2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS) | 979-8-3503-3277-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/DDECS57882.2023.10139331

Bits, Flips and RISCs

Nicolas Gerlin*§, Endri Kaja*§, Fabian Vargas†, Li Lu†, Anselm Breitenreiter†,
Junchao Chen†, Markus Ulbricht†, Maribel Gomez‡, Ares Tahiraga*,
Sebastian Prebeck*‖, Eyck Jentzsch‡, Miloš Krstić†¶, Wolfgang Ecker*‖
*Infineon Technologies AG, Germany – †IHP - Leibniz Institute for High Performance Microelectronics, Germany
‡MINRES Technologies, Germany – §Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau, Germany
¶University of Potsdam, Germany – ‖Technische Universität München, Germany

***Abstract*—Electronic systems can be submitted to hostile environments leading to bit-flips or stuck-at faults and, ultimately, a system malfunction or failure. In safety-critical applications, the risks of such events should be managed to prevent injuries or material damage. This paper provides a comprehensive overview of the challenges associated with designing and verifying safe and reliable systems, as well as the potential of the RISC-V architecture in addressing these challenges.**

We present several state-of-the-art safety and reliability verification techniques in the design phase. These include a highly-automated verification flow, an automated fault injection and analysis tool, and an AI-based fault verification flow. Furthermore, we discuss core hardening and fault mitigation strategies at the design level. We focus on automated SoC hardening using model-driven development and resilient processing based on sensing and prediction for space and avionic applications.

By combining these techniques with the inherent flexibility of the RISC-V architecture, designers can develop tailored solutions that balance cost, performance, and fault tolerance to meet the requirements of various safety-critical applications in different safety domains, such as avionics, automotive, and space. The insights and methodologies presented in this paper contribute to the ongoing efforts to improve the dependability of computing systems in safety-critical environments.

***Index Terms*—GNN, Hardening, Reliability, RISC-V, Safety, Verification**

I. Introduction

In recent years, the design of electronic systems has become increasingly complex, with systems becoming more integrated and incorporating advanced functionality. This complexity, coupled with the continuous technology scaling, has made it more challenging to ensure the safety and reliability of these systems. On the other hand, in high-reliability environments such as aerospace and automotive, where human lives can be at stake, more and more electronic systems are incorporated, and ensuring enough safety and reliability becomes thus critical. A single fault in this field can have severe consequences, such as a crash leading to injuries or death, highlighting the need for a comprehensive approach to designing, verifying, and validating such systems.

This paper presents possible solutions to the challenges associated with achieving safety and reliability in electronic systems, including approaches to verification and design for safety purposes. Section II discusses the relevant fault sources to consider for safety-critical systems and how they are modeled. Understanding these concepts is crucial to the development of fault-tolerant and resilient systems. In Section III, we introduce the RISC-V architecture, explaining why it is rapidly expanding and the challenges it faces regarding safety-critical applications.

In Section IV, we delve into some verification flows tailored to ensure a high-reliability level for SoCs:

1) We take as a use-case a RISC-V soft IP core family complying with ISO 26262: The Good Core (TGC). We show how its architecture can be customized and how the development flow has been mostly automated. The details about verification and how we can ensure high confidence in our released products are also discussed.
2) We explore efficient methods to run fault injection campaigns to assess fault coverage. Our approach is based on model-driven development and a mixed RTL/Gate-Level granularity design to inject faults into. We are evaluating our solution using both simulation and emulation.
3) We investigate using Machine Learning to predict fault simulation results and examine how it speeds up the fault susceptibility analysis process. We show how Graph Neural Networks (GNNs) and Graph Convolutional Networks (GCNs) can be used instead of traditional Neural Networks (NNs). We are also discussing using analytical methods and approaches based on combinatorial search problems.

In Section V, we dig into design hardening and fault mitigation techniques:

1) Despite being widely used, IP-reuse techniques applied to SoC hardening still require a significant amount of manual effort to be implemented, limiting the reuse of a design from one project to the other. We propose an automated approach for SoC hardening based on model-driven development to overcome this issue. We show how our flow automates resilience against data corruption based on safety requirements.
2) We see that static hardening techniques need more flexibility as they can lead to unnecessary overhead. We use TETRISC (TETra core system based on RISC-V) SoC as a use-case to overcome this issue. It is a highly reliable and resilient quad-core system based on the open-source PULPissimo architecture, which uses dynamic reliability adjustment and multiple fault source monitoring. We discuss the implementation of our system, the enablement of on-board reliability state monitoring,

2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)

the dynamic real-time system reconfiguration based on user requirements and multiple on-chip monitors, and the trade-off optimization between reliability, power consumption, and performance.

Finally, in Section VI, we conclude the paper with a summary of our findings and emphasize the importance of continued research for the development of highly resilient safety-critical systems.

II. Fault sources and models

Resiliency has become an increasingly relevant feature in recent years, as it addresses issues of reliability, safety, and security in a dynamic and predictive way. Resilient systems have various levels of reliability and are used in countless application domains, such as automotive (airbag, ABS braking, fuel injection, and electronic stability control systems), defense and aerospace (on-board aircraft navigation systems, attitude, telemetry and telecommand systems in satellites), energy (nuclear power plants), medicine (life-support devices in medical equipment), and transport (air and train traffic control systems).

The use of resilient systems is essential in high-availability, mission-critical, or even life-critical applications, typically operating in harsh environments such as those involving radiation, electromagnetic interference (EMI), and high temperatures. To achieve this, designers addressing resiliency, and in particular fault tolerance, must ensure that these systems will continue to operate correctly even in the event of failures due to one or more of its components. These faults can produce permanent system failure, such as those induced by aging and total ionizing dose (TID) radiation [1] [2], as well as transient faults, such as single-event upset (SEU) in flip-flops (FFs) and RAM memories, single-event transients (SET) in logic, EMI in power and data lines [3], and crosstalk among tracks carrying high-speed signals.

It is worth noting that in real, harsh environments, systems are exposed simultaneously to various sources of faults [4]. For instance, electronics for automotive applications must operate correctly while exposed to high electromagnetic fields and temperatures [5]. Additionally, the electronics in a Low-Earth Orbit (LEO) satellite passing through the Van Allen Belts (whose passage typically takes a few minutes) are exposed to ionizing radiation (SEU) and EMI simultaneously. Therefore, these systems are prone to transient faults induced by EMI (due to conducted noise on DC input power supply and data lines), aging (due to degradation mechanisms such as Bias Temperature Instability – BTI), and bit-flips in memory elements (due to the presence of high-energy particles).

