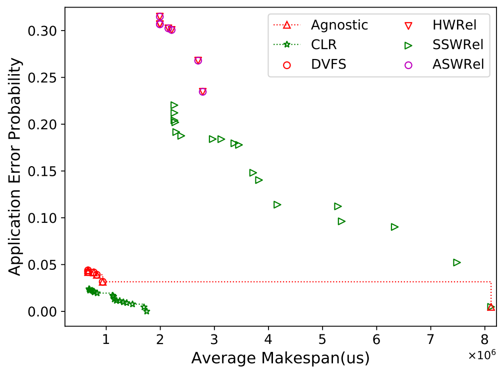


Figure 8: Comparison of Pareto-front obtained from cross-layer optimization and the combination of points obtained from single-layer optimizations.

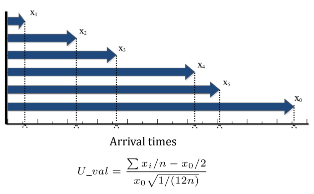
Figure 7a and 7b show the Markov Chain-based models for timing and functional reliability respectively, as a result of the combination of redundancy type—information, temporal and spatial at the application software (ASW) system software (SSW) and hardware (HW) layers respectively [35, 39]. As seen in the figure, the model accounts for factors such as imperfect fault-detection (cov_{Det}) and partial masking at HW and ASW layers (m_{HW}, m_{ASW}) along with implicit fault-masking at SSW layer ($m_{ImplSSW}$) [44]. Such Markov chain-based models allow the estimation of reliability metrics—both analytically and by using simulations [40].

4.2 Design-time Task-mapping

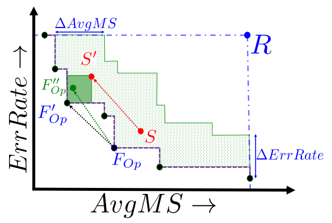
For embedded systems, task-mapping—the allocation of the limited compute resources to the system's workload—forms an integral aspect of system level design. Reliability-aware task-mapping has been an active field of research for quite a few years. However, CLR provides an additional scope for application's Quality of Service (QoS)-specific allocation of the systems resources to provide varying types and levels of redundancy. The corresponding DSE problem, however, has the additional challenge of joint optimization of fault-mitigation methods across multiple layers. Naive application and extensions of Multi-Objective Evolutionary Algorithms (MOEA)-based methods can be insufficient for such complex DSE problems [7]. Similarly, the traditional approach of *other-layer-agnostic* optimization can result in degraded quality of the solutions obtained in multi-objective optimizations during design/compile time. As shown in Figure 8, the joint optimization across DVFS and fault-mitigation at multiple layers shows better Pareto-front optimization than collecting the results from multiple other-layer-agnostic DSE runs. Similar CLR-aware design time optimization, when applied to hardware-hardware partitioning in Dynamic Partial Reconfiguration (DPR) enabled FPGA-based embedded systems, can result in improved lifetime reliability and performance [33, 36, 37].

4.3 Dynamic Run-time Adaptation

The DSE for CLR results in design points that can provide better reliability-performance trade-offs than single-layer approaches. Similarly, the joint optimization across multiple layers can result in a larger number of feasible design points at a finer granularity. However, the most efficient usage of such high quality Pareto fronts requires optimal adaptation of the system to dynamic variations



(a) Low-overhead aging estimation [43]



(b) Hybrid DSE for reconfiguration cost-aware dynamic adaptation [38]

