Fault-Tolerant Nanosatellite Computing on a Budget

Christian M. Fuchs*†, Nadia M. Murillo†, Aske Plaat*, Erik van der Kouwe*, and Todor P. Stefanov*
*Leiden Institute for Advanced Computer Science †Leiden Observatory;
Leiden University, The Netherlands

I. Introduction

Satellite miniaturization has enabled a broad variety of scientific and commercial space missions, which previously were technically infeasible, impractical or simply uneconomical. However, due to their low reliability, nanosatellites as well as light microsatellites are typically not considered suitable for critical and complex multi-phased missions and high-priority science. The on-board computer (OBC) and related electronics constitute a large part of such spacecraft, and have been shown to be responsible for a significant share of post-deployment failure [1]. Indeed, these components often lack even basic fault tolerance (FT) capabilities.

Due to budget, energy, mass and volume restrictions, existing FT solutions originally developed for larger spacecraft can not be adopted. In this paper we describe an multiprocessor System-on-Chip (MPSoC) that utilizes conventional hardware, providing FT for miniaturized satellites. The MPSoC is assembled from well tested COTS components, library logic (IP), and powerful embedded and mobile-market processor cores, yielding a non-proprietary, open architecture. Our key contribution is a fault tolerant OBC concept that consists only of extensively validated standard parts, and can be reproduced with minimal manpower and financial resources.

II. Background & Related Work

Aboard nanosatellites, subsystems are controlled by just one command & data handling system, whereas aboard a larger satellite these tasks are distributed across multiple dedicated payload and subsystem computers. This implies a varying OBC workload throughout a mission aboard nanosatellites, which traditional FT solutions only handle through over-provisioning. The tiled MPSoC design presented in this paper can efficiently handle faults through thread migration and partial reconfiguration. Major parts of our approach are implemented in software, allowing the OBC to deliver the desired combination of performance, robustness, functionality, or to meet a specific power budget. To enable strong FT with low-cost commodity hardware, we combine software-side fault detection, FPGA configuration scrubbing with other fault detection, isolation and recovery (FDIR) measures across the embedded stack.

Nanosatellites utilize almost exclusively COTS microcontroller- and application processors-SoCs, FPGAs, and combinations thereof [2], [3]. Due to manufacturing in fine technology nodes, and the use of extensively optimized standard IP, they offer superior efficiency and performance as compared to space-grade OBC designs. The energy threshold above which highly charged particles can induce faults (single event effects – SEE) in such components decreases, while the ratio of events inducing multi-bit upsets (MBU), and the likelihood of permanent faults increases. To adapt hardware-FT based concepts to protect such chips, additional FT-circuitry is required, inflating logic size and resulting in diminishing returns, limited scalability and low clock frequencies [4]–[6]. We can observe that traditional FT-concepts for space applied to COTS hardware yield no nanosatellite compatible architectures due to extreme cost and high overhead [7].

FPGA-based Soft-SoCs have been shown to offer excellent FDIR potential [8], as transients in large parts of the FPGA fabric can be corrected, and permanent faults may be compensated through reconfiguration [9] with differently routed configuration variants. Total ionizing dose becomes less of a problem with finer technology nodes used for recent generation FPGAs [10], [11]. Fine-grained, non-invasive fault detection in FPGA fabric, however, is challenging, and subject of ongoing research [12], [13]. Relevant FT-concepts thus rely on error scrubbing, which has scalability limitations and cover only parts of the fabric [12], [14]. In our technology project, however, we facilitate fault-detection in software using coarse-grain lockstep of weakly coupled cores.

Tiled architectures [15], [16] are often used for well paralellizable applications with many low-performance processor cores, but attempts have been made to utilize them for FT [17], albeit only for specially structured applications such as image processing, not for general purpose programming. Coarse-Grained Lockstep with weakly coupled cores [18] can be combined well with tiled MPSoC architectures, as the thereby implied compartmentalization into individual sub-SoCs enables or drastically simplifies the application of the utilized FDIR measures. This also enables us to inherit a considerable amount of testing from COTS components and logic.

III. A Hybrid Fault-Tolerance Approach

Conventional FT architectures require proprietary logic to facilitate fault detection and coverage. In contrast, the MPSoC described in this paper can offer strong FT using just from COTS components and standard logic. This is made possible through the use of the FT approach we presented in [18]. The high-level logic of this approach is depicted in Figure 1, and its consists of three interlinked fault mitigation stages implemented across the embedded stack:

Stage 1 implements forward error correction and utilizes coarse-grain lockstep of weakly coupled cores to generate a distributed majority decision across tiles. Fault detection is facilitated through application callback functions, without requiring deep modifications to an application or knowledge about intrinsics.

Stage 2 recovers defective tiles through reconfiguration. It assures the integrity of programmed logic and deploys configuration scrubbing as well as Xilinx SEM to correct transients in FPGA fabric. Its objective is to assure and recover the integrity of processor cores and their immediate peripheral IP through FPGA reconfiguration and the use of alternative configuration variants, thereby counteracting resource exhaustion.

Stage 3 is activated when too few healthy tiles are available, and re-allocates processing time to maintain reliability. To do so, thread-level mixed criticality is exploited, assuring sufficient compute resources are available to high-criticality applications by sacrificing performance of lower-criticality threads.

Further details including benchmark results are available in [18]. The main target in our project is the ARM Cortex-A53 application processor, which is today widely used in embedded and mobile-market devices.

