Microelectronics Reliability 148 (2023) 115173

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

The TETRISC SoC—A resilient quad-core system based on the ResiliCell approach

Markus Ulbricht [a,*], Li Lu [a], Junchao Chen [a], Milos Krstic [a,b]

[a] IHP - Leibniz Institute for High Performance Microelectronics, Im Technologiepark 25, Frankfurt (Oder), 15236, Germany
[b] University of Potsdam, Am Neuen Palais 10, Potsdam, 14469, Germany

A R T I C L E I N F O

Keywords:
RISC-V
Reliability
TETRISC SoC
Multicore
NMR
Adaptive lockstep system

A B S T R A C T

Resilient systems require monitoring and prediction of environmental and intrinsic conditions and the ability to adapt to changing circumstances to optimize the trade-off between performance, power consumption, and fault tolerance. This paper presents an approach for enabling a design to achieve resilience. By using a range of reliability sensors and the novel ResiliCells, we have developed the TETRISC System-on-Chip (SoC), which is a multiprocessor system based on the PULPissimo platform. The TETRISC SoC can operate its four cores in different performance and fault tolerance modes based on real-time data, making it ideal for use cases with dynamically changing and reliability critical requirements, such as avionics or aerospace. Additional in-depth studies on possible optimizations demonstrate the flexibility of the ResiliCell approach.

1. Introduction

Due to the steadily growing demand for real-time data processing, multi- and manycore systems are becoming increasingly important, especially in avionic and aerospace. Inherent to such systems is always a degree of redundancy that can be exploited to adapt the performance and fault tolerance to specific situations. Fault tolerance, in particular, plays a significant role in this respect. In space, for example, the flux of high energy particles and, thus, the number of transient errors can quickly fluctuate by several orders of magnitude [1]. In order to cope with such demanding environments, systems need to quickly adapt to different requirements and situations by allocating the available redundancy and without interrupting the service.

Over the years, considerable work has been done in the domain of fault tolerance [2], and many of these works have a strong focus on multi-core systems [3]. Adaptive fault tolerance methods, however, are a rather new topic that is gaining more interest in recent years. Kempf et al. for example, proposed a dual-core platform that is based on LEON3 cores [4]. The two cores are able to either run independently and offer maximum performance, or they act as a dual modular redundant (DMR) lockstep system and enable fault tolerance by cross-checking the computational results of the two cores. The distribution of either different or identical instructions to the cores is implemented in software, but the comparison in fault-tolerant mode is realized via an additional pipeline stage. Rogenmoser et al. furthermore presented an adaptive system [5] that is based on the PULP platform [6], an open, scalable hardware and software research and development platform that is based on the RISC-V ISA. Here, a clustered system with two times three cores either runs in performance mode with six independent cores or forms two triple modular redundant systems for soft-error tolerance. The distribution of either different or identical instructions also happens in software, and the error correction is achieved with an additional majority voter in hardware.

If such a system is equipped with respective sensors and the required intelligence, it even gains the ability to trigger the adaption autonomously and proactively. Simevski et al. for example, include temperature sensors in their adaptive PISA platform [7] that consists of four LEON2 cores. The cores are capable of running independently in high-performance mode, individually in destress mode or group-wise in different NMR lockstep combinations. Here, the distribution of the commands, the voting, and the control of the system state is done via a dedicated hardware-based framework controller. This controller also has access to the sensors and triggers the destressing mode if certain heat thresholds are exceeded. In addition to temperature, other types of sensors can be useful for autonomous adaptation too. For example, in-depth studies on on-chip sensors for single event upsets (SEUs) were conducted by Andjelkovic et al. and Chen et al. [8,9]. In this context, the integration of these sensors into the PISA design was discussed as well.

The main objective of our research was to develop a generalizable approach for implementing such a highly adaptive system that is able to deliver a dynamic trade-off between reliability, power, and

* Corresponding author.
E-mail address: ulbricht@ihp-microelectronics.com (M. Ulbricht).

https://doi.org/10.1016/j.microrel.2023.115173
Received 2 June 2023; Received in revised form 24 July 2023; Accepted 28 July 2023
Available online 14 August 2023

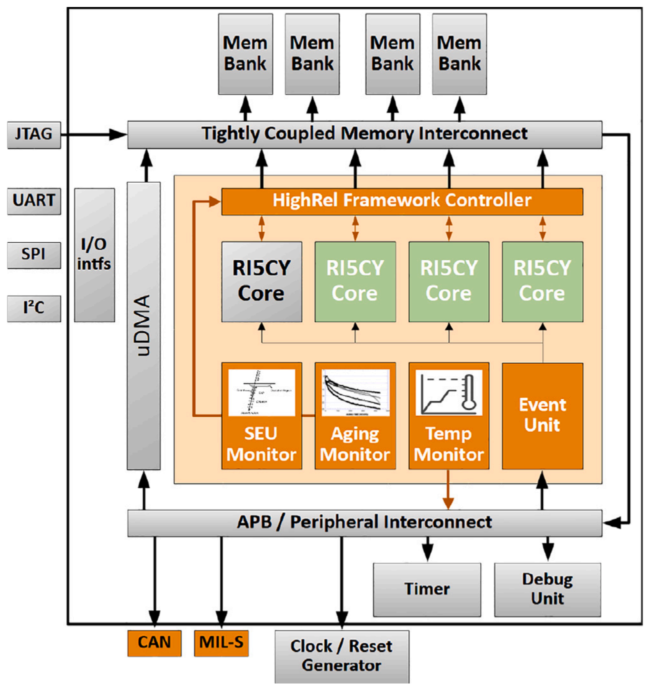

Fig. 1. Schematic architecture of the TETRISC SoC.

