Fault-tolerant satellite computing with modern semiconductors

Fuchs, C.M.

Chapter 11

Conclusions and Outlook

11.1 Conclusions

RQ1 In this thesis, we presented a satellite on-board computer (OBC) architecture that can offer strong fault tolerance with conventional, low-cost, modern semiconductors manufactured in small feature-size technology nodes. The correct functionality of this architecture is safeguarded through a set of inter-linked software-implemented fault tolerance measures combined with FPGA reconfiguration, which we described in Chapter 4. These concepts allow us to assure fault tolerance even for satellites with a very small form factor, which today can only utilize primitive or no fault tolerance measures at all, as traditional radiation-hardened satellite computer solutions can not be utilized due to volume, mass and power restrictions. We showed that through lockstep implemented in software, we can efficiently protect a system consisting of embedded and mobile-market components, and should ideally be implemented within an FPGA to exploit reconfiguration. We demonstrate that the performance cost of this lockstep mechanics is economical, and that its implementation is possible in a non-invasive manner. Its protective guarantees are run-time configurable, and fault tolerance can even be entirely deactivated at runtime if so desired.

RQ2 In Chapters 4 and 5, we showed that the logic of an FPGA-implemented MPSoC can be protected well from radiation effects through smart configuration management and off-chip diagnostics. We closed the fault-detection gap which prior research struggles to close through the multi-stage fault tolerance architecture described in Chapter 4. To safeguard an FPGA from transient faults, we showed that error scrubbing and FPGA reconfiguration can be used to detect and correct bit-upsets in the CRAM of an FPGA. As described in Chapter 4, permanent faults can then be mitigated through reconfiguration with alternative partition variants. This not only increases the capability to cover permanent faults, but as we show in Chapter 5, it also allows an OBC to adapted to the specific requirements during each phase of complex, multi-phased space missions. This allows a reduction of overall system complexity, reduces the need for spare processor cores and MPSoC infrastructure logic, and can drastically extend the lifetime of a COTS FPGA-based OBC.

RQ3 In space missions with a very long duration, parts of an FPGA's fabric will eventually no longer be recoverable through reconfiguration. This is due to accumulating permanent faults in the semiconductor the FPGA, and thus also the

MPSoC, are implemented in. Over time, this will result in an increasing number of the MPSoC's processor cores becoming unusable, gradually reducing the amount of processing time available to the lockstep, and the level of replication it can achieve for all applications. In Chapter 6, we showed that the run-time configurable nature of software-implemented fault tolerance enables an OBC to respond to this behavior in a way that can best be described as "graceful aging". By exploiting mixed criticality, it is possible to autonomously reallocate processing time between the different applications that are part of an OBC's flight software, allowing us to safeguard fault-tolerant operation for the flight software's core functionality. We showed that stability and availability of critical applications can be maintained by sacrificing performance of less important applications. In practice, this allows an OBC to age gracefully and adapt to a shrinking set of intact processor cores, instead of failing spontaneously as traditional systems do. A satellite operator can use this functionality to prioritize and dynamically trade system performance for increased fault coverage, power saving, or to maximize an OBC's functionality. Spare processor cores in traditional hardware-voting based systems remain idle until a fault occurs, but our lockstep can use them actively to run less critical parts of the flight software, until they are needed in practice to replace a failed processor core. This allow spare processor cores available throughout an MPSoC to be pooled and used more efficiently, thereby to overcoming the static nature of traditional static hardware-implemented fault tolerance measures. This allows an OBC to offer stronger fault coverage, and to more efficiently meet the changing performance requirements throughout complex multi-phased solar system exploration missions with much reduced over-provisioning and without requiring idle spares.

