Autonomous Fault-Tolerant Avionics for Small COTS Satellites: to Reality and Prototype

Dr. Christian M. Fuchs
The Fault Tolerant Satellite Computer Organization
Weigunystrasse 4, 4040 Linz, Austria
christian.fuchs@dependable.space

Dr. Nadia M. Murillo
RIKEN Cluster for Pioneering Research
Wako, Saitama 351-0198, Japan
nmurillo@starformation.space

ABSTRACT

In this contribution we present practical experiences from realizing a prototype of the first truly fault-tolerant and autonomously operating avionics suite for miniaturized satellite down to the size of a 2U CubeSat. Our initial demonstrator setup consists of a mix of COTS parts and FPGA development boards, which we gradually expanded in scope and capabilities. After four iterations of PCB development and manufacturing, we have condensed this design to a fully integrated custom PCB-based prototype. Our fourth architecture iteration is stackable and is designed to fit on an 80×80mm PCB footprint. It is furthermore capable of operating as generic satellite subsystem node, functioning in a distributed, fault-tolerant, interconnected manner together with other subsystems. Each node is fully replaceable by two or more neighboring subsystem-nodes. In consequence, we achieve a satellite bus setup which is in spirit similar to integrated modular avionics and modern fault-tolerant avionics network architectures used in other fields. We realize this setup through a high-speed chip-to-chip network in a compact CubeSat form factor.

INTRODUCTION

Until today, fault-tolerant avionics for satellites are costly, less flexible, consume considerably more power, and have a worse performance-per-watt ratio than conventional COTS components. Without exception, they are all dependent on proprietary RHBD and RHBM components, custom designed for space-use. These take decades to develop and also lag generations behind their conventional COTS counterparts.

This applies to all satellite classes. However, in contrast to smaller modern miniaturized satellites, multi-ton spacecraft, as well as SmallSats above 50 kg, can afford to fly high-power consumption avionics. For smaller, miniaturized satellites such as microsatellite and nanosatellites, this is not the case. Such small spacecraft can not afford to fly traditional fault-tolerant RHBD and RHBM components anymore due to practical and operational constraints. For most smaller spacecraft that enable the innovative mission concepts that have enabled the emergence of our new-space sector, the bottom line is simply that fault-tolerant avionics are just not yet possible with currently available technology.

Most smaller satellite form factors are thus constrained to space missions with a brief duration, of months to maybe a few years. This is due to the unpredictable survivability of commodity-electronics based hardware in the space environment, as well as the lack of a reliability-safety net, which larger spacecraft are specifically designed to include. Instead of a system of fault and failure mitigation techniques, a failure aboard most small satellites can very well be fatal. This must change.

Therefore, we developed novel fault-tolerance concepts. Today, these enable us to finally overcome this limitation. It is now possible to achieve strong fault-tolerance for missions with an extended duration, using the very same conventional COTS technology that is being used today aboard very small satellites. Combined, these concepts form an avionics architecture which was presented SmallSat 2019 and the proof-of-concept implementation of which we described in-depth in [1].

By design, satellite subsystems following this architecture are capable of autonomous failure handling, dynamic mid-mission reconfiguration, and adaptive, graceful aging under the consideration of permanent and persistent faults. By 2019, we produced proof-of-concept implementations for each part of this architecture. We had conducted fault-injection and system simulation at different levels to test the effectiveness of this architecture, and confirmed its practical feasibility under considerations of cost, power consumption, long-term

survivability, and mass. We tested each part of this concept extensively and systematically, and analyzed the architecture's performance overhead. Since then, we have advanced the maturity of this technology from proof-of-concept to the prototype stage.

In this paper, we share experiences and lessons learned in developing prototype hardware for this new technology, and we provide a status update on our development efforts. We optimized and streamlined the individual elements and fault-mitigation stages of the protective architecture we described in [1], stripping out features that we considered mainly academically viable, and achieved real time capabilities. We present our practical experiences from building the first autonomously operating avionics suite implementing this technology in the real world in hardware. Specifically, we outline our experience building several demonstrator setups, and eventually developing a first fully-custom and functional prototype system. This prototype is intended for use as development platform, as well as target for radiation testing and characterization, and can be shrunk to fit aboard SmallSats, Microsatellites, and even 2U+ CubeSats.

BACKGROUND AND RELATED WORK

In contrast to the initial generation of educational CubeSats, today fewer satellites fail due to practical design problems caused by inexperience [2]. Instead, Langer et al. in [3] showed that the a majority of these failures can be attributed to electronics heavy subsystems. Even traditional space industry actors with years of experience in large satellite design, who attempt to develop CubeSats satellites "by the traditional book" with quasi-infinite budgets today struggle to reach just 30% mission success [4].

The main source of failure there are environmental effects encountered in the space environment: radiation, thermal stress, and corruption of critical software components that can not be recovered from the ground, and failures caused by power electronics. Considering again Langer et al., [3], with increasing age mission duration, a broad majority of documented failures aboard CubeSats originate from OBCs, transceivers, and the electrical power subsystem. While functionally disjunct, these subsystems all have in common that they are heavily computerized and architecturally rather similar, built around one or multiple microcontrollers and memories.

A satellite must cope with challenging design constraints, and the effects of the space environment on electronics. The main source for faults within electronics in the space environment are highly charged particles from a variety of sources [5]. Particles interact with a spacecraft's electronics, and can induce different effects in a semiconductor depending on the type of particle and its charge. Among others, charged particles can corrupt logical operations or induce bit-flips within semiconductor logic and memory (single event effects - SEE), and may cause displacement damage (DD) at the molecular level, induce a latch-ups or cause functional interrupts – SEFIs. The cumulative effect of charge trapping in the oxide of electronic devices (total ionizing dose – TID) further impacts the lifetime satellite electronics.