Technology scaling has made electronics accessible and affordable for almost everyone on the globe, and has advanced integrated circuit (IC) and electronics performance since the sixties. However, this scaling has introduced new and major reliability challenges to the semiconductor industry. Although the deleterious effects of total ionizing dose (TID) on ICs are gradually diminishing as technology scales down (mainly due to the reduction of the sensitive volumes represented by thin gate oxide and thick insulation layers), it is well recognized that scaling has dramatically increased the occurrence of single-event upsets (SEUs) and single-event transients (SETs), especially in fin field-effect transistor (FinFET)-based technology nodes. In FFs and SRAM cells, this is due to the reduction of sensitive volumes used to store information in memory elements and the reduction of circuit internal nodes' capacitance. Moreover, the continuous reduction of power supply renders memory elements even more sensitive to SEUs, SETs, and especially electromagnetic interference (EMI) [6]. Crosstalk in ICs is a type of EMI between two signals propagating adjacently. As EMI and crosstalk in an IC increase, the system experiences an increase in glitches, errors, and timing problems. In short, as technology scales down, the error rate consistently increases.

The literature is populated with a large variety of solutions, based on hardware, time, and information redundancies, and on-chip sensing for error prediction/detection and recovery, to guarantee the fault tolerance (FT) systems' reliability. These solutions have side-effects that must be taken into account. Extensive reliability and safety verification is required already in the design process to define such mitigation techniques. The cornerstone of such verification is a fault injection process that evaluates the system's behavior in the presence of faults. Such a process can be implemented in different ways, as will be elaborated in the next sections. Nevertheless, the baseline for any efficient fault injection campaign is the definition of the fault model to be used. The assumed fault model depends on the reliability challenges relevant to the target application, as commented in this section. The usual fault models relevant to reliability issues are stuck-at models (namely, stuck-at-0, stuck-at-1, stuck-open, stuck-on, and bridging faults) that can be extended to evaluate other reliability challenges. As a result, delay fault models (that can be used for investigating aging and total dose effects) and transient fault models (to emulate SEU, SET, EMI effects, and laser security attacks) may also be considered.

In the following sections, we will analyze in detail the modern safety and reliability verification methods important for RISC-V-based systems and summarize the potential fault mitigation methods.

III. Introduction to RISC-V

The advantages of open-source developments are well reflected in RISC-V, which is an open-source instruction set architecture (ISA) developed at the University of California, Berkeley in 2010. It is a modern, modular, and extensible ISA designed to be easily integrated in devices ranging from small embedded systems to large-scale servers and high-performance systems. RISC-V International (https://riscv.org/) is the global non-profit organisation behind the open standard RISC-V ISA, related specifications and stakeholder community. It has no commercial interest in products or services and makes possible for organisations around the world to collaborate and seek for innovation.

RISC-V is based on the Reduced Instruction Set Computer (RISC) philosophy, which advocates for a simpler instruction

set that allows a more efficient processing. It is a load-store architecture, which means that all data processing is performed on registers rather than directly on memory. It has a small base integer ISA, usable by itself, and optional standard extensions and specialized variants, to support general-purpose software development. Along with the ratified extensions for integer and floating-point arithmetic and vectored, packed-SIMD and atomic operations, there is a growing pool of specialised extension for cryptography, bit manipulation and many more. There are also activities to standardize elements like external interrupt controllers, tracing capabilities, cache support, etc.

As an open standard, RISC-V can be used by anyone without paying licensing fees, which has made it popular among academics studying the architecture and developing new tools and applications based on it. The amount of industry professionals interested in developing custom processors or hardware accelerators is also growing. The ability of RISC-V to be extended by custom instructions is also adding to its growing popularity. Currently there are several key players, SiFive, Western Digital, Andes Technologies, NXP Semiconductors, Esperanto Technologies and Google to name a few, involved in the development, implementation, and promotion of the RISC-V architecture. Being an open standard enables sharing technical know-how and rapid creation with great design freedom. It also leverages contributions to strategic future developments.

The rapid growth and increasing adoption of RISC-V is echoed by the wide range of available applications and the rapidly growing RISC-V ecosystem, with more companies and organizations getting involved in the development and promotion of the architecture. This includes not only semiconductor companies, but also software companies, system integrators, and others. There are ongoing advancements in both RISC-V hardware and software. New RISC-V-based processors are being developed and released, and new software tools and development environments being created as well.

The current state of RISC-V is very promising, which poses as well the challenges the architecture is facing. RISC-V is still a relatively young architecture compared to more established architectures like ARM and x86. As such, there may be some concerns around the maturity and stability of the architecture, particularly for mission-critical or safety-critical applications. As an open standard, there are lots of different implementations and variations of the architecture. This leads to fragmentation and compatibility issues, particularly if different implementations don't adhere to the same standards.

Although the RISC-V ecosystem is rapidly growing, it is still not as mature or robust as some of the more established architectures. This can make it more challenging for developers and companies to find the hardware and software tools they need to work with RISC-V. This relates to another challenge with RISC-V, which is the lack of mature toolchain support. While there are many tools available for RISC-V development, they may not be as feature-rich or well-supported as the tools available for more established architectures. While RISC-V is an open standard, there are still some patents associated with the architecture. This can create some uncertainty around the use and adoption of RISC-V, particularly for companies that are concerned about potential patent infringement issues.

Overall, while RISC-V is a promising architecture with many potential benefits, there are still some challenges that need to be addressed in order to fully realize its potential. These challenges may require ongoing investment and development in the RISC-V ecosystem to ensure that it remains a viable and competitive option for developers and companies.

IV. Safety/Reliability verification of RISC-V in the design phase

During design process of resilient systems it is extremely important to verify the compliance of design to certain design rules (for example for safety), as well as to investigate susceptibility of the systems to faults (by using respective fault models). In the following chapter we will present different reliability and safety verification approaches of RISC-V processor based systems.

A. Robust verification flow

Designing a RISC-V core with safety and security aspects in mind requires not only well thought processes, from architectural exploration to IP release, but also taking care of aspects like the tools and technical environments involved. Fundamental are also the safety and security awareness and the competence of the team.

