PULP Fiction No More—Dependable PULP Systems for Space

Conference Paper

Author(s):
Ulbricht, Markus; Tortorella, Yvan; Rogenmoser, Michael; Lu, Li; Chen, Junchao; Conti, Francesco; Krstic, Milos; Benini, Luca

Publication date:
2023

Permanent link:
https://doi.org/10.3929/ethz-b-000627364

Originally published in:
https://doi.org/10.1109/ETS56758.2023.10174164

Funding acknowledgement:
877056 - A Cognitive Fractal and Secure EDGE based on an unique Open-Safe-Reliable-Low Power Hardware Platform Node (EC)

PULP Fiction No More— Dependable PULP Systems for Space

Markus Ulbricht*, Yvan Tortorella†, Michael Rogenmoser‡, Li Lu*, Junchao Chen*, Francesco Conti†, Milos Krstic*§, Luca Benini†‡
*IHP - Leibniz Institute for High Performance Microelectronics, Frankfurt (Oder), Germany
†DEI Department, University of Bologna, Bologna, Italy
‡Integrated Systems Laboratory (IIS), ETH Zürich, Zürich, Switzerland
§University of Potsdam, Potsdam, Germany
ulbricht|lu|chen|krstic@ihp-microelectronics.com, yvan.tortorella|f.conti@unibo.it, michaero|lbenini@iis.ee.ethz.ch

***Abstract*—Due to their flexibility and openness, the RISC-V ISA and processor architectures have emerged as notable contenders in various application domains. Their advantages over commercial solutions have attracted the interest of academia and industry and even led to their planned adoption in aeronautics and space. However, in these demanding environments, system reliability is of paramount importance. To address this issue, this paper presents an overview of several hardware-centric approaches for developing reliable systems based on the parallel-ultra low power (PULP) open-source RISC-V hardware platform. These approaches range from gate-level optimizations to system-level improvements and highlight the versatility of the PULP architecture and its potential as a viable architecture for developing various aerospace platforms.**

***Index Terms*—RISC-V, PULP, Reliability, Fault Tolerance, Fault injection, Selective Hardening, UVM, Machine Learning, Hardware Acceleration, ECC, Adaptivity, SEU**

I. Introduction

The RISC-V instruction set architecture (ISA) has gained significant traction in academia and industry over the past years. Its open-source nature has created a collaborative environment for hardware developers to innovate and develop new hardware systems [1]. The flexibility and versatility of the RISC-V ISA led to its use in various applications, including machine learning, automotive, and aerospace.

The increasing demand for open systems and the inherent adaptability for specific applications are fueling the growth of RISC-V-based hardware in industries. SiFive, for example, has developed a wide range of RISC-V-based processors that can be used in various applications such as AI, networking, and storage [2]. Western Digital supports RISC-V as part of its strategy to intensify the development of purpose-built architectures that bring memory closer to compute. One of their many highlights is the SweRV Core EH2, which they claim to support "the running of two simultaneous threads on top of its two-way superscalar architecture, enabling 6.3 Coremarks/Mhz simulated performance" [3]. As a European company, MINRES has developed a RISC-V soft IP core family called *The Good Core* [4], which is built according to the ISO 26262 standard and targets functional safety and reliability.

RISC-V has also become a popular research platform for exploring new computer architectures and applications in academia. Many universities and research institutions use RISC-V-based hardware to develop new systems and applications. For example, the Parallel Ultra-Low-Power (PULP) [5] platform developed at ETH Zürich and the University of Bologna is an open-source hardware and software platform under a liberal license which allows unconstrained use for research, education as well as in for industrial development that released several RISC-V-based processors [6]. The PULP platform aims to provide energy-efficient and high-performance solutions for the Internet of Things (IoT), wearable devices, and autonomous vehicles. In addition, the PULP IPs are highly configurable and enable the optimization of power and performance for specific applications. Another example of an academic project is the Chipyard framework, which illustrates the potential of RISC-V-based open-source hardware. It is a framework for designing and testing customizable RISC-V-based systems and provides a set of tools and libraries for designing processors, memory subsystems, and interconnects [7].

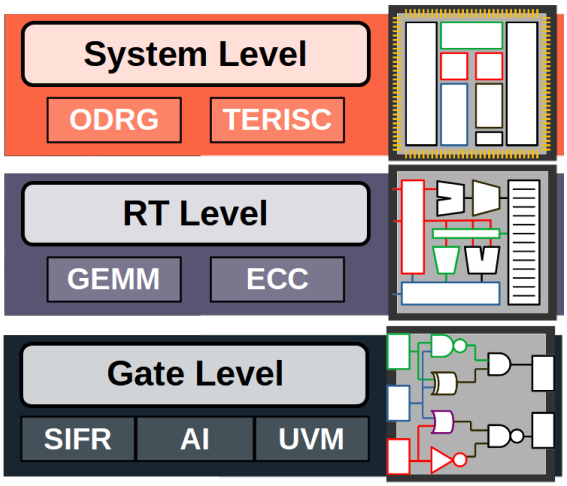

Fig. 1. addressed abstraction levels

As open-source hardware gradually finds its way into more critical applications, the need for reliable systems becomes increasingly important. In aerospace, reliability is critical due to the harsh and remote environments in which systems operate. As a result, developing reliable open-source hardware is an active area of research to ensure system reliability and safety and is making rapid progress. The NOEL-V processor by Frontgrade Gaisler [8], for example, was the first RISC-V-based processor that has started operation in a European nanosat in space [9]. The success of this has inspired many follow-up works that enrich the NOEL-V environment. De-RISC [10] is a European Horizon 2020 project to investigate a space-grade MPSoC and qualified hypervisor based on the RISC-V ISA.

Other research addresses dependable RISC-V systems on a more abstract level. Di Mascio et al. [11], for example, propose a framework to evaluate the fitness of a microarchitecture for environments where failure rates are dominated by soft errors and allow the inclusion of considerations on soft errors when selecting and configuring an open-source IP core (like RISC-V). Recently, even the main space agencies are recognizing and promoting the importance of dependable RISC-V-based systems. ESA has published a roadmap to leverage RISC-V in space applications [12] and is having workshops [13] to make them more known to the interested public. Furthermore, NASA has discovered the benefits of open-source hardware and plans to use it on future missions [14].