All these effects can result in spontaneous or drastically accelerated aging compared to ground applications, which must be handled efficiently throughout an entire space mission. To do so, traditional space-grade hardware makes heavy use of over-provisioning and tries to include idle spare resources (processor cores, components, memory, ...) where necessary. Traditional OBCs for large satellites realize fault tolerance using circuit-, RTL- [7], IP-block- [8], [9], and OBC-level TMR [10] through costly, space-proprietary IP. Circuit-, RTL-, and core-level measures are effective for small microcontroller-SoCs [11], [12], if they are manufactured in large feature-size technology nodes. More and more error correction and voting circuitry is needed to compensate for the increased severity of radiation effects with modern technology nodes [11]. This in turn inflates the fault-potential, requiring even more protective circuitry, making this approach ineffective for modern semiconductors.

Approximately 10 years ago, nanosatellites began to heavily utilize redundancy at the component level to achieve some form of fail-over, to provide at least some protection from failure. Unfortunately, several CubeSat bus manufacturers have in recent years decided to follow this approach, and began to add redundancy to everything. However, practical flight results show that such designs are complex and fragile, as compared to entirely unprotected ones [2], [13]. Entirely unprotected OBC designs are of course also no solution to the reliability of a vast majority of CubeSats, as they, in turn, may fail at any given point in time. However, today satellite designers are usually forced to simply accept this risk, leaving the hope that a satellite will by chance not experience critical faults before its mission is concluded. Risk acceptance is viable only for educational, and uncritical, low-priority missions with a very brief duration.

FROM PROOF-OF-CONCEPT BACK TO THE DRAWING BOARD

In this section, we will provide a brief summary of the base concept that we have now developed into a fully fledged hardware prototype. This section is meant only to provide a brief introduction to the concept, considerably more in-depth documentation, as well as testing and validation results can be found in [1] and [14].

Our objective is to provide practically viable and economical means of assuring fault-tolerance aboard

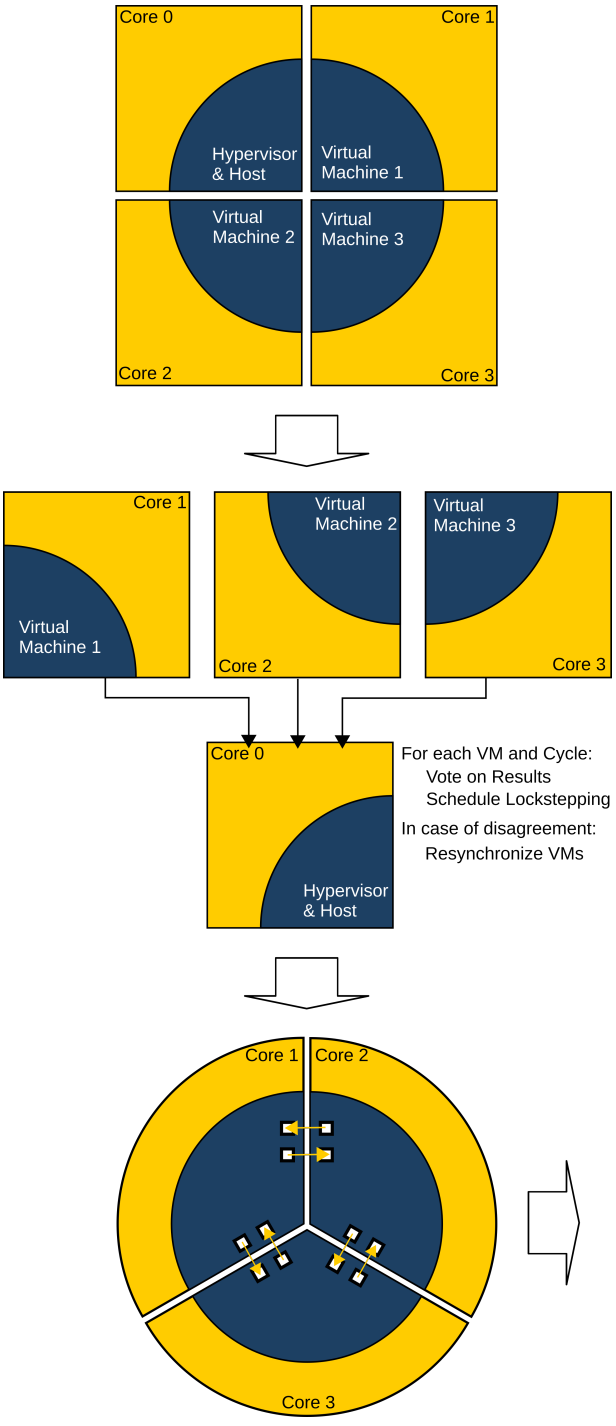

Figure 1: The early evolution the fault-tolerance concept we started out with. Initially, we built worked with a COTS quad-core MPSoC running KVM virtual machines and I/O voting. This concept later evolved into an FPGA-based MPSoC architecture consisting of multiple fault-tolerance measures.

small satellites, especially micro and nanosatellites down to the size of 2U CubeSats. We have developed this architecture from the ground up. We formulated an initial concept out of immediate need, when the first CubeSat we co-developed had failed after two months on-orbit [15]. The original concept is depicted in Figure 1. Since its infancy, we have continued to mature this technology. We began by establishing a sound scientific, theoretical foundation for its functionality, then started designing proof-of-concept implementations, and eventually developed these into breadboard proof-of-concept setups. We tested and validated this technology at each stage of technological maturity, and presented our results to the community, e.g., in [1]. At the time of writing, we have just finished a clean-slate iteration, and completely re-developed this concept from the ground up. In this process, we streamlined the resulting architecture and fault-tolerance mechanics considerably, and developed first a demonstrator setup, and most recently a custom-PCB based prototype.

Instead of utilizing classical radiation hardened semiconductors or custom TMRed processors, we combine software functionality with architectural and topological design features within a system-on-chip to achieve fault-tolerance to the effects of the space environment. This system-on-chip design then forms the core of a satellite computer, which can be freely adapted to a broad variety of use cases. It can be utilized either as command & data handling subsystem, or fulfill any other subsystem role within a satellite.

