IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 January 2024, accepted 12 February 2024, date of publication 16 February 2024, date of current version 1 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3366806

APPLIED RESEARCH

A RISC-V Fault-Tolerant Soft-Processor Based on Full/Partial Heterogeneous Dual-Core Protection

FRANCESCO VIGLI, MARCELLO BARBIROTTA, ABDALLAH CHEIKH, FRANCESCO MENICHELLI, ANTONIO MASTRANDREA, AND MAURO OLIVIERI, (Senior Member, IEEE)

Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00184 Rome, Italy

Corresponding author: Francesco Vigli (francesco.vigli@uniroma1.it)

ABSTRACT The low probability of single event upsets (SEU) within particular satellite orbits, makes Commercial-off-the-shelf (COTS) electronic components a viable solution for space system implementation, thanks to the introduction of design-level fault tolerance techniques at the expense of some performance/energy/area penalty. This paper illustrates the design and validation of a novel RISC-V dual-core architecture, based on a computing paradigm that we refer to as full/partial heterogeneous multi-core protection. The approach relies on a small, low-performance, fully fault-tolerant core (LP core) coupled with a high-performance partially fault-tolerant core (HP core). The computing paradigm assumes the failure-exposed HP core executes computation intensive routines for relatively short periods of time, making the occurrence of failures a statistically unlikely situation, while the fully fault-tolerant LP core operates in critical control tasks and manages the failure recovery of the high-performance core. The execution time percentage in the LP core varies from a minimum of 11.4% up to a maximum of 91.3% while in the HP core it is between 8.7% and 88.6%, depending on the application. In the proposed study, both the cores belong to the RISC-V compliant Klessydra core family. The dual-core architecture also includes a watchdog timer controlled by the LP core and monitoring the non-protected HP core, and a context switch FIFO that speeds up the code and data switch between the two cores during failure recovery. A dedicated run-time software environment coordinates the execution of tasks on the high-performance core in a resilient fashion. The dual-core processor has been validated through extensive RTL simulations running in an UVM-based fault-injection environment, which emulates SEUs at various rates. Experimental results illustrate the benefits and limits obtained by using a heterogeneous architecture with different levels of protection and performance. The failure probability assuming a SEU fault occurrence can be reduced by a factor between 10X and 30X with respect to the non-protected architecture, leading to an average failure rate of up to 4.00E-06 per second with respect to 1.80E-05 per second in the non-protected architecture.

INDEX TERMS Processor architecture, fault-tolerance, multi-core, RISC-V, interleaved multi-threading, heterogeneous computing, single event effects.

I. INTRODUCTION

In some specific operating environments, such as space and aerospace applications, electronic devices have to be specifically designed and/or fabricated to manage the occurrence of hardware faults and possibly temporary operation errors and maintain a high grade of reliability [1]. Yet, the demand for on-board computing power in satellites has increased over the years, requiring the operation of complex processors in the space environment and better battery lifetime [2], [3]. While components fabricated with radiation-hard technologies are available, they are affordable

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Nitin.

only to high-cost space satellites [4]. Commercial Off-The-Shelf (COTS) components can achieve a sufficient degree of resilience when coupled with ad-hoc design techniques [5]. Redundancy, at the logic and microarchitecture level or at the system level, represents the conceptual basis for obtaining computation resilience in a harsh environment by design. Triple-Modular-Redundancy (TMR) at the logic and microarchitecture level has been demonstrated as a primary technique to defeat faults, yet with expensive hardware resource usage [6]. Redundancy obtained at the microarchitecture level leveraging simultaneous Multi-Threading (SMT) and Multi-Core architectures have become the basis for many fault tolerant processors, not only for space applications [7], [8], [9], [10].

Our work addresses the possibility of exploiting microarchitecture heterogeneity in the context of fault tolerance (FT). The concept of heterogeneous dual cores has been successfully proven in embedded systems for power efficiency, with the *big.LITTLE* architecture being the most representative industrial example [11] and [12]. The basic idea of our work is to explore the use of a heterogeneous architecture based on a fully fault-tolerant, low-performance core coupled with a unprotected high-performance core. The rational behind the study is that using a high performance core can reduce the time window in which the core is susceptible to faults, while using a low-performance, yet fully protected core can guarantee a way for recovering from failures. Our goal is to analyze the FT performance obtained by partitioning the program code between the two computing resources, under the assumption of a supposed Single Event Upset (SEU) rate per bit.

Generally speaking, integrated circuit faults may result from fabrication defects, physical wear-out, and particle radiation. Normally, they can be divided into two main classes [13]:permanent faults, such as stuck-at-zero or stuck-at-one with a fixed logic value, and Transient faults, such as bit-flip, pulse, and delay. In this work, we analyze the fault effects originating from particle radiation, targeting Transient faults with Single Event Upsets. Permanent faults in our architecture would produce different impacts depending on the affected unit. In case of permanent faults in the sequential logic of the protected LP core, a failure would be exposed only if two redundant register bits are affected in the same TMR FF unit. In contrast, a permanent fault in the pipeline of the non-protected HP core would make the HP core unusable, forcing the software to operate only in LP mode. Detailed impact about permanent faults will be explored in future developements.

The target implementation technology underlying the proposed study is represented by low-cost FPGA devices. In the FPGA scenario, established commercial solutions are available for protecting the device from SEU-induced bit flips in the configuration RAM (via dedicated IPs [14]), and in the data contained in BRAMs (via automatically synthesized error correction codes [15]). Thus, architectural flip-flops in the synthesized design represent the open critical issue for FT. From this point of view, the focus of our work is the definition of an alternative solution to a resource-consuming full-TMR protection of all the flip-flop cells in the synthesized processor.

The contributions of our work can be summarized as follows:

- Defining and illustrating the structure and the mechanisms of the full/partial heterogeneous dual core architecture in detail;
- Quantitatively evaluating the achievable FT performance in terms of failure probability in case of a SEU, and average failure rate per program run assuming a given SEU rate per bit.

The proposed work primarily accounts for the resilience analysis of a practical design case, rather than a general theoretical model. The theoretical contribution of the work mainly resides in the mathematical analysis for the extraction of the failure probability from an ad-hoc fault injection simulation.

The rest of the article is organized as follows: in Sections II we discuss the related works and the foundation of the proposed approach; section III and IV describe the processor microarchitecture and related run-time software environment; section V describes the testing methodology; Section VI discusses the results, and Section VII reports our conclusions.

