ENGAGE-V: A RERI-Compliant RISC-V Module for RAS in Space Applications

Nicasio Canino, Daniele Rossi, and Sergio Saponara
Department of Information Engineering, University of Pisa, Pisa, Italy

***Abstract*—The adoption of the RISC-V Instruction Set Architecture (ISA) in space applications has gained momentum due to its open-source and royalty-free nature, offering flexibility and customization. This paper presents the design and implementation of the ENGAGE-V peripheral, a RERI-compliant (Reliability availability and serviceability Error record Register Interface) error logging and reporting module tailored for RISC-V-based systems in space environments. The ENGAGE-V module enhances the resilience and fault tolerance of the system by providing customizable and standardized error logging and reporting capabilities, supporting standard fault tolerant techniques in addressing the challenges posed by cosmic radiation, solar particle events, and extreme temperatures. Our approach leverages the modularity of RISC-V to create a robust solution that can be integrated into various space-related systems, from low-power embedded devices to high-end computing platforms. The exploration of the ENGAGE-V module's design variables focuses on key parameters that configure the error logging and reporting capabilities, for a balanced trade-off between resource constraints and error management needs. By implementing the ENGAGE-V module, designers can extend the operational life and reliability of space missions, reducing the need for redundant systems and minimizing mission-critical failures.**

I. Introduction / Topic Discussion

Over the last few years, the space industry and the research community have progressively adopted the RISC-V Instruction Set Architecture (ISA) to design novel and resilient computing systems targeting space applications. A key motivating factor is its royalty-free and open-source nature, which provides designers with a flexible and extremely customizable ISA. RISC-V-based systems can cover low-power to high-end space applications with address space configurations of 32-bit (RV32) and 64-bit (RV64), which are also structured to support many-core implementations [1]–[3]. The modularity of RISC-V allows for customization of processors for space applications, yet designers must prioritize system resilience and fault-tolerance by enhancing the computing system's Reliability, Availability, and Serviceability (RAS).

Resilient and fault-tolerant computing systems (high RAS) are crucial in space applications due to the inherently harsh and unpredictable conditions of the space environment. Spacecraft and satellites are constantly exposed to extreme conditions, which leads to a high incidence of soft and hard hardware errors, thus critically compromising the functionality and longevity of space missions [4]. In space applications, the main causes of hardware errors are cosmic radiation and solar particle events, which can penetrate electronic components and trigger effects such as Single-Event Upsets (SEUs), Single-Event Transients (SETs), and Single-Event Latch-ups (SELs). Additional factors such as electromagnetic interference and extreme temperatures increase the susceptibility of onboard systems. Therefore, key components such as processors, memories, and communication systems are among the most vulnerable to these errors. Incorporating resilient and fault-tolerant designs not only mitigates the risks associated with hardware errors but also extends the operational life of spacecraft.

Redundancy is a key approach to improving the system fault tolerance of the targeted components, distinguished in three main categories: Temporal redundancy repeats operations over time with the same HW unit to save costs but increase execution time; Spatial redundancy relies on parallel HW units to perform the same operations, often implementing Triple-Modular Redundancy (TMR) for quick error detection; and Information redundancy adds extra information generated by Error Correcting Codes (ECC) that can detect and even correct some errors [4]–[6]. To further increase fault tolerance and, more generally, RAS, these redundancy approaches should be supported by error logging and reporting features. Some works in this direction propose solutions that include these features, but they log and report the detected errors via custom interfaces, which are specific to the targeted problem. An alternative solution can come from the implementation of a standardized and flexible Hardware-Software Interface (HW-SW Interface) for Error Logging and Reporting. It is an additional module that monitors HW units that implement an error-checking solution, exploiting redundancy. When the monitored HW unit detects an error, this HW-SW interface will retrieve a set of error-related information that will be logged in an ad hoc set of registers (*Error Record*), which is accessible by SW through memory-mapped access. Then, depending on error severity, the detected error can be reported to the system SW via an interrupt signal.