At MINRES we develop our RISC-V soft IP core family (TGC - The Good Core) according to the standard ISO 26262 Road vehicles – Functional safety, which covers the whole life cycle of electronic systems used in the automotive area. TGC is developed as safety element out of context (SEooC), since it could be integrated in a variety of embedded and internet of things (IoT) or edge applications. For a SEooC, 4 of the 12 parts of the standard are particularly relevant:

- Part 2: Management of functional safety [7]
- Part 5: Product development at the hardware level Road vehicles [8]
- Part 8: Supporting processes [9]
- Part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses [10].

The safety project management, according to part 2 of the ISO 26262 standard, deals with the consecution of the project goals in the given time frame, which also guarantees the information flow to all involved people, the adequacy of the processes and the quality of the results. The resulting TGC life cycle processes are depicted in Fig. 1. It covers the planning, architecture, development and release phases, explicitly including the safety related aspects of each of them.

The planning phase starts with the collection of the functional requirements, which consistently and uniquely identify the TGC characteristics and the safety requirements, which indicate the safety features and goals that TGC must fulfill. Both, together with the project schedule, are gathered and managed in OpenProject, which is a project management tool and ticket system in a secure environment.

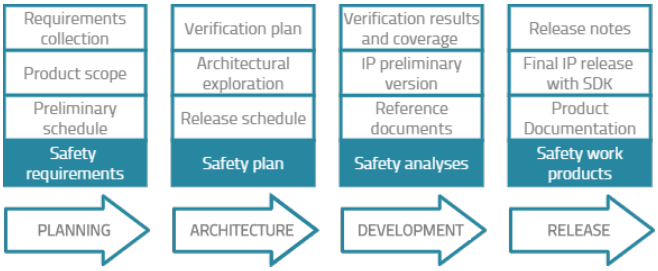

Fig. 1. TGC development life cycle

During the architecture phase, the generator-based TGC customization is analysed. This includes scaling and extending TGC to the appropriate number of pipeline stages and features, like bus interfaces, physical memory protection or support of caches, for the target application. Experimenting with custom instructions is also done in this phase. The firmware can be tried at this early development stage on the TGC instruction set simulator (ISS), which allows assessing the impact on performance of the customization decisions. The TGC ISS is part of the TGC virtual prototype (VP), which is delivered with a software development kit (SDK) containing a compiler that supports the custom instructions.

The highly automated development and verification flow used in the development phase is shown in Fig. 2. It follows the guidelines of part 5 and part 8 of the ISO 26262 standard. It is based on CoreDSL [11], a domain specific language to formally specify an instruction set architecture (ISA), as well as custom instructions. According on the TGC specific CoreDSL description, the ISS/VP and RTL artifacts as well as the tailored development tools are generated. Longnail and SCAIE-V [12], tools developed by the Technical University of Darmstadt, integrate the custom instructions into the TGC RTL.

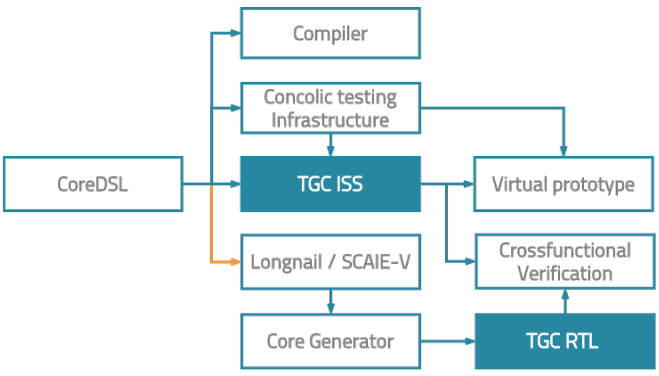

Fig. 2. TGC generation flow

The cross-level functional verification scheme, shown in Fig. 3, is a tightly-coupled co-simulation environment that consists of a SystemC UVM testbench with TLM connectors and a generator that produces a continuous stream of random instructions, which are fed into the TGC RTL and the TGC ISS, allowing result comparison between them. The random stream of instructions is reshaped on-the-fly to ensure the coverage of the whole parameter space and the execution of instruction in all possible sequences.

Functional coverage is realized by the coverage monitor, which collects the information gathered by the coverage points of the instructions. The coverage metrics analysis reveals whether all parameters of all instructions have been sufficiently exercised, including corner-cases, all hardware hazards have been correctly detected and avoided and all possible exceptions were raised.

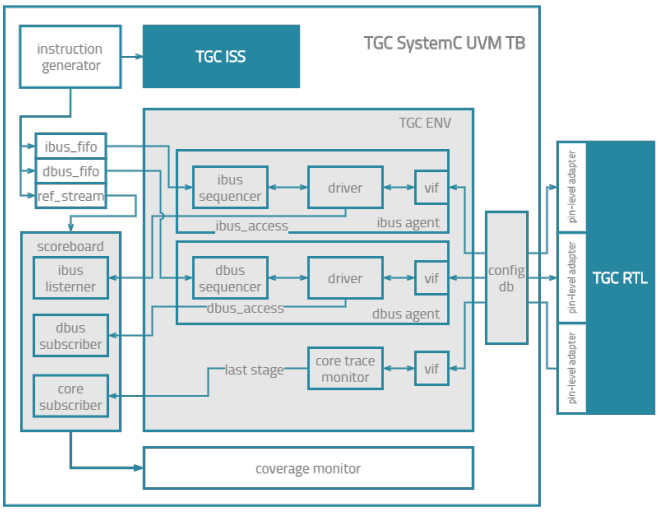

Fig. 3. TGC SystemC UVM testbench

The high level of confidence on the core functional correctness and coverage is obtained by running a daily regression on several TGC configurations covering all available features. The regression includes the random co-simulation with millions of instructions, various benchmarks:

- Dhrystone: a synthetic benchmank adequate for small cores, allowing comparison between different ISAs or the efficiency of different compilers on the same core. The performance is reported in seconds
- Coremark: which measures the core performance using basic data structures and algorithms, but avoiding pre-computed results. The compiler must be specified in order to obtain the final score
- Embench: a modern C benchmark with realistic workload that reports either the code speed, based on the cycle counter and the clock frequency, or the code area

and 3 RISC-V test suits:

- The RISC-V tests from UCB [13], which exercise the standard extensions
- The RISC-V design verification tests [14], a series of random tests tailored for each specific core
- The RISC-V architectural tests [15], for validating each core against the RISC-V ISA standard

In addition to the these verification steps, security vulnerabilities are detected by means of formal security analysis. According to part 9 of the ISO 26262 standard, safety analysis are carried out. They include the analysis of dependent failures,

the safety analyses and the confirmation measures. In particular, the FMEDA considers the hardware failures and diagnostic measures defined and gives an idea of the Automotive Safety Integrity Level (ASIL) that can be achieved by the SEooC. The achievement of the target ASIL must be nevertheless confirmed at the system level with the SEooC integrated.