Figure 9: Hybrid DSE for CLR-aware system design

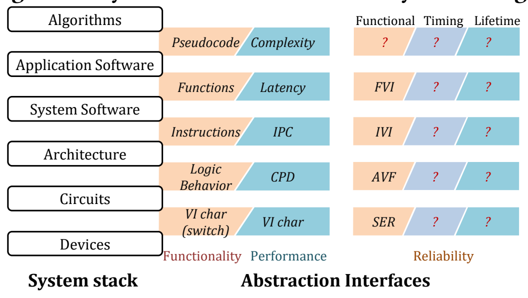


Figure 10: Designing Cross-layer Reliability: Abstractions and Interfaces [41]

in the embedded systems operating environment—both internal and external. Internal variations include the gradual wear-out of the systems hardware resources—Processing Element (PE), on-chip interconnects and memory elements. While the designer may include aging sensors [16] for a more accurate estimation, low-cost methods using the fault behavior of the PEs have also been proposed [11, 43]. Figure 9a shows the process of using the Centroid test [27] over the arrival times of the intermittent faults in each PE to estimate the aging of individual PEs and remap tasks to improve the system's useful lifetime.

In addition to aging of PEs, the system should also be able to adapt to changes in the application's QoS requirements by switching to the appropriate design point determined during design/compile-time DSE. However frequent changes in the system's run-time configuration can also result in high reconfiguration costs. As shown in Figure 9b, the design/compile-time DSE can be used to generate additional non-dominating design points that can result in lower average reconfiguration cost. Similarly, agent-based methods can be used to determine the appropriate long-term efficacy of the stored design points during hybrid DSE and perform dynamic adaptation accordingly [38].

4.4 Paths Forward

4.4.1 Methods and Paradigms. With electronic component becoming increasingly unreliable, considerable research is required for the development of low-overhead and tunable fault-mitigation methods. Modern approaches such as CLR work best in the presence of more degrees of freedom—both in terms of the number of methods available at each layer and the granularity of the parameters of those methods. Further, more recent paradigms of computing such as approximate computing can be used to reduce the overheads of the redundancy-based methods [50].

4.4.2 Modeling and Abstraction. While the Markov Chain-based models for timing and functional reliability provides a good first order method for analysis and estimation, more holistic approaches need to be developed for more complex cross-layer interactions such as integrating the impact of multiple reliability methods at each layer. As shown in Figure 10, having reliability interfaces, similar to those used for functionality and performance, can enable far better DSE than current state-of-the-art works [41].

4.4.3 Design Space Exploration. The development of more fault-mitigation methods with tunable efficacy will result in an exponentially increasing design space complexity. While MOEA-based methods provide a generic set of tools for such large search spaces, they will need to be augmented with more modern techniques such as Monte Carlo Tree Search (MCTS) [42] and Machine Learning (ML)-based estimation. Similarly, agent-based techniques can be used to provide more self-sufficient systems deployed in widely varying operating environments.

5 AGING AND COVERT CHANNEL ATTACKS AND MITIGATIONS

Aging and covert channel attacks can cause several damages such as financial, health and emotional. Therefore, timely detection of these attacks and their prevention is of utmost importance.

5.1 Aging Attack

To accelerate aging, planned obsolescence in industrial design and marketing is a notorious strategy to deliberately set product to malfunction within a certain period of time [8]. Quite a few consumer electronics vendors have been reported of engaging in such practice of obsolescence by downgrading their old products to drive customers to move to their newer versions of products [8].

A manufacturer often favours this type of obsolescence by either deliberately crafting malicious programs or inserting hardware backdoors that can be exploited to accelerate aging of key components, such as cores and some of the critical links in many-core chips [8] [19]. Targeting a hotspot node and making it age faster can certainly cause a product to obsolete fast. However, if the chip wears out before its warranty expires, the manufacturer is obligated to refund consumers or replace/repair the product, which undermines the purpose of planned obsolescence. In a simple word, blindly accelerating aging does not serve the best interest of the manufacturer. Instead, manufacturer prefers to have its component/product work perfectly at the customer end when it is still under the warranty, but it would wear out as soon as the warranty expires.

In what follows, a profit model of the manufacturer is first proposed, followed by the routing algorithms to launch planned obsolescence attack.