Previously, we developed the ENGAGE peripheral (Error loggiNG And reportinG architecturE), as a solution to provide error logging and reporting features in any computing system [7]. In this work, instead, we present and explore an updated version that specifically targets RISC-V-based systems, called ENGAGE-V. It offers standardized logging and reporting capabilities for HW errors, ensuring compliance with the ISA-agnostic RERI specification (RAS Error Record Register Interface), thus providing an additional layer of resilience for space-oriented systems [8]. Moreover, the ENGAGE-V peripheral offers significant flexibility and customization during

This research was partially supported by EuroHPC Joint Undertaking projects EPI-SGA2 (European Processor Initiative) and DARE (Digital Autonomy with RISC-V in Europe).

the implementation phase (maintaining the RERI compliance), enabling designers to accommodate various trade-offs related to system resource overhead and logging/reporting needs.

II. RERI-COMPLIANT ENGAGE-V MODULE

As anticipated, our proposed ENGAGE-V module provides error logging and reporting capabilities that comply with the RERI specification, a RISC-V task group [8]. These capabilities can only be provided to system modules that implement some sort of error-checking HW. In fact, our ENGAGE-V module monitors the protected units while waiting for an HW error to be detected.

A. RERI Specification

First, the RERI taxonomy for the detected errors is provided, which classifies errors depending on their severity:

- *Corrected Error (CE):* if the implemented redundancy technique detects and autonomously corrects the HW error, so no additional action is required. Different error correction techniques can be implemented, for example, ECC codes in memories.
- *Uncorrected Error Deferred (UED):* if the detected error is not autonomously correctable but does not disrupt the ongoing system functionality. The system operation can progress, and the error management can be deferred to when the affected data is actually used. This may happen with uncorrectable errors detected during memory scrubbing/refreshing, since the erroneous data is not consumed by any "processing" module.
- *Uncorrected Error Critical (UEC):* if the detected error is not autonomously correctable and requires immediate error management. Note that an UED can be escalated to UEC when its management cannot be further deferred.

Clearly, error taxonomy cannot consider undetected errors by definition. From a RAS point of view, for instance, the UED category can maximize the availability and serviceability of the system in space applications. By logging the information on the detected UED without disrupting the system operations, it is able to provide the desired service minimizing the interruptions and the system failures. Even though this consideration is applicable to non-space classical systems, it is greatly magnified in the harsh space environment, which is notoriously characterized by higher error rates. Therefore, keeping trace of also the UEDs provides a more comprehensive approach.

Secondly, the heart of the HW-SW interface that provides the error logging feature are the *Error Record Banks*. In particular, the key characteristics are the following.

- A system can implement one or more error banks, accessible via memory-mapped means;
- Each error bank can implement up to 63 Error Records (ER), independently of the others;
- Each monitored HW unit can be connected to multiple ERs;
- Each ER requires a 64-byte addressing space (set of eight 64-bit registers).

Table I lists the available registers within a bank. The two main groups are separated by a continuous line: the Bank Header, which stores information on the state of that bank, and the array of N ER, each containing information on a detected HW error. An ER can store a variety of information on the detected error, such as error severity (CE, UED, or UEC), error code, CE counter, address (if any), timestamp, or other implementation-specific information. Also, the logging capabilities can be tuned for the ERs in a bank because the RERI specification only mandates some register fields. This flexibility allows designers to add these error logging and reporting features to their RISC-V-based systems, covering from tightly constrained to high-end computing systems for space applications.

TABLE I
ERROR RECORD BANK MEMORY-MAPPED REGISTER LAYOUT.

Address Offset	Register Name	Size (Bytes)	Description
0	`vendor_n_imp_id`	8	Vendor and implementation ID.
8	`bank_info`	8	Error bank information.
16	`valid_summary`	8	Summary of valid ERs.
24	Reserved	32	Res. for future standard use.
56	Custom	8	Designated for custom use.
$64+64\cdot i$	`control_i`	8	Control reg of ER_i.
$72+64\cdot i$	`status_i`	8	Status reg of ER_i.
$80+64\cdot i$	`addr_i`	8	Address reg of ER_i.
$88+64\cdot i$	`info_i`	8	Information reg of ER_i.
$96+64\cdot i$	`suppl_info_i`	8	Supplemental info reg of ER_i.
$104+64\cdot i$	`timestamp_i`	8	Timestamp reg of ER_i.
$112+64\cdot i$	Reserved	16	Res. for future standard use.