The PULP architecture mentioned above, with its high configurability and heavy emphasis on low power consumption, builds a solid basis for the various computational needs of avionic or space applications, ranging from small sensing devices to Linux-capable processing systems. This paper provides an overview of different research approaches that target reliability as another vital characteristic to enable the PULP platform as a reliable companion for harsh environments. As a general introduction, the following chapter introduces the PULP platform and summarizes its main properties and architectural features. The subsequent sections then introduce our research at different abstraction layers.

Figure 1 summarizes the presented abstraction layers and respective methods. The lowest layer is the gate level, which we address with fault simulations in section III and selective hardening based on artificial intelligence in section IV. Methods that increase the fault tolerance on the more abstract register transfer (RT) level will follow in section V, which focuses on memory banks, and section VI, with insights on a fault-tolerant hardware accelerator. Sections VII and VIII then move on to our investigations at the system level and present two adaptive and resilient multicore systems that offer the performance, flexibility, and reliability required in harsh and remote environments that are encountered during high altitude flights or space missions. Section IX concludes the paper, with summarizing remarks and suggestions for future research, that will further enhance the reliability of PULP systems for space.

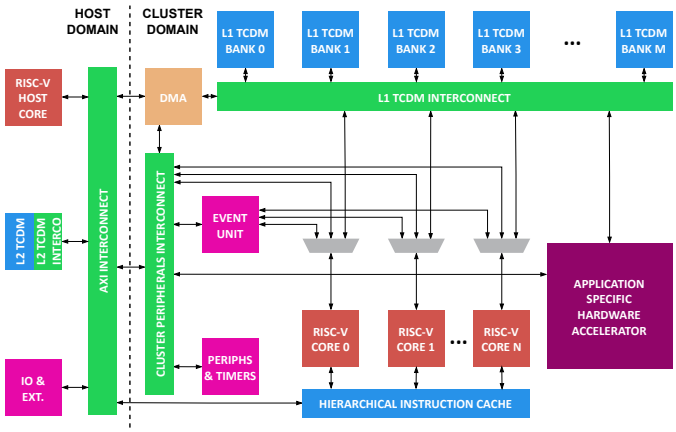

Fig. 2. PULP architecture.

II. The PULP System

The Parallel Ultra-Low-Power (PULP) [5] platform offers an open-source template and all the hardware IPs that can be used to build reliable digital computing systems ranging from small micro-controllers up to Linux-capable single-board computers, with several options for software- and hardware-based acceleration. Figure 2 depicts the architecture of the PULP platform, which is divided into two main domains.

The first domain is the host domain [15]. It is a system based on a single RISC-V controller core that can be either a 32-bit [16] core for the execution of ultra-low-power real-time applications or a 64-bit [17] Linux-capable core for mid-to-high-end applications, and both cores are industrially verified by the OpenHardware Group [18]. The host domain typically includes an L2 Tightly-Coupled Data Memory (TCDM), whose size typically ranges between 512 KiB and 2 MiB, and a ROM storing the boot code. Furthermore, the host domain includes a complete set of peripherals, such as Quad SPI, I2C, I2S, UART, and a DDR HyperBus interface to extend the size of the on-chip memory, and a JTAG interface that allows for system debugging.

The second domain of the PULP system is the cluster domain, which acts as a parallel accelerator to the host. The PULP cluster [19] features a parametric number of 32-bit RISC-V CV32E40P cores [16] varying from 2 to 16. The CV32E40P architecture is a simple in-order four-stage pipeline based on the Harvard template, extending the baseline *RV32IMC* ISA with *Xpulp* extensions for efficient digital signal processing. In the PULP cluster, the cores' instruction interface connects them to a hierarchical instruction cache [20] made of private (512 B per core) and shared (4 kiB) banks for improved application performance on parallel workloads. On the other hand, the cores connect to the rest of the system through a data interface. This interface allows connection to the L1 TCDM, whose size typically ranges between 64 and 256 kiB, and that features a parametric number of 32-bit word-interleaved memory banks accessible through a full crossbar

with single-cycle access latency, with a banking factor of two that allows for minimizing the memory contention probability on memory-intensive workloads.

The data interface also connects the cores to a peripheral interconnect to access memory-mapped devices within the cluster and the host domain. One of these peripherals is the Direct Memory Access Controller (DMAC), allowing for data transfers of up to 64 bit/cycle between the L1 TCDM and the L2 TCDM or other external memories. The cores can also access an event unit [21] for synchronization barriers and interrupt handling within the cluster. The communication within the PULP cluster and any other external domains, like the host domain, is allowed through a dedicated AXI crossbar in the PULP cluster's domain.

PULP clusters can be further extended with architectural heterogeneity, integrating application-specific hardware acceleration units that share the L1 TCDM with the cluster cores and can be configured through a memory-mapped register file [22], [23]. The PULP project provides standardized interfaces[1] to integrate these hardware accelerators without redesigning interfaces from scratch, reusing designs which can provide standard (32-bit) or high-bandwidth (up to 512-bit/cycle) access to L1 TCDM.

The host subsystem's controller nature and the cluster domain's multi-core nature provide the opportunity to be extended with fault-tolerant features to build reliable systems, exploiting architectural or component-level redundancy.

III. Fault simulation on Ibex with UVM testbench

As the scaling of feature sizes of circuits continuously decreases and the complexity of SoCs steadily grows, the probability of system failures caused by soft errors in microprocessors increases. Circuit reliability analysis during the early design stage is crucial to identify the vulnerable parts of a microprocessor. A commonly used approach is to use simulation-based fault injection [24], where Universal Verification Methodology (UVM) represents the most widely adopted simulation verification methodology in the industry. Testbenches based on UVM facilitate coverage-driven verification, which ensures a balance between verification completeness and minimum verification effort and time.

A. Workflow and Optimization

Work has already been done in the domain of fault simulation, also with a focus on RISC-V-based processors [25]. We propose an optimized workflow for conducting simulation-based fault injection at the gate level (GL) on the Ibex core [26] using the UVM testbench [27] provided by ETH Zurich and the University of Bologna. The testbench generates compiled instruction binaries using an open-source RISCV-DV random instruction generator. We simulate the generated instructions on an Ibex core and compare the core trace log to a *golden* ISS trace log to ensure the program is executed correctly. We use the Xcelium fault simulator provided by Cadence for fault simulation ([28]).