5.1.1 Profit Model. It has been shown in [32] that the routers in a NoC age differently at different paces as they handle different amount of traffic. In XY routing, central routers handle more traffic and can be referred to as "hotspot nodes", which experience accelerated aging due to the exponential nature of the NBTI effect.

Note that an NoC system's lifetime is bounded by the hotspot node, which is the router with the lowest MTTF. That is, the MTTF of the entire chip system, MTTF_{sys}, can be written as,

$$MTTF_{sys} = \min_{\forall i \in [1,N]} MTTF_i \tag{3}$$

where N is the number of routers.

The profit of the manufacturer for the i^{th}-version devices can be written as,

$$W_i = (P_i - C_i) \times S_i \tag{4}$$

where P_i, C_i, and S_i are the price, the cost, and the number of sold copies of the i^{th}-version devices, respectively. The sales volume S_i is written as,

$$S_i = \sum_{l=1}^{i-1} \rho_{i-l} \times S_{i-l} + S_0 \tag{5}$$

where ρ_{i-l} is the probability that customers who bought the $(i-l)^{th}$-version of the devices would switch to the new-version, and S_0 is the number of new clients who have never bought any previous version of the product.

Let event B be the occurrence of an event that a customer buys a new version of the device, and let event A be the occurrence of an event that a customer's old version device is worn out. In this case, ρ_{i-l} in Eqn. (5) can be further written as,

$$\rho_{i-l} = \rho(B|A) \times \rho(A) + \rho(B|1-A) \times \rho(1-A) \tag{6}$$

where $\rho(B|A) \times \rho(A)$ is the probability that a customer buys a new version of device given that his/her old-version device malfunctions, and $\rho(B|1-A) \times \rho(1-A)$ is the probability that a customer buys a new device, even though his/her old-version device still functions.

Taking the warranty into account, one can see that, with the version g as its latest model, the profit for the i^{th}-version can be calculated as,

$$\begin{aligned} W_i = \{ & \sum_{l=1}^{g} [p_{Ml} \times x_{i-l} + p_{Wl} \times (1 - x_{i-l})] \times S_{i-l} + S_0 \} \\ & \times (P_i - C_i) - \sum_{m=1}^{w} \sum_{n=0}^{w-m} (x_{i-m} \times S_{i-g+n}) \times C_i \end{aligned} \tag{7}$$

where x_i is the failing rate of the i^{th}-version of the device, S_0 is the number of new customers, w is the product warranty and p_{Ml} and p_{Wl} are the respective $(i-l)^{th}$-version device's probabilities, $\rho(B|A)$ and $\rho(B|1-A)$ as defined in Eqn. (6).

The failing rate x_{i-l} in Eqn. (7) is assumed to follow the Gaussian distribution denoted as G(μ,σ^2). Here the mean μ is obtained from Eqn. (3), and the variance σ^2 is a random variable. By setting $\alpha = 0.5$ as the testing condition, x_{i-l} is generated by randomly sampling from G(μ,σ^2). For the proposed planned obsolescence, the value of x_{i-l} depends on the triggering condition of aging acceleration.

5.1.2 Aging Attack Algorithms. The aging controller module is designed to use router aging acceleration during a time trigger threshold, and router aging deceleration after the time trigger threshold, which is set according to the terms of the warranty agreement.

Placed at each router, the control module, shown in Fig. 11, consists of the trigger and the routing computation (RC) submodules. When the product is still under warranty, the routers will adopt a routing strategy that helps decelerate aging. This module continuously compares the trigger threshold with the system clock, and forces routers to switch to a different routing strategy to accelerate aging, once the trigger threshold is reached.

The NoC is assumed to have a mesh topology. Once the trigger module activates the aging acceleration signal, the RC module selects the aging acceleration routing algorithm to route data packets, which stipulates the packets' routing paths to include the hotspot node. That is, the hotspot node is set as the temporary destination of a packet, and the packet is first transmitted to this hotspot node by XY routing, after which it will be delivered to the destination node.