Once TGC fulfills all requirements and they are verified and validated by means of a FPGA co-simulation board, the release phase begins. The automated processes of this phase gather together the TGC IP, all artifacts that belong to the software development kit (SDK), the product technical documentation and the safety work products into a customer release.

Operational integrity, state-of-the-art processes and competence assurance are the key elements of the TGC safety flow. They are ensured by the usage of highly automated state-of-the-art processes and quality management practices. End-to-end traceability, result reproducibility and transparency of all technical processes, including documentation, is ensured by version control.

B. Automated fault injection framework

A system is defined by Avizienis et al. [16] as an ”entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena.” The ever-growing size and complexity of the systems has made them more prone to failures due to different occurring faults. Thus, various techniques are utilized to achieve design robustness through fault tolerance and fault avoidance. Nevertheless, it is required to prove that the system is ”safe enough” through verification techniques. ISO26262, the international standard of automotive safety, strongly recommends *fault injection* to verify the dependability of safety-critical designs. Considering the complexity of the systems nowadays, fault injection process is cumbersome and prone to errors, therefore fast, automated, and reliable fault injection techniques are required.

Traditionally, fault injection is applied on the Gate-level (GL) representation of a design because of highly accurate fault models. However, GL fault injection is very slow and consequently different fault injection techniques are applied on a higher abstraction level such as Register-Transfer Level (RTL). RTL fault injection speeds up the process but lacks the accuracy of GL fault models. To tackle this drawback, the RTL generation flow [17] at Infineon enables generating the designs on a mixed RTL/GL granularity as presented by Kaja et al. [18]. Parts of the design which are subject of fault injection are kept on a GL granularity while the rest of the design is represented on the RTL granularity. As an example, let us assume a 5-stage pipelined processor and the Fetch-stage is the subject of fault injection. The Fetch-stage is described using the data from the netlist (on a GL) and the rest of the design is described using the original RTL. Saboteurs, i.e. small pieces of hardware that are able to inject a fault when activated through a control signal, are added to every signal of the design on the GL granularity by transforming the model of the design [18]. As a result, the designs are generated on a mixed RTL/GL representations with fault injection capabilities. This feature enables and facilitates a tool-agnostic fault analysis framework. Experimental results have shown a speedup of up to 8.4 times of mixed RTL/GL simulation compared to a full GL simulation.

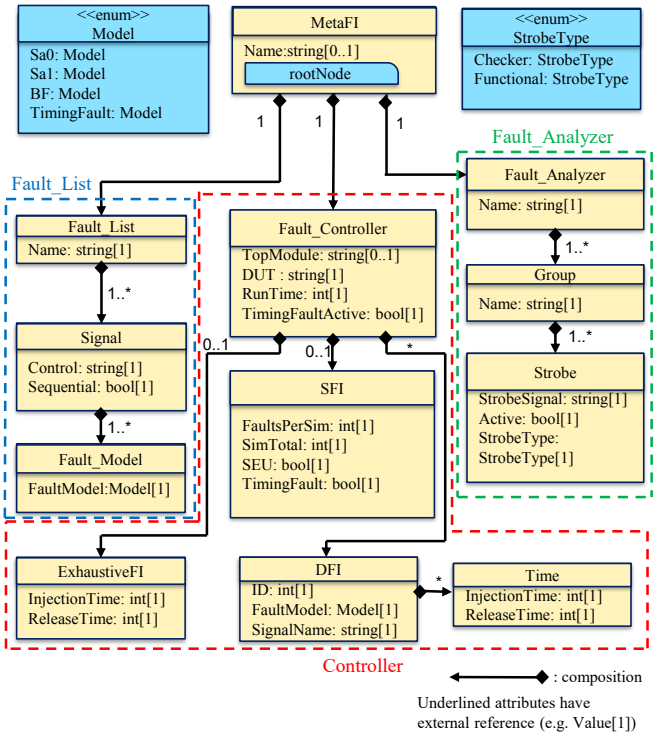

Fig. 4. Fault injection metamodel definition [19]

The core of the developed framework is the metamodel shown in Fig. 4. A metamodel represents the structure of a model and the relation of its elements while the model itself represents the system at a certain abstraction level [20]. The fault injection metamodel is composed of three main parts: *Fault List*, *Fault Analyzer* and *Controller*.

- Fault List component displays the model-based fault list. It describes the saboteur control line (*Control*), whether a signal is an input of a sequential cell (*Sequential*) and all possible fault models that could be injected to the particular signal (*Fault_Model*).
- Fault Analyzer represents the data that should be collected and analyzed while performing simulation/emulation. *Group* describes modules and submodules of the design which will be analyzed *Strobe* depicts the outputs and registers of the aforementioned modules. Furthermore, outputs are classified as functional strobes (output of the functional part of the design) or checker strobes (output of the safety mechanism(s)).
- Controller determines the total runtime, module to perform the fault injection as well as the particular fault injection campaign such as Statistical Fault Injection (SFI), Exhaustive Fault Injection (EFI) and Direct Fault Injection (DFI). SFI is utilized to provide statistical data for fault propagation and the user simply has to specify the total

amount of faults to inject and whether a particular fault model is utilized. EFI injects all stuck-at fault models and the user has to specify the injection and release time. In some cases it is required to inject faults only at particular locations and with specified fault models, and therefore, DFI can be utilized.

The integral features described above are deployed on a fault simulation and a fault emulation framework. A SystemVerilog/C++ generated testbench enables a simulator-independent fault simulation by setting the appropriate saboteur control lines according to the data from the model. Furthermore, a novel FPGA-based fault emulation framework [21] has been developed while deriving from the same metamodel. The applicability of the frameworks was demonstrated by performing fault injection on different modules of various RISC-V alternatives. The injected faults are classified as silent, failure or latent. Experimental results have shown that the emulation framework speeds up the fault injection up to 47.57 times compared to the simulation-based fault injection.

