Adding Dynamic Triple Modular Redundancy on a RISC-V Microarchitecture

Marcello Barbirotta[1]*, Abdallah Cheikh[1], Antonio Mastrandrea[1], Francesco Menichelli[1], Marco Ottavi[2,3] and Mauro Olivieri[1]

[1] Department of Information Engineering, Electronics and Telecommunications (DIET), "La Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy
[2] Department of Electronic Engineering, "Tor Vergata" University, Via del Politecnico 1, 00133 Rome, Italy
[3] Faculty of Electrical Engineering, Mathematics and Computer Science, "University of Twente", Drienerlolaan 5, 7522 Enschede, The Netherlands

Abstract

Fault Tolerance represents an important area for digital applications, so it has received recent acceleration in its development and evolution. Being able to understand how to protect electronic circuits, and in particular microprocessors, from the different types of SEE (Single Event Error) faults, frequent and internally divisible into other categories, is a very complex process [1], which sees the study and consequent implementation of these techniques for the hardware/software protection of the architectures under examination, making them more expensive and less performing than the respective non-redundant architectures. Safety and Reliability are, therefore, two key concepts in the technological world, and RISC-V plays an interesting role in this context for its inherent extendability and the availability of open-source microarchitecture designs.

Introduction

Among the various FT techniques developed, we find software methodologies, Double Modular Redundancy (DMR) and Triple Modular Redundancy (TMR) hardware methodologies, temporal redundancy techniques, and hybrid techniques based on previous ones. All techniques in the literature protect circuits by acting at different levels [2], with interesting research topics related to optimization, by inserting multiple circuits in the same circuit to obtain modular architectures with the required trade-off between performance and cost. 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. It is gaining increasing attention in computer architecture due to its open nature, flexibility, and potential for low power consumption. For the same reasons, one of the areas where RISC-V is particularly important is fault-tolerant computer architecture. Fault tolerance is critical in applications where system failures, such as space exploration, automotive systems, and military and safety-critical applications, can have serious consequences. RISC-V flexibility and customizable design make it well-suited to developing fault-tolerant systems. It enables developers to tailor the architecture to meet the application's specific needs allowing the creation of highly specialized systems that can withstand harsh environmental conditions and maintain functionality even in the event of system failures. This work summarizes the idea presented in [3], centered on the application of the DMR paradigm within an Interleaved Multi-Threading (IMT) RISC-V architecture, gaining the low overhead advantages of the DMR technique, and yet overcoming the cost of saving checkpoints and restoring the software state using Dynamic TMR (DTMR) protection, which actually implies the behavior of a TMR only in the case of error detection. This work also demonstrates the concept of Dynamic TMR and how it can be applied to an existing RISC-V IMT core, opening to performance evaluation and future fault-injection (FI) simulation campaign.

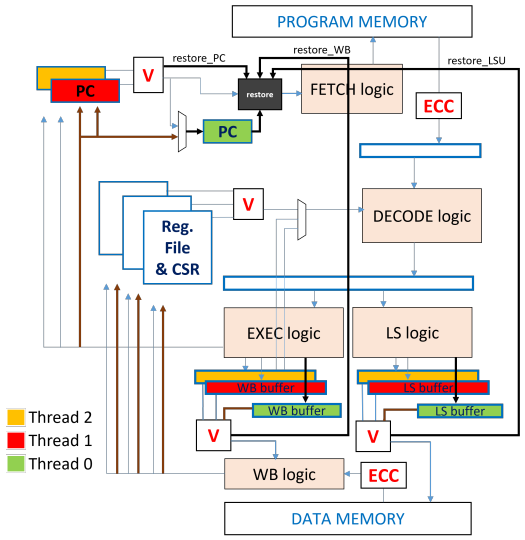

Figure 1: *Klessydra-dfT03 microarchitecture. Blue arrows: Normal mode; black arrows: Restore mode; brown arrows: End Restore Phase.*