[1] http://hwpe-doc.rtfd.io

As can be seen in later parts, the cost for the high accuracy of the gate-level simulations is an extremely long runtime. An important goal of the research on hand is to optimize this from three perspectives: 1) Performing the simulations at the RTL, rather than GL, because RT level simulations are much faster, 2) optimal test case selection based on functional coverage and 3) fault pruning to reduce the number of simulations. We will describe the details of the optimized approach in the following subsections.

1) Test Case Selection: To reduce simulation time, we first analyze the contributions of the test cases provided in [27] to the functional coverage using Integrated Metrics Center (IMC) by Cadence. We select the cases with higher contributions and modify or merge them to reduce the overall number from 39 to 6 to cover about 98.2% functions. The selected test cases and their descriptions are listed in Table I.

2) Fault Injection Flow, SIFR and Test Case Pruning: The simulation-based fault injection is then moved from GL, for which the test cases were originally created by the UVM, to the RTL to increase simulation speed. We conduct the simulations for each test case in the order of the coverage contribution from high to low. Despite aiming for single event upsets (SEUs), we initially inject permanent faults (i.e., stuck-at-1 or stuck-at-0) into all flip-flops. If the fault in a flip-flop is detected in the output interface of the Ibex core, we subsequently inject multiple bit-flips with an even distribution over the time window from the activation time to the end of the simulation, to imitate the effect of SEUs. The SEU-induced failure rate (SIFR) of an FF under a certain test load describes the probability of the SEU propagating to the primary outputs of the Ibex core. If n faults were injected during the time window, and m were detected, the SIFR for the FF under the given load was calculated using Equation 1.

$$SIFR = \frac{t-x}{t} \times \frac{m}{n} \tag{1}$$

If neither of the two permanent errors of an FF was detected, its SIFR is set to 0 right away. We set flip-flops as *critical* when their SIFRs are larger than a certain threshold that should be specified in the safety requirements of the system (in our case, we set it to 0,1). When the simulation of one test case is finished, we remove the identified critical flip-flops from the fault list of the following test case and repeat the process until the simulation of all test cases is completed.

B. Results

Table I summarizes the simulation time and number of runs needed for each test case, which gradually decreases. The time for the overall fault simulation is given in the last row. It is based on a single-core Intel Xeon processor E5-4627V2 and can be reduced considerably by parallelization on multiple cores.

In addition to the simulation time, we were able to extract other interesting information on the criticality of the flip-flops in an Ibex core during our work. The number of critical flip-flops identified by each test case in each module in the Ibex

TABLE I
FAULT SIMULATION ON IBEX.

Test ID	Test Name	Descriptions	Simulation Time (ns)	Number of Runs	Number of Detected Critical Flip-flops	Time (hours)
1	rand_jump_test	Jump among large number of sub-programs.	59,400,814	23314	590	5196
2	pmp_basic_test	Basic physical memory protection (PMP) test.	3,611,144	8044	62	81
3	debug_branch_jump_test	Randomly assert debug requests, insert branch instructions and subprograms into the debug rom to make core jump around within the the debug rom	21,970,704	6640	53	338
4	mem_error_test	Randomly insert instruction fetch or memory load/store errors.	9,818,104	5942	327	206
5	debug_ebreakmu_test	Set dcsr.ebreakm at the beginning of the test upon the first entry into the debug rom.	13,274,244	2627	101	80
6	invalid_csr_test	Boot core into random privilege mode and generate csr accesses to invalid CSRs.	2,261,544	2258	0	23

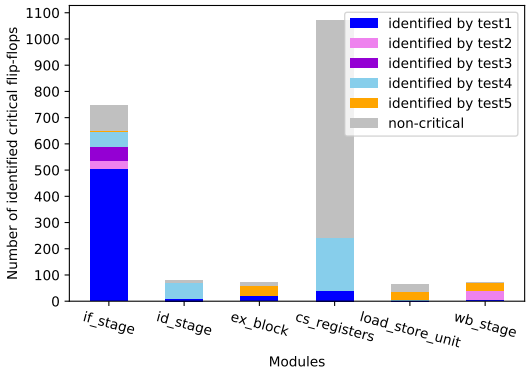

Fig. 3. The number of critical flip-flops on Ibex

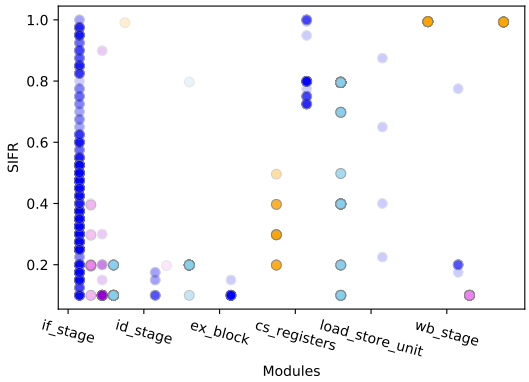

Fig. 4. The SIFRs of critical flip-flops. Each point represents the failure rate of a flip-flop, with the color representing the method (as in figure 3) and the x-coordinate representing the processor's pipeline stage of the flip-flop.

core can be seen in figure 3 and the respective SIFRs in figure 4. The module of *if_stage* has the most significant number of critical flip-flops, and most of them are identified by the *test1* which is *rand_jump_test* (see Table I). The SIFRs of critical flip-flops recognized by *test5* (represented by orange points, the color is the same as in figure 3) in the module *wb_stage* and *load_store_unit* is close to 1, which means they are significantly critical.

IV. IDENTIFYING CRITICAL FLIP-FLOPS WITH MACHINE LEARNING

Selective hardening [29] is a widely employed technique to enhance the reliability of integrated circuits (ICs). It is based on selectively protecting only the critical circuit components rather than the entire circuit to reduce the resources involved. However, the identification of the critical components in circuits is a challenging task. A commonly used approach is simulation-based fault injection. Unfortunately, this is time-consuming and impractical for complex circuits running complicated workloads. To overcome this challenge, we are exploring using machine learning (ML) as an alternative method to identify critical flip-flops in circuits.

Previous studies have applied ML to reliability analysis, including using ML algorithms to determine the relationship between fault injection outcomes and the characteristics of applications and platforms [30], predicting functional de-rating (FDR) [31], [32], and anticipating hardware defects at the transistor level [33]. Our work aims to utilize ML to predict critical flip-flops in circuits.