C. AI-based fault verification flow

Important part of the fault verification flow is the fault susceptibility analysis. Fault susceptibility analysis deals with the prediction of the likeliness of certain errors in digital circuits. The error propagation can be inhibited by electrical, temporal and logic masking effects. Electrical and temporal masking is addressed on different levels of abstraction, and logic masking is in focus in such analysis. Logic masking depends on the logic state of the circuit as well as on the state of the primary inputs.

In order to analyze the error propagation in sequential circuits, even the sequence of internal states and input values has to be considered for as long as the circuit is in an erroneous state. The number of all possible sequences is huge for a complex chip, so that an accurate analysis is impossible with nowadays methods.

Three classes of methods could be used for fault susceptibility analysis: methods based on fault-injection simulation, analytical methods and methods based on solving combinatorial search problems.

Simulation-based fault injection is a commonly used reliability analysis technique. It enables us to recognize critical components in a circuit in the early design stage. As a result, we could choose to conduct fault mitigation techniques to improve the system reliability. However, simulation-based fault injection approach is very time-consuming, and it is even impossible to exhaustively inject all possible faults for a complex circuit running a complicated workload. On the other hand, it is possible to use machine learning (ML) to address this problem. The process could be accelerated if we can use machine learning to predict fault simulation results rather than conduct simulations on a target circuit.

In some studies, machine learning has been used for reliability analysis. In [22], Da Rosa et al. use ML algorithms to determine the relationship between fault injection outcomes and the characteristics of applications and platforms. T. Lange et al. use ML in [23] to predict functional de-rating (FDR), which refers to the visible soft errors at the application level. K. Khalil et al. [24] employ ML to anticipate hardware defects at the transistor level, such as aging, short-circuiting, etc. However, one could use ML instead of fault simulation to predict critical flip-flops in a circuit.

Most researchers applying ML for reliability analysis use models like Linear Regression, Support Vector Regression (SVR), and Neural Networks (NNs). These models are unable to learn the connection information between components in circuits and can only learn from the characteristics of individual components. Structural characteristics of circuits are crucial to learn the patterns of fault propagation. Graph Neural Networks (GNNs) are ML models designed to learn graphs. They use the features of a node's neighbors in the graph as well as itself to calculate the feature of the node during training. And therefore, they can learn from connection information.

We propose a method to use GNNs to identify critical flip-flops in circuits and validate the approach on the RISCY core in Pulpissimo platform [25]. To utilize GNNs, the circuit should be first converted into a graph. Each flip-flop should be taken as a node in the graph and Breadth-First Search (BFS) used to search the shortest path between the flip-flops to create edges. The distance between two flip-flops could be used as the number of combinational gates on the shortest path between them. To reduce the noise in the graph model, the edges are removed from the graph, when the distance of two flip-flops is larger than the specified maximum distance.

GCN (Graph Convolutional Network) [26] are selected as GNN model. GCN is one of the most popular GNNs, and many advanced GNNs are developed based on it. With GCN, the prediction accuracy we achieved on the dataset based on RISCY core is 91.1%, about 3% higher than the accuracy achieved on NNs [27].

Analytical methods arithmetically propagate signal and fault probabilities along the circuit netlist. Although being less computational intense than fault-injection simulation, they require significant computational effort, so that they are only applicable to circuits of limited complexity.

Third class of methods are based on the solution of combinatorial search problems. The most popular example of this kind might be the problem of propositional satisfiability (SAT). It answers the question of satisfiability for a given boolean expression. Elaborate algorithms exist for the solution of this problem. However, to transform the analysis of the error propagation probability to a SAT problem, extensive preprocessing is required. For this reason, Answer Set Programming could be used, which results from the combination of Stable Model Semantics and Logic Programming. It can solve the same problems as a SAT solver, but it comes with a more powerful input language. This enables us to reduce preprocessing to a minimum and to efficiently narrow down the analysis to more special error scenarios.

However, an accurate analysis of the error propagation in a circuit still requires the enumeration of all possibilities which fails due to computational limitations. As an extension,

Approximate Model Counting to deal with this limitation [28]. The basic idea is, to split the solutions in equal parts, count the solutions in just one part and multiply by the number of parts. By this approach, we are able to reduce computational effort by orders of magnitudes.

V. Core Hardening and Fault Mitigation by Design

A wide range of mechanisms on a wide range of design and implementation levels can be considered to implement cores more robust, i.e less sensitive to bit flips, for instance. They may rank from physical design, e.g. larger Flip-Flops, to the implementation level as the use of dual chips in the system or software re-execution to name only some. This section describes two approaches on the design level, namely, *automated hardening SoC hardening* and *resilient processing based on sensing and prediction for Space/Avionic applications*.

A. Automated SoC hardening

Safety standards (ISO 26262, DO-178, ECSS, ...) define methodologies to classify each task depending on its criticality. The nature and amount of safety mechanisms to be implemented vary according to the standard and the criticality level. While IP-reuse attempts to reduce the development effort, selecting the ideal trade-off between cost and reliability still requires a thorough design analysis, making it challenging to follow state-of-the-art techniques.

A noticeable point about the process of adding safety mechanisms is that while it is hardening a given design, it is not affecting its functionality. Model-driven development and design-centric modeling of digital hardware have already shown a significant productivity increase [29]. Fig. 5 shows the different steps of this generation flow. From formalized specifications and a design template written in Python representing all possible configurations, we generate a Model of Design (MoD). This MoD describes the micro-architecture of the design while being independent of the RTL language used. From there, we generate a Model of View (MoV) that is the base for producing RTL files. We enhance this approach by adding functional safety features within the MoD layer; safety constraints are applied to the original MoD to obtain a transformed one with various safety mechanisms against data corruption.

Our model-driven transformation approach takes a base MoD as input and transforms its flip-flops according to the desired safety constraints to generate a hardened MoD. The safety constraints consist of safety groups representing the modules in which the flip-flops we want to harden are included, as shown in Fig. 6. Each safety group has one safety method associated. As of now, we support Dual Modular Redundancy (DMR), Triple Modular Redundancy (TMR), Error Detection Code (EDC) with Parity (PAR), and Error Correction Code (ECC) with Hamming code for Single Error Correction (SEC) and extended Hamming code for additional Double Error Detection (SECDED). The targeted flip-flops are then defined in the critical component group. If only specific bitfields of a flip-flop need to be protected, they can be specified within the bit range class.