RQ5 All these operational and system-design improvements are possible due to the coarse-grain lockstep concept described in Chapter 4, which we utilize to achieve forward error correction. We implement this lockstep within the OS kernel of an operating system (RTEMS, FreeRTOS, and experimentally also on Linux) or as part of baremetal software, where it assures synchronization between multiple thread-replicas run on the processor cores of an MPSoC. To test and validate our architecture, in Chapter 8, we conduct fault-injection into an emulated system and into a SystemC-implemented MPSoC model. In this chapter we describe the two fault injection campaigns we conducted against implementations of our lockstep: In the first campaign, we utilized the QEMU-based FIES fault injection framework to inject faults into an RTEMS implemented variant of our lockstep run on a Cortex-A system. In the second campaign, we modeled a triple-core model of our MPSoC using RISC-V cores in ArchC, and injected faults using SystemC simulation. Few software-implemented fault tolerance concepts described in literature have been practically implemented and validated. Therefore this chapter is also intended as practical guide for fellow researchers, to make proper testing of software-implemented fault tolerance measures less challenging and time consuming.

RQ4 Relying on software-implemented fault tolerance measures also require special care to be taken to assure the integrity of the flight-software in which they are implemented. Hence, in Chapter 7, we explored how unprotected volatile and non-volatile COTS memory can be retrofitted with strong error correction and protected from bit-upsets and SEFIs in control logic. We showed that error scrubbing for volatile memory can be combined with allocation-time integrity checking and blacklisting for defective pages in widely-used operating systems such as Linux. To safeguard the logic

of our lockstep and a full firmware image, we showed that a file system can be equipped symbol-based erasure coding and can use memory protection to mitigate the impact of faults in control logic. To protect payload data, we described that a composite erasure coding system can be combined with RAID-like functionality to efficiently protect data stored within high-density NAND-flash and phase change memory. We showed that software measures can guarantee strong fault tolerance, the NAND-flash industry has in even begun to adopt the same erasure coding systems we proposed in this paper as part of a solid-state drives embedded software-stack, e.g., in [286]. Simple erasure coding for caches and other on-chip memories at the time of writing is a standard feature in Xilinx library IP, and supported in all currently available model-market devices [119]. Security vulnerabilities such as Rowhammer and an increased need for yield enhancement have prompted the adoption of ECC also for protecting main memory [362], and in combined with software-implemented memory testing and scrubbing described in this chapter, sufficient protection can be assured even for LEO CubeSat missions with an extended duration of 2-5 years.

RQ6 Much of today's fault tolerance research proposes interesting and novel concepts. But in practice, the majority of these concepts can not be applied to protect a critical system as it exists in the real world. To show that our architecture is effective in practice, in Chapter 9 we developed an MPSoC design which provides an ideal platform for the software-mechanics used to assure fault tolerance. It is the result of a hardware-software co-design process and assures a high-degree of logic and data isolation for software run on the individual processor cores of the OBC within compartments. It can be implemented with just currently available COTS hardware and extensively validated FPGA-vendor library IP, requiring no proprietary logic or costly, custom space-grade processor cores. This design demonstrates that our architecture can not just protect a satellite OBC in theory, but also that a suitable computer architecture is feasible, and that no space-proprietary logic or IP is required.

In Chapter 10, we described the practical implementation of this MPSoC for a variety of Xilinx Ultrascale and Ultrascale+ FPGAs as proof-of-concept. To show how a practical OBC implementation for this MPSoC can look like, we developed a series of MPSoC implementations and a breadboard proof-of-concept of this architecture on Xilinx VCU118 (with 2 DDR memory channels) and KCU116 boards (with 1 channels due to board constraints) in conjunction with TI-MSP430FR development boards. We described the component-level setup of this architecture for CubeSat-use, for which an MPSoC implementation on a KU3P FPGA is possible with just 1.94W total power consumption. This demonstrates that a practical implementation of our architecture can be achieved, which stays well within the power budget range available aboard current 2U CubeSats.