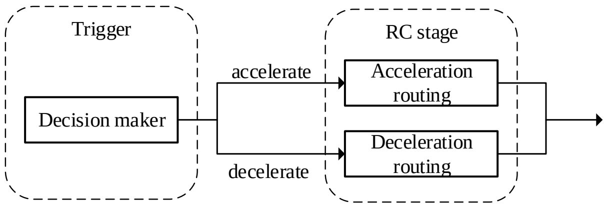


Figure 11: Architecture of the control module

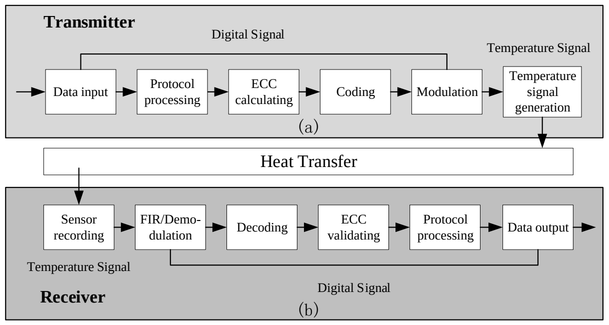


Figure 12: The flow of a baseline thermal covert channel transmission from the (a) transmitter to the (b) receiver end.

Once the trigger module activates the aging deceleration signal, the RC module selects the aging deceleration routing algorithm, which diverts packets to be routed along the paths away from the hotspot nodes. A specific region that runs YX routing algorithm, named as YX region, is composed of given by Eqn. (8), assuming the hotspot node is (x_0, y_0). The YX region forms a triangle shape, and it shall be able to generate more evenly distributed traffic [32].

$$|y_0 - y| < x_0 - x \tag{8}$$

The planned aging can be achieved with the two proposed routing algorithms: accelerated aging when the warranty expires, and decelerated aging when the warranty is still in effect. Adding the product warranty into the aging model, a manufacturer can maximize its profit by altering the sales volume, the price, the cost, the warranty, and the network size of a device.

5.2 Thermal Covert Channel Attack and Countermeasure

5.2.1 Thermal covert channel attack. The baseline thermal covert channel (TCC) attack [21] links a transmitter and a receiver, as shown in Fig. 12. The transmitter and receiver run on different cores or on different hardware threads of a physical core if the processor supports multi-threading. The transmitter sends the sensitive data via the heat transfer. The receiver records the temperature signal by reading its thermal sensor, and decodes the signal to recover the original data.

In [51] an enhanced attack using communication protocol is proposed to dynamically change the transmission frequency with a very low implementation overhead, and without involving any extra channel. They use three detection mechanisms to detect whether the channel is jammed. Once the channel is found to be jammed, it will trigger a dynamic frequency changing module to change the transmission frequency to avoid jamming. In essence, It is a polling based frequency changing protocol and the available transmission frequencies are stored by both the transmitter and receiver in advance. In the worst case that the channel is severely jammed, both

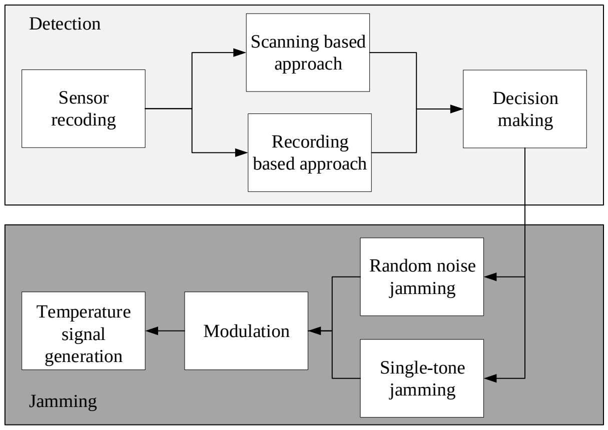


Figure 13: Workflow of the proposed countermeasure.

the transmitter and the receiver poll the next available frequency iteratively and send a series of packets as an attempt to set up their connection over that channel. Once they find a channel available for connection, communication resumes in this new channel.