A. Methodology overview

We use Neural networks (NNs) as our primary ML model. We extract features such as the number of loads and the types of connected gates for each flip-flop from the circuit's netlist to build the dataset. The flip-flops in the dataset are annotated as critical or non-critical based on the results of simulation-based fault injection. NNs are trained on the dataset to search the pattern between the features and the annotations. However, NNs are limited in that they cannot comprehend the inter-component connections within circuits and can only learn from the characteristics of each individual component. The structural characteristics of circuits are essential for understanding the patterns of fault propagation. Graph Neural Networks (GNNs) are capable of learning from connection

information. They have been developed as machine learning models that can learn from graphs by utilizing the features of a node's neighbors in the graph and the node's own features during training. To utilize the structural features of circuits, we also train GNNs on our datasets to compare their prediction performance to NNs.

To use GNNs, our initial step is to transform the circuit into a graph. Each flip-flop in the circuit is represented as a node in the graph, and we employ the Breadth-First Search (BFS) algorithm to identify the shortest path between flip-flops and establish edges between them. The distance between flip-flops is determined by the number of combinational gates on the shortest path that connects them. To enhance the clarity of the graph model and eliminate unnecessary noise, we remove edges from the graph when the distance between two flip-flops exceeds a specified maximum distance. In the end, we represent the graph model using three tensors. The feature matrix contains the features of each flip-flop in the circuit or each node in the graph. The adjacency matrix refers to the interconnections between nodes. Lastly, the edge tensor defines the properties of the combinational gates on the edges that connect the flip-flops in the graph. GNN models learn from these three tensors to get the capacity to predict the critical flip-flops in a circuit based on them.

B. Preliminary results

Our work [34] compares the performance of NNs and GNNs on the dataset built on the RI5CY core [16]. The highest prediction accuracy we achieved on NN is 88.2%. The accuracy of GNNs is 91.1%, about 3% higher than NNs. Our current research efforts are focused on improving the accuracy of our predictions. By incorporating more advanced GNNs, augmenting our feature set to include additional features extracted from the Value Change Dump (VCD) waveforms, and optimizing the method to use edge features, we have achieved a prediction accuracy that exceeds 97%. Additionally, we are actively working to validate our proposed methodology on other open-source RISC-V cores, such as Ibex.

V. Protecting On-Chip Memory Banks

Memory is a fundamental component of any digital system and is responsible for storing both program and data [11]. In line with other state-saving elements such as flip-flops, errors in memory banks are persistent until corrected and can lead to incorrect behavior in the surrounding system. Furthermore, memories generally occupy large areas within an SoC at a high density, which leads them to be the main source of errors within the system in a hazardous environment.

While there are several ways to protect bits from corruption due to radiation-induced single-event upsets, static memory can greatly benefit from information redundancy. This generally relies on adding additional bits to each data word, allowing errors to be detected and corrected using error-correcting codes (ECC). For single-error correction and double-error detection (SECDED), a Hamming code with parity or a Hsiao code is often used. With ECC applied to the data word before storage,

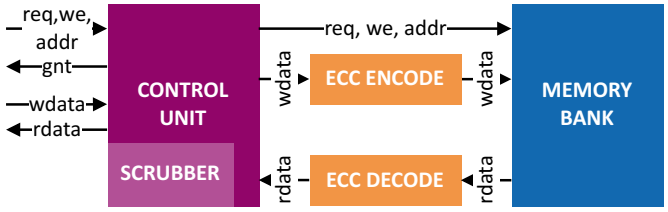

Fig. 5. Block Diagram of the ECC-protected memory bank.

any individual errors occurring within the memory can be detected and corrected upon reading back this memory word, ensuring correct operation.

The PULP system is based on the 32-bit RISC-V ISA, which, like many other ISAs, allows for memory access at a byte granularity. This can pose some difficulty when implementing ECC for memory protection: As the additional bits required scales logarithmically with the data word bits, larger granularities are usually used, such as protecting a full 32-bit word. Therefore, as the additional bits in the codeword depend on all bits of the data word, storing only individual bytes, e.g., using a byte strobe, requires recalculating the codeword.

To reduce the impact on the PULP memory system, we implement a read-modify-write approach directly at the memory bank shown in Figure 5, keeping up with the single-cycle access latency of the system's memory banks. Upon storing less than a word, the first cycle reads the stored protected word, directly responding to the requester with an error code derived from decoding the protected word. In the following cycle, the stored and to-be-stored parts of the word are strobed appropriately, and the protection codeword is calculated. Any request to this memory bank in this cycle is stalled so the memory can store the data.

To analyze the performance impact of this approach, we first investigate a representative workload such as the Coremark benchmark. Analyzing the simulation instruction trace of an ibex core executing the benchmark, we find that 90% of store operations store full words, with 9% storing halfwords and 1% storing individual bytes. To verify that the performance does not degrade within the PULP system, even for a workload using individual bytes more intensely than Coremark, we execute an 8-bit parallel matrix-matrix multiplication on the PULP cluster in RTL simulation, finding that the performance impact is less than 0.5%.

Finally, to ensure that latent errors within the memory system are corrected even if the data is not accessed, we implement a memory scrubber, sequentially reading the stored data within a single memory bank. If another actor in the system accesses the memory bank, the scrubber pauses its operation, resulting in no impact on the system's performance. If the scrubber detects a correctable error, it writes back a corrected word to memory.

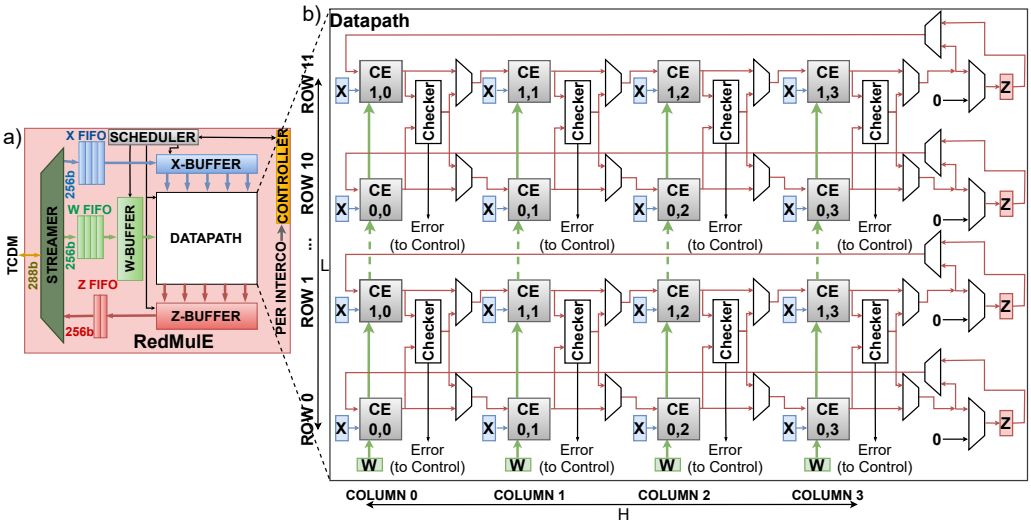

Fig. 6. a) RedMulE architecture; b) redundant datapath implementation.