11.2 Discussions

Traditional fault-tolerant computer architectures intended for space applications struggle against technology, and are ineffective for embedded and mobile-market components manufactured in technology nodes with a fine feature size. In this thesis we showed that the solution to this limitation is the use of software-implemented fault tolerance measures, which can be utilized to systematically protect each component of an OBC as depicted in Figure 75. Through the architecture we developed originally as OBC for the MOVE-II satellite, we show that it is possible to efficiently

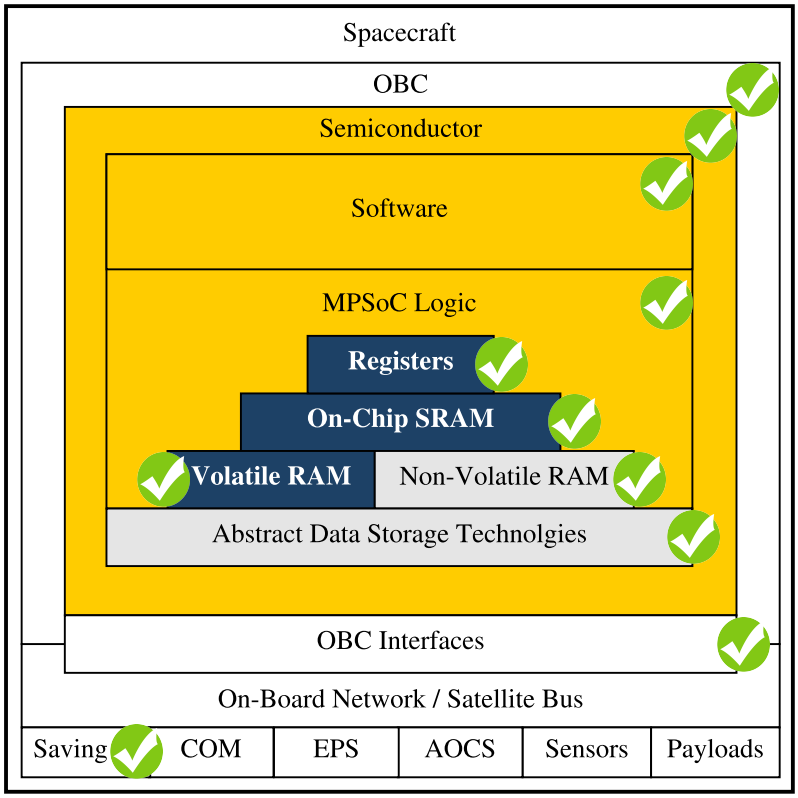

Figure 75: A component-level model of a satellite OBC, components for which the research presented in this thesis offers protection are indicated with checkmarks.

protect modern COTS semiconductors effectively, and make them usable for critical space applications. To realize such an architecture, we do not require any space-grade components, fault-tolerant processor designs, or other custom and proprietary logic. The OBC architecture we developed from this approach can be replicated with just standard design tools and library IP, which are available commercially and even free-of-charge to designers in academic environments. Our architecture scales with technology, instead of struggling against it. It benefits from performance and energy efficiency improvements that can be achieved with modern mobile-market hardware, and can be scaled up to include more, and more powerful processor cores.

In Chapter 10, we showed as practical example that our architecture can achieve beyond 50% power saving even between two generations of Xilinx FPGAs, one being manufactured in 16nm FinFET and the prior generation in a 20nm planar technology node. In this regard, we eagerly await the release of the next generation of FPGAs manufactured in EUV-based technology nodes with 7nm or 5nm feature size. Compared to 16nm FinFET and 20nm planar manufactured devices, we expect that next generation FPGAs manufactured in these technology nodes will offer further power saving, will allow much higher clock frequencies to be achieved for an MPSoC implemented in configurable logic, while the reduced feature size of the semiconductor logic would further reduced the likelihood for radiation to affect.

A comparison of our OBC architecture to traditional space-grade solutions and contemporary CubeSat computing seems unfair. Today, miniaturized satellite developers are limited to use low-performance microcontrollers and MPSoCs implemented in ASIC or FPGA. Considering the few CubeSat compatible low-performance microcontrollers that have been shown robust under radiation, our implementation can offer drastically more performance. At the time of writing Chapter 4, we estimated that our architecture run on modern MPSoC and FPGAs can offer a beyond factor-of-5 performance improvement as compared to these microcontrollers. Since 2017, within a time-span of just two years, mobile market MPSoCs have advanced drastically, and a beyond factor-of-10 improvement seems more realistic. At the time of writing in mid-2019, most mobile-market devices can offer almost twice the clock speed and a better performance per clock cycle as compared to their counterparts in 2017. Same applies to the upcoming generation of FPGA which will benefit greatly from technology scaling.