In the environment with fixed-frequency jamming noise, the packet error rate (PER) of enhanced TCC [51] is 5%, while the PER of baseline TCC [21] is over 85%. The reason is that the jammer detects the thermal covert channel and emits the noise signal with the same transmission frequency as the thermal covert channel. The receiver of the baseline TCC [21] reads both the noise and the packets from the transmitter, which results in decoding errors. However, the dynamic frequency changing protocol can detect the jamming and changes the transmission frequency dynamically to avoid being jammed. Once the transmission frequency moves to a new frequency, the fixed-frequency jammer loses its target to jam.

5.2.2 Countermeasure with Scanning and Channel-aware Jamming. To countermeasure the above thermal covert channel, an intuitive approach is to use a full-band jamming, that that thermal noise will flood the entire band $[f_l, f_h]$ occupied by the thermal covert channel, where f_l is the lowest frequency in the entire band, f_h is the highest frequency. However, such a naive approach causes excessive power consumption. Instead, A lightweight countermeasure [51] is proposed as shown in Fig. 13 which periodically scans the frequency spectrum to check if there exists a potential thermal covert channel attack or not. If such a channel is identified, a noise is emitted with the same frequency of the covert channel. With a high scanning speed, all the thermal covert channels shall be able to be detected and jammed. Two detection approaches are used for detecting TCC. (1) **Scanning-based approach**. In this approach, the jammer changes the center frequency of the Finite Impulse Response (FIR) filter to scan the entire frequency spectrum of interest. The linear scan method changes the center frequency sequentially from the lowest frequency f_l to the highest frequency f_h. The difference between two adjacent center frequencies is referred as the frequency increment Δf. A larger Δf results in fewer scanning steps and higher probability of failing to detect the covert channel, but with a lower runtime cost. For each core i, an FIR with center frequency f_{center} and bandwidth of Δf is used to filter the thermal signal. If the maximum amplitude of the filtered signal is higher than threshold T_r, it is deemed as a malicious core involved in the thermal covert channel attack. The center frequency of FIR sweeps from f_l to f_h with an incremental of Δf. (2) **Recording-based approach**. In this approach, the temporal temperature signal is recorded for 1 second, and transformed into frequency domain using fast Fourier transform (FFT). The frequency spectrum of the signal is checked whether there is a potential attack or not. The advantage of this approach is that it can check the full spectrum. However, since recording a long temperature sequence takes time, the attack might already transmit sensitive data before being detected.

Given the output of the FIR filter (approach 1) or the spectrum (approach 2), a decision should be made by comparing the signal amplitude with a given threshold T_r. If the amplitude exceeds the threshold, it can conclude that an attack has been discovered, and the signal frequency and core number (f_{detect} and L), are passed to the jamming process for action. Once the thermal covert channel is detected, a random bit sequence of '1's and '0's is generated as noise at **the random noise generation module**. This random noise sequence $s(t)$ can be expressed as

$$s_{jam}(t) = r_n g_{jam}(t - nT_{jam}) \tag{9}$$

where T_{jam} is the period of one bit (symbol width), $g_{jam}(t)$ is the baseband pulse waveform with duration of T_{jam}, and r_n is the value of the n-th bit of the random sequence which is defined as follows:

$$r_n = \begin{cases} 1 & with\ a\ probability\ of\ 0.5 \\ 0 & with\ a\ probability\ of\ 0.5 \end{cases} \tag{10}$$

The single-tone noise generation module, unlike random noise, continuously sends bit sequence of '1's, *i.e.*, $r_n = 1$.