B. ENGAGE-V Architecture

As illustrated in Figure 1, the architecture of the developed ENGAGE-V module comprises the following main building blocks: Error Mux, Error Synchronization Interface, Logging Controller, Error Record Bank, and Interrupt Request (IRQ) Generator. Our ENGAGE-V module can monitor M HW units that have to be protected by error-checking HW (e.g., ECC protected memories, or even modules with DMR or TMR). We designed the peripheral in SystemVerilog to be fully compliant with the RERI specification, targeting RISC-V systems [7].

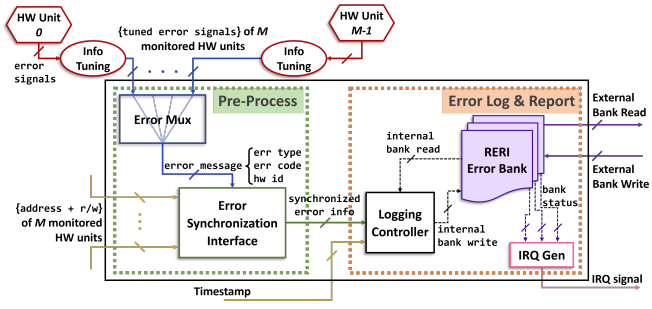

Fig. 1. Internal architecture of the developed ENGAGE-V module.

In summary, the architecture can be divided into two stages. The first one, *Pre-Process stage*, includes the Error Mux and the Error Synchronization Interface. It gathers and pre-processes the output of the M error-checking HW, which contains the information on the detected error. Note that the blocks *Info Tuning* contain minimum implementation-dependent combinational logic, which is fundamental to correctly interface each HW unit with the Error Mux.

This information is then interfaced via a circular FIFO (First-In First-Out) buffer to the second stage, *Error Log & Report stage*, which consists of the Logging Controller, the Error Record Bank, and the IRQ Generator. This stage collects and stores the pre-processed error information in one of the N ERs of the Bank. In particular, the Logging Controller determines in which ER to store the new information. If all ERs already contain valid information, it determines whether to discard the new one or to overwrite a valid record, depending on the severity and priority of the new error against the stored ones. In addition, the IRQ Generator triggers an interrupt signal according to the error log stored in the ER_i and the run-time configuration of its control register, `control_i`.

From a system perspective, multiple instances of our ENGAGE-V module can be implemented in the system, and each RAS hub can monitor a customized number of HW units (in line with the RERI specification). Some examples are shown in Figure 2, illustrating different combinations of M (number of HW units) and N (number of ER). It should be noted that each RAS hub will handle a single error bank and all its ER will have the same error logging capabilities.

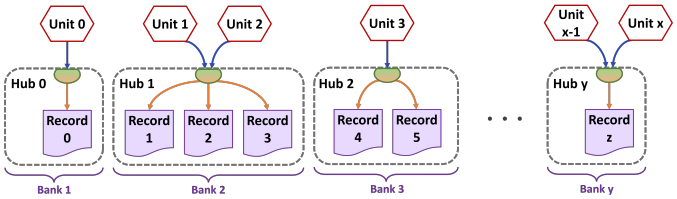

Fig. 2. Example of configurability of the ENGAGE-V modules monitoring different HW units and handling different ER.

By modifying the configuration parameters in the SystemVerilog wrapper module of the specific RAS hub, different trade-offs between hardware requirements and logging and reporting capabilities can be achieved. This configurability allows designers to provide well-balanced error logging and reporting features in any space-related system, as error-checking HW (e.g., information redundancy) is always implemented to ensure fault-tolerance and resilience.

III. Design Space of ENGAGE-V Module

The configuration of the single ENGAGE-V module is determined by three main variables within the design space, in addition to the number of instances to be integrated into the system ($N_{RAS\text{-}hub}$). The following three design parameters can be independently adjusted for each RAS hub:

- The depth of the FIFO buffer in the pre-processing stage (N_{FIFO}), fundamental for aligning the error data with the pertinent address, if applicable.
- The number of ER contained in the error bank of that RAS hub (N_{ER}).
- The register fields implemented in each ER within that RAS hub, concerning the provided error logging and reporting features (e.g., severity of error logged, CE counter, timestamp, etc.).

