(Article begins on next page)

Real-Time Dynamic Hardware Reconfiguration for Processors with Redundant Functional Units

Randolf Rotta, Raphael Segabinazzi Ferreira, Jörg Nolte
Brandenburg University of Technology, Cottbus, Germany
{rottaran, R.SegabinazziFerreira, Joerg.Nolte}@b-tu.de

***Abstract*—The tiny logic elements in modern integrated circuits increase the rate of transient failures significantly. Therefore, redundancy on various levels is necessary to retain reliability. However, for mixed-criticality scenarios, the typical processor designs offer either too little fault-tolerance or too much redundancy for one part of the applications. Amongst others, we specifically address redundant processor internal functional units (FU) to cope with transient errors and support wear leveling. A real-time operating system (RTOS) was extended to control our prototypical hardware platform and, since it can be configured deterministically within few clock cycles, we are able to reconfigure the FUs dynamically, at process switching time, according to the specified critically of the running processes. Our mechanisms were integrated into the Plasma processor and the Plasma-RTOS. With few changes to the original software code, it was, for example, possible to quickly change from fault-detecting to fault-correcting modes of the processor on demand.**

***Keywords*—Reconfiguration, Run-Time, Modular Redundancy, Mixed-Criticality.**

I. Introduction

The demand for higher computing throughput, lower power and energy consumption necessitates smaller manufacturing technologies for the processors. This, however, increases the impact of aging effects and the susceptibility to external interference such as radiation, electromagnetic noise, and voltage fluctuations, effectively limiting the expected life-time of the processors and its reliability.

At the same time, due to the ever growing amount of ”smart” software-controlled components and embedded logic in modern machinery, there is a desire for consolidation of multiple applications into fewer processors and hardware components. This mix of applications comes with a wide range of reliability and performance requirements. In this context, less critical tasks might afford to fail without detecting any fault or can afford the run-time overhead of restarting the application to recover from detected faults. On the other hand, highly-critical tasks cannot afford this and may require, for example, a triple modular redundant execution for on-the-fly fault correction.

This paper presents a platform to enable modular redundancy on-demand. With the focus in mixed-criticality scenarios, the idea is to combine controllable fine-grained replication of functional units in the processor, such as the ones presented in [1], with software-based dynamic configuration strategies explored in this paper. This platform offers configuration flexibility to the criticality mix of its running tasks, for example, configure redundancy minimizing fault effects in functional units, but also wear-leveling reducing aging of electronics by not using all the units all the time for non-critical tasks.

One of our main contributions is the possibility to trade fault tolerance and aging of electronics. For that, we are addressing soft faults caused by charged particles hitting the hardware design and intermittent faults that increase, in number, as the electronics start to age. So, the fault model considered over this paper is the single bit flip that may happen in the internal signals of processor functional units due to the faults just mentioned.

II. Methodology

For this paper, we used the 32-bit, MIPS-I compliant, synthesizable Plasma microprocessor [2]. It was extended with a low-latency reconfiguration logic and configurable DMR/TMR scheme similar to the platform presented in [1]. Additionally, the RTOS taken as the base for our software implementations was the Plasma-RTOS [2]. The next subsections describe these implementations.

A. Reconfigurable Processor Design

This section summarizes the prototypical design of a simple processor core with dynamically configurable redundancy. Instead of replicating the complete core for lockstep execution, the design aims for fine-grained replication on the level of *Functional Units (FUs)* such as arithmetic logic units (ALUs), floating-point units (FPUs), integer multipliers and dividers, and address generation units (AGUs). These perform most of the software's computations, take up large portions of the die area, and contain most of the complex logic with critical propagation path delays. Hence, covering these with modular redundancy and other fault detection mechanisms should effectively increase the processor's fault tolerance against soft faults.

Figure 1 shows the replication of a single functional unit (FU) and its interaction with the added components. The instruction decoder was extended to add new instruction opcodes. These provide read/write access to the configuration state and fault counters. The output of each replica FU is passed through a Muller C-Element with a configurable delay line (MCE), similar to the scheme proposed in [3], so it can correct small glitches in the logic signals. Then, the voter performs a majority vote across the active FUs replica. When

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 722325.