Mobile-market MPSoCs used aboard CubeSats today seldom include any fault tolerance capabilities. Only sometimes to CubeSat designers implement custom homebrew component-level failover concepts, which has been shown to inflate complexity and failure potential. Our OBC architecture is based upon the same type of commercial technology, but through software-measures and a smart MPSoC design, we assure long-term fault coverage with a component-wise simple setup. Comparing this OBC architecture with traditional solutions for larger spacecraft, even our current FPGA-based proof-of-concept exceeds the single-core performance of the latest generation of space-grade ASICS-SoCs such as an GR740 (250MHz vs 300MHz+). On top of that, our architecture can offer fault tolerance at a fraction of the cost. It can do so without suffering from the tight technological constraints of this classical technology and the archaic development tools used there. All this is possible while still using COTS hardware, without being impacted by the legal constraints of components that are subject to ITAR or other export control laws.

11.3 Outlook and Future Work

As of early 2019, Xilinx has began to introduce a new generation of FPGA-equipped devices manufactured in a 7nm FinFET+ technology node, in which the design issue causing latch-up in Ultrascale+ should be mitigated [299]. With this node, Xilinx's foundry TSMC expects an around 65% reduction power consumption as compared to the 16nm FinFET node used for Ultrascale+ FPGAs [360]. Even if only half of this expected power reduction would manifests, in combination with FPGA-fabric optimizations, we can expect to achieve approximately 1W power consumption with our MPSoC implemented on a next-gen Xilinx FPGA. While these expectations based on experiences with the current 20nm Planar and 16nm FinFET manufactured Xilinx FPGAs, future FPGA generations released within the next decade will, with near certainty, allow our architecture to even become usable aboard 1U CubeSats.

At this point in time, I have validated this OBC architecture to the extent that this is possible for a single researcher in an academic environment. As next step to validate it, I therefore plan to develop a prototype implementation. Since 2018, I have therefore collaborated with and contributed to the Xilinx Radiation Testing Consortium in the creation of a Kintex Ultrascale KU60 device-test card to reduce the cost and time required for constructing this prototype. As of 12.09.2019, we, the

XRTC infrastructure team, have finalized the KU60 card's design and schematics, and after routing and a final review pass, the KU60 DuT-card will go into production later this year.

Once the XRTC KU60 DuT-card becomes available, I plan to implement a matching daughterboard carrying DDR-SDRAM, MRAM, and PCM components as well as a supervisor MSP430FR, to then conduct radiation testing. Radiation testing will then increase the maturity of this architecture to TRL4, and also serves as intermediate step to then realize a full custom-PCB based prototype. This prototype can then for the first time be used to demonstrate the full capabilities of this architecture at TRL5, without the constraints present in a development-based breadboard setup.

There is considerable potential for improvements considering the proof-of-concept that I have developed before and during my PhD: The relaxed cost, energy, and size constraints aboard microsatellites and larger spacecraft would allow an implementation of this OBC architecture spanning multiple FPGAs and with a drastically higher number of compartments. Such an OBC would not only offer better scalability and fault-isolation than a single-FPGA system, but can then also tolerate chip-level defects and SEFIs. Application replicas in lockstep could then be distributed across multiple FPGAs, allowing non-stop operation even if an individual FPGA would have to be reset, if or full reconfiguration is necessary.