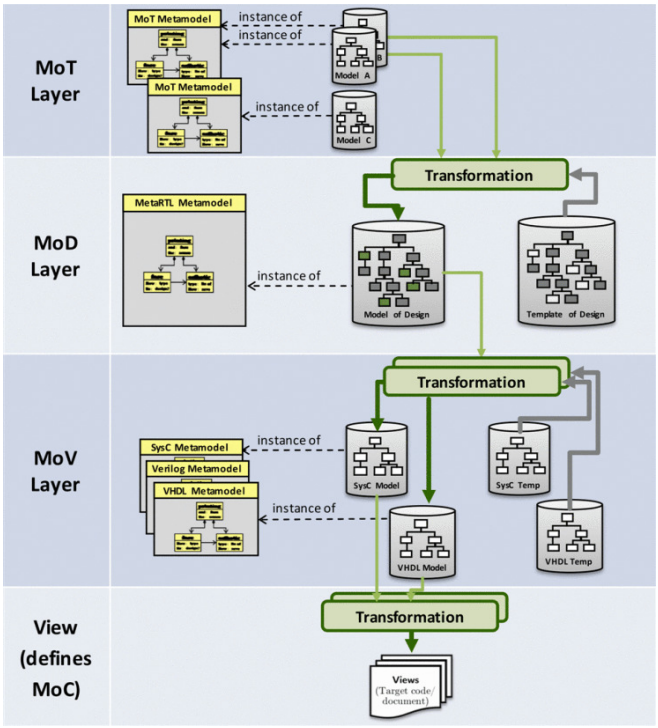

Fig. 5. RTL generation flow [20]

Once the safety constraints have been set as desired, a Python script will locate the so-called safety groups in the base MoD and the selected flip-flops and bitfields inside them. For each of the latter, we create a wrapper containing a copy of it with the additional safety mechanism. The original flip-flop is then deleted and replaced by the corresponding wrapper. After every safety constraint has been treated, the output is a hardened MoD that can be integrated into our generation flow to generate RTL code.

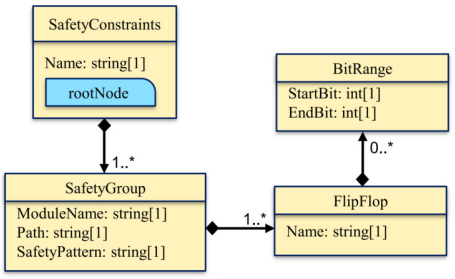

Fig. 6. Safety transformation metamodel

Additionally to the above-mentioned available features, each protected flip-flop can be provided with a Built-In Self-Test (BIST) against stuck-at faults. A central control unit is created to manage these tests and is automatically connected to the wrappers. Similarly, an error detection signal is automatically propagated from the wrappers to a central alarm unit.

Our approach reduces the manual effort to protect a design, as

the only requirement is to select which part we want to harden and how. Noticeably, we do not need to modify the original design; it can therefore be reused in different applications. The cost of the safety mechanisms is also easy to assess, as the original design does not include any of them.

Besides the flexibility offered by our solution, the total time spent from writing the safety constraints to the final RTL code is in the range of 5-10 minutes for a 5-stage RISC-V core. The RTL generation can be parallelized so that this time can be reduced to 1-5min to specify each safety constraint and about 5 minutes to generate all hardened MoDs and RTL codes. Many implementations can therefore be tested for fault coverage and cost, increasing the chances of finding the best trade-off. Another advantage is that the original design can be functionally verified before transformation, hence having a shared verification process between several applications. We are leveraging this by using formal verification tools to perform equivalence checking between the hardened designs and the original one, making the whole verification process faster and independent of the mechanisms used.

B. Resilient processing based on sensing and prediction

With the scaling of CMOS technology into the deep nanometer range and the aggressive reduction of supply voltage, the design of Integrated Circuits (ICs) for safety- and mission-critical applications in fields such as aviation, space, and automotive, is encountering an increasing number of reliability challenges related to faults. Among the various sources of faults, radiation-induced effects, aging, and temperature are particularly significant during the operational use. Failure to address these effects can result in degraded performance, data corruption, and even catastrophic failures. However, conventional methods for system protection often involve a series of static hardening techniques that may lead to unnecessary overhead and lack the flexibility to adapt to dynamic working conditions. Therefore, the integration of flexible mitigation measures and robust system status monitoring is essential for enhancing the reliability of ICs in complex operating environments. As we have already mentioned, RISC-V ISA has raised significant attention on the academic and industry side in the last years. One of the promising application field is in the reliability critical domains, such as avionic, space etc. The open question is how to achieve resilient but still adaptive operation of such processing system.

A highly reliable resilient multiprocessing system, named TETRISC (TETra Core System based on RISC-V) SoC, has been proposed to address the dynamic reliability adjustment of the system in harsh environments, while simultaneously monitoring multiple fault sources. The TETRISC SoC is a quad-core multiprocessing platform that utilizes the open-source PULPissimo [25] single-core architecture. The block diagram of the implemented SoC is shown in Figure 7, where the green and orange blocks are newly implemented or upgraded components and the remaining gray blocks are from the original PULPissimo platform. The proposed design has four primary objectives: (1) to implement a RISC-V

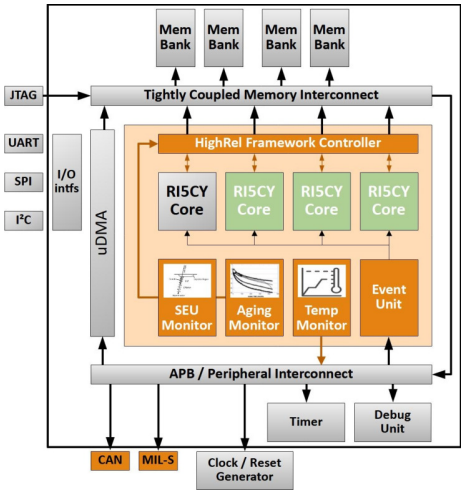

Fig. 7. Block diagram of the proposed TETRISC SoC

based resilient multiprocessing system, (2) to enable on-board reliability state monitoring, (3) to dynamically reconfigure the system utilizing different operating modes in real-time based on multiple on-chip monitors and user requirements, and (4) to achieve optimized switching between different operating modes enabling trade-off between reliability, power consumption and performance. The proposed design as a whole contains two main resilient modules: an on-chip detection system composed of multiple monitors, and a HighRel Framework Controller (HFC) designed to adjust the operating mode. These subsystems will be thoroughly elaborated in the following paragraphs.