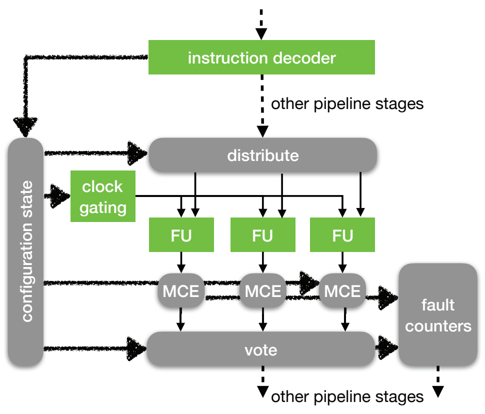

Fig. 1. Extensions to the processor design to enable configurable modular redundancy and fault detection. Rounded boxes mark the extensions.

faults are detected by the MCEs or the voter, these are counted for the respective replica and a signal is raised to let the system software performs the appropriate action.

B. The Operating System and its extensions

As part of a co-design approach, the operating system was extended to control the reconfigurable platform mentioned in section II-A. The RTOS is then responsible for monitoring the health status of the FUs, check the criticality requirements of its running processes, and configure the HW design to match these two parameters. In case of modifications in the health status of the FUs, the RTOS functions are also responsible to update the hardware configuration, as well as the FUs, attributed to each process.

To do so, we mainly extended the operating system processes data structure and the process switching mechanism. The first, to attribute the criticality parameter to each process and, the second, to enable units reconfiguration at process switching time.

As a result, we have a very flexible platform, which enables real-time design reconfiguration on-demand either to the fault-tolerant mode or to the aging avoidance mode.

C. Software Design Space Exploration

Although pure software approaches for fault tolerance demonstrated very good results, they have been showing limitations in its effectiveness. Which means that joint methods, involving hardware and software approaches, become necessary for further improvement.

Thanks to reconfigurable designs, software approaches can keep their role, but additionally, use this feature to improve their fault tolerance capability as it demands to, without wasting unnecessary hardware resources. The design just present in the sections above, enable configuration on-demand of fault tolerant measures, e.g. double or triple modular redundancy.

Taking the mixed-criticality scenario as its application domain, such reconfigurable platform can provide, for example, triple modular redundancy while executing a highly critical task or, in the other case, a simple execution using a single unit when no critical task is been executed, saving some power and minimizing aging over the electronics.

Additionally, the design reconfiguration can be explored as such to provide fault correction on fault detection. It means, once a fault is detected, for example, using double modular redundancy, software approaches can be designed to enable simple instruction re-execution over the same units, instruction re-execution using different units and re-execution using full TMR scheme. To enable such features software approaches like branch on-fault, interrupt on-fault, trap on-fault can be implemented.

III. Evaluation

The final system is composed of the Plasma processor design, its modification, and the Plasma-RTOS equipped with the proposed extensions running on top of it. The system was simulated using the Xilinx Vivado 2018.2 simulator tool.

We evaluate the latency incurred in the process switching mechanism due to the modification to perform the reconfiguration at process switching time. To do so, we first executed the plain process switch with no additional code. Next, we measured the execution time after adding all the necessary code to verify processes criticalities and perform the run-time reconfiguration. The difference between these two executions resulted in an overhead of 5.1% of execution time in the process switching mechanism.

Concerning area overhead, the Plasma processor has its FUs covering about to 60% of the original processor design. So, a triplication of all of these units would generate an overhead of 120%. Moreover, a combinational logic necessary to enable such configurations would take an additional 18% of the area for this particular design, as it is suggested by [1].

Although simple fault injections were performed in the functional units internal signals, confirming the fault tolerance effectiveness of the TMR configuration, long and comprehensive fault injections campaigns are still to be done.

IV. Conclusion

This paper presented a reconfigurable platform to cope with the criticality mix of its application scenario. Our prototype consists of both extended versions of the Plasma processor design and the Plasma-RTOS. Its focus is to provide fault tolerance on-demand and minimizes the aging of electronics while executing non-critical tasks. The prototype shows a rather small overhead of 5% on the latency of the system's task switching mechanism to configure the core for the next task. Finally, the incurred overhead, for triplicating the functional units plus the combinational logic, still bellows the full core replication when compared to core lockstep schemes.

References

[1] R. Segabinazzi Ferreira and J. Nolte, "Low latency reconfiguration mechanism for fine-grained processor internal functional units," in *2019 IEEE Latin American Test Symposium (LATS)*, (Santiago/CL), 2019.

[2] OpenCores.org, "Plasma - most MIPS I(TM) Overview," in *https://opencores.org/projects/plasma, visited Dec. 6th*, 2019.

[3] S. Mitra, M. Zhang, N. Seifert, T. Mak, and K. Kim, "Soft Error Resilient System Design through Error Correction," in *2006 IFIP International Conference on Very Large Scale Integration*, pp. 332–337, IEEE, 10 2006.