A concept flowchart of the approach we utilize to achieve fault detection, isolation, recovery, and reporting is depicted in Figure 2. Among others, our architecture combines software-enforced lockstep concepts,

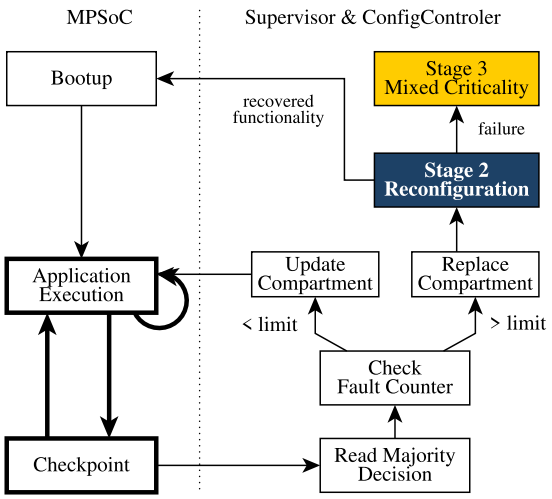

Figure 2: The same concept, now implemented in software through lockstep within an MPSoC on an FPGA.

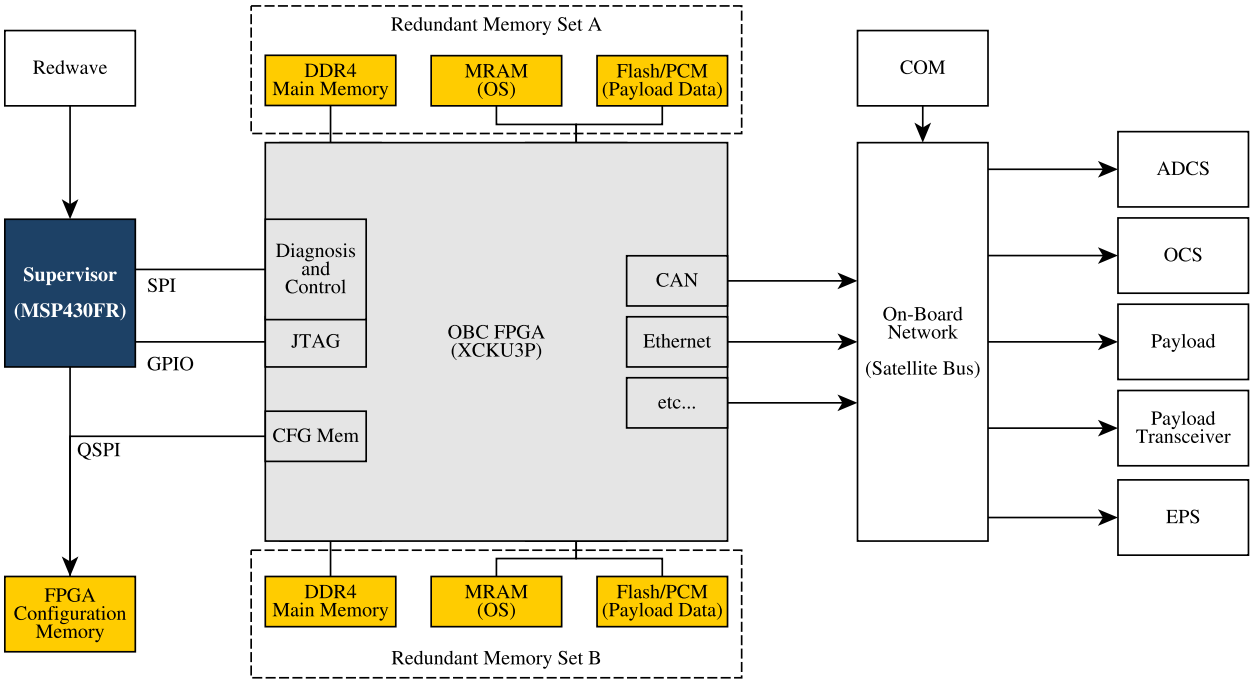

Figure 3: A component-level diagram of a proof-of-concept implementation of our OBC architecture.

distributed decentralized voting, FPGA reconfiguration, component-level self testing, as well as adaptive application scheduling using mixed criticality aspects. The combination of several of these mechanisms acts mutually amplifying, thereby increasing the protective strengths of the system far beyond of what usually would be achievable if these measures were applied independently. This fault tolerant system architecture can be implemented purely using commodity hardware and COTS components that are being flown today aboard miniaturized satellites. It requires only standard library IP to achieve fault-tolerance for virtually any firmware, operating system, and software without enforcing a vendor lock-in.

A subsystem utilizing this architecture can be dynamically reconfigured during a mission and trade compute performance for reduced energy consumption or increased robustness at run time. It enables a satellite computer, or any other kind of subsystem utilizing this architecture to best meet performance requirements during different mission phases and different tasks. The architecture allows a satellite to age gracefully under the effects of the space environment, instead of failing spontaneously. Note that it provides actual fault tolerance, it does not simply suppress faults until all redundancy has been expended. It does not prevent damage but allows a system to adapt to chip level degradation and accumulating permanent faults, even during space missions with a very long duration.

To test our implementation, we conducted fault injection at different levels, which we documented in [16] and [14]. Next, we developed a multi-core model of our MPSoC also in ArchC/SystemC to conduct further fault-injection close-to-hardware, which is further described in [17]. A more detailed description of these validation steps is described in detail in [14]. To achieve worst-case performance estimations, we measured the worst-case performance cost of this approach, which are also described further in [18]. These benchmark results were generated based on code derived off a CCD readout program used for astronomical instrumentation.

At the time of our last smallsat contribution [1], we had constructed a proof-of-concept OBC setup based on an FPGA development board in conjunction with an MSP430FR development board. Following this publication, the first author earned his PhD 2 weeks before the COVID pandemic, in late December 2019. In early 2020, we began the next stage in developing this technology, with a clean slate approach. At his point in time, we had developed a specific architectural flavor for several years in an academic environment.

Hence, we began by taking a step back from the results we had developed and started fresh. We evaluated every aspect of the concepts, software, hardware, and HDL implementations we had developed over the past years. Subsequently, we re-developed a new fault-tolerance concept based on functionality that we liked in our original concepts, but not in direct continuation of the

results obtained. We did this to avoid locking ourselves into design decisions we ourselves had considered proper from an academic point of view, not for an industrial perspective. This was necessary to achieve not just a functional prototype, but also an implementation that in reality works efficiently aboard a CubeSat.