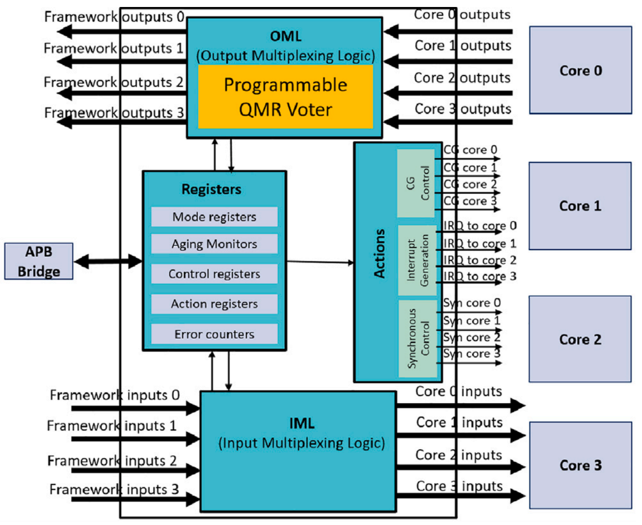

Fig. 2. Block diagram of the HiRel Framework Controller.

performance depending on environmental and intrinsic conditions and trends. In this paper, we describe how we successfully achieved this goal by a combination of lessons learned from prior work and a novel approach based on the so-called ResiliCell. As a proof of concept, we implemented our approach in a state-of-the-art open-source compute platform, which is now suited to be used in rough environments. The remainder of this paper will be divided into two parts. The first one will give deeper insights into the implementation of the resulting platform and will briefly introduce the ResiliCell approach as part of the general architecture. The second part will then focus on the design of the ResiliCells and investigate different implementations to optimize the trade-off between hardware overhead and reliability.

2. The TETRISC SoC

To avoid any ambiguity, we first want to define two concepts that have essential importance for this work — fault tolerance and n-modular redundancy. In terms of fault tolerance, we adhere to the definition by Stolte et al. [10] and refer to any kind of redundant execution as "fault-tolerant”, covering both error-detecting and -correcting configurations. The term n-modular redundancy (NMR) furthermore often refers to a configuration with an odd number of identical systems, but we would not consider this a general definition. Shooman et al. [11], for example, state the following (ch. 4.4.1, p. 153): "In general, N is an odd integer; however, if we have additional information on which systems are malfunctioning and also the ability to lock out malfunctioning systems, it is feasible to let N be an even integer”. A prominent example they provide with an even number of N is the space shuttle control system (ch. 5.9.3, p. 266 ff). In this case, the system consists of four identical processors, forming a 2-out-of-4 system. Based on this, we would extend the term n-modular redundancy to systems with any number of identical systems.

Now let us continue with the description of our system. After thorough investigations on possible RISC-V based platforms, we decided to use the PULP [6] as a basis. But unlike Rogenmoser et al. with OpenPULP and IBEX cores [5], we chose the less complex PULPissimo framework with the more powerful RI5CY core as the better candidate for our cause. The resulting architecture has been named TETRISC (TETra Core System based on RISC-V) SoC and is shown in Fig. 1. The gray blocks illustrate the original IPs of the PULPissimo SoC. The newly added components are shown in green and orange.

2.1. *PULPissimo as multicore*

As a first step, we extended the PULPissimo to a quad-core processor structure (see Fig. 1). Three additional RI5CY cores, shown in green, were inserted into the system and connected to the correspondingly expanded interfaces. We chose to implement the TETRISC SoC as a quad-core system because this offers enough redundancy to withstand even the high radiation environment in space [12] and it also delivers sufficient freedom for implementing a wide range of system configurations, which will be explained later in this section.

In addition to the cores, the memory interface had to be adapted to offer equal access to all four cores over the entire address space. The program and data memory are separated on the software side. As with the original PULPissimo platform, interrupts are handled by a (to the cores) external event/interrupt unit. For TETRISC, this was extended accordingly so all four processors could receive interrupts separately.

2.2. *Reliability-sensors and configurability*

For a sensor-based proactive reconfiguration of the system, three types of sensors were selected that, in combination, provide a comprehensive overall picture of acute or future threats: sensors to determine the temperature, which is a major accelerator of various failure effects; aging detectors for measuring different wear-out related effects on the chip [13], as well as the SEU monitor/solar particle event (SEP) predictor by Chen et al. [9].

2.3. *HiRel framework controller*

To allow hardware-driven reconfigurability and fault tolerance, we extended the design by the High-Reliability Framework Controller (HFC), depicted in Fig. 2. The HFC is based on the framework controller described by Simevski et al. [7] and serves as the main control unit in the SoC. It manages the inputs and outputs across the four cores and enables three different operational modes: high-performance, power-saving (destress), and fault tolerance. The states are realized with either core-level NMR or clock-gating (CG) strategies and are shown in Fig. 3. The circles correspond to the cores, while different colors

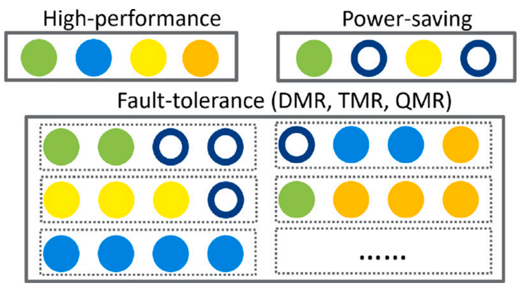

Fig. 3. Operation modes for the system.

represent different tasks running on the cores. Hollow circles represent the clock-gating of a core.

The main component of the HFC is a binary matrix-based programmable NMR majority voter designed for multiprocessors. It allows the inclusion of each processing core into the voting process and, with the help of the Input Multiplexing Logic (IML) and Output Multiplexing Logic (OML), manages the routing of inputs and outputs across all cores within the selection. During fault-tolerant mode, the outputs of the cores are gathered and compared by the programmable voter within the OML. Based on this setup, the system offers various lockstep combinations: double-modular redundancy (DMR) - two cores, single error detection; dual-DMR (D-DMR) - two independent DMR systems with two cores, single error detection each; triple-modular redundancy (TMR) - three cores, single error correction; quadruple-modular redundancy (QMR) - four cores, double error correction (if not identical errors); The QMR configuration enables a degree of fault tolerance, which is comparable to the system by Rogenmoser et al. [5], but with altogether only four cores instead of six. The current operating mode and NMR group are specified in the mode register, which controls the programming bits of the programmable voter. The action registers steer different measures initiated upon voting discrepancies and voter errors (e.g., IRQs, synchronous, clock on/off). The voter outputs increment the four error counters, thereby counting the detected errors for each core.