II. RELATED WORKS

Many fault-tolerant techniques were developed over the years based on multi-thread and multi-core architectures [16]. In Multi-Core (MC) architectures, some approaches were adapted from the Simultaneous Multi-Threading (SMT) field [1]. For example, the chip-level redundant threading (CRT) approach [17] implements the Simultaneous Redundant Thread (SRT) technique using two cores, having a shared branch prediction queue and a store/load value queue. The authors of [18] also implement a ''fingerprint'' to compress the architecture status reducing the comparison overhead. The most common idea in MC architectures [8], [9] is represented by lockstep cores, in which synchronized processor replicas execute the same instruction in the same clock cycle or with a timing offset of a few cycles. Authors in [19] introduced a configurable computing cluster for dual-core and triple-core lockstep execution, with Error-Correcting-Code (ECC) protected registers to restore the state of the computing cores, along with features to define explicit portions of code as safety-critical sections. The work in [20] describes the NMR-MPar method used to improve the reliability of applications running in multi-/many-core processors as a generic software approach using partitioning, spatial redundancy and redundancy in data. Authors in [7] propose an FPGA methodology combining two different cores with different performance, creating a heterogeneous Dual-Core Lockstep (DCLS) processor, featuring a 666 MHz ARM A9 hard-core and a 25 MHz LowRISC soft-core on Zynq-7000. Each core receives the

same input, and additional hardware compares the output. The system can stop, restart, restore from a checkpoint, or continue execution. The overall perforamnce is dictated by the slow core. Authors in [8] designed an offline scheduler synthesis framework for MC processors running real-time applications, which can drive the system to a safe execution state, even in the case of transient faults. In [21], the authors present CEVERO, a RISC-V System-on-Chip that implements FT on a PULP platform [22], incorporating two Ibex cores running in a lockstep mode and compared to each other via an FT hardware module. The work reported in [10] explores how a reliability-aware dynamic scheduler can improve soft error resilience in a non-fault-tolerant heterogeneous MC processor. The approach aims at showing the FT improvement with respect to a generic scheduling mechanism, evidencing an average 26% reduction in the failure probability.

The classic concept of time or space redundancy behind fault-tolerant architecture implies a compromise in terms of resources, which could be about performance or area [1]. For this reason, recent research has started looking at partial FT support, which generally provides less reliability but is susceptible to optimization by an offline analysis of the program code. In many applications, the workload of an embedded processor can be divided into heavy-load tasks and light-load tasks, from a computational point of view. In harsh environment applications, where fault resilience plays a crucial role, we can define an additional subdivision between critical tasks and non-critical ones. As a critical task, we consider every portion of code where an error could generate dramatic side effects for the entire system. As non-critical tasks, we consider all the parts that may be re-executed again in case of errors, with performance side effects only. Fig. 1 illustrates a graphical representation of this scheme.

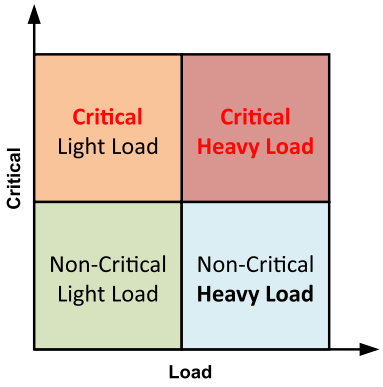


FIGURE 1. Software application partitioning principle scheme.

A previous work resembling this idea proposed a multi-threaded single core, in which the processor threads intrinsically have different fault tolerant capabilities [23]. Critical tasks are executed in the thread with full FT support while non-critical ones are executed on other threads.

III. HETEROGENEOUS DUAL-CORE SOFT PROCESSOR ARCHITECTURE

A. CONCEPTUAL BASIS

Our work aims at studying the behaviour of a non-uniformly protected dual-core in a harsh environment characterized by a given SEU rate. Assuming the software task partitioning defined in the previous section, we focused on executing critical code on a protected core, and reducing the execution time of the non-critical code (both light and heavy loads) on a fast but unprotected core, to reduce the overall system failure probability without compromising performance. The underlying conjectures are the following:

- The fault distribution is uniform in time - with respect to the execution duration of an application program on the processing cores - and in space - with respect to the physical area of the computing device;
- The probability of a fault occurrence in a processing core is proportional to its hardware resource occupation (area), yet the failure probability does not increase proportionally, since the physical site and time instant of a fault affects its actual impact on the application behavior;
- The probability of a fault occurrence in a processing core decreases by reducing the working time of the core, and so does the failure probability.

B. TOP LEVEL MICROARCHITECTURE VIEW

The Hydra Heterogeneous Computing Architecture (Hydra-HCA) is a dual-core soft-processor, in which the two cores differ in performance, hardware resource occupation, and computing features. Both the cores belong to the Klessydra family, which is fully compatible with the PULPino platform, an open-source embedded platform supporting 32-bit RISC-V cores [24]. The Hydra-HCA processor platform extends the PULPino architecture by featuring a small, low performance single-thread core (LP-Core), along with a second relatively high-performance and multi-threading core (HP-Core) (Fig.2). The LP-core is a Klessydra FT core, fully protected from SEUs via TMR at flip-flop level. The HP-Core is a Klessydra T03 core, which supports a wider RISC-V instruction set extension and provides higher performance, but has no hardware protection from SEUs at all. In applications that use a common set of instructions, the instruction per cycle (IPC) rate in the HP-core is at least 3 times higher than in the LP-core.

The memory sub-system of the Hydra-HCA platform (Fig. 2 - arrows indicate address paths, for both instructions and data) is composed of a shared program RAM and a shared data RAM. Each memory is a dual-port RAM in which port B has lower priority access than port A. Port B in each memory is directly accessed by the HP core. Port A is accessed through a multiplexer having one input connected to the LP-core and the other to an AXI4 interconnect, via a bridge, for programming/boot-loading purposes. The HP-core, connected using port B, always has lower priority

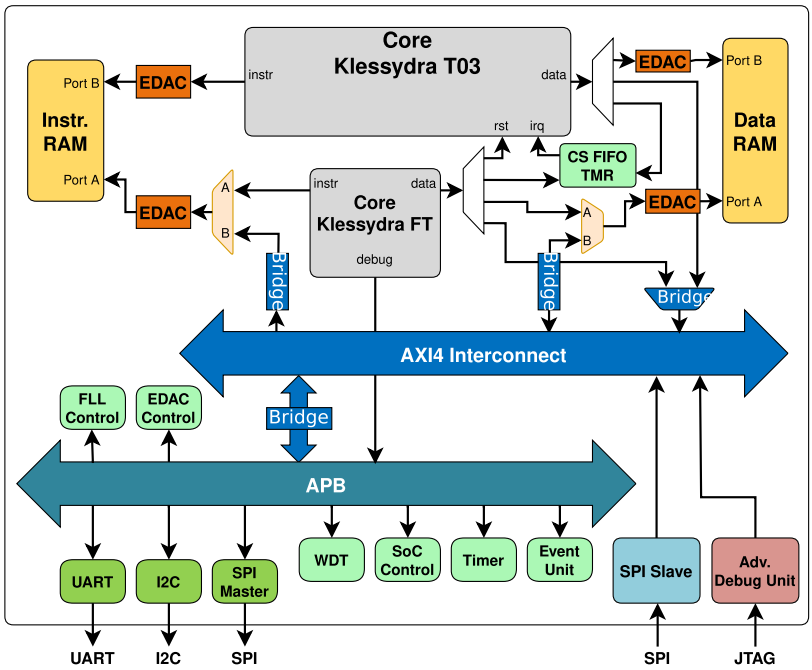


FIGURE 2. Hydra-HCA top level architecture view.