After several weeks of study, we ended up with a fresh take at the concept we had originally developed for protecting the MOVE-II CubeSat. We omitted several aspects of the architecture we described in [1], especially those that made sense for academic research, but not for immediate practical use aboard a satellite. To clarify, we had expected that this would be necessary already several years earlier when starting the project which funded our research between 2016 and 2019.

First Steps: A Functional Demonstrator

In the second half of 2020, we had constructed a first fully functional development-board based demonstrator following the reworked architecture we had just developed. This breadboard-prototype for the first time allows us to run a full Supervisor-MPSoC setup within an Ethernet-based satellite bus testbed, instead of individual proof-of-concept setups for different parts of the architecture. This setup is depicted in Figure 4.

The test setup consisted of a fault-tolerant quad-core MPSoC with an on-chip ICAP Microblaze-MCS controller, which carried out platform management, temperature sensor readout, and partial reconfiguration tasks. This ICAP controller enabled us to perform rapid partial reconfiguration while remaining within the limited IO-capabilities of our MSP430FR development board supervisor, and not forcing us to design custom PCBs at this stage. We constructed a first breadboard-based shields to interface our MSP430FR development board with MPSoC carrier.

Each compartment was outfitted with an Ethernet-controller, a UART interface, I2C, as well as two SPI interfaces (one master, one slave). One of these SPI interfaces was multiplexed and then made accessible to the off-chip supervisor. All other interfaces were exposed on the second FMC port available within the system, which we simply looped back to the system as depicted in Figure 4. Ethernet was realized through the Xilinx AXI-Ethernet Core in RGMII mode coupled with a trial-TEMAC core. Each compartment's ethernet interface was fully independent and attached to one of

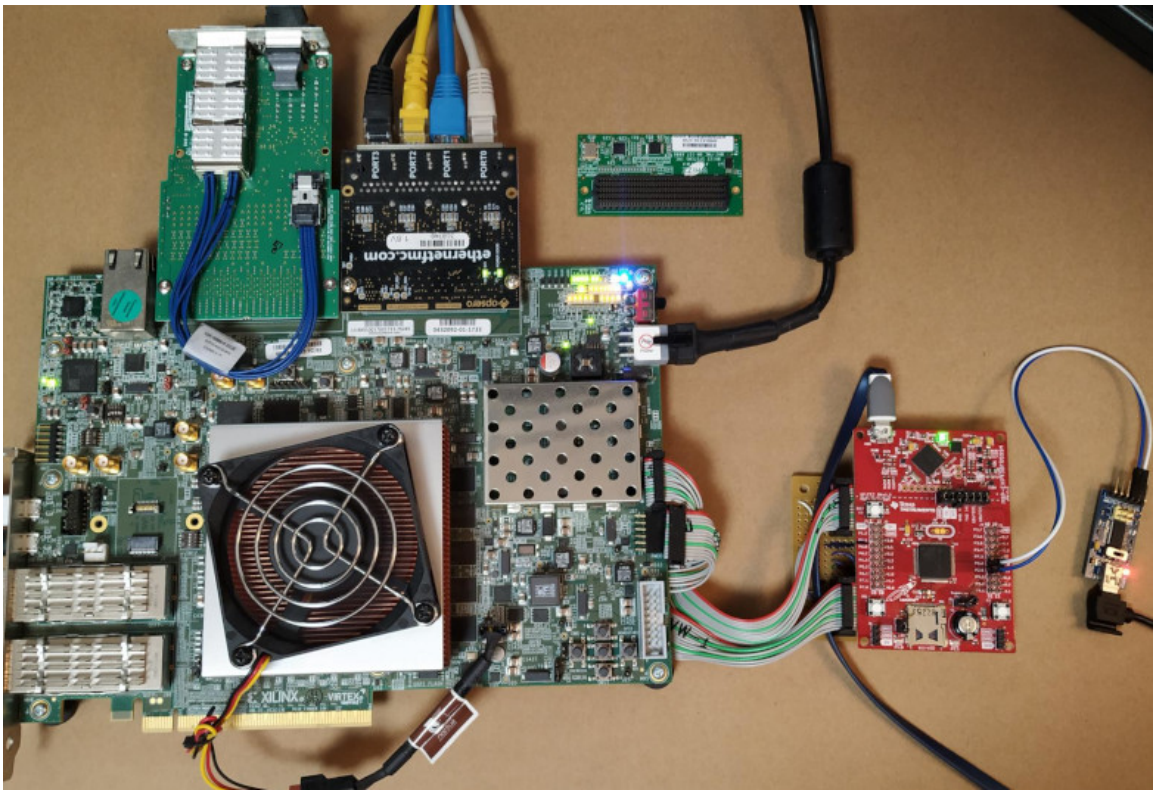

Figure 4: Our first functional all-in-one breadboard setup the off-chip lockstep supervisor implemented within the red MSP430FR board shown on the left, one Ethernet interface for each processor compartment realized through a quad-port ethernet FMC card, and other compartment IO in loopback on the left FMC port.

the dedicated Phys available on our COTS quad-ethernet FMC card.

At this point in time, we had reached the limit of functionality possible with just development boards, and hand-soldered interface components. Hence, as next step we began to scale up our test setup, and started gradually introducing custom-manufactured PCBs.

EXPANSION AND DETOUR TO THE ZYNQMPSOC

With our first batch of custom manufactured hardware, we broke-out of our MPSoC development card to a much larger and more capable supervisor-scrubber-MPSoC setup. This setup, while still based on development boards, now allowed direct access to the MPSoC-FPGA's configuration interface via FMC. The MPSoC remained within our VCU118 development board, but supervisor, logging, and satellite bus-emulation functionality was now implemented through a ZynqMPSoC development board.