VI. Redundant Hardware Accelerators

The criticality of the space environment requires space systems such as satellites and spacecraft to feature a tight link between onboard processing, communication, sensing, and actuation elements [35], providing enough performance to complete their tasks in a reasonable amount of time. The onboard computing capabilities of spacecraft play a crucial role in reducing the overhead given by raw data transmission, allowing the data processing to be directly performed onboard a spacecraft (for example, for image processing or fault detection, isolation and recovery through machine-learning-based techniques), sharing only valuable information on the communication line. Thus, the onboard processing systems must be highly reliable and perform well to process data while recovering from incurring faults [36].

General-purpose computing systems might not provide enough performance and efficiency to accelerate specific kernels needed for deep learning execution [37], [38], requiring coupling with application-specific hardware accelerating units. Even though machine learning has some capabilities of resisting faults, reliability in the acceleration of neural networks' execution is still a requirement that makes machine learning accelerators benefit from features that guarantee the reliability of their operation [39].

Figure 2 shows that a PULP cluster can be enhanced by introducing application-specific hardware accelerators that provide higher efficiency and performance during specific kernel execution over general-purpose cores. Since intense matrix multiplications are widespread in machine learning and deep learning, we integrate a **Red**uced-Precision Matrix **Mul**tiplication **E**ngine, RedMulE [40], to introduce up to 22× better performance and 5× better energy efficiency on the execution of 16-bits floating-point matrix multiplication kernels over the parallel execution on the general-purpose RISC-V cores. Furthermore, we enhanced RedMulE with fault-tolerant capabilities to tackle safety-critical satellite onboard computing.

RedMulE accelerates the execution of matrix multiplication of the kind $\mathbf{Z} = \mathbf{X} \times \mathbf{W}$, and Figure 6 shows its internal RedMulE architecture. It features internal buffers to store input/output matrices tiles, a specialized streaming memory interface that translates the TCDM protocol into streaming one to feed the accelerator, dedicated control logic, and a datapath. The datapath features a bi-dimensional array of Computing Elements (CE) organized in rows and columns. Each CE contains a Fused Multiply-Add module adapted from the industrially-verified Trans-Precision Floating-Point Unit [41] and has a private copy of elements from the $\mathbf{X}$-matrix. The elements from the $\mathbf{W}$-matrix are broadcasted among all CEs in a column so that all the CEs in a single column operate in lockstep. Then, each CE provides the intermediate result of its computation to the CE of the adjacent column in the same row. The private copies of the X-elements of each CE remain static for a given amount of time, while the W-elements are shifted cycle-by-cycle within RedMulE's datapath. The internal computations of each row of CEs are reused within the accelerator until an entire tile of the result matrix is completed, reducing the required memory accesses to store intermediate computations and maximizing internal reuse.

RedMulE can be configured so that its internal CEs are grouped in a Dual Modular Redundancy (DMR) fashion, allowing for the reliable execution of intense workloads if the application requires reliability. In the redundant configuration, each couple of adjacent CEs in a column receives the same X and W input elements, so it is supposed to produce the same output intermediate results. Each pair of CEs outputs is provided as input to a DMR checker that compares the results. If the results are consistent, they are propagated to the next column of CEs for further computation. On the other hand, if the checker detects a mismatch in the results provided by paired CEs, it raises an error signal propagating it to

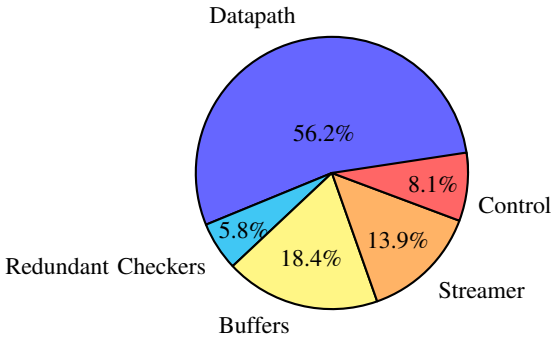

Fig. 7. Area breakdown of the fault-tolerant RedMulE accelerator.

the control logic of RedMulE, forcing the datapath to repeat the last computation. Figure 7 shows the area breakdown of RedMulE, highlighting that the additional hardware required for redundant grouping accounts for just 6% of the accelerator data occupation.

VII. TETRISC SoC

In this section, the TETRISC (TETra Core System based on RISC-V) SoC, an adaptive and resilient quad-core system developed on the PULPissimo platform [15], is introduced. TETRISC SoC is designed to dynamically adjust system reliability in harsh environments while monitoring multiple fault sources. The proposed system integrates a RISC-V-based SoC, onboard reliability monitoring, and real-time switching of operating modes to achieve a balance between system reliability, performance, and power consumption. Figure 8 presents the block diagram of the implemented SoC, where the green and purple blocks represent newly implemented or upgraded components, and the gray blocks originate from PULPissimo. The proposed design comprises three main subsystems: the quad-core subsystem, the on-chip reliability monitor subsystem, and the adaptive control subsystem. These subsystems are discussed in detail in the subsequent paragraphs.

In order to achieve the necessary level of redundancy, the TETRISC SoC architecture extends the Pulpissimo platform to a quad-core processor architecture, with the addition of three RI5CY cores connected to an extended interface. The memory interface has been redesigned to provide equal memory access across the address space for all four cores, and the separation of program and data memory is managed at the software level. As with the original Pulpissimo platform, interrupts are handled by an external event/interrupt unit, which has been extended for TETRISC to ensure that all four processors can receive interrupts independently.