At **the modulation module**, the sequence generated by the noise generation module is modulated by the carrier that has the same frequency as the transmission frequency f_{detect} occupied by the detected covert channel. The output of the modulation module, $e_{jam}(t)$, is thus

$$e_{jam}(t) = s_{jam}(t) \times c(t) \tag{11}$$

where $s_{jam}(t)$ is the noise generated by the noise generation module, and $c(t)$ is the carrier with a frequency of f_{detect}.

Finally, the thermal signal is generated following the same temperature signal generation approach as baseline TCC.

The detection error of the two detection approaches is less than 3Hz. After the signal is detected, the jamming process can increase the packet error rate to 85%. Experimental results confirmed that when the countermeasure is applied, its PER jumps to 85%, effectively shut down thermal covert channel attacks even with enhanced capabilities.

5.3 Paths Forward

Based on the observations and expectations, we see the following paths forward for emerging aging and covert channel attacks.

Increasing types of aging and covert channel attacks. With advancement of defence mechanisms for one type of attack, attackers are able to devise new types of attacks [51, 54]. It is expected that this trend is going to continue and thus design challenges arising will need continued innovations in defence mechanisms to address them.

It is also evident that there is widespread adoption of embedded AI in various application domains, e.g. audio [28] and video [10] analytics. Thus, a new or enhanced type of attacks on such systems are also effected. This will drive development of novel defence mechanisms while addressing involved challenges.

Joint consideration of various attacks. It is expected that an embedded system can encounter various attacks at the same time, e.g. aging and cover channel, and thus a joint consideration is important. With this, it is evident that the complexity of defence mechanisms and involved challenges to be addressed are expected to rise.

Cost-effective countermeasures. Hardware based countermeasures are being extensively explored but they are costly. Based on the anticipated increasing attacks, we expect that security solutions will jointly consider software and hardware to achieve the best protection mechanism at a minimal cost. This will require addressing several challenges like identifying the defence components to be run on the hardware and software, and their adaptation on them.

6 CONCLUSIONS

The paper provides insight into design challenges and paths forward for embedded systems. Various aspects of embedded systems design are considered from both industrial and academic point while reviewing state-of-the-art approaches. Though the design aspects are considered separately due to their importance in a given application domain and research focus by different groups, we expect the future lies in joint consideration of all possible aspects, i.e. metrics, needed for an application domain.

ACKNOWLEDGMENTS

We acknowledge financial support from the following: the UK Engineering and Physical Sciences Research Council under Platform grant EP/P010164/1, National Natural Science Foundation of China 61971200, Open Research Grant of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences CARCH201916, and Zhejiang Lab 2021LE0AB01.

REFERENCES

[1] Rodriguez Arreola et al. 2018. RESTOP: Retaining External Peripheral State in Intermittently-Powered Sensor Systems. *Sensors* 18, 1 (2018), 172.

[2] Karunakar R Basireddy et al. 2019. AdaMD: adaptive mapping and DVFS for energy-efficient heterogeneous multicores. *IEEE TCAD* 39, 10 (2019).

[3] Aruna Prem Bianzino et al. 2012. A Survey of Green Networking Research. *IEEE Communications Surveys Tutorials* 14, 1 (2012), 3–20.

[4] Samuel Wong Chang Bing et al. 2018. An Energy-Driven Wireless Bicycle Trip Counter with Zero Energy Storage. In *Proceedings of the SenSys.* 404–405.

[5] C. Bormann et al. 2014. *Terminology for Constrained-Node Networks.* RFC 7228.

[6] Adriano Branco et al. 2019. Intermittent asynchronous peripheral operations. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems.* 55–67.

[7] A. Das et al. 2014. Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs. In *DATE.* 1–6.

[8] Sourav Das et al. 2018. Abetting planned obsolescence by aging 3D networks-on-chip. In *IEEE/ACM Int'l Symp. Networks-on-Chip.*

[9] Timothy Daulby et al. 2020. Comparing NVM technologies through the lens of Intermittent computation. In *Proceedings of the ENSsys.* 77–78.