With the Zynq MPSoC platform, we had hoped to achieve an all-in-one single-chip implementation of our MPSoC+Supervisor tandem. However, after several months of reverse engineering firmware and platform bring-up code, we discovered that the architectural features facilitating platform bring-up on the Zynq platform makes a truly fault-tolerant single-chip on-board computer implementation unfeasible. The bring-up process of the ZynqMPSoC can in practice not be protected in a single-chip setup, regardless of the fault-tolerance measures used on top or surrounding the platform. This is due to the fact that the bring-up process is dependent upon several features which act as single points of failures. These can neither be protected, nor can failures affecting them be mitigated except for measures taken in silicone. Hence, we stepped away from pursuing a single-chip MPSoC setup with our architecture, regardless of its charm and energy efficiency. Instead, we utilized the ZynqMPSoC platform to drastically expand our breadboard demonstrator and then slowly began introducing custom manufactured hardware.

In the process of investigating the Zynq MPSoC's platform, we also worked on mitigating the latch-up issues present in the Xilinx 16nm Ultrascale+ and newer families. It turned out that simple current and chip-internal temperature monitoring is sufficient to solve the destructive part of the latch-ups encountered, and to efficiently and rapidly mitigate the non-destructive share of latch-ups. This process is described in detail at a high level in [19], and covered in detail in [20] where also the voltage threshold required to resolve this latch-up is documented. The process has been validated through radiation testing, and is a flavor of the traditional latch-up mitigation process employed in some CubeSat COTS-EPS. These results have tentatively been confirmed independently also though a latch-up mitigation experiment in a radiation test campaign we supported in May 2021 at LBL. Note that this radiation test campaign is still ongoing at the time of writing, no final report has been released as to date, but we hope it will be made available to the public.

THINGS GETTING REAL: TOWARDS A PROTOTYPE

Since mid-2020, our development approach became increasingly iterative due to limited component availability, process optimization, logistical constraints during a pandemic, and the step-by-step manner in which we could merge-in new demonstrator functionality. We began by designing and manufacturing custom breakout PCBs in Q2/2020.

Initially, we replaced the simple hand-soldered interface shields we had built earlier. In the second half of 2020, we had completed our first fully-featured platform demonstrator setup, which is depicted in Figure 5. This demonstrator was capable of autonomous operation and fault-mitigation, fully automated logging and fault injection. Naturally, due to still utilizing COTS development boards instead of custom designed-hardware, this setup was relatively large and included an abundance of unnecessary and unused functionality (approximately 80% of each board's functionality). As payload we utilized a variety of different test applications, all running within an RTOS. We used this demonstrator's fault injection capabilities to test and optimize its ability to respond to and handle a wide array of different failure modes.

In autumn 2020, custom hardware development was well underway, and our first fully custom-designed board prototypes were assembled under extreme conditions during the peak of the COVID-19 pandemic. Due to being stuck on an island in the pacific with only parts of the first author's personal electronics lab in reach, we also were in acute need of an SMD assembly, reflow, and testing lab. None of the local facilities were available to individual customers for low-quantity prototype development, so we developed our own SMD processing line. We will not describe this setup in detail here, but will only briefly mention some of the tools we built and the tool set we have been relying upon for the past year.

We started with basic microsoldering setup, most notably including tools such as an HDMI-inspection microscope, temperature controlled hotplate, SMD rework tools, logic analyzer, and other test and measurement equipment, etc. We constructed an SMD reflow oven based on a commercial toaster oven, and controlled it with raspberry-pi driven setup. We optimized and reworked this setup over the course of 2020 and the first half of 2021, and rebuilt and refined our reflow oven design serveral times. To control this setup, we utilized a

largely rewritten version of picoreflow [1]. Specifically, we replaced the temperature control loop of picoreflow and integrated support for multiple thermocouples to track different temperature in different PCB regions.

This enabled us to assemble, reflow, and test manufacture the first two generations of custom designed hardware. We developed and manufactured several custom utility cards, such as the one depicted in Figure 7. At the time of writing, we are in the middle of the fourth iteration of prototype hardware development. In the first prototype development iteration, we replaced the early hand-soldered breadboard interface we had constructed in 2020 with custom PCBs. This allowed us to achieve much higher clock frequencies and gave us the design freedom necessary to make our demonstrator's interfaces fully functional. The remainder of the demonstrator remained largely identical, though it allowed us to fully automatize the setup.

In the second iteration, we added support for external FPGA scrubbing, and removed the need of the supervisor component to interact with an FPGA's configuration interface through an FPGA-implemented configuration controller. In practice, this iteration yielded a considerably more powerful implementation of the safeing subsystem we had developed for MOVE-II in [21], the logic of which is depicted in Figure 6. Furthermore, we used this hardware-design iteration to prototype high-speed interface design. At this point in time, we also required more sophisticated measurement tools that exceed what a personal electronics lab usually includes, and we sourced more sophisticated measurement equipment to do precise signal integrity analysis, debug DDR memory