The HFC contains four aging monitors [13] to track the effects of Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) on each core. It should be noted that while the registers of these aging monitors are logically integrated with the HFC, their sensory components are physically dispersed within each core to detect core aging. For instance, should a core be clocked off, the corresponding sensory element of the aging monitor is likewise deactivated.

The collaborative efforts of the HFC and the onboard monitor system enable the implementation of two operational mode reconfiguration strategies: user-defined- and sensor-defined-NMR. User-defined NMR allows users to activate predefined modes by writing into the registers of the framework controller. It is designed to provide fault-tolerant protection for critical software operations, such as satellite altitude control processes. On the other hand, sensor-defined-NMR is intended to enhance the resilience of all tasks during challenging conditions as they are measured by the sensors, such as periods of solar particle events that the radiation monitor network can detect.

2.4. The ResiliCell approach

As an important, fully novel feature, we aimed to ensure that the processors can switch to the different NMR modes and back within a few clock cycles without losing the processor state. For this purpose, every flip flop (FF) of the RI5CY cores was replaced by a so-called ResiliCell (RC), which extends the "original" FF by a second Slave FF and some control logic. The ResiliCells were originally inspired by shadow registers but in this case, they are not used to store the processor state for recovery but to mirror the system state of another processor into a cores logic paths. This can be seen in Fig. 4, where identical RCs of two different cores are depicted. In performance mode, the two processors run in parallel on different tasks. Once more fault tolerance is required, the HFC programs the content of every original FF (F) of core one into the Slave FF (S) of core two without interrupting the operation of core two. Afterward, the HFC fully switches into redundant mode by rerouting all signals from F to S in all ResiliCells, thereby changing the entire system state of core two. Together with supplying identical inputs to both cores and cross-checking the primary outputs, this instantly forms a DMR system. Once the system requires more performance, the HFC restores the original processor state of core two by switching back the original FFs into their respective path.

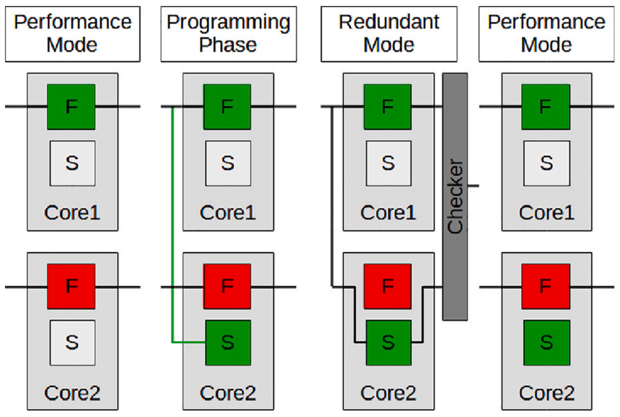

Fig. 4. General principle of the ResiliCells to achieve redundant task execution.

2.5. Hardening peripherals

In addition to the previously mentioned fault mitigation methods, we have implemented TMR (Triple Modular Redundancy) flip-flops to protect all components outside the cores from radiation effects. This approach, proposed by Schrape et al. [14], helps mitigate Single Event Effects (SEE) caused by energetic particles. Based on established research [15], SRAM, the most susceptible element which accounts for nearly 40% of the target chip's area, necessitates sufficient hardening strategies. Our design incorporates the HSIAO (39,32) Single Error Correction and Double Error Detection (SEC-DED) code within the SRAM blocks to safeguard the memory contents. Additionally, we have implemented a scrubbing technique in the memory blocks, which helps prevent fault accumulation and reduces the risk of data corruption within the memory banks.

Furthermore, various hardening systems can be implemented at different layers of the system, employing various fault mitigation methods [16]. At the circuit layer, which forms the foundational level, fault occurrence primarily depends on circuit design and inherent physical properties. For example, the LEAP-DICE hardened flip-flop design utilizes Dual-interlocked cell (DICE) and Error Aware Transistor Positioning (LEAP) technology. Moving up to the logic layer, which encompasses different gates and memory elements, various Error Correcting Codes (ECCs) can be employed for fault management. In the architectural layer, common strategies include data and control flow checking methods and module-level redundancy. Progressing to the software layer, fault tolerance is typically achieved through software-based techniques such as code and data replication for comparison, as seen in duplication with comparison technique. Alternatively, Algorithm-Based Fault Tolerance (ABFT) strengthens the system by integrating detection or correction mechanisms at the algorithmic layer.

Table 1
TETRISC SoC details.

Chip area	4356 mm^2 (6.6*6.6 mm)
Nominal clock frequency	30 MHz
Power consumption	<1 W (estimated)
Memory	4*8192*40 Bit SRAM
Pads	81 Signal pads, 35 others

3. Improvements to the ResiliCell approach

In order to fully evaluate the design with respect to adaptability and fault tolerance, we implemented the TETRISC architecture in IHPs SG13S technology, which is our current 130 nm node. The final chip's most significant properties are given in Table 1. The comparatively low limit in clock frequency will be improved in future versions.

In the following section, we want to focus on optimizing the approach by looking deeper at the ResiliCells. Since they replace every FF of every core in the system, their design and number greatly impact the overall system. Therefore, we will first present different layout options for the cells, each offering distinct advantages. Then, we will optimize the ResiliCells for a trade-off between hardware overhead and reliability by identifying an optimal number of master cores on the one hand and identifying and selectively hardening "critical" cells on the other.