These variables depend on the number of HW units to be monitored ($N_{HW\text{-}unit}$) and the resource constraints of the system. Indeed, standard computing systems can be categorized as embedded, application, and HPC/cloud. In space applications, they are typically further categorized by their ability to address unique challenges such as radiation, extreme temperatures, and limited power. Therefore, we can consider that the more resilient and fault-tolerant the architecture of the system, the higher $N_{HW\text{-}unit}$ in total. In addition, depending on the target space application, the error rate may vary significantly. Once $N_{RAS\text{-}hub}$ has been determined, the remaining design variables can be selected for the different RAS hubs. In the following analyses, $N_{HW\text{-}unit}$ will refer to a single RAS hub.

A. Design Exploration of the RAS Hub

1) FIFO Buffer configuration: Firstly, the depth of the FIFO buffer can be determined depending on $N_{HW\text{-}unit}$ specifically monitored by that RAS hub. Since the *Logging Controller* cannot immediately retrieve the error information stored in the buffer, a reasonable curve for N_{FIFO} should also consider the conditional probability of distinct errors detected in consecutive clock cycles ($P_{Con\text{-}Err}$). In this situation, the buffer will have a temporary peak in the write rate (one per clock cycle), whereas the read rate will be limited to one every two clock cycles. This conditional probability rapidly decreases for each additional consecutive error that should be detected in one of the monitored HW units. However, it increases slightly for greater $N_{HW\text{-}unit}$. The following function traces some guidelines for the the estimation of a sufficient depth of the buffer:

$$N_{FIFO} = \delta + \lfloor \alpha \cdot \log_2(N_{HW\text{-}unit}) \rfloor \quad (1)$$

where α should be tuned according to the application environment and the respective HW error probability; $\delta \geq 2$ sets a minimum value for the FIFO buffer depth.

2) Error Records configuration: Secondly, the number of Error Records (N_{ER}) to implement within a single RAS hub should be determined. In this case, N_{ER} varies according to three main factors:

1) Number of HW units to monitor ($N_{HW\text{-}unit}$), since the higher this number, the higher the probability that a new HW error is detected and should be recorded.
2) Reducing N_{ER}, the likelihood that each ER in the bank already holds valid data increases, requiring either overwriting existing records or discarding new entries. To minimize information loss due to hardware errors, it is essential to limit the overwrite/discard rates.
3) The frequency at which the system SW can access the logged ERs, which can then log information of new errors, thus minimizing the overwriting/discarding events.

The function considered, $N_{ER} = f(N_{HW\text{-}unit}, \delta, \alpha)$, has the same expression as Equation 1, where α has an equivalent functional meaning, while δ now defines a minimum number of ERs that are deemed acceptable according to the application considered ($\delta \geq 1$).

3) Error Logging & Reporting configuration: Lastly, once the HW units to be monitored have been defined, the error logging and reporting capabilities of that RAS hub should be defined. This step is crucial to optimizing the resource requirements of the entire module. Not all bank registers

(listed in Table I) must be physically implemented with 64-bit registers to comply with the RERI specification. Specifically, each ER requires an equivalent number of 1-bit registers $REG_{1\text{-}bit} \in [18; 325]$. Therefore, designers may physically implement only the required register fields, thus optimizing the total number of registers REG_{TOT}; for example, if the CE counter is not required, the relative 16-bit field of the register `status_i` will not be physically implemented.

B. Impact of Design Parameters on System Resources

The design parameters discussed in the previous subsection affect primarily one of the two internal stages of the ENGAGE-V module. For the sake of clarity, we evaluated the impact of $N_{HW\text{-}unit}$ and N_{ER} on the resources required to implement a single RAS hub. Different syntheses have been performed with the NanGate Open-Cell library, designed with $45nm$ FreePDK. For all syntheses, the ERs have been configured to implement complete logging and reporting features.

Firstly, Figure 3 shows how the resource requirements vary as a function of $N_{HW\text{-}unit}$ for the entire RAS hub and the internal Pre-Process stage. Note that this parameter mainly affects this internal stage, since $total[i] - pre\text{-}process[i] \simeq 6$ kGE for any considered value of monitored HW units.

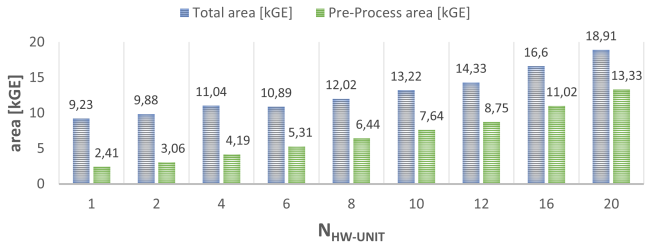

Fig. 3. Synthesis of a RAS hub with $N_{HW-unit} \in [1; 20]$ and $N_{ER} = 1$.