*Corresponding author: marcello.barbirotta@uniroma1.it

Methodologies

Klessydra-fT03 [4] [5] is a fault-tolerant RISC-V processor core that uses an Interleaved Multi-Threading (IMT) architecture as a basis for the implementation of a radiation hardening technique called Buffered TMR, using its intrinsic spatial redundancy and temporal redundancy without adding additional memory locations to save data produced by the redundant instructions. Klessydra-fT03 is based on an open-source RISC-V softcore family, namely Klessydra-T, which interleaves three or more hardware threads in a round-robin fashion on a four-stage in-order pipeline that is fully compatible with the PULPino open-source microcontroller platform [6] [7]. The principle of the Dynamic TMR is the implementation of a DMR instead of a triple one without adding overhead for recovery and checkpointing mechanisms typically visible in these environments, turning the Buffered TMR into a DMR technique to reduce power consumption and increase speed by leveraging the multi-threaded architecture to use only two replicated threads instead of three. From that idea, we built a new core named Klessydra-dfT03 (microarchitecture in Figure 1), where "d" stands for dynamic, and we introduced a single register that saves the address of the last correct instruction and restores it in the PC with a latency overhead of four clock cycles, without any changes in the data memory or the Register File [3]. We use three hardware threads, numbered Thread 2, Thread 1, and Thread 0 (blue, red, and green colours in Figure 1), and we leave only threads 2 and 1 active while turning off Thread 0, which we call the auxiliary thread that is activated only in case of fault detection, and does not take part in the pipeline's normal operations fetching instructions, having no dynamic consumption for its dedicated hardware units. The operation are organized into three modes [3]:

- **Normal or "*Buffered DMR*" mode:** Threads 2 and 1 work in interleaved mode (blue arrows in Figure 1), executing the same instructions and thus implementing spatial and temporal redundancy, with a *buffered* voting mechanism implemented in the critical units PC, Register File, Write Back unit, and Load Store Unit, that check for the correctness of the program execution.
- **Restore or Recovery mode:** If the voting logic gives a negative result due to a fault, specific control signals named *restore_* signals (Figure 1) are asserted, and the core enters the recovery mode. Notably, a fault is always detected before the Register File would be updated with a wrong result using the faulted instruction. Following the black arrows in Figure 1, the *restore_* signals activate the Restore block (black unit in Figure 1), which wakes up the auxiliary sleeping thread. As the new thread enters the IMT pipeline, it fetches the last successfully executed instruction indicated by the dummy PC register (see next section), while the other threads are stalled.
- **End of Restore Phase:** Once the recovered instruction is completed, the produced result is compared with the results previously produced by the other two mismatching threads (brown arrows in Figure 1), thus obtaining a majority voting similar to a TMR system, and writing back the correct value into the Register File. The recovery procedure ends with the suspension of the auxiliary Thread 0, and the loading of the address of the next instruction in the PCs of Threads 2 and 1, so that they restart from the instruction following the one that faulted.

Discussion

In this work, we discussed the creation of a new hardening-by-design DTMR technique starting from an open-source IMT RISC-V microprocessor core, exploiting the advantages of both DMR and TMR techniques obtaining a system that can dynamically switch from one to another in case of faults [3].

References

[1] AH Johnston, GM Swift, and DC Shaw. "Impact of cmos scaling on single-event hard errors in space systems". In: *1995 IEEE Symposium on Low Power Electronics. Digest of Technical Papers*. IEEE. 1995, pp. 88–89.

[2] Alessandro Bernardini, Wolfgang Ecker, and Ulf Schlichtmann. "Where formal verification can help in functional safety analysis". In: *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM. 2016, pp. 1–8.

[3] Marcello Barbirotta et al. "Evaluation of Dynamic Triple Modular Redundancy in an Interleaved-Multi-Threading RISC-V Core". In: *Journal of Low Power Electronics and Applications* 13.1 (2023). ISSN: 2079-9268. DOI: 10.3390/jlpea13010002. URL: https://www.mdpi.com/2079-9268/13/1/2.

[4] Marcello Barbirotta et al. "Design and Evaluation of Buffered Triple Modular Redundancy in Interleaved-Multi-Threading Processors". In: *IEEE Access* 10 (2022), pp. 126074–126088.

[5] Marcello Barbirotta et al. "A Fault Tolerant soft-core obtained from an Interleaved-Multi-Threading RISC-V microprocessor design". In: *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE. 2021, pp. 1–4.

[6] Davide Rossi et al. "PULP: A parallel ultra low power platform for next generation IoT applications". In: *2015 IEEE Hot Chips 27 Symposium (HCS)*. IEEE Computer Society. 2015, pp. 1–39.

[7] Vladimir Herdt et al. "Extensible and configurable RISC-V based virtual prototype". In: *2018 Forum on Specification & Design Languages (FDL)*. IEEE. 2018, pp. 5–16.