The on-chip fault detection system includes three existing monitor designs: the Single Event Upset (SEU) monitor [30], aging monitor, and temperature monitor. The SEU monitor is an SRAM-based on-board radiation monitor that can detect and correct radiation-induced transient bit upsets or permanent faults at a negligible cost. Its main objective is to enable real-time detection of on-board radiation threats and forecast future radiation conditions. This feature is crucial in space applications and other contexts where radiation levels may fluctuate significantly. The proposed monitor [30] utilizes an Error Detection and Correction (EDAC) method, a scrubbing module, and a dedicated detection process to identify all upsets and distinguish the type of faults that occurs in each SRAM word. The SEU monitor system is integrated into the memory interface controller and arbitration tree of the Tightly Coupled Memory Interconnect (TCMI) (as shown in Figure7), which can take control of the access of mem banks. Each memory word comprises 40 bits, with 32 bits used for data storage and the remaining 8 bits for the corresponding EDAC parity bit. To prevent access hazards during the detection of SRAM faults, the TCMI has been designed with a specific control mechanism that can enable or disable memory block access. In addition, the SEU detection process can be periodically triggered by the SoC or directly manipulated by the user through control registers. Furthermore, as discussed in reference [31], this monitor can also be used to forecast future radiation conditions by utilizing collected historical error rate data and running

a customized program from a pre-trained machine learning model.

The aging monitor is designed to detect variations in aging of the target module, thereby preventing degradation and aging imbalance. The implemented aging monitor utilizes standard library cells to create a simple and flexible design that can be integrated with the target module. It works by detecting the increase in transistor input and output delays that occurs due to aging degradation. Furthermore, this monitor can generate an "aging code" that reflects the degradation of the module's performance over time. Each core in the system has its own aging monitor, which is integrated into the HFC platform. Therefore, the system can easily acquire the aging information of cores and take countermeasures, such as clock-gating, if one or more cores are deemed too old. Furthermore, temperature can also have a significant impact on ICs, including effects on system performance, leakage current, and material degradation. Therefore, it is essential to monitor the on-board temperature. An on-chip analog temperature monitor has been integrated into the design, with an Analog-to-Digital Converter (ADC) implemented for real-time data processing and analysis.

The primary function of the HFC is to regulate the input and output of the four cores, thereby enabling a diverse set of operational modes based on core-level N-Module Redundancy (NMR) and clock-gating techniques. Under normal conditions, the four cores can independently execute different programs. However, in high-reliability situations, the HFC can facilitate parallel execution of the same program by two, three, or all four cores, thereby enabling varying levels of fault tolerance in the system. The main component in the HFC is a binary matrix-based programmable NMR majority voter for multiprocessors [32], where each processing core could be selected whether it takes part in voting. Moreover, this voter can provide the status of inputs and can change from 2MR to 4MR systems with any combination of active processors, which is an NMR on-demand system. Additionally, as the primary control component of the SoC, it internally integrates various control registers, which provides users with the convenience of controlling operating modes and obtaining real-time system status information. Moreover, the design includes a custom-designed shadow register for the cores, allowing for rapid switching and synchronization between different modes and core tasks. This approach enables synchronization of tasks between different cores during NMR modes to be achieved in just two clock cycles.

The TETRISC SoC chip is fabricated using IHP 130 nm technology and utilizes a standard cell library for four RISC-V cores, with a rad-hard cell library employed for the remainder of the design. The chip area is 43.56 mm2, with 39.17% allocated to four 8192 x 40-bit shared L2 SRAM blocks that can also serve as sensing elements for radiation monitoring. The chip has been recently fabricated and it is currently under test.

VI. Conclusion

This paper has provided a comprehensive overview of the challenges and state-of-the-art techniques in designing safe and reliable computing systems for safety-critical applications. We have explored the various fault sources and models that impact system reliability and the potential of the RISC-V architecture in addressing these challenges.

Several safety verification methodologies have been presented. We first looked into the TGC development lifecycle, showing how we use an Instruction Set Simulator (ISS) to customize its architecture and how CoreDSL, our domain-specific language, enabled the automation of the ISA and custom instructions modeling. Using a SystemC UVM testbench with a random stream of instructions, we validated our design by comparing the results from the RTL with the ISS ones. We assess functional correctness and fault coverage with daily regression tests, benchmarks, RISC-V test suits, and safety analysis. The final release of an IP includes the TGC itself with SDK artifacts, technical documentation, and safety work products. Overall, the TGC safety flow relies on highly automated processes, quality management practices, and version control to ensure operational integrity, traceability, and transparency. Later, we investigated an efficient fault injection technique based on model-driven development and mixed RTL/Gate-Level granularity. Our metamodel is built with three main parts: a fault list, a fault analyzer, and a fault controller. We tested our framework on both simulation and emulation systems, showing significant speedup compared to traditional approaches. The emulation framework proved to be 47.57 times faster than the simulation one, opening new possibilities for safety verification. Then, we explored using GCN models to predict fault simulation results and speed up the fault susceptibility analysis process. We showed that GNNs learn from connection information to identify critical FFs, and our method outperforms the traditional NNs, with a prediction accuracy of 91.1% on the RISCY core dataset. We also discussed using analytical methods and approaches based on combinatorial search problems such as SAT solvers and Answer Set Programming (ASP). We used Approximate Model Counting to overcome computational limitations in analyzing error propagation.

Furthermore, core hardening and fault mitigation techniques at the design level have been discussed. We first proposed an automated SoC hardening flow based on model-driven development. Our method demonstrated higher flexibility than the standard IP-reuse approach by clearly separating the design functionality and its safety requirements. This approach fosters the reuse of designs from one project to another. Also, it eases verification: the original design can be functionally verified, and the hardened designs only need to be proved equivalent to the original one with formal verification tools without any change required. The hardened design now only needs to be tested for fault coverage. Lastly, we showed with the TETRISC use-case that the integration of flexible mitigation measures and robust system status monitoring helps to overcome unnecessary overhead created by standard static hardening techniques. Our system is suitable for harsh environments and incorporates dynamic reliability adjustment and multiple fault source monitoring. We discussed the implementation

of our system, the enablement of on-board reliability state monitoring, the dynamic real-time system reconfiguration based on user requirements and multiple on-chip monitors, and the trade-off optimization between reliability, power consumption, and performance. The on-chip detection system, including multiple monitors, and the HighRel Framework Controller (HFC), switching operating modes, work together to monitor faults and make necessary adjustments to maintain system reliability.