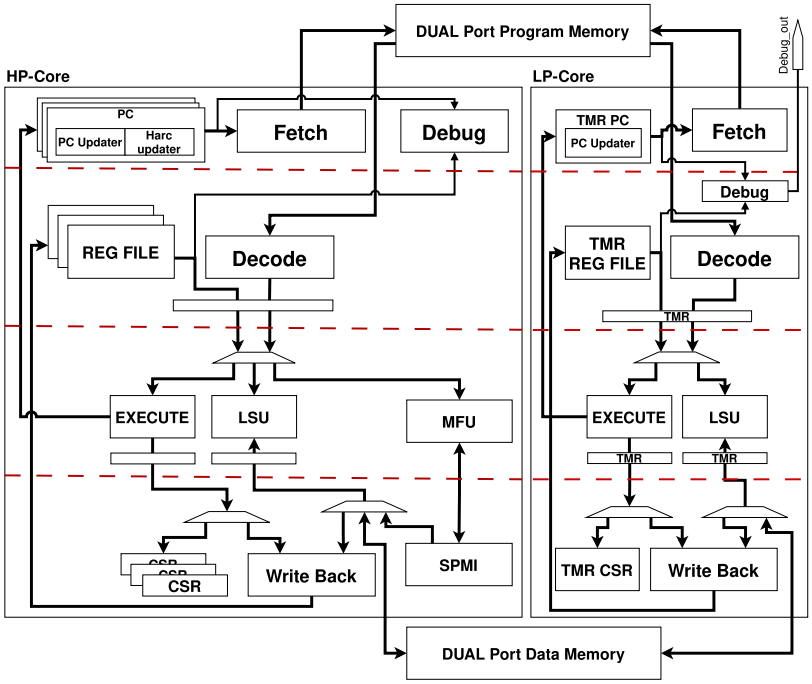


FIGURE 3. Detailed microarchitecture of the dual core section of Hydra-HCA.

access to the memories with respect to the fault-tolerant LP-core. Moreover, to increase the reliability of the architecture, the HP-core has no write permission to the program RAM.

The data ports of the two cores can access the AXI bus via a dual-port multiplexing AXI bridge (Fig. 2 - right), with the LP-core having higher priority access. This multiplexing bridge can also be configured by the LP-core to selectively disable the access of the unprotected HP-core to external peripherals.

The LP-core can control at hardware level the HP-core. In fact, while the LP-core is directly connected to the system reset logic, the HP-core hardware reset line is controlled by the LP-core through a memory-mapped register. During

power-on and after the system reset the HP-core is in reset state by default, so that the LP-core can wake up the HP-core only after the system configuration is complete. Furthermore, through another memory mapped register, the LP-core controls the fetch-enable line of the HP-core, which can put HP-core in sleep-state. Finally, the LP-core has the ability to send a hardware interrupt to the HP-core.

While in a DCLS microarchitecture both cores run the same program thread to spot an error by analysing the corresponding outputs, in Hydra-HCA the two cores run different parts of the software workload, balancing performance and reliability. In DCLS, every fault resulting in an output mismatch must be managed by reloading a safe execution checkpoint, with a computational overhead due to both the restoring operation and especially the checkpoint state periodic saving. In Hydra-HCA, the software execution is preemptively partitioned between the cores to exploit their different features and achieve the best compromise between performance and reliability, at the expense of a larger effort in the software design.

C. HP-CORE

Fig. 3 provides a detailed illustration of the dual core microarchitecture section. The HP-core implemented in Hydra-HCA is Klessydra T03 [24], [25]. Klessydra-T03 is a four-stage pipeline in-order processor that interleaves three or more hardware threads (Harts), with a shared memory scheme. Thread interleaving allows avoiding pipeline stalls thanks to the intrinsic absence of data or branch hazards [26]. It supports the RV32IMA instruction set [24], thus including the ''M'' (integer multiply/divide) and ''A'' (Atomic) extensions, the latter being used in thread synchronization [27].

D. LP-CORE

The LP-core in Hydra-HCA is the Klessydra F01, a single-thread fault-tolerant processor [23], [28] based on a four stages pipeline that supports the RV32I instruction set, designed to maximally privilege simplicity and resilience over performance. The pipeline implements stalls to avoid register access hazards as well as branch hazards. All the registers in the core are implemented as TMR components, to guarantee full protection from SEUs.

E. CONTEXT SWITCH TMR FIFO

The Context Switch TMR FIFO (CS-FIFO Fig. 4) is a fast bridge memory that allows fast data transfer from LP-core to HP-core. The purpose of the CS-FIFO is to send state information produced by a safe processing environment, composed of the register file and the Control&Status Registers (CSR) of the protected core, to the unprotected core in order to create a safe starting point for execution resume. The CS-FIFO is based on a full-TMR First-In-First-Out (FIFO) buffer and is directly connected to the data memory of each core through two dedicated ports. It works with three different addresses:

- Configuration address: writable only from LP-core and readable by both;
- Data in/out address: enabling the FIFO data input and data output port respectively for the LP and HP core;
- Command address: writable only from the LP-core and used to send the transfer-abort command.

The FIFO is managed by a controller unit that contains the configuration register and executes the write requests depending on the availability of empty elements in the FIFO. A second unit manages the read requests and it can be configured to provide the same element from the FIFO to multiple sequential requests, which is used when sending the same context to all threads in the HP-core. The last component is the interrupt controller that manages the interrupt requests to the HP-core through dedicated lines. It can be configured to send multiple interrupts, one for each Hart in the HP-core. The FIFO operates on the basis of a configuration word that sets up and enables the component for the data transfer. The configuration word contains the Hart destination ID, that enables the corresponding interrupt line, the amount of data to be transferred and the transmission modes. The component can work in two different modes:

- Single Hart: the configuration word contains the destination Hart number;
- Multi-Hart: the configuration word contains a bit that, when set, will cause the first valid data to trigger an interrupt to all the Harts.

In other words, the component can be used to send different states to the Harts in the HP core (single Hart mode), or can be used to initialize all the Harts in the HP coren with the state. The latter mode may be exploited for comparing the results produced by the same code executed in the three harts, so to implement a software TMR in the HP core, if needed.

F. CORE-CONTROLLED WATCHDOG TIMER

The Core-Controlled Watchdog Timer (CC-WDT Fig. 5) is a Hydra-HCA dedicated peripheral derived from the Thread-Controlled WDT (TC-WDT) first described in [23]. The component is connected to a classic AMBA APB interconnect, accessible to both cores, and to a dedicated port accessed the LP-core only, to avoid that failures in the HP core prevent the protected LP-core from controlling the CC-WDT. The CC-WDT can only be reset by the LP-Core, while the HP-core can set dedicated HP-flags inside the CC-WDT status register. The LP-core periodically checks the CC-WDT flags and it resets the CC-WDT count only if the flags are correctly set. If the Harts in the HP-core delay or omit the flag setting request, the LP-core may send a hardware reset to the HP-core and restart the specific Hart that omitted setting the flags, and then reset the CC-WDT count. In extreme cases, the CC-WDT will time out, which will reset the entire system.

IV. CORE CONTROL RUN-TIME SOFTWARE LIBRARY

The resilience of the proposed architecture is enhanced by the availability of a run-time software library, specifically

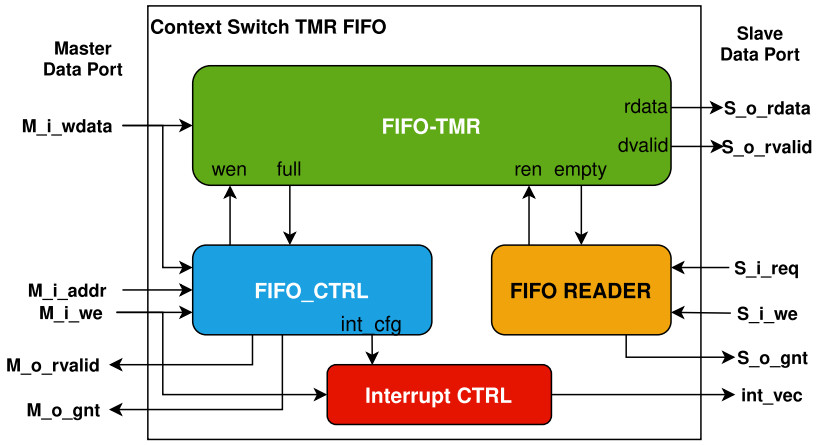


FIGURE 4. Details of the context Switch TMR FIFO.

