Reliability in High-Performance Computing: Insights from a RISC-V Vector Processor

Marcello Barbirotta[1], Francesco Minervini[2], Carlos Rojas Morales[2], Adrian Cristal[2], Osman Unsal[2] and Mauro Olivieri[1] *

[1]Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome 00184, Italy
[2]Barcelona Supercomputing Center, Barcelona 08034, Spain

Abstract

High-Performance Computing (HPC) systems are designed for large-scale processing and complex data analysis, utilizing scalability, efficiency, and parallelism, often with specialized hardware like Vector Processing Units (VPUs). RISC-V plays an interesting role in this context for its inherent extendability and the availability of open-source microarchitecture designs. Still, as these systems become more complex, their susceptibility to errors and failures poses significant challenges. Our research addresses this by implementing advanced fault tolerance techniques in the Vitruvius+ architecture, a partial out-of-order RISC-V VPU. Notably, we present the first full RTL-level implementation of instruction replication in an HPC-class vector processor for reliability. We explore redundancy mechanisms in critical architectural units, achieving a 75% reduction in non-silent faults leading to system failure, supported by extensive fault injection simulations, with only a 7.5% hardware overhead and minimal clock frequency variation.

Introduction

RISC-V is an open-source instruction set architecture (ISA) developed and designed to be highly flexible and customizable, allowing various implementations across various hardware platforms. One of the areas where RISC-V is particularly important is Fault Tolerant (FT) computer architecture. These techniques that emerged during the space age to address faults from cosmic rays have become crucial in designing current and future High-Performance Computing (HPC) systems, as they operate in parallel across thousands of nodes. Due to high hardware complexity and performance demands, these systems often rely on software-level FT techniques, like software checkpointing with rollback recovery [1], which is complex due to its integration with libraries, programming models, and data storage [2]. These methods can lead to significant performance drops, with latency overheads reaching 6X - 8X [3]. As a result, software-level replication has been used to ensure application reliability despite individual component failures, proving more efficient than extensive checkpointing in petascale and exascale systems. However, it still incurs substantial overhead. This work explores hardware support for instruction-level replication within HPC cores, which aims to mitigate this overhead while protecting critical components like the vector processor pipeline and floating-point units from soft errors. The Vitruvius+ Architecture [4] is compliant with RISC-V vector extension (RVV version 1.0) and is a decoupled vector accelerator with lightweight out-of-order execution capabilities boosted by vector register renaming and concurrent execution of memory and arithmetic instructions. Vitruvius+ mainly targets HPC applications characterized by long vectors; it supports up to 16384 bits in a vector register, and it can manage any vector length of up to 256 Double-Precision (DP) 64-bit floating-point elements.

*Corresponding author: marcello.barbirotta@uniroma1.it

Methodologies

The Fault Tolerant RISC-V Vitruvius+ Architecture [5], depicted in Figure 1, maintains the same vector characteristics of the original version, plus some modifications and adjustments. The approach is based on temporal redundancy in conjunction with ECC. Generally speaking, the main advantage of temporal redundancy (instruction replication) techniques is related to the almost null hardware overhead at the expense of time overhead. The green blocks in the Figure 1 are related to the temporal redundancy hardware support, which implements the Instruction Duplication for all the arithmetic instructions within the Front End and the Reorder Buffer blocks. Every duplicated instruction (called EVEN and ODD) is written inside the ECC buffer as the parity code word produced by the logic. If the ECC comparison reports a mismatch, the instruction is not committed, and a kill / roll-back hardware procedure is activated to recover the status of the VPU before the execution of the faulty instruction.

The ECC calculation (red blocks in Figure 1) gives the advantage of protecting the VRF and comparing the calculated values of each replicated instruction result before writing it in the VRF rather than comparing the whole replicated result vectors. To reduce potential performance issues from latency in error correction and data rewriting in the VRF, the rewriting can be delayed while immediately using corrected data in subsequent instructions. For VRF correction, the