3.1. Different versions of the ResiliCells

To provide an impression of the current hardware overhead, we have listed the area of the design in Table 2 after synthesizing it with IHP's 130 nm library. Column 0 (Orig) shows the original area of PULPissimo. Column 1 (Base) summarizes the area of TETRISC without ResiliCells, including the three additional RI5CY cores, sensors, interfaces, and the HFC. The third column (v0) gives data about the TETRISC, including ResiliCells, as depicted in gray in Fig. 5. The other columns will be explained soon. The first row lists the area of a memory cell. Here the size of the "original" FF in Orig and Base can be compared to the basic ResiliCell in column v0. It can be seen that the area is more than quadrupled due to the additional Slave FF and multiplexers. The size of the RI5CY core grows between Orig/Base and v0 by about 54%. This growth comes from the 3041 FFs that our ResiliCells replace. The area of the overall SoC increases by 23% from Orig to Base and by 30% from Orig to v0. This moderate increase also shows that the area of the SoC is very much affected by the other components (SRAM, controller, IOs) outside of cores. The last row displays the relative size compared to the Base variant.

An attractive property of the ResiliCell approach is that it can be easily adjusted to require less hardware or to include higher degrees of fault tolerance by optimizing the cell design. Fig. 5 depicts the different adjustment options. The basic cell in gray is the original implementation (v0). It enables mirroring the system state of every core to every other core, allowing every combination of NMR subsystems. By removing (some of) the multiplexers marked with "vI", one reduces the number of possible input FFs and thereby the number of "master" cores of the NMR subsystems, but also decreases the hardware overhead.

A second adjustment ("vII") allows daisy-chaining of the ResiliCells during the programming phase, which results in an execution delay of the cores in the NMR system of 1 cycle. Together with corresponding input and output buffers, this implements a delayed lockstep similar to the one deployed in the MPC5744P core [17] and offers protection against common cause failures.

The third variant ("vIII") adds an additional FF with a delay element on the clock line. It, therefore, implements a simplified version of the radiation-hardened TMR FF by Schrape et al. [14]. This way, an SEU cannot "silently corrupt" the system state in the original FF while disabled. The implementation overhead for the different versions can be taken from Table 2, columns vI, vII, and vIII.

Table 2
Area comparison of the different Extensions.

	Orig	Base	v0	vI	vII	vIII
Cell (um^2)	32,1	32,1	130,4	103,9	143,6	190,8
Core ($e^{-3}mm^2$)	552,1	552,1	850,9	770,5	891,2	1034,8
Core (% growth)	0	0	54,1	39,5	61,4	87,4
SoC (mm^2)	17,3	21,2	22,4	22,1	22,6	23,1
SoC (% growth)	0	22,8	29,7	27,8	30,6	33,9
SoC (% growth)	$-18,5$	0	5,6	4,1	6,3	9

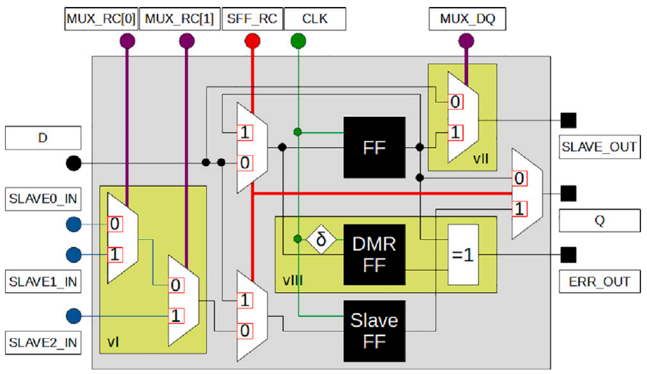

Fig. 5. Schematic design of the ResiliCell (gray) with adapted versions I, II and III (green).

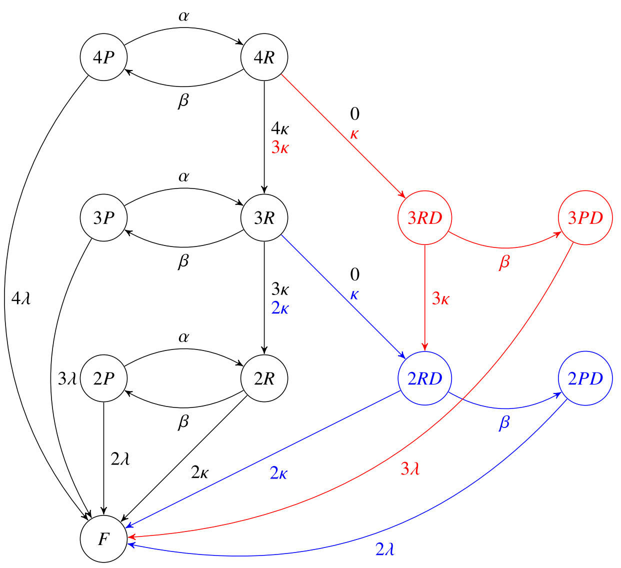

Fig. 6. Markov Chain.

3.2. Optimizing the number of masters

To find an optimal implementation of the ResiliCell, we first want to look at vI. As mentioned above, this version minimizes the number of multiplexers, thereby limiting the number of "master" cores of which the state can be written into the Slave FF. With an overall reduction of the hardware overhead by about 15% per core and roughly 1.5% for the SoC, this improvement does not have the greatest impact. Nevertheless, finding the minimal amount of masters is a very compelling task.

In principle, one single master would suffice to make full use of the system's configurability and adaptiveness, as only one single task can be executed redundantly at a time (or two in DDMR). Critical tasks of other cores could be scheduled to the master as required. Still, this would cost additional time. More masters, on the other hand, would lead to more hardware — but less scheduling overhead and possibly more freedom if one of the masters fails permanently.

To investigate the long-term effects on functionality, we developed a reliability model based on the Markov chain approach that models the main aspects of our system. The result is depicted in Fig. 6. The basic system with four masters is depicted in black and starts off at state “4P”, where all four cores run in parallel. Each core might fail with a rate of λ, and since in this state, there is no method for fault tolerance active, we consider the system as failed (state “F”). If the sensors report an incident that requires more fault tolerance, the system switches into the “4R” state. For simplicity, we will assume that this is a QMR state. We also assume those incidents occur randomly with rate parameter α. Once the situation normalizes, the system returns to the “4P” state with rate β. We assume that transient faults are also covered by β and repaired by returning to performance mode. If, on the other hand, one of the cores fails with a permanent error, the system now degrades into the TMR mode “3R” with a rate κ for each core. κ will generally be bigger than λ due to the incident reported by the sensors. Now, analog to the “4R” state, the system might switch back to performance mode with three functioning cores (“3P”) with rate parameter β or further degrade with κ. From state “2R” or “2P”, no further degradation is possible, and we consider the entire system failed with the respective rates.