The operational phase of a chip can be affected by various sources of failure, including radiation-induced effects, aging, and temperature. These factors can result in performance degradation, data corruption, and even catastrophic failures. To comprehensively monitor these potential threats, this design proposes the use of three on-chip monitors, each dedicated to monitoring one of the three sources of faults. These monitors

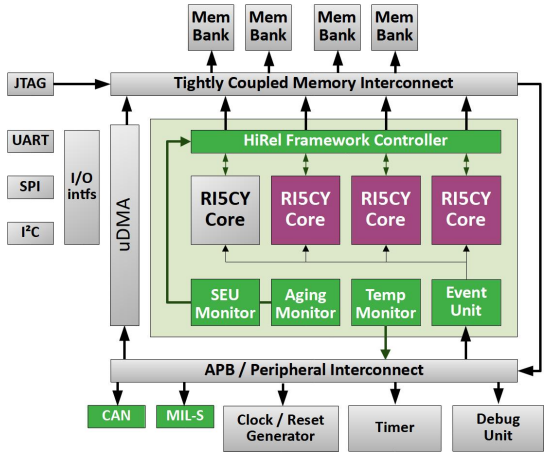

Fig. 8. The system architecture of the TETRISC SoC.

provide a comprehensive view of the current and future threats to the system's reliability.

- The Single-Event Upset (SEU) monitor is an efficient, SRAM-based radiation monitor that can detect and correct radiation-induced transient and permanent faults at a negligible cost [42]. This feature is crucial in space applications and other contexts where radiation levels may fluctuate significantly. Moreover, it can forecast future radiation conditions by analyzing historical error rate data and applying a pre-trained machine learning model. The SEU monitor uses an error detection and correction code, a scrubbing module, and a dedicated detection process to identify and classify faults in SRAM words. It is integrated into the memory interface controller and arbitration tree to control memory bank access, while a specific control mechanism manages access hazards.
- The aging monitor detects aging variations in target modules, preventing degradation imbalances [43]. Its simple, flexible design utilizes standard library cells and measures aging degradation through transistor input and output delays. The monitor generates an "aging code" to reflect module performance degradation over time, and each core has its own aging monitor integrated into the HiRel Framework Controller (HFC) platform. This allows the system to access aging information and take countermeasures if needed.
- Additionally, the temperature can have a significant impact on ICs, such as effects on system performance, leakage current, and material degradation. Therefore, it is essential to monitor the onboard temperature. The design includes an on-chip analog temperature monitor with an integrated Analog-to-Digital Converter (ADC) for real-time temperature data processing and analysis.

The HFC serves as the main adaptive control subsystem of the TETRISC SoC, allowing for hardware-based reconfigurability and fault tolerance. The HFC allows for diverse operational modes based on core-level N-Module Redundancy (NMR) and clock-gating techniques, providing efficient and

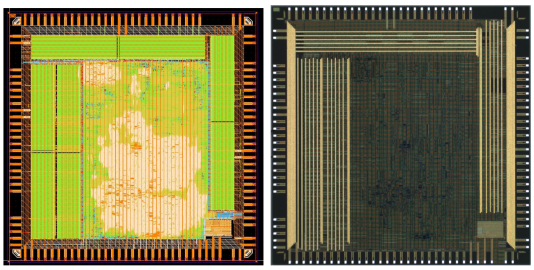

Fig. 9. Layout and die photos of the TETRISC SoC.

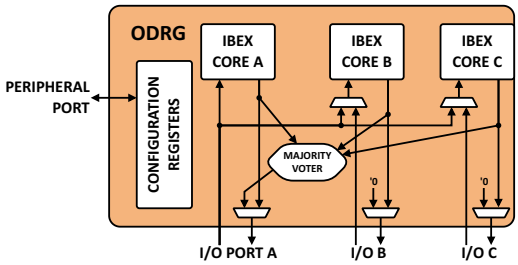

Fig. 10. Block diagram of ODRG, wrapping 3 cores with a configurable voter

reliable performance in various operating environments [44]. Under normal operating conditions, the four cores can independently execute different programs. However, in high-reliability scenarios triggered by either the monitor system or the user, the HFC can enable the parallel execution of the same program by two, three, or all four cores, thereby allowing for various levels of fault tolerance. The HFC's primary component is a binary matrix-based programmable NMR majority voter for multiprocessors. The voter is able to select processing cores to participate in the voting and can monitor the results. It can also transition from a 2MR to a 4MR system with any combination of active processors, creating an NMR on-demand system. As the primary control component of the SoC, the HFC integrates various control registers, providing users with the ability to manage operating modes and obtain real-time system status information. The design also includes custom-designed shadow registers for cores, enabling rapid switching and synchronization between different modes and core tasks. This approach ensures that task synchronization between different cores during NMR modes can be achieved in just two clock cycles.

To accurately assess the system's reliability under various forms of harsh conditions, a thorough investigation of the resiliency of such an adaptive system was conducted in [45]. Additionally, the TETRISC SoC was fabricated using IHP's 130 nm CMOS technology, as depicted in Figure 9, which displays the design layout and die photos. The chip employs a standard cell library for its four RISC-V cores, while a rad-hard cell library is utilized for the remaining design. Operating at a clock frequency of 30 MHz, the chip encompasses an area of 43.56 mm^2, with 39.17 % allocated to four shared 8192 x 40-bit L2 SRAM blocks that also function as sensing elements for radiation monitoring. A future version of the chip is expected to aim for an enhancement of the currently limited clock frequency.

VIII. On-Demand Redundancy Grouping (ODRG)

The multicore PULP cluster, discussed in Section 2, offers a unique advantage when designing a fault-tolerant system: multiple cores for parallel calculations are already available. While the tight coupling and integration with the memory system significantly improve the processing performance of parallelizable tasks, these cores can also be used for the same calculation, offering a redundancy copy to detect errors.

Extending the hardware, on-demand redundancy grouping (ODRG) [46] combines three cores in the PULP cluster into a triple-core lockstep (TCLS) group. In an 8-core cluster, this allows for two TCLS groups, and 2 remaining individual cores. In a TLCS group, the three cores receive identical inputs to ensure identical calculations. If an error occurs internally, all core outputs are voted with bitwise majority voters to correct errors in flight. This allows all processing to continue without requiring a reset to determine the correct result, such as for a Dual-Core Lockstep mode.