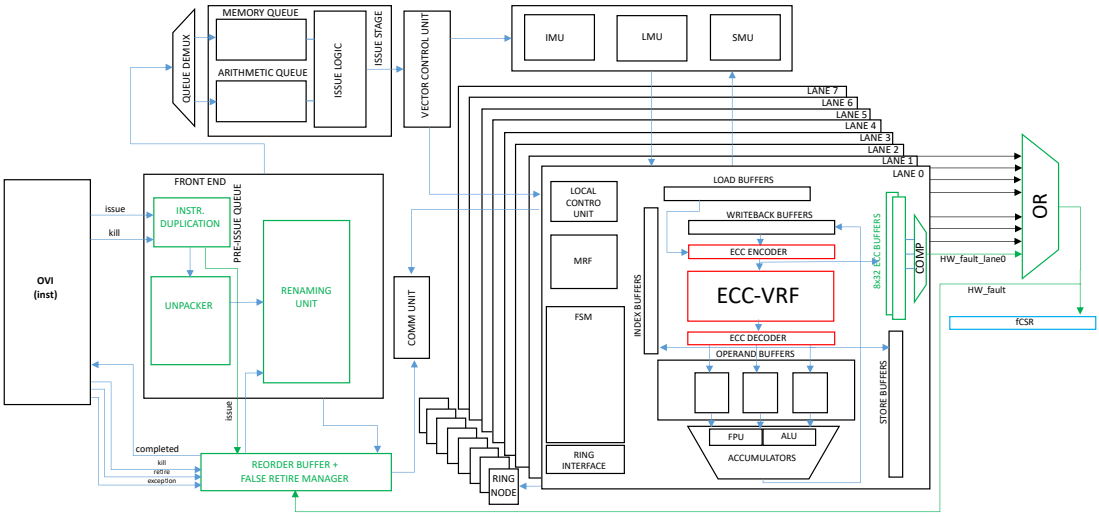

Figure 1: *Fault tolerant - Vitruvius+ Architecture*

error signals are counted in a custom CSR. When the count reaches a set threshold, the OS activates a software routine to recover the faulty data by performing a read-write cycle.

Impact on Hardware Resources

The synthesis results are depicted in Table 1. In the typical corner case (TT / 0.80 V / 25°C), the fault-tolerant design achieved a maximum frequency of 1.47 GHz, slightly lower than the 1.51 GHz of the original design. In the slow corner case (SS / 0.72 V / 125°C), the frequencies were 1.04 GHz and 1.08 GHz, respectively. The area occupation results show a hardware overhead of 7.5% for the fault-tolerant design, primarily due to the implementations in the Register File (6.83%) and instruction duplication (0.67%). The single lane, particularly the FPU and VRF, consumes most area, with only 10% of VPU signal distribution allocated to non-lane blocks, making lane faults more probable during fault injection simulations.

Table 1: *Synthesis results using Cadence Genus Synthesis Solution 19.11 on the 8T GF22FDX library.*

	VPU Area	VPU Frequency (typ)	VPU Frequency (slow)
No ECC - No HW-ID	1.49 μm^2	1.51 GHz	1.08 GHz
ECC - HW-ID	1.61 μm^2	1.47 GHz	1.04 GHz

Fault Injection Results

The fault injection test campaign was done using a UVM environment able to collect four different types of information by means of a comparison with the RISC-V Instruction Set Simulator Spike. It was chosen to analyze only the 0-index bit of all the signals belonging to Lane 0, for a total of approximately 3400 faulty bits per simulation run with an average of 1 fault every 300 clock cycles to avoid multiple faults hitting two copies of the same instruction (EVEN and ODD). The introduced features reduce the occurrence of non-silent undetected faults that would result in application failure by 75%.

Conclusion

This work presents the design and validation of a fault-tolerant RISC-V VPU for HPC systems, building on the Vitruvius+ architecture, and incorporating instruction-level hardware Temporal Redundancy. This design shows a 75% reduction in non-silent undetected faults and improves upon traditional software checkpointing methods with only a 7% area overhead and minimal performance degradation due to instruction duplication. The computational overhead from instruction duplication impacts performance primarily for latency-sensitive and throughput-sensitive tasks. In the worst-case scenario with all arithmetic instructions, execution time could double, yet this overhead remains stable in multi-node setups.

References

[1] Jack Dongarra, Thomas Herault, and Yves Robert. *Fault tolerance techniques for high-performance computing*. Springer, 2015.

[2] Franck Cappello. "Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities". In: *The International Journal of High Performance Computing Applications* 23.3 (2009), pp. 212–226.

[3] Odorico M Mendizabal, Fernando Luís Dotti, and Fernando Pedone. "Analysis of checkpointing overhead in parallel state machine replication". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 2016, pp. 534–537.

[4] Francesco Minervini et al. "Vitruvius+: an area-efficient RISC-V decoupled vector coprocessor for high performance computing applications". In: *ACM Transactions on Architecture and Code Optimization* 20.2 (2023), pp. 1–25.

[5] Marcello Barbirotta et al. "Enhancing Fault Tolerance in High-Performance Computing: A Real Hardware Case Study on a RISC-V Vector Processing Unit". In: *IEEE Open Journal of the Computer Society* 5 (2024), pp. 553–565. DOI: 10.1109/OJCS.2024.3468895.