To support larger MPSoCs with more than 8 compartments efficiently, a more scalable interface between compartments and memory controller sets should be used. This can be achieved by replacing the 2-level AXI crossbar the MPSoC is built around today with a Network-on-Chip (NoC). A NoC offers improved scalability [329], can also be used to enable fault-tolerant routing [349], backwards error correction through re-transmission, and quality-of-service support [359]. When implementing this architecture with a NoC, the shared memory controller sets would be implemented on one NoC layer, while the state-exchange network described in Chapter 9 would exist as second layer. NoC routers can also be outfitted with error correction themselves [93]. Unfortunately, the few NoC-specialized experts I encountered while conducting this research had little interest in implementing their research practically. Hence I hope incorporate NoC into this MPSoC design in the future in collaboration with those who are willing to do so.

I designed this OBC architecture specifically to utilize and exploit the powerful fault-recovery capabilities of modern FPGAs. However, this OBC architecture could very well be realized also on ASICs manufactured in radiation-robust COTS manufacturing processes such as FD-SoI [144]. This would allow much reduced energy consumption, and drastically higher clock speeds to be achieved. An ASIC variant would be less susceptible to transients and more robust to permanent faults, while loosing the capability to mitigate permanent faults through FPGA reconfiguration. However, due to the drastically increased development costs of an ASIC implementation, the resulting OBC would not be viable for miniaturized satellite applications anymore. We see this as a "big-space" variant of this approach with its own advantages, but it would no longer offer fault tolerance "on a budget".

This research began as a one-person project, but towards the end of my PhD, it has become clear that it has today outgrown the capacity of just a single researcher. In all regards, the end of my PhD is actually the beginning of something new, and more important. I know that in the coming years, I must gather a research group to advance this research and develop it further in a suitable environment. Where I will do this

remains yet to be seen. At the end of the second year and the beginning of the final year of my time as PhD researcher, I therefore began to explore ways for conducting long-term testing for this OBC architecture to appropriately consider the time-component that is introduced in testing hardware-software-hybrid systems. In this processes, I have had the pleasure collaborate with several international experts in the fields of radiation testing, space engineering, and semiconductor testing. Promising test environments for long-term testing include the close proximity of a radiation source, the Exposed Facility aboard the ISS (JEM-EF), or the vicinity of the Fukushima Daiichi site. Naturally, all these test setups require considerable preparation time, and preparing a prototype for deployed, e.g., aboard ISS is a highly competitive and certification-heavy undertaking. Therefore, I aim to conduct in parallel to long-term testing also on-orbit validation aboard a CubeSat, which is possible more rapidly and at reduced cost than e.g., through an ISS experiment. After all, on-orbit technology demonstration and validation is one of the prime use-cases for CubeSats today, and also one of their most successful applications.

On-orbit validation aboard a CubeSat also closes a circle that began with the early failure of the FirstMOVE CubeSat, and that initiated my satellite fault tolerance research. I started this research, searching for a way to realize a better, fault-tolerant satellite bus architecture for the MOVE-II CubeSat project. Back then, it became clear that there were simply no fault-tolerant OBC architectures or products in existence that could even theoretically be used to assure fault tolerance and guarantee reliable operation for long-term CubeSat mission. At the start of this thesis, we raised the question:

RQ0 *Can a fault tolerance computer architecture be achieved with modern embedded and mobile-market technology, without breaking the mass, size, complexity, and budget constraints of miniaturized satellite applications?*

This hard question arrose at the beginning of the development process of the MOVE-II CubeSat. I approached this research without a specific architecture or solution in mind, and even briefly considered a highly experimental, academic VLIW platform. Three years, many published research papers, and several catastrophes later, it is now possible to answer this question in the following way:

RQ0 Yes. A fault-tolerant computer architecture for miniaturized satellites is technically feasible with contemporary COTS technology. Once fully implemented as a prototype, it can be used to expand the reliable lifetime of modern day CubeSats drastically, thereby enabling their use in critical and long-term space missions. With contemporary COTS components, this OBC architecture can be applied to satellites as small as 2U CubeSats. Advances in semiconductor manufacturing in the upcoming generation of FPGAs will make this approach also usable for smaller spacecraft, and even more appealing as it scales with technology. It can improve efficiency and scalability when implemented aboard heavier spacecraft that we use today for high-priority science and solar system exploration. And maybe in the future, hopefully, we can explore even what lies beyond its boundaries.