Enhancing Maintainability and Reliability in Space Applications: A RISC-V Based System Controller for High-Performance Data Processing Units for Small Satellites Using LiteX

Daniel Garbe
Fraunhofer Institute for High-Speed Dynamics, Ernst-Mach-Institut, EMI
Freiburg, Germany
daniel.garbe@emi.fraunhofer.de

Clemens Horch
Fraunhofer Institute for High-Speed Dynamics, Ernst-Mach-Institut, EMI
Freiburg, Germany
clemens.horch@emi.fraunhofer.de

Konstantin Schäfer
Fraunhofer Institute for High-Speed Dynamics, Ernst-Mach-Institut, EMI
Freiburg, Germany
konstantin.schaefer@emi.fraunhofer.de

Abstract – The presented System Controller (SC) is based on an implementation of failure detection, isolation, and recovery (FDIR) for a high-performance data processing unit (DPU) in small satellites. The SC monitors and isolates affected components in case of abnormal behavior. To enhance reliability, the design shifted to a RISC-V approach implemented on an FPGA. Built-in error correction and upset management features protect against radiation effects, while FDIR measures focus on critical components such as GPIOs and data interfaces. With LiteX, a Python-based SoC framework, rapid development of configurable interfaces has been achieved. The FPGA-based design improves response times and reliability through autonomous error-handling reactions.

Keywords – System Controller, FDIR, Redundancy, COTS, FPGA, LiteX, Softcore, RISC-V

I. INTRODUCTION

The high-performance data processing unit (DPU) by Fraunhofer EMI [1], based on a heterogeneous MPSoC, includes redundant data storage and interfaces. While the SRAM-based FPGA features single-event effect (SEE) mitigation for configuration memory (CRAM) protection, it does not cover permanent effects, transient issues, or upsets affecting internal states. This necessitates additional techniques like dual or triple modular redundancy (DMR or TMR), error correction codes (ECC), interleaving, and software redundancy, which are discussed in this paper's motivation.

To enhance the reliability of the DPU, a distributed FDIR approach is implemented through a synergistic collaboration between the DPU and its System Controllers (SC), cf. Figure 1. Each SC interacts with its MPSoC and other SC, enabling both, a hot and a cold redundancy. By monitoring power supply rails and operational states, the SC can react autonomously or software-controlled to error syndromes with user-defined actions. Syndromes and actions are in-space configurable, ranging from isolation and recovery of certain components or subsystem by power cycling, up to swapping operations to the redundant DPU instance. This integration ensures a more robust response to faults and enhances overall system reliability, provided interfaces for configuration, measurements and control signals are fault tolerant. To achieve this, FDIR of the SoC and CPU focusses on GPIOs and data interfaces.

In small satellites commercial-off-the-shelf (COTS) products are commonly found. A previous SC design of the DPU used a low-power microcontroller unit (MCU) with COTS interfaces and therefore lacked the required flexibility, maintainability and most important immediate and guaranteed reaction to faults. A shift to a low-power Certus-NX FPGA with SEE-protected CRAM and a soft MCU aligns better with the DPU's demands, allowing for self-monitoring extensions and improved reliability.

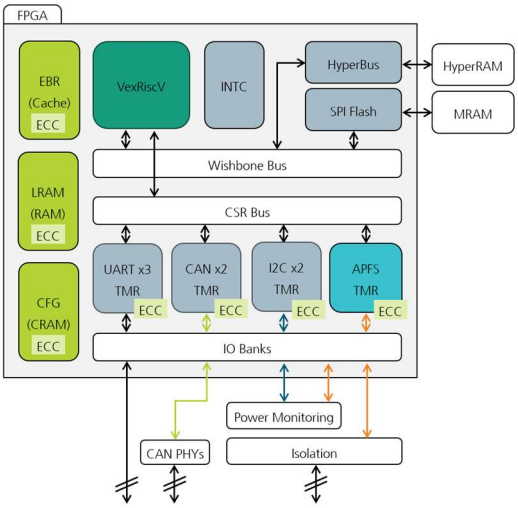

Figure 1 SC SoC design

RISC-V, as an open instruction set architecture (ISA), enabled the creation of customizable and scalable softcore ecosystems, such as VexRiscV (SpinalHDL), making it a favorable option for DMR/TMR designs and space applications [2]. The Python-based hardware-description (HDL) framework LiteX [3] supports various RISC-V implementations, including VexRiscV with support for the real-time operating system Zephyr, and

facilitates simplified migration across platforms. Despite these advantages, implementing modular redundancy and other FDIR techniques requires careful analysis and choice of measures, motivating the FDIR concept explored in this paper.