Similarly, Figure 4 illustrates how area occupation varies with N_{ER}, highlighting the area required for the Log & Report stage. It should be noted that this parameter affects not only this stage, but also the AXI4 logic required for the memory-mapped accesses. Indeed, the difference in the area between the two cases in Figure 4 increases with N_{ER}.

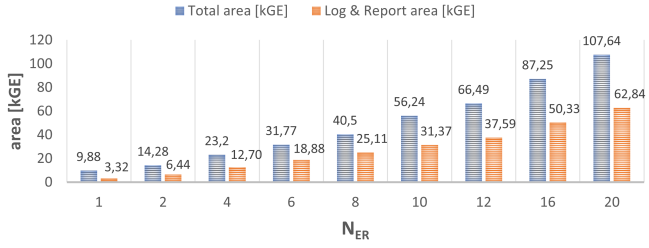

Fig. 4. Synthesis of a RAS hub with $N_{HW-unit} = 2$ and $N_{ER} \in [1; 20]$.

These synthesis results highlight that some parameters influence overall resource consumption more than others, for example, the case $N_{ER} = 4$ in Figure 4 requires a total of 23.2 kGE, whereas, even with $N_{HW\text{-}unit} = 20$ in Figure 3 the total area is only 18.91 kGE.

By adjusting these parameters, designers can tailor the ENGAGE-V module to meet the unique requirements of various space applications, balancing resource constraints with the need for robust error logging and reporting capabilities.

IV. Conclusion

In this work, we have outlined the architecture of our ENGAGE-V peripheral. It is a RERI-compliant error logging and reporting module tailored for RISC-V-based systems, which could increase system resilience in space applications. The ENGAGE-V module enhances the resilience and fault tolerance of computing systems by providing customizable and standardized error logging and reporting capabilities. Our approach addresses the unique challenges posed by the harsh space environment, such as cosmic radiation and extreme temperatures, which significantly increase the incidence of HW errors.

The modular and flexible nature of the ENGAGE-V module allows its integration into various space-related systems, from low-power embedded devices to high-end computing platforms. By implementing the ENGAGE-V module, designers can achieve a balanced trade-off between system resource overhead and the need for robust error management, ultimately extending the operational life and reliability of space missions.

Future research and development will focus on further optimizing the performance of the ENGAGE-V module and exploring additional fault-tolerance techniques to enhance the reliability of RISC-V systems in space. This includes investigating new redundancy approaches, improving error detection and correction mechanisms, and expanding the module's capabilities to support emerging space applications.

References

[1] L. Cassano *et al.*, "Is RISC-V Ready for Space? A Security Perspective," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, IEEE, 2022, pp. 1–6.

[2] N.-J. Wessman *et al.*, "De-RISC: the First RISC-V Space-Grade Platform for Safety-Critical Systems," in *2021 IEEE space computing conference (SCC)*, IEEE, 2021, pp. 17–26.

[3] D. A. Santos *et al.*, "A Low-Cost Fault-Tolerant RISC-V Processor for Space Systems," in *2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, IEEE, 2020, pp. 1–5.

[4] N. Koca *et al.*, "Exploring Error Correction Circuits on RISC-V based Systems for Space Applications," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2024, pp. 1–5.

[5] M. Barbirotta *et al.*, "Dynamic triple modular redundancy in interleaved hardware threads: An alternative solution to lockstep multi-cores for fault-tolerant systems," *IEEE Access*, 2024.

[6] A. M. P. Mattos *et al.*, "Using HARV-SoC for Reliable Sensing Applications in Radiation Harsh Environments," in *2023 9th International Workshop on Advances in Sensors and Interfaces (IWASI)*, IEEE, 2023, pp. 227–232.

[7] N. Canino *et al.*, "HW-SW Interface Design and Implementation for Error Logging and Reporting for RAS Improvement," *IEEE Access*, pp. 60 081–60 094, 2024.

[8] *RERI (RAS Error-record Register Interface) task group*. [Online]. Available: https://lists.riscv.org/g/tech-ras-eri.