[10] Somdip Dey et al. 2020. Temporal motionless analysis of video using cnn in mpsoc. In *IEEE ASAP.*

[11] L. A. R. Duque et al. 2015. Improving MPSoC reliability through adapting runtime task schedule based on time-correlated fault behavior. 818–823.

[12] Kevin Elphinstone and Gernot Heiser. 2013. From L3 to SeL4 What Have We Learnt in 20 Years of L4 Microkernels?. In *Proceedings of the SOSP.* 133–150.

[13] Radio Technical Commission for Aeronautics. 2012. *Software Considerations in Airborne Systems and Equipment Certification.*

[14] International Organization for Standardization. 2018. *Road vehicles — Functional safety.*

[15] Soheil Ghiasi et al. 2002. Optimal Energy Aware Clustering in Sensor Networks. *Sensors* 2, 7 (2002), 258–269.

[16] Adam S. Hartman and Donald E. Thomas. 2012. Lifetime Improvement through Runtime Wear-Based Task Mapping. 13–22.

[17] Josiah Hester et al. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the SenSys.* 1–13.

[18] Hengli Huang et al. 2021. Detection of and Countermeasure against Thermal Covert Channel in Many-core Systems. *IEEE TCAD* (2021).

[19] Naghmeh Karimi and Xueyang Wang. 2015. MAGIC: Malicious aging in circuits/cores. In *ACM Trans. Architecture Code Optimization.*

[20] Gerwin Klein et al. 2009. SeL4: Formal Verification of an OS Kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles.* 207–220.

[21] Zijun Long et al. 2018. Improving the efficiency of thermal covert channels in multi-/many-core systems. In *DATE.*

[22] Brandon Lucia et al. 2017. Intermittent Computing: Challenges and Opportunities. In *2nd Summit on Advances in Programming Languages*, Vol. 71. 8:1–8:14.

[23] Kiwan Maeng et al. 2017. Alpaca: intermittent execution without checkpoints. *Proceedings of the ACM on Programming Languages* 1, OOPSLA (2017), 96:1–96:30.

[24] Michele Magno et al. 2016. Design, Implementation, and Performance Evaluation of a Flexible Low-Latency Nanowatt Wake-Up Radio Receiver. *IEEE Transactions on Industrial Informatics* 12, 2 (2016), 633–644.

[25] Geoff V. Merrett and Bashir M. Al-Hashimi. 2017. Energy-driven computing: Rethinking the design of energy harvesting systems. In *DATE.* 960–965.

[26] Matteo Nardello et al. 2019. Camaroptera: A Batteryless Long-Range Remote Visual Sensing System. In *Proceedings of the ENSsys.* 8–14.

[27] Patrick P. O'Connor and Andre Kleyner. 2012. *Practical Reliability Engineering* (5th ed.). Wiley Publishing.

[28] Jordi Pons and Xavier Serra. 2019. Randomly weighted cnns for (music) audio classification. In *IEEE ICASSP.* IEEE.

[29] Benjamin Ransford and Brandon Lucia. 2014. Nonvolatile Memory is a Broken Time Machine. In *Proceedings of the MSPC.* Article 5, 3 pages.

[30] Vijeta Rathore et al. 2020. Longevity Framework: Leveraging Online Integrated Aging-Aware Hierarchical Mapping and VF-Selection for Lifetime Reliability Optimization in Manycore Processors. *IEEE Trans. Comput.* 70, 7 (2020).

[31] Alberto Rodriguez et al. 2017. Intermittently-powered energy harvesting step counter for fitness tracking. In *2017 IEEE Sensors Applications Symposium (SAS).*

[32] Nezam Rohbani et al. 2017. LAXY: A location-based aging-resilient xy-yx routing algorithm for network on chip. In *IEEE TCAD.*

[33] S.S. Sahoo et al. 2018. Multi-objective design space exploration for system partitioning of FPGA-based Dynamic Partially Reconfigurable Systems. *Integration* (2018).