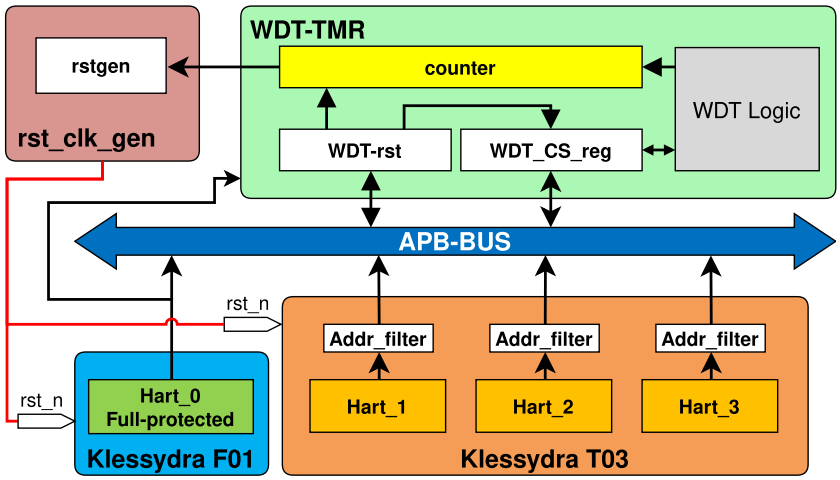


FIGURE 5. Details of the Core-Controlled Watchdog Timer connections.

designed to control the dedicated Hydra-HCA hardware that manages failure recovery and inter-core context switch. The run-time library includes a set of routines that are exposed as C functions to the programmer:

- hp_core_starting(): this function controls the reset input pin of the HP-core, in order to wake-up the HP-core from the reset state or to send it into the reset state. The wake-up time depends on the environment boot code. When in reset state, the HP-core is intrinsically insensitive to SEU and SET events;
- hp_core_fetch_toggle(): this function controls the fetch input pin of the HP-core. It is used to wake-up the HP-core from sleep-state. The wake-up time from sleep-state is 1 clock cycle. When in sleep state, the HP-core is not affected by SEU and SET events;
- hp_core_sleep_mode(): this function is the companion routine of the hp_core_fetch_toggle() and is used to send the HP-core in sleep-state.
- fifo_send_task(): this function sends the LP-core context to the CS-FIFO. After the data transfer, the CS-FIFO sends an interrupt to one or more Harts of the HP-core to transfer the context to them, depending on the CS-FIFO configuration.

The above functions are essential to implement the program partitioning between the LP-core and the HP-core. By means of inter-Hart interrupts, shared data memory, atomic semaphore operations and the CS-FIFO, the LP-core can set up and launch the execution of a safe task on the HP core. This implies transferring the LP-core context (data registers and CSRs) into the context of a Hart in the HP-core. In this way, it is possible to create a safe-start environment from which the HP-core can run the computational intensive code portion. As previously mentioned, the HP-core should execute tasks that are not critical for the system.

The run-time library functions can also implement several FT mechanisms. The choice of the most appropriate one depends on the way the software has to be partitioned in the architecture, based on the knowledge of which part of the application is considered critical for the system. Examples of methods to use the runtime library functions for FT are:

- HP-core reset-and-run: The HP-core reset signal is controlled by the LP-Core using the hp_core_starting()

function. If, for any reason, it is not possible to pass the context from the LP-core to the HP-core, the former can reset the second after writing the HP-core program routine in the instruction RAM immediately following the HP-core boot routine. After rebooting, the HP-core will run the required routine.

- HP-core fault-tolerant processing: It is possible to take advantage of the interleaved multi-threading architecture of T03x by using the three threads to implement a temporal triple redundancy, similar to [5]. In this processing flow, the LP-core loses the role of main and shall only send the context and PC, without the necessity of a complex preparation phase. The HP-core executes the same code portion in all its Harts, at the end the three intermediate results are stored in different Hart-dedicated memory regions and the LP-core takes the role of result voter.
- HP-core fault-detecting processing: Similarly to the previous case, the LP-core may launch two identical threads on the HP core to implement a temporal double redundancy. At the end of the execution the two intermediate results are stored in two different Hart-dedicated memory locations, the LP-core compares the results in order to reset/restart the HP-core in case of mismatch.
- CC-WDT fault-detection: the LP-core runs a routine on a Hart of the HP-core and in case the CC-WDT detects an anomalous behaviour, the LP-core resets the HP-core, or even the entire system.

V. TEST AND VALIDATION

We analyzed the proposed heterogeneous dual-core concept through Register Transfer level (RTL) fault injection simulations. The approach, together with gate-level fault injection simulation, is the first-choice technique for fault-resilience characterization, as it allows analyzing the behavior of the system when subject to faults with clock-accurate signal-accurate results [29], [30], [31]. The detection of failures is consistent with the standard RTL simulation flow used for digital design verification. An experimental analysis instead, based on physical error injection (proton or neutron radiations),is planned as a final activity complementary to fault simulation, but it cannot produce the detailed information needed for design characterization, as it only provides a global view of the resilience of the countermeasured chip.

We evaluated the FT effectiveness of the proposed approach by comparing the average failure rate (FR) of a program run on the HCA platform and the FR of the same program run on a platform equipped with the sole HP-core (i.e. Klessydra T03 core). The aim of the analysis is to evaluate the average failure rate reduction achieved by the heterogeneous dual-core approach per se, thanks to workload distribution between the cores, even when no recovery mechanism is present. For this reason, the failure recovery methods were not activated during the tests. The assessment of the FR was obtained from a simulation-based fault injection (FI) campaign implemented in a Universal Verification Methodology (UVM) environment. For the purpose of our analysis, we simulated the occurrence of SEUs targeting any of the FF register cells of the cores, i.e. the occurrence of a flip on a HDL bit signal that is going to be synthesized as a flip-flop cell in the FPGA implementation (the analysis of faults outside the cores, as memories and peripherals, being out of the scope of this work). Specifically, the UVM test environment reads a previously provided list of the architecture signals, iteratively injecting bit-flip faults on the signals using the *uvm_hdl_deposit* function. In the described setup, the LP-core operation in the HCA architecture is never affected by SEU-induced failures, thanks to the full TMR protection. The FI simulation environment allowed us to estimate the probability that a SEU occurring on the cores causes a program failure, namely $P_{FAIL} = \Pr\{\text{failure} \mid \text{SEU on a FF of the processor}\}$.

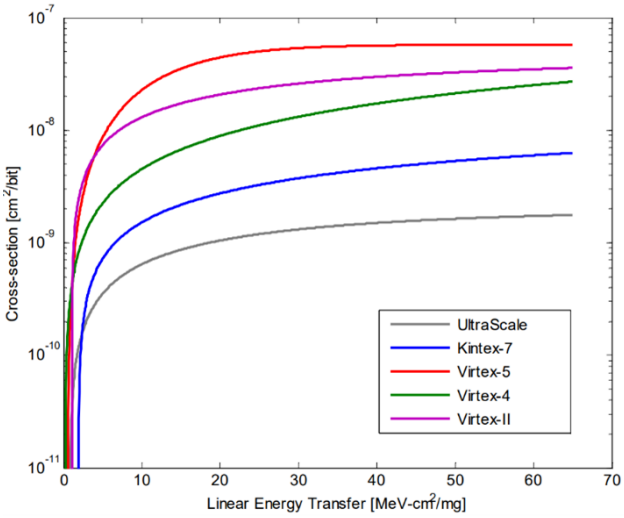


FIGURE 6. FPGA scaling Trends. Adapted from [33].