Suppose the number of multiplexers and, thereby, the number of masters is reduced. In that case, the system gradually loses its ability to transition from performance to a redundant state when the remaining master cores permanently fail. This is depicted in the blue and red “sidearms”. With a single master, there is a chance of 3κ that the system degrades and the master is still functional. But if the master fails with rate κ, the system reaches the “3RD” state. From here, it can further degrade in redundant mode with rate κ for each remaining core or move to performance mode with rate β. But since there is no functioning master, it cannot move back to a redundant mode and fails as soon as one of the remaining cores fails. Starting with two masters, the system can only enter the blue sidearm after the two masters have failed in succession.

We derive the transition matrix from the graph in Fig. 6 and compute the steady-state probability of all states with the help of the numerical approach presented in [18] with an initial state probability of 1 for state “4P”. For defining the parameters α, β, λ and κ, we refer to the results presented by Chen [12]. In his thesis, he determines the time a space-born system would be subject to increased radiation (in which it would move to fault-tolerant mode) and provides an amount of 220 h or 2.5% per year. Based on this, we set $\alpha = 0.025$ and $\beta = 0.975$. The radiation rate moves from $2.8 * 10^{-6}$ upsets/bits/day, which is the lower limit for moving into a fault-tolerant state, to $1.32 * 10^{-3}$ upsets/bits/day, which is the average of the highest radiation events during the measurement period. To further include the additional hardware for implementing multiple masters, we introduce another factor, δ. This factor scales the respective fault rate according to the percentage of hardware overhead per core because, with increasing hardware, the susceptibility to faults rises. In reality, hardware susceptibility strongly depends on the nature of different fault effects as well, but we will accept this simplification for now. Thus, we set $\delta_4 = 1, \delta_3 = 0.96, \delta_2 = 0.92, \delta_1 = 0.84$ for 4, 3, 2, and 1 masters, respectively.

The results of this investigation are depicted in Fig. 7. We have plotted the reliability of the RI5CY-based QMR systems for the different numbers of masters with the aforementioned assumptions and parameters. The implementations with 3 or 4 masters have the lowest reliability, which shows the heavy impact of the additional hardware overhead. The preferable implementation is the one having two masters. It is characterized by the highest reliability and more flexibility than the one with a single master.

3.3. Identification of critical FFs

To further optimize the hardware–reliability trade-off, we aim to identify the set of ResiliCells that are most critical and thus require

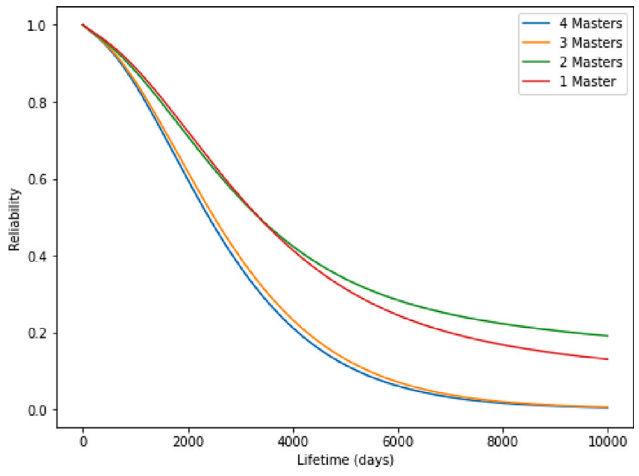

Fig. 7. The reliability of the RI5CY-based QMR systems with different numbers of masters.

the highest level of fault tolerance. Therefore, we decided to follow a structured approach based on the vulnerability of a core presented in [19]. It puts the susceptiveness of a (partial) design to faults in relation to a particular simulation’s overall runtime. It, therefore, delivers information on how vulnerable a design really is.

To apply this to the ResiliCells, we first generated random instructions representing approximately 90% of the instruction set as a test set by using a program based on the assembler software generator riscv-torture [20]. The resulting programs included around 5000 instructions with an execution time of about 13.6k cycles. After that, simulations based on these routines were performed using the Xcelium fault simulator [21], widely used in industrial applications.

A fault injection campaign followed this. First, at time 0, two types of permanent faults (SA0 and SA1) were injected into all FFs of the original RI5CY core. If the simulator did not detect the fault at the primary outputs of the design, we conclude that it is never activated. Therefore, the fault could be ruled out of the following injection of SEUs and the overall amount of simulations could be reduced. If, on the other hand, the error was detected, the time of the detection was measured. Starting at this time x, where $x = Min(Detection_{SA0}, Detection_{SA1})$ and thus equals the activation time of the FF, SEUs (bitflips) were injected into the FF several times until the simulation ended at time t. The injection frequency was evenly distributed over the time window from x to t. The so-called SEU-induced failure rate (SIFR) of a FF under a certain test load hence describes the probability of the SEU propagating to the primary outputs of the design. Thus, if n faults were injected during the time window from x to t and m were detected, the SIFR for the FF under the given load was calculated as follows:

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

If neither of the two permanent errors of a FF was detected, its SIFR was set to 0 right away.

The SIFR for all flip-flops of the RI5CY core is given in Table 3. The first and third rows contain the rounded SIFR, and the second and fourth rows the number of FFs that fall in this value range. The last two cells on the bottom right present the average SIFR and number of all FFs in the core. It can be seen that about half of the FFs have a SIFR of 0. This means that the software never activated the respective s-a-0/1 faults or bit-flips and did not propagate to the primary outputs of the design. The other half has a higher susceptibility, which goes up to 1 for 407 FFs, where the faults were always visible.

Table 3
SIFR of the FFs in the design.