The internal state within the cores, such as the program counter, register file, or control and status registers, is stored internally in registers and is also vulnerable to faults. While these can lead to an error at the interface, the error is persistent within the registers and is not corrected with the majority voters at the interface. This can lead to a single error creating an erroneous state within the cores continuously, as the cores are no longer guaranteed to ensure triple modular redundancy. To mitigate this, a re-synchronization routine is implemented in software to correct any latent errors within the cores: When an error is detected, the TCLS control unit hardware raises an interrupt signal to the cores. This stops the cores and starts a software routine that saves the internal state into memory through the voters, correcting any errors that may persist. Upon completion, the cores can be reset, clearing any remaining errors within and ensuring an identical starting condition across the cores. The previously stored state can then be reloaded from the cores with additional software, ensuring correct operation of the interrupted task can continue. In the implemented PULP cluster, this re-synchronization requires around 700 cycles, significantly faster than restarting the entire computation if an error is detected.

While enforcing the locked TCLS mode allows for the correct execution of tasks, it does so at a significant penalty in area, requiring $> 3\times$ the area to replicate the processing cores and add majority voters. However, the PULP cluster is designed for many parallel processing cores, and not all tasks require the high level of reliability guaranteed with TCLS. Therefore, ODRG enables the reliability configuration to be switched, allowing the cores to operate in the described lock-step configuration and the standard individual configuration without active reliability hardware. This switching is achieved with multiplexers controlled by memory-mapped configuration registers, where the implemented setup is shown in figure 10.

Switching between a reliable mode and a high-performance mode is especially useful for systems where certain tasks require absolute correctness, such as the control operations of a satellite, and other tasks do not need this level of reliability, such as image processing algorithms on already noisy data. When switching to the individual configuration, the system can benefit from an up-to $2.9\times$ speedup over the locked mode for non-critical tasks, tested here with matrix multiplication. Overall, for a 6-core PULP cluster, ODRG adds less than 1% area overhead with negligible impact on timing.

IX. Conclusions

In this paper, we illustrated several techniques to increase the fault resilience capabilities of digital systems in order to prepare them for demanding applications such as space and avionics. All the proposed approaches rely on the openness of the RISC-V Instruction Set Architecture, particularly on the PULP platform, that uses industrially verified IP cores. We showed how it is possible to use UVM-based test benches and AI-based approaches to perform gate-level fault injection within RISC-V cores to identify the design's critical flip-flops and further perform selective hardening. Then we were able to show how redundant grouping can be applied to the internal processing elements of application-specific accelerators, guaranteeing the reliable execution of deep learning algorithms. Additionally, we presented architectural-level approaches based on modular redundancy applied to simple microcontrollers, or multi-core computing clusters, grouping multiple cores for lockstep execution, allowing for increased reliability of their operation. Our achieved results show that the PULP platform can be modified to a reliable and adaptable, highly performant system for air- or space-borne applications.

Future research needs to address certain aspects of our work that we were not able to investigate yet. This is the trade-off between the approaches on the one hand. Modular redundancy leads to comparatively low area and performance optimization, but selective hardening on flip-flop level introduces other low-level issues in regard to timing, etc., and requires detailed simulations. It might make sense to prefer one approach to the other depending on the application, or even to develop hybrid approaches to deliver more optimal results. On the other hand, the presented approaches strongly focus on hardware-based methods and only briefly touch upon the software side, but software-based methods for fault tolerance are well established. This needs to be further addressed in order to deliver an optimal interaction between the different methods.

Acknowledgment

The projects Scale4Edge and VE-HEP, on which this report is partially based, were funded by the German Federal Ministry of Education under grant numbers 16ME0134 and 16KIS1339K. The responsibility for the content of this publication lies with the authors.

The presented work was supported by Thales Alenia Space.

We acknowledge support by the EU H2020 Fractal project funded by ECSEL-JU grant agreement #877056.

We acknowledge support by TRISTAN, which has received funding from the Key Digital Technologies Joint Undertaking (KDT JU) under grant agreement nr. 101095947. The KDT JU receives support from the European Union's Horizon Europe's research and innovation programmes.

References

[1] B. Bailey, "Why RISC-V Is Succeeding," Feb. 2022. [Online]. Available: https://semiengineering.com/why-risc-v-is-succeeding/

[2] "Leading the RISC-V Revolution." [Online]. Available: https://www.sifive.com/

[3] "Western Digital Brings Memory Closer to Compute With New RISC-V Innovations and Strategic Partnerships." [Online]. Available: https://www.westerndigital.com/company/newsroom/press-releases/2019/2019-12-10-western-digital-brings-memory-closer-to-compute-with-new-risc-v-innovations

[4] "The Good Folk Series." [Online]. Available: https://www.minres.com/products/the-good-folk-series/

[5] A. Pullini, D. Rossi, I. Loi, G. Tagliavini, and L. Benini, "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 7, pp. 1970–1981, 2019.

[6] "PULP platform." [Online]. Available: https://pulp-platform.org/

[7] "Welcome to Chipyard's documentation (version "1.8.1")! — Chipyard 1.8.1 documentation." [Online]. Available: https://chipyard.readthedocs.io/en/stable/#

[8] J. Andersson, "Development of a NOEL-V RISC-V SoC Targeting Space Applications," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Jun. 2020, pp. 66–67, iSSN: 2325-6664.

[9] N. Flaherty, "First RISC-V processor starts operation in orbit," Aug. 2022. [Online]. Available: https://www.eenewseurope.com/en/first-risc-v-processor-starts-operation-in-orbit/

[10] N.-J. Wessman, F. Malatesta, S. Ribes, J. Andersson, A. García-Vilanova, M. Masmano, V. Nicolau, P. Gomez, J. L. Rhun, S. Alcaide, G. Cabo, F. Bas, P. Benedicte, F. Mazzocchetti, and J. Abella, "De-RISC: A Complete RISC-V Based Space-Grade Platform," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2022, pp. 802–807, iSSN: 1558-1101.

[11] S. Di Mascio, A. Menicucci, E. Gill, G. Furano, and C. Monteleone, "Open-source IP cores for space: A processor-level perspective on soft errors in the RISC-V era," *Computer Science Review*, vol. 39, p. 100349, Feb. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013720304494

[12] G. Furano, S. Di Mascio, A. Menicucci, and C. Monteleone, "A European Roadmap to Leverage RISC-V in Space Applications," in *2022 IEEE Aerospace Conference (AERO)*, Mar. 2022, pp. 1–7, iSSN: 1095-323X.