[34] Siva Satyendra Sahoo et al. 2016. Cross-layer fault-tolerant design of real-time systems. In *DFTS.*

[35] Siva Satyendra Sahoo et al. 2018. CLRFrame: An Analysis Framework for Designing Cross-Layer Reliability in Embedded Systems. In *VLSID.*

[36] Siva Satyendra Sahoo et al. 2018. Lifetime-aware Design Methodology for Dynamic Partially Reconfigurable Systems. In *ASP-DAC.*

[37] S. S. Sahoo et al. 2018. QoS-Aware Cross-Layer Reliability-Integrated FPGA-Based Dynamic Partially Reconfigurable System Partitioning. In *FPT.*

[38] S. S. Sahoo et al. 2019. A Hybrid Agent-based Design Methodology for Dynamic Cross-layer Reliability in Heterogeneous Embedded Systems. In *DAC.* 6.

[39] S. S. Sahoo et al. 2020. CL(R)Early: An Early-stage DSE Methodology for Cross-Layer Reliability-aware Heterogeneous Embedded Systems. In *DAC.*

[40] Siva Satyendra Sahoo et al. 2020. Markov Chain-based Modeling and Analysis of Checkpointing with Rollback Recovery for Efficient DSE in Soft Real-time Systems. In *DFT.*

[41] Siva Satyendra Sahoo et al. 2021. Reliability-Aware Resource Management in Multi-/Many-Core Systems: A Perspective Paper. *JLPEA* 11, 1 (2021), 7.

[42] Siva Satyendra Sahoo and Akash Kumar. 2021. Using Monte Carlo Tree Search for CAD - A Case-study with Designing Cross-layer Reliability for Heterogeneous Embedded Systems. In *VLSI-SOC.*

[43] Siva Satyendra Sahoo, Akash Kumar, and Bharadwaj Veeravalli. 2016. Design and Evaluation of Reliability-oriented Task Re-Mapping in MPSoCs using Time-Series Analysis of Intermittent faults. In *DATE.* IEEE.

[44] T. Santini et al. 2015. Evaluation of Failures Masking Across the Software Stack. *MEDIAN* (2015).

[45] Amit Kumar Singh et al. 2017. A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems. *ACM Computing Surveys (CSUR)* 50, 2 (2017).

[46] Amit Kumar Singh et al. 2019. Collaborative adaptation for energy-efficient heterogeneous mobile SoCs. *IEEE Trans. Comput.* 69, 2 (2019).

[47] Amit Kumar Singh et al. 2020. Dynamic energy and thermal management of multi-core mobile platforms: A survey. *IEEE Design & Test* 37, 5 (2020).

[48] Sivert T. Sliper et al. 2020. Energy-driven computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 378 (2020).

[49] Sivert T. Sliper et al. 2020. Fused: Closed-Loop Performance and Energy Simulation of Embedded Systems. In *ISPASS.* 263–272.

[50] Shyamsundar Venkataraman et al. 2016. A Flexible Inexact TMR Technique for SRAM-based FPGAs. In *DATE.* IEEE.

[51] Jiachen Wang et al. 2020. Combating Enhanced Thermal Covert Channel in Multi-/Many-Core Systems With Channel-Aware Jamming. *IEEE TCAD* 39 (2020).

[52] Samuel C.B. Wong et al. 2020. Energy-aware HW/SW Co-modeling of Batteryless Wireless Sensor Nodes. In *Proceedings of the ENSsys.* 57–63.

[53] Li Zhang et al. 2018. Effectiveness of HT-assisted sinkhole and blackhole denial of service attacks targeting mesh networks-on-chip. *JSA* 89 (2018).

[54] Yinyuan Zhao et al. 2021. An enhanced planned obsolescence attack by aging networks-on-chip. *Journal of Systems Architecture* 117 (2021).