II. Motivation and Requirements

Protecting the DPU and itself from permanent damage, requires autonomous reactions with high reliability. For commanding and housekeeping services, interfaces pose equally high priority for configuring rail monitors, controlling power and signal switches, monitor readings and behavioral analysis via CAN/UART. It becomes clear, that interfaces, reaction management and configuration are crucial parts of SC functionality, thus requiring FDIR and if possible, redundancy. Shifting to a FPGA-based design parallelization in hardware of protection tasks, interface handling and software services, frees computational resources for more elaborated FDIR tasks,

Existing FDIR mechanisms [4] [5] [6] face challenges with LiteX. Although LiteX supports symmetric multiprocessing (SMP), it is still experimental and relies on pre-generated Verilog cores. This necessitates post-synthesis and specialized design tools, like Synplify [7] [8] [9]. for implementing fine- and coarse-grained DMR/TMR approaches. Integrating these tools or modifying pre-generated cores may require extra maintenance due to LiteX's modular design and increase FPGA resource demands [5]. Furthermore, interlock hazard circuitry in many LiteX cores will require updating the core pipeline with modifications to the pre-generated Verilog.

LiteX's HDL is based on modular classes and inheritance. Two major interconnect-design patterns within the SoC design are control and status registers (CSR) and Wishbone. The quest for long-term maintainability has been identified in augmenting these concepts, without relying on hardware-specific libraries.

Encouraged by radiation reports [10] of the chosen FPGA platform, the presented approach embraces ideas from instruction mimicking [6], interleaving threaded redundancy [11] and software diversity [12] playing along with enhancing LiteX design patterns.

Existing modules are inheriting from a custom LiteX TMR class. Applying to interfaces, state machines and GPIO in combination with password protected CSRs configuring these by the softcore needs to be fault tolerant. With TMR in hardware being out of scope for LiteX SoCs, a shift to virtualizing TMR concepts was taken. As granularity requires each thread locks after certain tasks by counting semaphores, which yield task sensitive contexts. At each lock step a software majority compares and restores erroneous thread contexts. Whenever interfaces or GPIOs are modified, a single majority vote thread could misbehave. Here, a triple majority vote and triplicated CSR architecture helps to reduce error propagation. While intermittent errors occurring in one thread are detected, multiple execution of the same code is prone to permanent errors. Adopted from instruction mimicking, predefined inputs are processed by a reference code and compared to known results and behavior. Cross referencing with execution on associated MPSoC and other SC cores, this yields a distributed hybrid approach to the FDIR concept.

III. Fault Analysis

A fault and risk analysis has been conducted for fabric, mass storage devices, peripherals and power rails involved.

As only the fabric is powered by the core voltage of 1.0V, mass storage with 1.8V and any other peripheral by 3.3V, power rail monitoring is greatly simplified, as priorities, error syndromes and actions are grouped without many different consumption and IO behaviors interfering.

By splitting the radiation caused faults into the commonly known SEEs, we derive design decision for the proposed implementation. Single-event upsets (SEU) and multi-bit upsets (MBU) effects configuration and internal memory states. The FPGA's FDIR can handle single errors and detect double errors in CRAM and memory states, when located in EBR or LRAM. This is beneficial for registers and FIFO implementations which are very commonly used by CSR and Wishbone modules. External memory either needs to be ECC protected or intrinsic radiation tolerant. In Chapter IV, we will explore how this can be evaded to a certain extent. Critical peripherals like power monitors, load and signal switches need other measures, i.e. plausibility check and supervising behavior.

Digital single-event transients (DSET) can be seen as transmission error between memories and registers. With faster bus clocks, transients begin to transform into SEU propagating through combinatorial logic [13], falsely encoded then by ECC. Therefore, replacing typical methods as electrical, logical or temporal masking [14] with an ECC augmented bus design can reduce the susceptibility and analysis uncertainty, freeing EBR resources.

Analog transients (ASET) in the scope of this paper are considered to affect just the power monitors. These transients can be overcome by oversampling and employing plausibility tests against measurements in the path of the power supply tree model. If not recoverable, immediate or graceful mitigation decision based on severeness can be configured to take place.

Single-event function interrupts (SEFI) on the other hand are considered pertaining complex interface logic and cores. As such protocol verifying transmitted

messages and software measures as discussed in Chapter II are mandatory.

Any hard SEEs like single-event latch-ups (SEL) can be understood as not recoverable. An increased SEU detection rate or unrecoverable error detection by SECDEC in CRAM implies the necessity for power cycling. Damage to the fabric or memories by i.e. single-event burnout (SEB) or gate rupture (SEGR) can lead to constant triggering of the internal FDIR mechanism. In consequence reconfiguration keeps failing and is unrecoverable in most cases.