SIFR	0	0,1	0,2	0,3	0,4	0,5
# FFs	1561	100	65	99	99	17
SIFR	0,6	0,7	0,8	0,9	1	**ø 0, 33**
# FFs	153	153	185	203	407	**Σ 3041**

3.4. Optimizing the number of ResiliCells with variant vIII

Based on the distribution of the SIFR among the FFs, the ResiliCells can be further optimized concerning the trade-off between the additional hardware overhead for vIII and the respective gain in reliability. Not adding redundancy to ResiliCells, of which the output is always masked or never used, and protecting the cells with a SIFR of 1 is a good starting point. But what about ResiliCells with a SIFR that is somewhere in between? To investigate this, we first collected the necessary hardware cost to protect all the cells above a certain SIFR range. Let Γ be the different ranges given in Table 3, lines 1 and 3, and Δ the number of ResiliCells in this range as given in lines 2 and 4. Table 4 contains the cost of protecting all cells from $\Gamma_0 = 1$ down to and including the given Γ_i value with the additional DMR FF, XOR, and delay element of vIII.

Table 4
Relative size of the implementations for vIII with different Γ_i.

Version	v0	$\Gamma_0 = 1.0$	$\Gamma_1 = 0.9$	$\Gamma_2 = 0.8$
Growth %	0	2.88	4.33	5.64
Version	$\Gamma_3 = 0.7$	$\Gamma_4 = 0.6$	$\Gamma_5 = 0.5$	$\Gamma_6 = 0.4$
Growth %	6.73	7.82	7.94	8.64
Version	$\Gamma_7 = 0.3$	$\Gamma_8 = 0.2$	$\Gamma_9 = 0.1$	$\Gamma_10 = 0$
Growth %	9.34	9.81	10.52	21.61

To investigate the reliability gain of the different implementations of vIII, we develop a reliability model of the ResiliCells that includes both the measured SIFR value and the number of cells that need additional protection. To achieve this, we first model the reliability of the "original" Flip-Flop (R_{FF}) in two phases: a low-stress phase (ls), where the system is typically in performance mode, and a high-stress phase (hs), in which the system is in redundant mode. The reliability in the ls phase (R_{ls}) is characterized by a constant fault rate λ, which lasts for a percentage α ($0 \leq \alpha \leq 1$) of time t. The fault rate λ is scaled by the SIFR-factor Γ_i, as presented in Table 3. The reliability in the hs phase (R_{hs}) is analogous with constant fault rate κ, duration β ($\alpha + \beta = 1$) and SIFR-factor Γ_i:

$$R_{ls}(i,t) = e^{-\lambda\alpha\Gamma_i t}, \quad R_{hs}(i,t) = e^{-\kappa\beta\Gamma_i t} \tag{2}$$

$$R_{FF}(i,t) = R_{ls}(i,t) * R_{hs}(i,t) \tag{3}$$

The reliability of all FFs that do not implement vIII (R_{v0}) can now be described as a series system of the reliability of all FFs in the respective SIFR-ranges (Δ_i) as taken from Table 3:

$$\mathbf{R_{v0}(j,k,t)} = \prod_{i=j}^{k} (R_{ls}(i,t))^{\Delta_i} * \prod_{i=j}^{k} (R_{hs}(i,t))^{\Delta_i} \tag{4}$$

To model the implementation of vIII, we must include two additional aspects: the influence of the combinational logic (XOR, delay element) and the two flip-flops forming a DMR subsystem. Concerning the two logic gates, one must consider a different size compared to the FF and a different nature regarding the susceptibility to certain faults. To take this into account, we assume a different but proportional constant fault rate that we will express by the factor δ. The reliability of the additional logic in the low-stress (R_{xls}) and high-stress (R_{xhs}) phases is then calculated as follows:

$$R_{xls}(i,t) = (R_{ls}(i,t))^{\delta}, \quad R_{xhs}(i,t) = (R_{hs}(i,t))^{\delta} \tag{5}$$

With respect to the two FFs, we consider the system failed if both the "original" and the DMR FF are faulty (because then they produce the same erroneous result). The reliability of the two flip-flops can then be modeled with the well-known formula for k-out-of-n-systems for the ls (R_{knls}) and hs (R_{knls}) phases as follows with $n = 2$ and $k = 1$:

$$R_{knls}(i,t) = \sum_{k}^{n} \binom{n}{k} R_{ls}(i,t)^{k}(1 - R_{ls}(i,t))^{n-k} \tag{6}$$

$$R_{knhs}(i,t) = \sum_{k}^{n} \binom{n}{k} R_{hs}(i,t)^{k}(1 - R_{hs}(i,t))^{n-k} \tag{7}$$

The reliability of all FFs that implement vIII (R_{v3}) can now be described as a series system of the logic gates and of the reliability of all FFs in the respective SIFR-ranges (Δ_i):

$$\mathbf{R_{v3}(j,k,t)} = \prod_{i=j}^{k} (R_{xls}(i,t) * R_{knls}(i,t))^{\Delta_j} * \prod_{i=j}^{k} (R_{xhs}(i,t) * R_{knhs}(i,t))^{\Delta_j} \tag{8}$$

Finally, the reliability of all FFs that either are or are not implemented in vIII can be calculated as follows:

$$\mathbf{R_{sys}(j,k,l,t) = R_{v0}(j,k,t) * R_{v3}(k,l,t)} \tag{9}$$

The results of our investigations in this regard can be seen in Fig. 8. We set $\alpha = 0.975$, $\beta = 0.025$, $\lambda = 2.8 * 10^{-6}$ and $\kappa = 1.32 * 10^{-3}$. We ran the calculations twice to take into account the different natures of the logic regarding fault susceptibility. First (left), we set $\delta = 0$. This erases