[1] https://github.com/apollo-ng/picoReflow

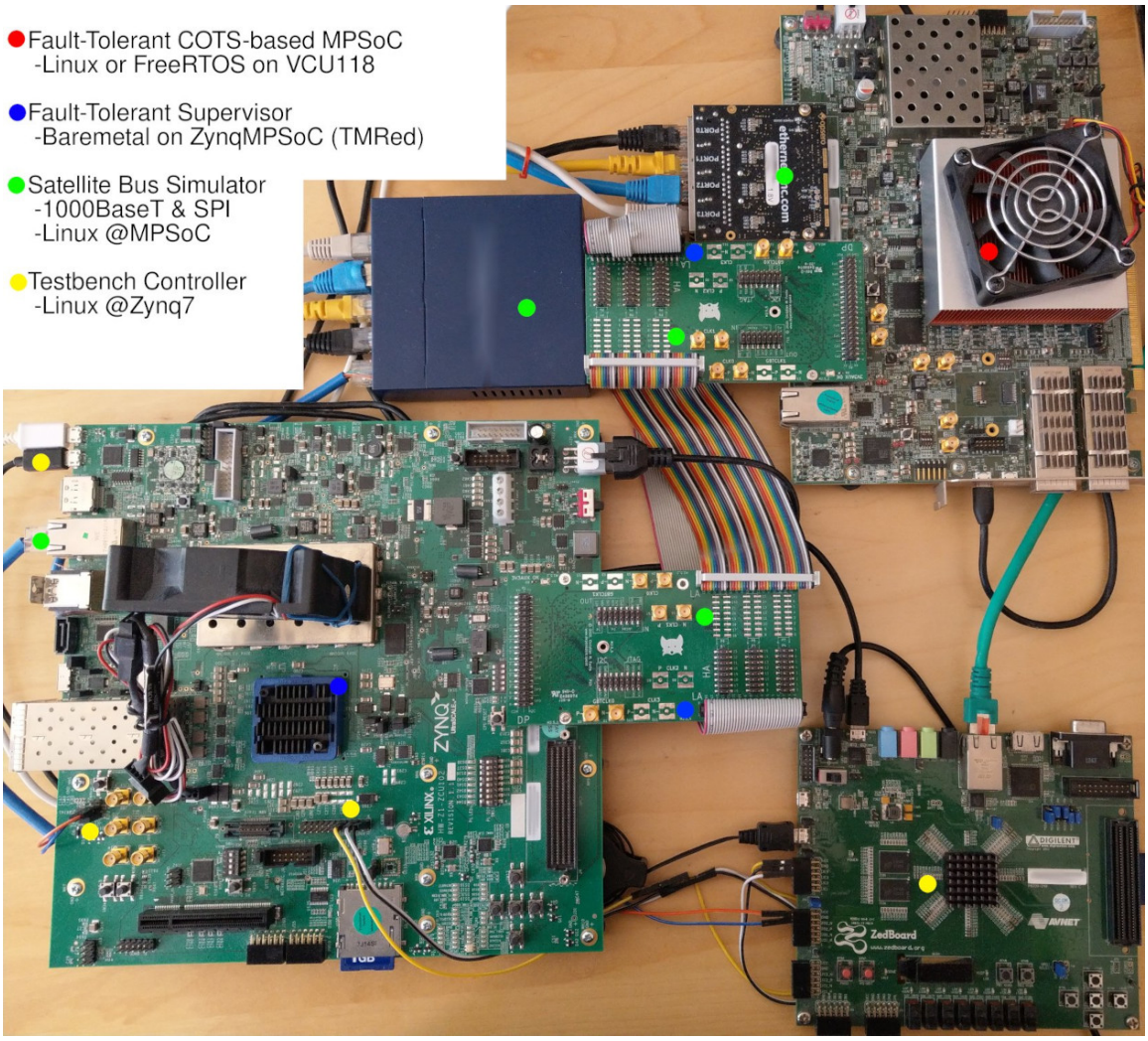

Figure 5: Our first fully functional development-board based emulator setup including some early custom designed breakout hardware (center). Different functional components of the setup are colorcoded.

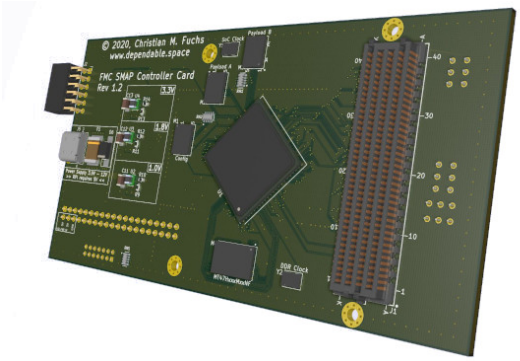

Figure 7: An example of a one of the pieces of hardware we developed in the past year on our way to achieve a full prototype implementation of our architecture.

and similar interfaces at the bandwidths at which they operate.

Our third custom-hardware manufacturing iteration can be considered a bugfix and lessons-learned step. In the previous development steps we had learned important lessons regarding high-speed transceiver design, signal integrity, and SMD process optimization. We also lost components that were never found. The hardware developed in this third iteration integrated all these results and experiences, and added functionality we had chosen to not integrate in previous prototype hardware. Also, we omitted a considerable amount of legacy interface support we had included in our generation 1 and 2 hardware.

In our fourth prototype development iteration, we have produced our first fully functional on-board computer prototype implementation on a single carrier-board. With this iteration, for the first time we no longer require COTS development boards or breakout cards, though still support them to simplify development. Instead, all functionality is now implemented within a form factor that we designed to be shrunk to fit aboard a CubeSat.

Evolution from OBC to Satellite Bus

Our initial objective over the past years always had been to provide a CubeSat-compatible COTS based satellite on-board computer that offers strong fault-tolerance using only commercial commodity components. In the progress of our clean-slate concept re-evaluation, we wanted to not just prototype a fault-tolerant satellite on-board computer, but aimed to provide fault coverage for an entire CubeSat bus or subsystem interconnect. Hence, we developed different demonstrator and prototype iterations throughout 2020 and 2021, increasingly evolving beyond "just" being an on-board computer prototype. As of the second iteration of our demonstrator that included satellite bus simulation, it became clear that the only real distinction between our OBC MPSoC and its simulated subsystem counterparts was the number of compartments implemented within the different modules, and the software deployed.

Our fourth architecture iteration is designed to be stackable, and the condensed footprint of each individual board easily can be fit on an 80×80mm PCB footprint. To illustrate the basic notion of how our fourth iteration boards stackup, we depict such a setup in Figure 8. Each PCB labeled as subsystem node is designed to be a