IV. Methods and Implementation

Equipped with the results from the fault analysis, a conceptional FDIR architecture has been elaborated, which is presented in this chapter.

Interleaving threaded redundancy depends on reliable majority votes over results produced for the same task by different threads. Counting semaphores and locks are used to step through the thread's machine code. In each step the thread context is accessed by a voter, determining source address and length for comparison. Contexts with the overlapping memory regions are generally not allowed, as this corrupts the independent computation assumption.

A single voting thread is the least safe option but is the fastest voting in software. Minimizing corruption three voting threads which update their associated context at each lock step, render this approach safer but are complex and introduce higher latency, especially if inter-processor lock steps are used. Depending on the task errors, dead locks or uncertain delays in function calls may cause indeterministic behavior. In this case a timeout for lock step should be employed by the developer.

Furthermore, a special context for inter-processor locks is envisioned, as these require communication before voting can be conducted. This context is used in tasks concerning cross-check computations with other SCs' cores and architectures like the MPSoC. Together with reference input and implementations, crucial parts of the core functionalities, like branching, arithmetic operations and memory access are supervised, indicating abnormal modes.

From hardware design perspective, task interactions with interfaces and GPIO are transfers between core and CSRs or Wishbone FIFOs. CSR interconnect is used i.e. by GPIO, UART, SPI and CAN LiteX modules. HyperRAM and SPI Flash modules are accessed by Wishbone FIFOs and CSR. The majority vote for CSR interactions is hardware-accelerate, requiring triplication of registers. As Wishbone is reserved to high throughput interfaces, in which case triplicating FIFO stages is not feasible.

Implementing TMR in LiteX is very similar to its Verilog counterpart. However, using Python allows for recursive modification and application of the TMR pattern to submodules, except for FSM, which LiteX inherits from an underlying HDL Python library proving to be non-trivial. Depending on the used toolchain synthesis constraints may be required to avoid optimization of redundant logic to a single instance. As LiteX supports various toolchains already, the necessary abstraction layer to add these constraints is already prepared.

LiteX CSR registers do not by default offer ECC but supplies the necessary logic already. Replacing the default implementation with a custom by default ECC enabled one, all LiteX modules are using ECC protected CSR register automatically. Again, LiteX maintains a target (FPGA derivate) aware infrastructure, enabling for overrides. By this it is possible to specify, to use for example the ECC protected EBR of the chosen FPGA instead.

Similarly, LiteX's Cache class is overwritten, which is optionally used by the Wishbone interfaces. Again, either generated or by the FPGA's block RAM provided ECC protection can be employed. As SDRAM (L2-Cache located in Wishbone interface), HyperRAM and SPI flash implementations in LiteX rely on Wishbone interface and therefore underlying unprotected FIFOs, thus usefulness of caches are greatly reduced in the current concept and system specification.

With regards to synchronous logic, clock glitches and failures are of importance. As these can desynchronize TMR design with internal states, re-synchronization or reset circuitry is needed. At the time of writing no re-synchronization concept generally applicable in LiteX has been found and is implemented whenever necessary manually.

Clock sources also affect how other important fail-safe mechanisms like watchdogs (WDT) perform. Therefore, a clock supervisor module has been introduced consisting of TMR watchdogs for each hardware base clock available (external 12 MHz oscillator and internal oscillator). If any clock cycles are missing or glitches occur, w. r. t. the other clock, an error signal and status register flag is set. Elongated malfunctions are detected by a maximum allowable number of deviations per defined period referenced to the other clock. This allows to distinguish between temporal and permanent clock failure.

Based on the previous SC design and experiences an autonomous power and fault supervisor module is presented. Although, clocked by any active clock available, error syndromes requiring immediate action are unaffected by clock failures. Error syndromes are fault signal vectors, comprising of external alert pins and internal fault signals. Bounding the number of known error syndromes to a power of two, Python libraries can be used to construct a divide and conquer based severeness arbitration to device which action to take

immediately or enqueue into a FIFO. Each error syndrome register has a corresponding severeness register and action register.

Arbitration for the priority queue: most severe syndrome in current path wins. Action is written to action register, controlling power rails and other control signal GPIOs. A masking register permanently shuts down for example damaged components.

Severeness is implemented as integer vector. Defining lowest value has highest severeness, allows for the convention to structure the severeness bits into group numbered 1 to 3, staring from LSB:

1. Response Timing: 0 – Immediate 1 – Signal, Delayed Action, 2 – Signal and Await Response with Timeout, 3 – RFU
2. Priority lowest value has highest priority
3. Path depth: components effecting leave-node components prioritized

Hence, fault affecting many subsequent can have highest severeness, resembling the structure from failure mode and effects analysis tree, subdivided by priorities.