Assuming an average SEU rate per bit r due to the operating environment, by multiplying this value by the total flip-flop bit count NFF and by the failure probability obtained from the UVM FI tests, we get an estimation of the FR as

$$FR = r \cdot NFF \cdot P_{FAIL} \tag{1}$$

By defining D as the program execution time, the average event count per bit during the program run is $r \cdot D$. The average number of failures per program run is expressed by

$$FPP = r \cdot D \cdot NFF \cdot P_{FAIL} = FR \cdot D \tag{2}$$

A. REFERENCE SEU RATE SCENARIO

The average SEU rate is a function of the average of the linear energy transfer (LET) of heavy ions, measured in $MeV \cdot cm^2/mg$, the flux of that particles per cm^2 at the target orbits, and the sensitivity of the device or cross section σ [32].

The value of σ for FPGAs is widely studied and can be found directly on vendor websites [33], [34]. An example is reported in Fig. 6. According to the data reported in [32], the rate of 1 event per year per bit, which is equivalent to

$r = 3.2 \cdot 10^{-8}$ events per second/bit, represents an upper bound to the estimated SEU rate for Virtex II FPGA devices. Without loss of generality we assumed such value as the reference point for evaluating the *FR* and the *FPP* values of the platforms under analysis.

B. TIME FRAME SPAN METHODOLOGY

The approach adopted to compute the failure probability P_{FAIL} was defined in [35]. The technique overrides the limits of a generic statistical Monte-Carlo analysis based on random fault distribution, by subdividing the total program execution time in uniform time intervals (time frames), injecting a large amount of deterministic faults on a target bit of the micro-architecture during a specific time frame, and checking the correctness of the program results. The procedure is repeated for each target bit to be analysed in the processor core. In the SEU analysis, the target bits are all the output pins of flip-flop cells.

Referring to the number of execution time frames as m, for a given j_{th} bit of the microarchitecture there will be $m_F(j)$ time frames in which at least one fault on the jth bit causes a system failure. The characterization of such critical time frames for each j_{th} bit, can be used to calculate the global failure probability P_{FAIL}. Since the spatial location of SEUs and the time instant of SEUs are independent and uniformly distributed random variables, we can elaborate as follows:

$$\begin{aligned} P_{FAIL} &= \Pr\{\text{ failure } \mid \text{SEU on a FF of the processor }\} \\ &= \sum_{j=1}^{NFF} P_{SEU}(j) \cdot P_e(j) \end{aligned} \tag{3}$$

where:

NFF = number of Flip-Flop cells in the architecture,

$P_{SEU}(j) = \Pr\{$SEU occurring on j^{th} FF $\mid$ SEU on a FF of the processor $\} = 1/NFF$,

and

$P_e(j) = \Pr\left\{\text{ failure } \mid \text{ SEU on } j^{\text{th}} \text{ FF}\right\} \leq \Pr\{$ SEU occurring in critical time frame for j^{th} FF$\} = m_F(j)/m$.

The value $P_e(j)$ gives an upper bound rating of the sensitivity of the system to a SEU occurring on bit j of the microarchitecture [35], related to a specific application program execution. In particular, only the bits for which $m_F(j) \neq 0$ and therefore $P_e(j) \neq 0$ are critical for the program execution (Architecturally Correct Execution - ACE - bits [36]) and are relevant for the fault resilience analysis.

C. BENCHMARK PROGRAM SETUP

In the HCA architecture, the software application program must be designed ad-hoc or properly modified to assign parts of the code to the LP-core and parts to the HP-core. This is accomplished by using the run-time library functions that control the context switch and the operating state of the HP-core. At present, no automated procedures have been implemented for this step.

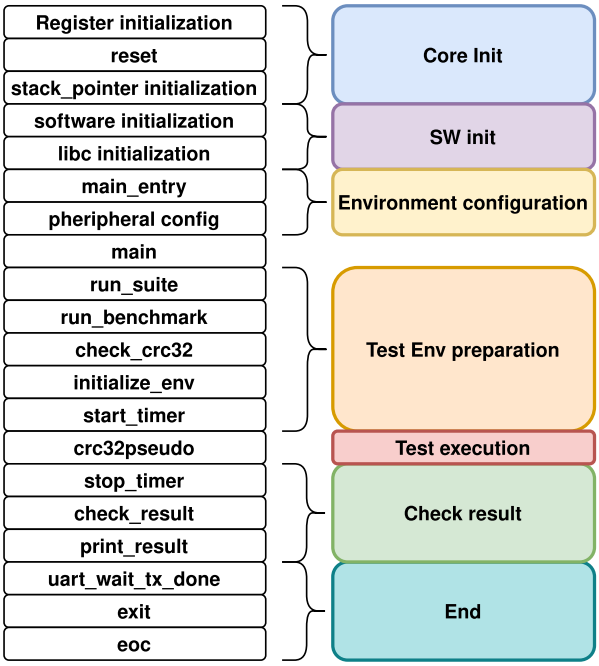


FIGURE 7. CRC32 software section diagram.

The code partitioning criteria are based on the analysis of the level of criticality under two points of view: the system point of view and the core point of view. The first is based on the severity level of the propagation of a failure to the system. This part of the analysis is up to the software designer, who knows which parts of the application program execution have no margins for errors. The core point of view only depends on the processor architecture and the software running on it, and can leverage the a-priori characterization of the resilience of each core through the FI UVM test environment [35]. The results of the FI analysis give an assessment of the failure probability of specific program portions and allow identifying the parts that are more susceptible to a SEU-induced failure, suggesting avoiding the execution of those parts in the unprotected core.

As an example of the implemented approach, Fig. 7 reports the code partitioning for a CRC32 test program. The software structure has been divided in macro-sections that perform different tasks. Fig. 8 reports the code partitioning and the consequent execution handovers between the two cores. The different parts are compiled toward the respective target, i.e the supported ISA extension of each core. All the environment configuration, peripheral control, and communication are managed by the LP-core. The CRC32 test environment preparation and result check are also performed by the LP-core. The HP-core, however, manages the majority of the CRC32 computation, taking advantage from multi-thread execution and the extended ISA.

VI. EXPERIMENTAL RESULTS

The benchmarks we used to perform FI tests are the following computation kernels, representative of embedded application workload:

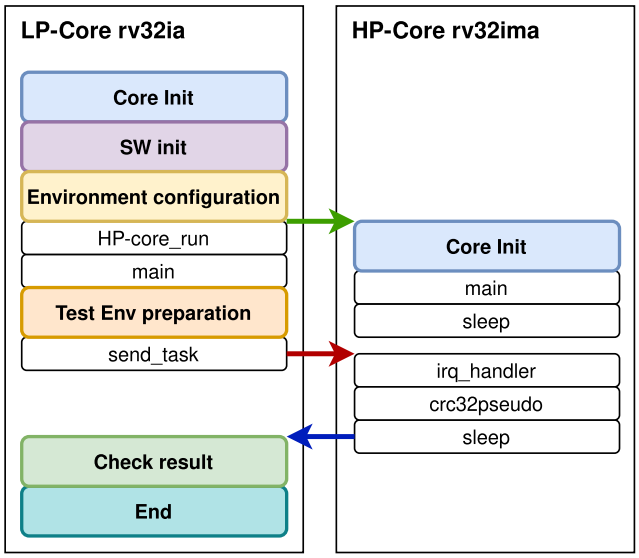


FIGURE 8. HCA partitioning of the CRC32 software sections.