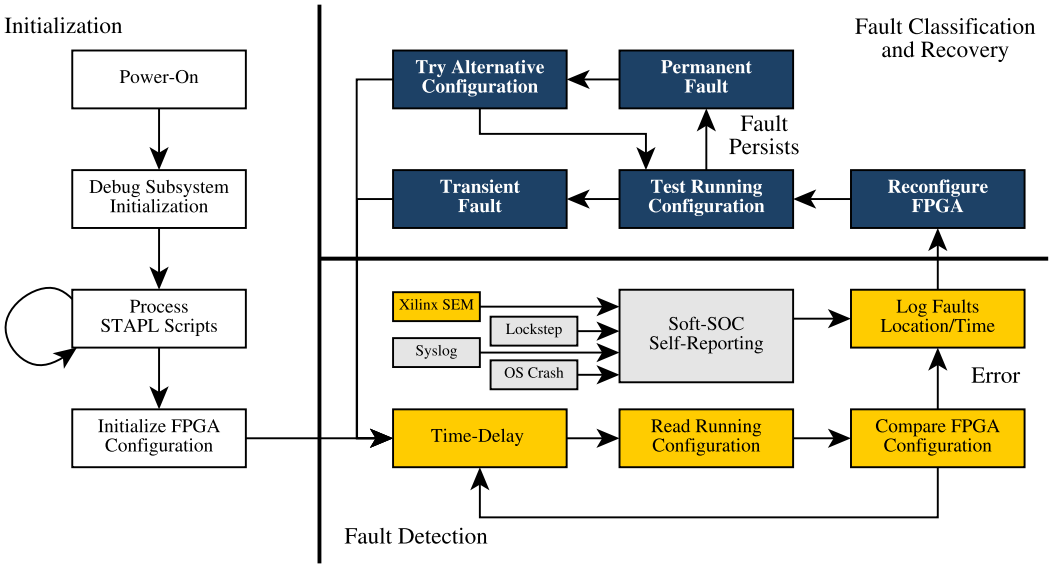

Figure 6: High-level function overview over the saving subsystem we developed for the CubeSat MOVE-II, which today is part of our prototype setup and which we run on the card depicted in Figure 7.

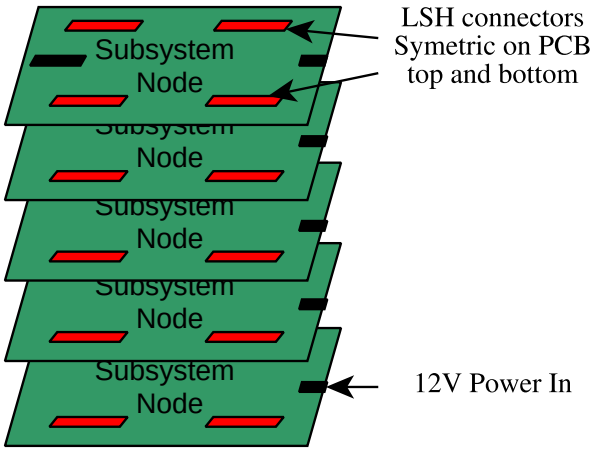

Figure 8: A 5-board stackup of our fourth-iteration prototype board. Note that this image merely depicts the general idea and is not drawn to scale or proportion. Each board represents a fully independent prototype board, however, each board can also be interfaced freely and supply power to siblings located above and below itself.

full satellite computer prototype instance, and we utilize a top-bottom symmetric hermaphroditic connector to achieve stackability. Notably, these boards are powered independently, and each includes a full set of power electronics which would usually exist only once in a satellite's EPS subsystem. We could also utilize these connectors for power delivery, but currently do not yet use this functionality.

For the sake of completeness, our fourth-iteration prototype is intentionally designed for a larger PCB footprint to enable convenient access debug headers, and allow for easier testing and probing. Furthermore, the larger spacing we are using in this generation is required to be able to still hand-assemble the PCB with tweezers. A more compact CubeSat form factor would require assembling using a pick&place machine, access to a commercial SMD manufacturing facility, or a way to finance turnkey manufacturing with larger quantities than what we currently produce.

The platform we implement our prototype for is designed to hold 4 or more compute compartments and operate as central on-board computer controlling all aspects of a satellite. At this point in time, we support the following Xilinx FPGA families:

- 16nm Ultrascale+ (except Zynq MPSoC)
- 20nm Virtex and Kintex Ultrascale
- Series 7 Artix and Kintex FPGAs

We support specifically only Xilinx devices at this point in time, as we have continuously worked with Xilinx FPGAs since 2014 and possess the tooling and experience. However, our architecture can very well be expanded to other systems, device types, and platforms.

For subsystem-versions of our prototype, no large 4-compartment MPSoCs are necessary and are also not desirable aboard tightly power-budget constrained satellites. This is especially true if we not only consider the role of our prototype as central satellite on-board computer, but as one subsystem node within a distributed satellite architecture with a varying set of sensors, memories, and in general, functionality attached and services by each subsystem node. There, fault-detection and mitigation and state-synchronization between compartments on different nodes still operates analogous to the mechanics we described and implemented in Section . However, satellite subsystem nodes operate in a distributed, interconnected manner where each node is fully replaceable by two or more neighboring subsystem-nodes. If we consider a stack of 5 or more instances of our fault-tolerant architecture, we can very well dedicate individual nodes to primarily service communications tasks, handle ADCS tasks, data storage and processing tasks, to scientific data acquisition, and to power management.

Naturally, the interfaces and workload of a single subsystem-node requires a much reduced amount of system resources as compared to "everyone" operating as fully independent, monolithic satellite-computers within a cluster[2]. Hence, each of these nodes can also be implemented on much smaller and more energy conserving FPGAs or ASICs, which each can still handle all or parts of the workload previously handled by a failed subsystem node. We interface compartments with subsystem peripherals (e.g., on- and off-chip driver logic, sensors, memories, ...), in a manner so that the compartment in each node can access peripherals of another device in a controlled and side-effect free manner. We decouple the interface logic from the immediate MPSoC design to assure this is possible, even if an entire compartment should fail. In consequence, we arrive at a satellite bus setup which is in spirit similar to integrated modular avionics (IMA) [22] and modern fault-tolerant avionics buses such as AFDX [23] used in atmospheric aerospace. However, we realize a chip-to-chip network in a compact CubeSat form factor, without the need to consider the impact of hundreds of kilometers of airplane cable harness on real-time guarantees.

Conclusions & Future Work

In this contribution we presented our practical experiences from realizing our prototype of the first truly fault-tolerant and autonomously operating avionics suite

[2]We mention this merely to make a point. In reality outfitting every single satellite subsystem with a *full* on-board computer instance would be unrealistic and wasteful regarding especially power budget and system complexity, and thereby also failure potential.

for CubeSats. In the process of going from several independent proof-of-concept implementations to prototype, we reworked the previously presented architecture [1]. We re-developed our architecture in a clean-slate approach, and optimized and streamlined the individual elements and fault-mitigation stages of our architecture. This enabled us to remove features that we considered mainly academically viable, which in turn enabled our implementation to achieve better real time capabilities.