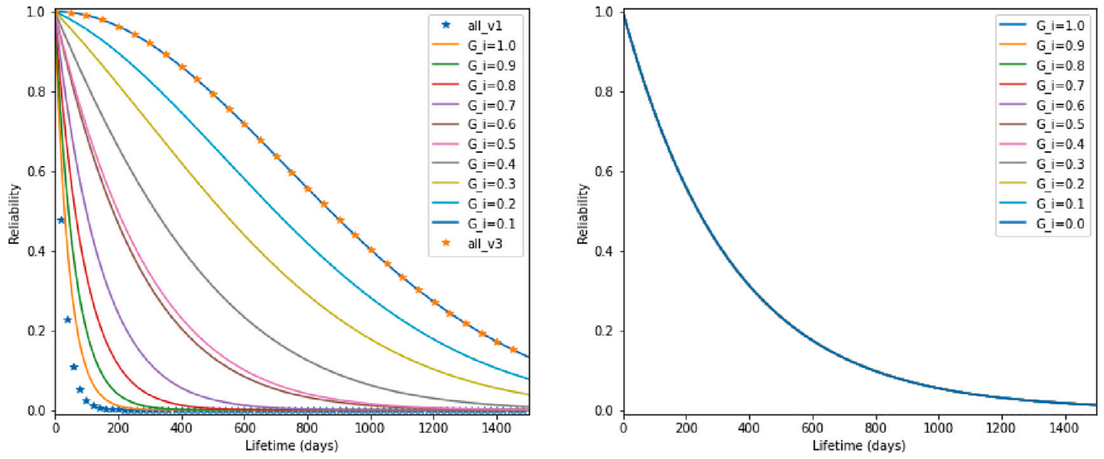

Fig. 8. Reliability of the implementations for vIII with different Γ_i for transient (left) and permanent (right) faults.

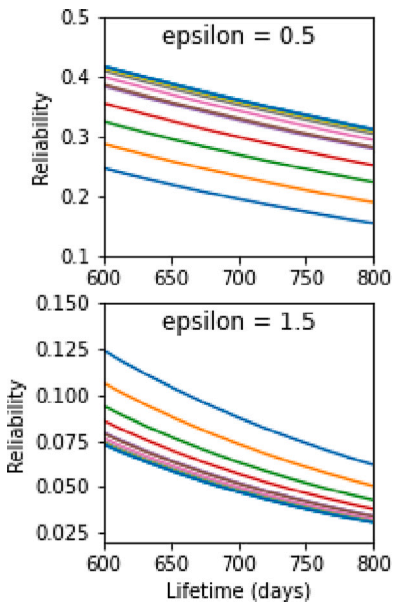

Fig. 9. "Turning point" for the reliability at $\approx \delta = 1$.

the influence of the logic altogether and therefore models the behavior of the ResiliCells concerning SEUs only. Looking at the graph, it can be seen that having only ResiliCells without vIII (all_v1) delivers the lowest reliability. Including more cells increases the reliability (Γ = 1.0 – Γ = 0.1), reaching its maximum if only those with $\Gamma = 0.0$ are excluded or if all ResiliCells are implemented with vIII (all_v3). The last two cases are identical because of a SIFR factor of $\Delta = 0$ for $\Gamma = 0$.

In the second run (right), we set $\delta = 1$ because the overall area of the XOR and the delay element roughly equal the area of the FF and change the value of κ to the same value as λ. This models the effects of long-term static faults that affect the entire ResiliCell (logic and FFs) at the same constant rate during both phases. It can be seen that the reliability is now lower and (nearly) equal for all cases. This leads to the conclusion that the negative impact of the additional logic equalizes the gain in the reliability of the DMR implementation. Interestingly, $\delta = 1$ roughly marks the turning point between the DMR systems improving the reliability and the (unprotected) combinational logic worsening it. This can be seen in Fig. 9. In the top graph, with $\delta = 0.5$, adding redundancy improves the reliability, while in the bottom graph with $\delta = 1.5$, more ResiliCells with vIII only decrease it.

4. Conclusion and outlook

This paper presented an approach for implementing a highly configurable system that is able to dynamically adapt to internal or external conditions as they are measured by relevant sensors. By inserting the HFC and the ResiliCells, we enabled the system to run the four cores either independently in high-performance mode, exclusively for destressing/power reduction, or in different NMR subsystems for fault tolerance without interrupting the service.

To further improve the approach, we presented several possible designs for the ResiliCell that optimize hardware overhead or add additional fault tolerance. We developed a system reliability model based on a Markov chain and determined an optimum of 2 master cores. On top of this, we introduced the SIFR, a unit of measure to state the criticality of flip-flops under a given load. Based thereupon, we investigated the impact of selectively hardening the ResiliCells on hardware overhead and reliability. Unfortunately, we cannot provide clear guidelines for design decisions in this case. The results strongly depend on the selected fault model and other requirements, such as hardware overhead or power consumption. Still, for our target areas in avionics or space, where SEUs represent a dominant fault effect, it is preferable to implement vIII for every ResiliCell with a SIFR > 0.

For future research, some interesting points can be addressed. First of all, our approach is very hardware-centric. Adding a software perspective, especially for scheduling the tasks between the masters, will deliver interesting results. Second, we have concentrated on the cores and mostly ignored the peripherals or the board as sources of errors. Widening the view will add to the overall reliability of the system. Third, the models we developed and the variables we used as a basis should be refined to reflect reality better. We have chosen them to the best of our knowledge, but there are still things open for discussion.

CRediT authorship contribution statement

Markus Ulbricht: Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Visualization. **Li Lu:** Writing – original draft, Writing – review & editing, Investigation. **Junchao Chen:** Writing – original draft, Writing – review & editing, Investigation. **Milos Krstic:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

The project on which this report is based was funded by the German Federal Ministry of Education under grant number 16ME0134. The responsibility for the content of this publication lies with the authors. During the preparation of this work, the authors used the large language model GPT-3.5 in order to improve readability and language. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] R. Glein, F. Rittner, A. Heuberger, Detection of solar particle events inside FPGAs, in: 2016 16th European Conference on Radiation and Its Effects on Components and Systems, RADECS, 2016, pp. 1–5, http://dx.doi.org/10.1109/RADECS.2016.8093159.