- 2D convolution (Conv2D), 10 × 10 matrix size, 5 × 5 coefficient kernel size, 5 repetitions;
- CRC32 standard algorithm, 256 element table size, 3 repetitions;
- testALU, a standard test program for integer arithmetic instructions from the RISC-V toolchain, which executes 30 iterations of addition, subtraction, logical bitwise operations, shift operations.

The evaluation setup is summarized in Table 1. We compared the 6 different architectures listed below:

- HCA1: a single-thread platform featuring a fault-tolerant F01x core and a non-protected T01 core, supporting the baseline RV32I instruction set;
- HCA2: a platform featuring an F01x core and a T02 core, the latter supporting the execution of two interleaved threads and the RV32IA instruction set (atomic operation extension);
- HCA3: a platform featuring an F01x core and a T03 core, the latter supporting the execution of three interleaved threads and using the RV32IA instruction set;
- HCA3+: a platform featuring the same architecture as HCA3, yet supporting the RV32IMA instruction set (multiplication and division extension, atomic operation extension);
- T03x: a platform featuring a single non-protected T03 core, supporting the execution of three interleaved threads and using the RV32IMA instruction set;
- F01x: a single-thread platform featuring a single fault-tolerant F01 core supporting the baseline RV32I instruction set;
- F03x+: a platform featuring a single fault-tolerant F03 core supporting the execution of three interleaved threads and using the RV32IMA instruction set.

The target clock frequency for the FPGA synthesis was set to 100 MHz for all the architectures, even if the different cores may be synthesized at higher frequencies. This choice is dictated by the fact that the critical path of the platforms is not inside the processing cores, and it is compatible with a frequency slightly above 100 MHz for all the analyzed platforms. Moreover, there is a consensus in limiting the clock frequency of space-qualified processors in order to exclude the possibility of Single Event Transient (SET) propagation from combinational cells to sequential elements [37], [38].

All the fault simulation tests were carried out with $m = 10$ frames according to the chosen methodology [35]. Table 2 reports the execution data along with the total failure probability P_{FAIL} assuming a fault hits one of the computing cores, obtained for the different architectures on each benchmark according to (3).

As the study assumes single fault events, the fully-protected TMR architectures indicated as F01x and F03x obviously result to have zero failure probability. Yet, F01x pays a significant performance overhead and F03x pays a significant hardware overhead.

For the dual-core HCA architectures, in general, a longer execution on the unprotected HP-core is more likely to be hit by a fault which may result in a program failure. Yet, the higher speed of the HP-core balances this effect by reducing the execution time of the HP-core program sections. The total duration D as well as the percentage of execution time spent on HP-core and LP-core are impacted by the multi-thread support in the HP-core and the different ISA hardware support.

Overall, the HCA approach reduces the failure probability in case of SEU by a factor between 10X and 30X with respect to the non-protected T03x architecture. In addition to the reduced failure probability, one should also consider that in the T03x core no failure detection and recovery is available, while the HCA architectures provide features for failure detection and recovering, which have not been used in the experiments.

In comparison with the full-TMR-protected F03 architecture, the HCA architectures offer the advantage of less hardware overhead. Looking at the comparison among the different HCA versions, the results show that - although scaling to higher performance implies a larger hardware area exposed to faults - when the architecture grows in number of supported threads in the HP core and in the instruction set, its failure probability decreases. This behaviour depends on the fact that an individual flip-flop cell is sensitive to a SEU for the application program only for a limited time and operating range, and having a faster HP-core decreases the time during which the core is exposed to possible failures.

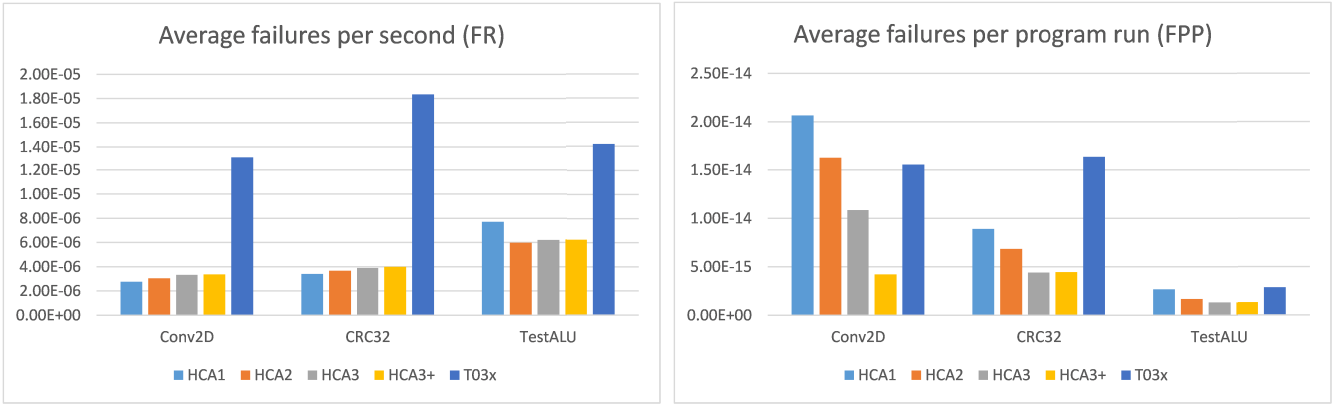
The resulting average failure rate per second FR for the assumed SEU rate per bit is reported in Fig. 9a, while the average failures per program run FPP - which is impacted by the total program execution duration D - are shown in Fig. 9b. On the one hand, the diagrams give evidence of the fact that a relatively slow HP core compromises the resiliency advantage of the HCA architecture; on the other hand, a fast HP core without a companion LP core equipped

TABLE 1. FPGA Synthesis Results. Hardware resource usage refers to the processing cores only.

configuration name	HCA 1	HCA 2	HCA 3	HCA3+	T03x	F01x	F03x+
ISA	RV32I	RV32IA	RV32IA	RV32IMA	RV32IMA	RV32I	RV32IMA
Architecture configuration	dual core F01x + T01	dual core F01x + T02	dual core F01x + T03	dual core F01x + T03	single core T03	single core F01	single core F03
Max threads	1	2	3	3	3	1	3
FFs	7960	9230	10499	10732	4489	6192	14863
LUTs	10666	12737	14803	15500	5527	8290	21117

TABLE 2. Execution performance results along with application failure probability assuming a SEU hits the cores.