If no syndrome requires an immediate action, arbitration is used to fill the priority queue. This process is clocked and may fail if no clock is present. Otherwise, arbitration overrides any other non-immediate action, by-passing the FIFO. Here arbitration is or'ing action bits disabling power and signal switches.

V. Conclusion and Future Work

A concept for enhanced reliability and maintainability by using design patterns based on LiteX has been proposed in context of the presented SC. Reusing FDIR mechanism for self-protection of the SC, similarities to the actual to be protected MPSoC infer and enhancing a synergetic usage of these mechanisms, analog to monitoring power rails of the MPSoC and SC itself. By choosing a FPGA based RISC-V architecture over existing radiation-hardened processors, an independency of FDIR measures to software and hardware faults of the processor is achieved.

Modifications to the LiteX framework has been depicted and tested in case-studies which uncovered open challenges arising from applying TMR FDIR techniques to the existing framework's module classes.

Proceeding steps include an implementation of these remaining challenges, to fully review and analyze the reliability of the proposed SC and FDIR design.

References

[1] C. Horch, D. Garbe and K. Schäfer, "Redundant imaging payload data processing system based on a heterogeneous MPSoC," *2023 European Data Handling & Data Processing Conference (EDHPC),* pp. 1-4, 2023.

[2] R. Weigand, "RISC-V First Steps Into Space," *15th ESA Workshop on Avionics, Data, Control and Software Systems,* 2021.

[3] F. Kermarrec, S. Bourdeauducq, J.-C. Le Lann and H. Badier, "LiteX: an open-source SoC builder and library based on Migen Python DSL," May 2020. [Online].

[4] E. M. Aguilar, F. Benevenuti and F. L. Kastensmidt, "Hardening a RISC-V Softcore for Embedded Aerospace Applications in SRAM-based FPGA," *2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI),* pp. 1-5, 2024.

[5] N. A. Koca, C. H. Chang, A. T. Do and V. P. Nambiar, "Exploring Error Correction Circuits on RISC-V based Systems for Space Applications," *2024 IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 1-5, 2024.

[6] A. Sreekumar, B. S. Shankar and B. N. K. Reddy, "Integrating error correction and detection techniques in RISC-V processor microarchitecture for enhanced reliability," *Integration,* vol. 100, p. 102282, 2025.

[7] S. Habinc, "Functional Triple Modular Redundancy (FTMR) - VHDL Design Methodology for Redundancy in Combinatorial and Sequential Logi," European Space Agency, 2022.

[8] "Using Synplify to Design in Microsemi Radiation-Hardened FPGAs - Application Note AC139," May 2012. [Online]. Available: https://ww1.microchip.com/downloads/aemdocuments/documents/fpga/ApplicationNotes/ApplicationNotes/synplifyrh_an.pdf. [Accessed 19 March 2025].

[9] A. E. Wilson and M. Wirthlin, "Fault Injection of TMR Open Source RISC-V Processors using Dynamic Partial Reconfiguration on SRAM-based FPGAs," *2021 IEEE Space Computing Conference (SCC),* pp. 1-8, 2021.

[10] A. Wilson, S. Larsen, C. Wilson, C. Thurlow and M. Wirthlin, "Neutron Radiation Testing of a TMR VexRiscv Soft Processor on SRAM-based FPGAs," *IEEE Transactions on Nuclear Science,* vol. PP, pp. 1-1, 2021.

[11] 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," *Journal of Low Power Electronics and Applications,* vol. 13, no. 1, 2022.

[12] T. Lovric, "Systematic and design diversity - Software techniques for hardware fault detection," *Dependable Computing - EDCC-1,* pp. 307--326, 1994.

[13] V. Ferlet-Cavrois, L. W. Massengill and P. Gouker, "Single Event Transients in Digital CMOS—A Review," *IEEE Transactions on Nuclear Science,* vol. 60, no. 3, pp. 1767-1790, 2013.

[14] ESCC, "ECSS-E-HB-20-40A – Engineering techniques for radiation effects mitigation in ASICs and FPGAs handbook," 11 October 2023. [Online]. Available: https://ecss.nl/wp-content/uploads/2023/10/ECSS-E-HB-20-40A(11October2023).pdf. [Accessed 25 March 2025].

[15] SpinalHDL, "GitHub SpinalHDL/VexRiscV," [Online]. Available: https://github.com/SpinalHDL/VexRiscv. [Accessed 21 March 2025].

[16] J. M. Johnson and M. J. Wirthlin, "Voter insertion algorithms for FPGA designs using triple modular redundancy," *FPGA '10: Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays,* pp. 249 - 258, 2010.