[13] "RISC-V in space." [Online]. Available: http://microelectronics.esa.int/riscv/rvws2022/index.php

[14] J. Morra, "RISC-V to Launch into Space," Sep. 2022. [Online]. Available: https://www.electronicdesign.com/technologies/embedded-revolution/article/21250236/electronic-design-riscv-to-launch-into-space

[15] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "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.

[16] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini, "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017.

[17] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, Nov 2019.

[18] O. Group, "OpenHW Group | OpenHW Group," 2023. [Online]. Available: https://www.openhwgroup.org/

[19] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini, "PULP: A parallel ultra low power platform for next generation IoT applications," in *2015 IEEE Hot Chips 27 Symposium (HCS)*. Cupertino, CA, USA: IEEE, Aug. 2015, pp. 1–39.

[20] J. Chen, I. Loi, E. Flamand, G. Tagliavini, L. Benini, and D. Rossi, "Scalable Hierarchical Instruction Cache for Ultralow-Power Processors Clusters," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–14, 2023, conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[21] F. Glaser, G. Tagliavini, D. Rossi, G. Haugou, Q. Huang, and L. Benini, "Energy-Efficient Hardware-Accelerated Synchronization for Shared-L1-Memory Multiprocessor Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 633–648, Mar. 2021, conference Name: IEEE Transactions on Parallel and Distributed Systems.

[22] F. Conti, P. D. Schiavone, and L. Benini, "XNOR Neural Engine: A Hardware Accelerator IP for 21.6-fJ/op Binary Neural Network Inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[23] A. Garofalo, Y. Tortorella, M. Perotti, L. Valente, A. Nadalini, L. Benini, D. Rossi, and F. Conti, "DARKSIDE: A Heterogeneous RISC-V Compute Cluster for Extreme-Edge On-Chip DNN Inference and Training," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 2, pp. 231–243, 2022.

[24] H. Ziade, R. A. Ayoubi, R. Velazco *et al.*, "A survey on fault injection techniques," *Int. Arab J. Inf. Technol.*, vol. 1, no. 2, pp. 171–186, 2004.

[25] E. Kaja, N. Gerlin, M. Vaddeboina, L. Rivas, S. Prebeck, Z. Han, K. Devarajegowda, and W. Ecker, "Towards Fault Simulation at Mixed Register-Transfer/Gate-Level Models," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2021, pp. 1–6, iSSN: 2765-933X.

[26] P. D. Schiavone, F. Conti, D. Rossi, M. Gautschi, A. Pullini, E. Flamand, and L. Benini, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 2017, pp. 1–8.

[27] E. Zurich and U. of Bologna, "Ibex UVM testbench," https://ibex-core.readthedocs.io/en/latest/03_reference/verification.html, Last accessed on 2023-03-20.

[28] "Incisive Functional Safety Simulator." [Online]. Available: https://www.cadence.com/ko_KR/home/tools/system-design-and-verification/simulation-and-testbench-verification/incisive-functional-safety-simulator.html

[29] I. Polian and J. P. Hayes, "Selective hardening: Toward cost-effective error tolerance," *IEEE Design & Test of Computers*, vol. 28, no. 3, pp. 54–63, 2010.

[30] F. R. da Rosa, R. Garibotti, L. Ost, and R. Reis, "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.

[31] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "On the estimation of complex circuits functional failure rate by machine learning techniques," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*. IEEE, 2019, pp. 35–41.

[32] T. Lange, A. Balakrishnan, M. Glorieux, D. Alexandrescu, and L. Sterpone, "Machine learning to tackle the challenges of transient and soft errors in complex circuits," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 7–14.

[33] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "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.

[34] L. Lu, J. Chen, M. Ulbricht, and M. Krstic, "A Methodology for Identifying Critical Sequential Circuits with Graph Convolutional Networks," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2022, pp. 20–25.

[35] A. T. Klesh, J. W. Cutler, and E. M. Atkins, "Cyber-Physical Challenges for Space Systems," in *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*, Apr. 2012, pp. 45–52.

[36] A. D. George and C. M. Wilson, "Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 458–470, 2018.

[37] Y. S. Shao, S. Xi, V. Srinivasan, G.-Y. Wei, and D. Brooks, "Toward cache-friendly hardware accelerators," in *HPCA Sensors and Cloud Architectures Workshop (SCAW)*, 2015, pp. 1–6.

[38] P. Dhilleswararao, S. Boppu, M. S. Manikandan, and L. R. Cenkeramaddi, "Efficient Hardware Architectures for Accelerating Deep Neural Networks: Survey," *IEEE Access*, vol. 10, pp. 131 788–131 828, 2022.

[39] K. Cho, I. Lee, H. Lim, and S. Kang, "Efficient Systolic-Array Redundancy Architecture for Offline/Online Repair," *Electronics*, vol. 9, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/2/338

[40] Y. Tortorella, L. Bertaccini, D. Rossi, L. Benini, and F. Conti, "RedMulE: A Compact FP16 Matrix-Multiplication Accelerator for Adaptive Deep Learning on RISC-V-Based Ultra-Low-Power SoCs," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1099–1102.

[41] S. Mach, F. Schuiki, F. Zaruba, and L. Benini, "FPnew: An Open-Source Multi-Format Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 4, pp. 774-78*, 2020.

[42] J. Chen, T. Lange, M. Andjelkovic, A. Simevski, L. Lu, and M. Krstic, "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.

[43] A. Simevski, R. Kraemer, and M. Krstic, "Low-complexity integrated circuit aging monitor," in *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2011, pp. 121–125.

[44] A. Simevski, O. Schrape, C. Benito, M. Krstic, and M. Andjelkovic, "PISA: Power-robust Multiprocessor Design for Space Applications," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2020, pp. 1–6.

[45] J. Chen, "A self-adaptive resilient method for implementing and managing the high-reliability processing system," doctoralthesis, Universität Potsdam, 2023.

[46] M. Rogenmoser, N. Wistoff, P. Vogel, F. Gürkaynak, and L. Benini, "On-Demand Redundancy Grouping: Selectable Soft-Error Tolerance for a Multicore Cluster," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Nicosia, Cyprus: IEEE, Jul. 2022, pp. 398–401, iSSN: 2159-3477.