Benchmark	Result	HCA 1	HCA 2	HCA 3	HCA3+	T03x	F01x	F03x+
Conv2D	Exec. Time D[ms]	22.458	10.676	3.251	1.645	1.181	6.825	7.241
	% Exec. on HP core	88.66%	84.1%	73.9%	32.3%	-	-	
	% Exec. on LP core	11.34%	15.9%	26.1%	67.6%	-	-	
	P_{FAIL}	0.0109	0.0104	0.01	0.0099	0.0922	0.0000	0.000
CRC32	Exec. Time D [ms]	7.814	3.735	1.126	1.117	0.894	2.618	2.84
	% Exec. on HP core	86.23%	80.78%	68.48%	49.84%	-	-	
	% Exec. on LP core	13.77%	19.22%	31.52%	50.16%	-	-	
	P_{FAIL}	0.0135	0.0125	0.0117	0.0117	0.1285	0.0000	0.000
TestALU	Exec. Time D[ms]	3033.4	1996.1	998.0	998.0	993.1	1011.0	1351.2
	% Exec. on HP core	21.47%	15.25%	8.74%	8.74%	-	-	
	% Exec. on LP core	78.53%	84.75%	91.26%	91.26%	-	-	
	P_{FAIL}	0.0036	0.0038	0.004	0.0039	0.0994	0.0000	0.000



a) b)

FIGURE 9. a) Average failure rate for the assumed rate of 1 SEU per bit/year; b) Average failure per program run for the assumed rate of 1 SEU per bit/year.

with full-TMR protection - which is the case of the pure T03x architecture - results in worsening the FT performance. This behavior suggests that an optimal balance between the LP and HP core performance exists, tailored to a specific application domain.

VII. CONCLUSION

We demonstrated the reduction in failure probability achieved by a HCA processor, composed of a fully protected low performance core and a non-protected high performance core, with respect to the utilization of the sole high-performance core. The approach allows exploring a balance among hardware resource optimization, performance and resilience. The method presently requires porting the software application to the HCA computing platform, by partitioning the execution between the two cores, leveraging a small and versatile runtime library that was described in the article. The tests show a significant decrease of the failure rate, without using any of the possible failure recovery techniques available in the architecture. The overhead in terms of hardware resources is much less than the direct application of TMR techniques at flip-flop level. A future development may be the investigation of the resilience achievable through different computation partitioning criteria, introducing also vector acceleration in the HP core.

REFERENCES

[1] M. Barbirotta, A. Cheikh, A. Mastrandrea, F. Menichelli, and M. Olivieri, "Design and evaluation of buffered triple modular redundancy in interleaved-multi-threading processors," *IEEE Access*, vol. 10, pp. 126074–126088, 2022.

[2] J. Zhang, C. Huang, M.-Y. Chow, X. Li, J. Tian, H. Luo, and S. Yin, "A data-model interactive remaining useful life prediction approach of lithium-ion batteries based on PF-BiGRU-TSAM," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 1144–1154, Feb. 2024.

[3] J. Zhang, Y. Jiang, X. Li, H. Luo, S. Yin, and O. Kaynak, "Remaining useful life prediction of lithium-ion battery with adaptive noise estimation and capacity regeneration detection," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 2, pp. 632–643, Apr. 2023.

[4] R. Ginosar, "Survey of processors for space," in *DASIA 2012—DAta Systems in Aerospace* (ESA Special Publication), vol. 701, Aug. 2012, p. 10.
[5] M. Barbirotta, A. Cheikh, A. Mastrandrea, F. Menichelli, F. Vigli, and M. Olivieri, "A fault tolerant soft-core obtained from an interleaved-multi-threading RISC-V microprocessor design," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2021, pp. 1–4.
[6] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Develop.*, vol. 6, no. 2, pp. 200–209, Apr. 1962.
[7] C. Rodrigues, I. Marques, S. Pinto, T. Gomes, and A. Tavares, "Towards a heterogeneous fault-tolerance architecture based on arm and RISC-V processors," in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc.*, vol. 1, 2019, pp. 3112–3117.
[8] L. Osinski, T. Langer, and J. Mottok, "A survey of fault tolerance approaches on different architecture levels," in *Proc. 30th Int. Conf. Archit. Comput. Syst.*, 2017, pp. 1–9.
[9] I. Oz and S. Arslan, "A survey on multithreading alternatives for soft error fault tolerance," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–38, Mar. 2019.
[10] A. Naithani, S. Eyerman, and L. Eeckhout, "Optimizing soft error reliability through scheduling on heterogeneous multicore processors," *IEEE Trans. Comput.*, vol. 67, no. 6, pp. 830–846, Jun. 2018.
[11] A. Butko, F. Bruguier, A. Gamatié, G. Sassatelli, D. Novo, L. Torres, and M. Robert, "Full-system simulation of big.LITTLE multicore architecture for performance and energy exploration," in *Proc. IEEE 10th Int. Symp. Embedded Multicore/Many-Core Syst.-Chip (MCSOC)*, Sep. 2016, pp. 201–208.
[12] H. D. Cho, K. Chung, and T. Kim, "Benefits of the big.LITTLE architecture," Samsung, White Paper, Feb. 2012. [Online]. Available: http://www.arm.com/files/downloads/Benefits_of_the_big.LITTLE_architecture.pdf
[13] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Self-healing hardware systems: A review," *Microelectron. J.*, vol. 93, Nov. 2019, Art. no. 104620. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026269219302782
[14] *Soft Error Mitigation Controller v4. 1 (PG036)*, Xilinx, San Jose, CA, USA, 2015.
[15] *Block Memory Generator*, Xilinx, San Jose, CA, USA, 2021.
[16] M. Barbirotta, A. Cheikh, A. Mastrandrea, F. Menichelli, M. Ottavi, and M. Olivieri, "Evaluation of dynamic triple modular redundancy in an interleaved-multi-threading RISC-V core," *J. Low Power Electron. Appl.*, vol. 13, no. 1, p. 2, Dec. 2022.
[17] M. Gomaa, C. Scarbrough, T. N. Vijaykumar, and I. Pomeranz, "Transient-fault recovery for chip multiprocessors," in *Proc. 30th Annu. Int. Symp. Comput. Archit.*, Jun. 2003, pp. 98–109.
[18] J. C. Smolens, B. T. Gold, B. Falsafi, and J. C. Hoe, "Reunion: Complexity-effective multicore redundancy," in *2006 39th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2006, pp. 223–234.
[19] M. Rogenmoser, Y. Tortorella, D. Rossi, F. Conti, and L. Benini, "Hybrid modular redundancy: Exploring modular redundancy approaches in RISC-V multi-core computing clusters for reliable processing in space," 2023, *arXiv:2303.08706*.
[20] V. Vargas, P. Ramos, J.-F. Méhaut, and R. Velazco, "NMR-MPar: A fault-tolerance approach for multi-core and many-core processors," *Appl. Sci.*, vol. 8, no. 3, p. 465, Mar. 2018.
[21] I. Silva, O. do Espírito Santo, D. do Nascimento, and S. Xavier-de-Souza, "CEVERO: A soft-error hardened SoC for aerospace applications," in *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*. Porto Alegre, Brazil: SBC, 2020, pp. 121–126, doi: 10.5753/sbesc_estendido.2020.13100.
[22] 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 *Proc. IEEE Hot Chips Symp. (HCS)*, Aug. 2015, pp. 1–39.
[23] L. Blasi, F. Vigli, A. Cheikh, A. Mastrandrea, F. Menichelli, and M. Olivieri, "A RISC-V fault-tolerant microcontroller core architecture based on a hardware thread full/partial protection and a thread-controlled watch-dog timer," in *Applications in Electronics Pervading Industry, Environment and Society*, S. Saponara and A. De Gloria, Eds. Cham, Switzerland: Springer, 2020, pp. 505–511.
[24] M. Olivieri, A. Cheikh, G. Cerutti, A. Mastrandrea, and F. Menichelli, "Investigation on the optimal pipeline organization in RISC-V multi-threaded soft processor cores," in *Proc. New Gener. CAS (NGCAS)*, Sep. 2017, pp. 45–48.
[25] A. Cheikh, G. Cerutti, A. Mastrandrea, F. Menichelli, and M. Olivieri, "The microarchitecture of a multi-threaded RISC-V compliant processing core family for IoT end-nodes," in *Applications in Electronics Pervading Industry, Environment and Society*, A. De Gloria, Ed. Cham, Switzerland: Springer, 2019, pp. 89–97.
[26] A. Cheikh, S. Sordillo, A. Mastrandrea, F. Menichelli, and M. Olivieri, "Efficient mathematic accelerator design coupled with an IMT RISC-V microprocessor," in *Applications in Electronics Pervading Industry, Environment and Society*, S. Saponara and A. De Gloria, Eds. Cham, Switzerland: Springer, 2020, pp. 529–539.
[27] M. Barbirotta, A. Cheikh, A. Mastrandrea, F. Menichelli, M. Angioli, S. Jamili, and M. Olivieri, "Fault-tolerant hardware acceleration for high-performance edge-computing nodes," *Electronics*, vol. 12, no. 17, p. 3574, Aug. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/17/3574
[28] L. Blasi et al., "An FPGA-based RISC-V computer architecture orbital laboratory on a pocketcube satellite," *Adv. Astron. Sci.*, vol. 173, pp. 587–593, 2020.
[29] L. A. B. Naviner, J.-F. Naviner, G. G. dos Santos, E. C. Marques, and N. M. Paiva, "FIFA: A fault-injection-fault-analysis-based tool for reliability assessment at RTL level," *Microelectron. Rel.*, vol. 51, nos. 9–11, pp. 1459–1463, Sep. 2011.
[30] W. Sheng, L. Xiao, and Z. Mao, "An automated fault injection technique based on VHDL syntax analysis and stratified sampling," in *Proc. 4th IEEE Int. Symp. Electron. Design, Test Appl.*, Jan. 2008, pp. 587–591.
[31] S. R. Seward and P. K. Lala, "Fault injection in digital logic circuits at the VHDL level," in *Proc. 9th IEEE On-line Test. Symp.*, Jul. 2003, p. 161.
[32] J. Engel, K. S. Morgan, M. J. Wirthlin, and P. S. Graham, "Predicting on-orbit static single event upset rates in Xilinx Virtex FPGAs," Fac. Publications, Tech. Rep. 1307, 2006. [Online]. Available: https://scholarsarchive.byu.edu/facpub/1307
[33] M. V. O'Bryan, K. A. LaBel, E. P. Wilcox, D. Chen, M. J. Campola, M. C. Casey, J. M. Lauenstein, E. J. Wyrwas, S. M. Guertin, J. A. Pellish, and M. D. Berg, "Compendium of current single event effects results from NASA Goddard Space Flight Center and NASA electronic parts and packaging program," in *Proc. IEEE Radiat. Effects Data Workshop (REDW)*, 2017, pp. 1–9, doi: 10.1109/NSREC.2017.8115432.
[34] D. S. Lee, G. R. Allen, G. Swift, M. Cannon, M. Wirthlin, J. S. George, R. Koga, and K. Huey, "Single-event characterization of the 20 nm Xilinx Kintex UltraScale field-programmable gate array under heavy ion irradiation," in *Proc. IEEE Radiat. Effects Data Workshop (REDW)*, Jul. 2015, pp. 1–6.
[35] M. Barbirotta, A. Mastrandrea, F. Menichelli, F. Vigli, L. Blasi, A. Cheikh, S. Sordillo, F. Di Gennaro, and M. Olivieri, "Fault resilience analysis of a RISC-V microprocessor design through a dedicated UVM environment," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2020, pp. 1–6.
[36] N. J. George, C. R. Elks, B. W. Johnson, and J. Lach, "Transient fault models and AVF estimation revisited," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2010, pp. 477–486.
[37] A. Duncan, V. Srinivasan, A. Sternberg, L. Massengill, B. Bhuva, and W. Robinson, "The effect of frequency and technology scaling on single event vulnerability of the combinational logic unit in the LEON2 SPARC V8 processor," in *Proc. Hardened Electron. Radiat. Technol. Conf.*, 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:2751231
[38] ECSS. (Sep. 2016). *ECSS-Q-HB-60-02a—Techniques for Radiation Effects Mitigation in Asics and FPGAS Handbook*. [Online]. Available: https://ecss.nl/hbstms/ecss-q-hb-60-02a-techniques-for-radiation-effects-mitigation-in-asics-and-fpgas-handbook-1-september-2016-published/