The advances in verification and design hardening methodologies presented in this paper demonstrate the potential for improving the development of resilient systems used in safety-critical domains, especially with RISC-V-based systems. By combining these techniques with the inherent flexibility of the RISC-V architecture, designers can develop tailored solutions that balance cost, performance, and fault tolerance to meet the requirements of various applications in avionics, automotive, and space. As technology continues to evolve, further advancements will be crucial in ensuring the dependability of computing systems. In this regard, our future work includes scaling ASP-based verification to entire processing systems and investigating the use of formal verification methods for fault injection techniques. Furthermore, we will develop our TETRISC platform with a focus on performance, AI engine, and verification in radiation environments, and we will couple our metamodel-based transformation flow with a RISC-V CPU to be used as a fault handler.

VII. Acknowledgements

Part of the work has been performed in the project ArchitectECA2030 under grant agreement No 877539. The project is co-funded by grants from Germany, Netherlands, Czech Republic, Austria, Norway, France and Electronic Component Systems for European Leadership Joint Undertaking (ECSEL JU). Part of the work described herein is also funded by the German Federal Ministry of Education and Research (BMBF) as part of the research project Scale4Edge (16ME0122K).

References

[1] B. Halak, *Ageing of Integrated Circuits: Causes, Effects and Mitigation Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2019.

[2] J. Srour *et al.*, "Radiation effects on microelectronics in space," *Proceedings of the IEEE*, vol. 76, no. 11, pp. 1443–1469, 1988.

[3] M. Ramdani *et al.*, "The Electromagnetic Compatibility of Integrated Circuits-Past, Present, and Future," *IEEE Transactions on Electromagnetic Compatibility*, vol. 51, no. 1, pp. 78–100, Feb. 2009. [Online]. Available: https://hal.science/hal-02523680

[4] R. Goerl *et al.*, "Combined ionizing radiation & electromagnetic interference test procedure to achieve reliable integrated circuits," *Microelectronics Reliability*, vol. 100-101, p. 113341, 2019, 30th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271419305098

[5] J. Benfica *et al.*, "Conducted EMI susceptibility analysis of a COTS processor as function of aging," *Microelectronics Reliability*, vol. 114, p. 113884, 2020, 31st European Symposium on Reliability of Electron Devices, Failure Physics and Analysis, ESREF 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271420305667

[6] ——, "Analysis of SRAM-based FPGA SEU sensitivity to combined EMI and TID-imprinted effects," *IEEE Transactions on Nuclear Science*, vol. 63, no. 2, pp. 1294–1300, 2016.

[7] "ISO 26262-2:2018 Road vehicles — Functional safety — Part 2: Management of functional safety," https://www.iso.org/standard/68384.html.

[8] "ISO 26262-5:2018 Road vehicles — Functional safety — Part 5: Product development at the hardware level," https://www.iso.org/standard/68387.html.

[9] "ISO 26262-8:2018 Road vehicles — Functional safety — Part 8: Supporting processes," https://www.iso.org/standard/68390.html.

[10] "ISO 26262-9:2018 Road vehicles — Functional safety — Part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses," https://www.iso.org/standard/68391.html.

[11] "CoreDSL 2.0 - Xtext project to parse CoreDSL files," https://www.minres.com/work/coredsl/.

[12] "SCAIE-V: an open-source SCAlable interface for ISA extensions for RISC-V processors," https://dl.acm.org/doi/10.1145/3489517.3530432.

[13] "Unit tests for RISC-V processors," https://github.com/riscv-software-src/riscv-tests.

[14] "RISCV-DV SV/UVM based open-source instruction generator," https://github.com/chipsalliance/riscv-dv.

[15] "RISC-V Architecture Test SIG," https://github.com/riscv-non-isa/riscv-arch-test.

[16] A. Avizienis *et al.*, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[17] J. Schreiner *et al.*, "Design centric modeling of digital hardware," in *2016 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, 2016, pp. 46–52.

[18] E. K. et al., "Towards fault simulation at mixed register-transfer/gate-level models," in *IEEE International Symposium on DFT in VLSI and Nanotechnology Systems*, 2021, pp. 1–6.

[19] E. Kaja *et al.*, "MetaFS: Model-driven Fault Simulation Framework," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022, pp. 1–4.

[20] W. Ecker *et al.*, "Introducing model-of-things (mot) and model-of-design (mod) for simpler and more efficient hardware generators," in *2016 IFIP/IEEE International Conference on VLSI-SoC*, Sept 2016, pp. 1–6.

[21] E. Kaja *et al.*, "Fast and Accurate Model-Driven FPGA-based System-Level Fault Emulation," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2022, pp. 1–6.

[22] F. R. da Rosa *et al.*, "Using machine learning techniques to evaluate multicore soft error reliability," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2151–2164, 2019.

[23] T. Lange *et al.*, "Machine learning clustering techniques for selective mitigation of critical design features," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2020, pp. 1–7.

[24] K. Khalil *et al.*, "Machine learning-based approach for hardware faults prediction," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3880–3892, 2020.

[25] P. D. Schiavone *et al.*, "Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX," in *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2018, pp. 1–3.

[26] T. N. Kipf *et al.*, "Semi-supervised classification with graph convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02907

[27] L. Lu *et al.*, "A methodology for identifying critical sequential circuits with graph convolutional networks," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022, pp. 20–25.

[28] A. Breitenreiter *et al.*, "Fast error propagation probability estimates by answer set programming and approximate model counting," *IEEE Access*, vol. 10, pp. 51 814–51 825, 2022.

[29] J. Schreiner *et al.*, "Design centric modeling of digital hardware," in *2016 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, 2016, pp. 46–52.

[30] J. Chen *et al.*, "Design of SRAM-Based Low-Cost SEU Monitor for Self-Adaptive Multiprocessing Systems," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 2019, pp. 514–521.

[31] ——, "Solar Particle Event and Single Event Upset Prediction from SRAM-Based Monitor and Supervised Machine Learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 564–580, 2022.

[32] A. Simevski *et al.*, "Scalable design of a programmable NMR voter with inputs' state descriptor and self-checking capability," in *2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2012, pp. 182–189.