[2] S. Safari, M. Ansari, H. Khdr, P. Gohari-Nazari, S. Yari-Karin, A. Yeganeh-Khaksar, S. Hessabi, A. Ejlali, J. Henkel, A survey of fault-tolerance techniques for embedded systems from the perspective of power, energy, and thermal issues, IEEE Access 10 (2022) 12229–12251, http://dx.doi.org/10.1109/ACCESS.2022.3144217, URL https://ieeexplore.ieee.org/document/9684471/.

[3] H. Mushtaq, Z. Al-Ars, K. Bertels, Survey of fault tolerance techniques for shared memory multicore/multiprocessor systems, in: 2011 IEEE 6th International Design and Test Workshop, IDT, 2011, pp. 12–17, http://dx.doi.org/10.1109/IDT.2011.6123094, ISSN: 2162-061X.

[4] F. Kempf, T. Hartmann, S. Baehr, J. Becker, An adaptive lockstep architecture for mixed-criticality systems, in: 2021 IEEE Computer Society Annual Symposium on VLSI, ISVLSI, (ISSN: 2159-3477) 2021, pp. 7–12, http://dx.doi.org/10.1109/ISVLSI51109.2021.00013.

[5] M. Rogenmoser, N. Wistoff, P. Vogel, F. Gürkaynak, L. Benini, On-Demand Redundancy Grouping: Selectable Soft-Error Tolerance for a Multicore Cluster, 2022, http://dx.doi.org/10.48550/arXiv.2205.12580.

[6] PULP platform. URL https://pulp-platform.org/.

[7] A. Simevski, O. Schrape, C. Benito, M. Krstic, 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, (ISSN: 1942-9401) 2020, pp. 1–6, http://dx.doi.org/10.1109/IOLTS50870.2020.9159716.

[8] M. Andjelkovic, J. Chen, A. Simevski, Z. Stamenkovic, M. Krstic, R. Kraemer, A review of particle detectors for space-Borne self-adaptive fault-tolerant systems, in: 2020 IEEE East-West Design & Test Symposium, EWDTS, (ISSN: 2472-761X) 2020, pp. 1–8, http://dx.doi.org/10.1109/EWDTS50664.2020.9225138.

[9] J. Chen, T. Lange, M. Andjelkovic, A. Simevski, L. Lu, M. Krstic, Solar particle event and single event upset prediction from SRAM-based monitor and supervised machine learning, IEEE Trans. Emerg. Top. Comput. 10 (2) (2022) 564–580, http://dx.doi.org/10.1109/TETC.2022.3147376, Conference Name: IEEE Transactions on Emerging Topics in Computing.

[10] T. Stolte, S. Ackermann, R. Graubohm, I. Jatzkowski, B. Klamann, H. Winner, M. Maurer, Taxonomy to unify fault tolerance regimes for automotive systems: Defining fail-operational, fail-degraded, and fail-safe, IEEE Trans. Intell. Veh. 7 (2) (2022) 251–262, http://dx.doi.org/10.1109/TIV.2021.3129933, Conference Name: IEEE Transactions on Intelligent Vehicles.

[11] N-modular redundancy, in: Reliability of Computer Systems and Networks, John Wiley & Sons, Ltd, 2002, pp. 145–201, http://dx.doi.org/10.1002/047122460X.ch4, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/047122460X.ch4.

[12] J. Chen, A Self-Adaptive Resilient Method for Implementing and Managing the High-Reliability Processing System, Universität Potsdam, 2023, http://dx.doi.org/10.25932/publishup-58313, URL https://publishup.uni-potsdam.de/frontdoor/index/index/docId/58313.

[13] A. Simevski, R. Kraemer, M. Krstic, Low-complexity integrated circuit aging monitor, in: 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2011-04, pp. 121–125, http://dx.doi.org/10.1109/DDECS.2011.5783060.

[14] O. Schrape, M. Andjelković, A. Breitenreiter, A. Balashov, M. Krstić, Design concept for radiation-hardening of triple modular redundancy TSPC flip-flops, in: 2020 23rd Euromicro Conference on Digital System Design, DSD, 2020, pp. 616–621, http://dx.doi.org/10.1109/DSD51259.2020.00101.

[15] J. Chen, M. Andjelkovic, A. Simevski, Y. Li, P. Skoncej, M. Krstic, Design of SRAM-based low-cost SEU monitor for self-adaptive multiprocessing systems, in: 2019 22nd Euromicro Conference on Digital System Design, DSD, 2019, pp. 514–521, http://dx.doi.org/10.1109/DSD.2019.00080.

[16] E. Cheng, J. Abraham, P. Bose, A. Buyuktosunoglu, K. Campbell, D. Chen, C.-Y. Cher, H. Cho, B. Le, K. Lilja, S. Mirkhani, K. Skadron, M. Stan, L. Szafaryn, C. Vezyrtzis, S. Mitra, Cross-layer resilience in low-voltage digital systems: Key insights, in: 2017 IEEE International Conference on Computer Design, ICCD, 2017, pp. 593–596, http://dx.doi.org/10.1109/ICCD.2017.103.

[17] Freescale Semiconductor Inc, Safety manual for MPC5744P, 2014, URL https://www.nxp.com/files-static/microcontrollers/doc/ref_manual/MPC5744PSM.pdf.

[18] V. Knight, Timing Data for Algorithms for Calculating Steady State Distributions of Continuous Time Markov Chains, Zenodo, 2018-12-19, http://dx.doi.org/10.5281/ZENODO.2429025, Type: dataset, URL https://zenodo.org/record/2429025.

[19] D. Asciolla, L. Dilillo, D. Santos, D. Melo, A. Menicucci, M. Ottavi, Characterization of a RISC-V microcontroller through fault injection, in: S. Saponara, A. De Gloria (Eds.), Applications in Electronics Pervading Industry, Environment and Society, in: Lecture Notes in Electrical Engineering, Springer International Publishing, Cham, 2020, pp. 91–101, http://dx.doi.org/10.1007/978-3-030-37277-4_11.

[20] P. Adelt, B. Koppelmann, W. Mueller, C. Scheytt, Register and instruction coverage analysis for different RISC-v ISA modules, in: MBMV 2021; 24th Workshop, 2021, pp. 1–8.

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