FRANCESCO VIGLI received the master's (Laurea) degree (cum laude) in electronics engineering from the Sapienza University of Rome, Italy, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering, Electronics and Telecommunications. Since 2019, he has been a Digital Hardware Designer with ELT Group. His current research interests include developing fault-tolerant techniques and designing microprocessor architectures for space applications.

MARCELLO BARBIROTTA received the Master's (Laurea) degree (cum laude) in electronics engineering from the Sapienza University of Rome, Italy, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering, Electronics and Telecommunications. His current research interests include analysis techniques and models for fault resilience within digital microprocessor architectures, targeting RISC-V cores, and microcontrollers.

ABDALLAH CHEIKH received the B.S. and M.S. degrees in electrical engineering from Rafik Hariri University, Lebanon, in 2014 and 2016, respectively, and the Ph.D. degree in computer architecture from the Sapienza University of Rome, Italy, in 2020, with a focus on computer organization and design. He is currently a Postdoctoral Researcher with the Sapienza University of Rome. His research interests include designing, implementing, and verifying a wide range of microprocessor architectures, vector accelerators, and morphing processors.

FRANCESCO MENICHELLI received the M.S. (cum laude) and Ph.D. degrees in electronic engineering from the Sapienza University of Rome, Italy, in 2001 and 2005, respectively. He is currently an Assistant Professor with the Sapienza University of Rome, Italy. He has coauthored more than 40 publications in international journals and conference proceedings. His research interests include low-power digital design and, in particular, system-level and architectural-level techniques for low-power consumption, power modeling, and simulation of digital system platforms.

ANTONIO MASTRANDREA received the M.S. (Laurea) degree (cum laude) in electronics engineering and the Ph.D. degree from the Sapienza University of Rome, Italy, in 2010 and 2014, respectively. He is currently a Research Assistant with the Department of Information Engineering, Electronics, and Telecommunications, Sapienza University of Rome. His current research interests include digital system-on-chip architectures and nano-CMOS circuits oriented toward high-speed computation.

MAURO OLIVIERI (Senior Member, IEEE) received the M.S. (Laurea) (cum laude) and Ph.D. degrees in electronics and computer engineering from the University of Genoa, Italy. From 1995 to 1998, he was an Assistant Professor with the University of Genoa. In 1998, he joined the Sapienza University of Rome, Italy, as an Associate Professor. He is currently a Visiting Researcher with the Barcelona Supercomputing Center in the European Processor Initiative Project. He has authored more than 100 articles and a textbook in three volumes on digital VLSI design. His research interests include microprocessor core designs and digital nanoscale circuits. He has been a member of the TPC of IEEE DATE. He was the General Co-Chair of IEEE/ACM ISLPED'15. He is an Evaluator of the European Commission in the ECSEL Joint Undertaking.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza" 2' within the CRUI CARE Agreement