Our initial demonstrator setup consisted of a mix of COTS parts and custom designed hardware. At the time of writing, we have condensed this design to a fully integrated custom PCB-based prototype. This design can then be directly used for characterization / radiation testing of the avionics suite, and is intended to be miniaturized for space use aboard SmallSats, Microsatellites, and even 2U+ CubeSats.

We hope the insight we shared in this contribution regarding the current state of our technology development efforts, prove to be of value to the avid reader. We especially hope that experiences with the Zynq MPSoC platform and latch-up mitigation are valuable to the small satellite community at large, and that they may aid in avoiding pitfalls that we discovered, some of which we expended considerable time and research on.

At this stage, the scope of our architecture and prototype implementation has begun to increasingly shift away from implementing just a fault-tolerant on-board computer. Our architecture evolved into an entire fault-tolerant satellite bus backbone, which increasingly covers all aspects of satellite bus design. We had originally planned to conduct these development steps as part of open academic research, but being stuck on an island for the duration of a global pandemic has interfered with this objective. Hence, we sought new ways to bring this new technology into the real world, and have now been making it practically and commercially usable for the small satellite community in the near future. At this point in time it is clear that we will in due time be able to field a prototype that can be flown aboard a future CubeSat. Where this is, however, we do not know yet.

ACKNOWLEDGMENTS

This research and contribution was not supported by a funding entity or institution. The work described in this paper has in its entirety been done by the first author. However, we consider writing a scientific/technical paper in first person singular inappropriate and would like to avoid adding feline co-authors, and therefore chose to use first person plural within this paper. The work described has been supported by N.M.M. in her private spare time, entirely outside of her capacity and funding as RIKEN special postdoctoral researcher.

REFERENCES

[1] C. M. Fuchs, N. M. Murillo, P. Chou, J.-J. Liou, Y.-M. Cheng, X. Wen, S. Holst, A. Tavoularis, G. Furano, G. Magistrati, K. Marinis, S.-K. Lu, and A. Plaat, "Fault tolerant nanosatellite computing on a budget," in *AIAA/USU Conference on Small Satellites*. AIAA, 2019.

[2] J. Bouwmeester, M. Langer, and E. Gill, "Survey on the implementation and reliability of CubeSat electrical bus interfaces," *CEAS Space Journal, Springer*, 2017.

[3] M. Langer and J. Bouwmeester, "Reliability of CubeSats – statistical data, developers' beliefs and the way forward," in *AIAA/USU SmallSat*, 2016.

[4] M. Swartwout, "You say "Picosat", i say "CubeSat": Developing a better taxonomy for secondary spacecraft," in *2018 IEEE Aerospace Conference*, 2018.

[5] J. Schwank *et al.*, "Radiation Hardness Assurance Testing of Microelectronic Devices and Integrated Circuits," *IEEE Transactions on Nuclear Science*, 2013.

[6] M. D. Berg, K. A. LaBel, and J. Pellish, "Single event effects in FPGA devices 2014-2015," in *NASA NEPP/ETW*, 2015.

[7] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," *Xilinx Application Note XAPP197*, 2001.

[8] K. Reick *et al.*, "FT design of the IBM Power6 microprocessor," *IEEE micro*, 2008.

[9] M. Hijorth *et al.*, "GR740: Rad-hard quad-core LEON4FT system-on-chip," in *Eurospace DASIA*, 2015.

[10] K. D. Safford *et al.*, "Off-chip lockstep checking," Jun. 26 2007, uS Patent 7,237,144.

[11] A. Fedi *et al.*, "High-energy neutrons characterization of a safety critical computing system," in *IEEE DFT*. IEEE, 2017.

[12] X. Iturbe *et al.*, "A triple core lock-step ARM Cortex-R5 processor for safety-critical and ultra-reliable applications," in *IEEE DSN-W*, 2016.

[13] M. Swartwout, "The first one hundred CubeSats: A statistical look," *Journal of Small Satellites*, 2014.

[14] C. M. Fuchs, "Fault-tolerant satellite computing with modern semiconductors," Ph.D. dissertation, Leiden University, 2019.

[15] M. Langer, C. Olthoff, J. Harder, Fuchs, C. M., M. Dziura, A. Hoehn, and U. Walter, "Results and lessons learned from the CubeSat mission First-MOVE," in *Symposium on Small Satellites for Earth Observation*. IAA, 2015.

[16] C. M. Fuchs *et al.*, "Towards affordable fault-tolerant nanosatellite computing with commodity hardware," in *IEEE ATS*, 2018.

[17] J.-T. Xiao, T.-S. Hsu, C. M. Fuchs, Y.-T. Chang, J.-J. Liou, and H. H. Chen, "An isa-level accurate fault simulator for system-level fault analysis," in *IEEE ATS*, 2020.

[18] C. M. Fuchs *et al.*, "Bringing fault-tolerant gigahertz-computing to space," in *IEEE ATS*, 2017.

[19] P. Maillard, J. Arver, M. J. Hart, and J. K. Jennings, "Using LVAUX mode in XQ ruggedized UltraScale+ devices for airborne systems," *Xilinx User Guide UG584*, 2019.

[20] ——, "Mitigation of single event latchup," 2017, uS Patent 9,793,899.

[21] C. M. Fuchs *et al.*, "Enhancing nanosatellite dependability through autonomous chip-level debug capabilities," in *Small Satellites, System & Services Symposium 2015 (4S)*. ESA, 2016.

[22] S. Cevher, A. Mumcu, A. Caglan, E. Kurt, M. K. Peker, I. Hokelek, and S. Altun, "A fault tolerant software defined networking architecture for integrated modular avionics," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE, 2018.

[23] Aeronautical Radio, INC, *ARINC Specification 664: Avionics Full Duplex Switched Ethernet (AFDX)*, 2005.