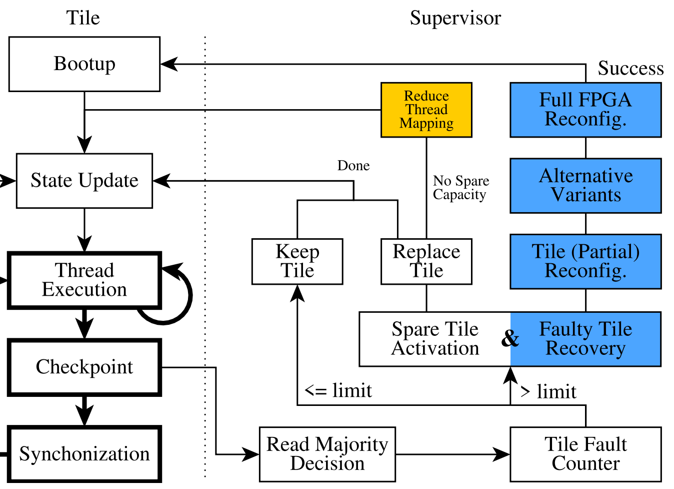

Fig. 1: Stage 1 assures fault detection and short-term coverage, while Stage 2 (blue) and 3 (yellow) counter resource exhaustion and adapt to reduced system resources.

However, this research is processor and ISA independent. In this paper, we describe an MPSoC design consisting of Microblaze processor cores and Xilinx Library IP, which can be reproduced in Xilinx Vivado 2017.1 and later.

IV. Supervision & Reconfiguration

Stage 1 can be implemented one a single chip, but we utilize an off-chip supervisor to facilitate FPGA reconfiguration and transient fault scrubbing in the running configuration. The outlined multi-stage FT approach puts only minimal load on the supervisor, and it can thus be again implemented using a traditional radiation hardened microcontroller. We deployed configuration error mitigation through Xilinx Soft-Error-Mitigation (SEM) in combination with supervisor-side scrubbing to safeguard logic integrity. However, SEM and scrubbing only detect faults in specific components of the FPGA fabric (e.g. not in BRAM), leaving significant parts of the design unprotected unless logic-side ECC is used.

These measures alone, thus, do not provide sufficient protection for fine-feature size FPGAs. Thus, we utilize the mentioned coarse-grain lock step functionality to detect faults and afterwards solve them using reconfiguration. We place tiles in separate configuration partitions to enable partial reconfiguration of individual tiles, without affecting the rest of the system.

To allow supervisor access to a tile and its address space, each tile is equipped with a AXI debug-bridge. The supervisor can induce a reset and execute self-test functionality run within a tile to detect faults in peripherals. It can also trigger an adjustment of a tile's thread allocation as part of Stage 1 and 3, making the MPSoC's computational performance, robustness and energy consumption adjustable at runtime.

V. Tile Architecture

Our MPSoC design implements multiple isolated SoC-compartments accessing shared main memory and operating system code. Even though the purpose and func-

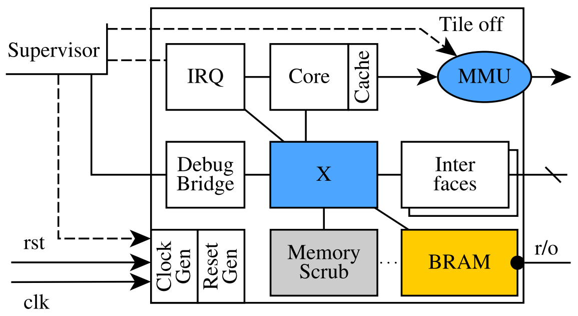

Fig. 2: The logic-side architecture of a tile. Access to local IP bypasses the cache, while access to global memory passes is cached for performance reasons.

tion of these compartments is different, the topology resembles a tiled architecture instead of a conventional MPSoC design, in which cores share infrastructure and peripherals. This topology allows to maximize Stage 1's fault-coverage capacity and allows task mapping for general-purpose software. Each such tile contains a processor core, local interconnect, and peripheral IP-cores and interfaces as depicted in Figure 2, resides in its own clock domain, and can be reset independently. Allocating a clock domain to each tile enables improved timing, reduce logic-overlap and interdependencies between tiles. Furthermore, we can then also utilize partial utilization and frequency scaling for each tile.

A tile executes a set of thread replicas, and its loss can be compensated by the rest of the system. To assure a failed tile can not cause performance degradation in the rest of the system e.g. by continuously accessing DDR or program memory, it can be disconnected off the global interconnect by the supervisor. Non-masked faults (due to radiation, ageing, and wear) disrupt the data or control flow of software running on a tile (state). Stage 1 builds upon this capability at the thread-level, as state difference can be detected by other tiles and often even by the malfunctioning tile itself [18].

Tiles are equipped with the same interfaces, with peripherals being mapped to identical locations in address ranges. The tile address space layout is uniform across the system and tiles are indistinguishable for software. Hence, application code and data structures are portable between tiles, simplifying thread migration drastically. This allows us to reduce the computational cost and complexity of the software-lockstepping.

Thread allocation and information relevant to the coarse-grain lockstep is stored in a dedicated dual-ported on-chip BRAM on each tile. One port is accessible to the tile's processor core, while the other is read-only accessible to the system, allowing low-latency information exchange between tiles without requiring inter-tile cache-coherence or main memory access.

VI. Interconnect Topology & Shared Memory

Figure 3 depicts the MPSoC's high-level topology with clock domains, reset lines and supervisor access facilities. Our MPSoC design utilizes an AXI interconnect in crossbar mode to allow tiles access to shared main and non-volatile memory controllers, though we are currently reworking our MPSoC to instead use a NoC [17].

Main memory is shared between tiles, as SD- and DDR memory controllers are too large and require too much I/O to instantiate for each tile. Each tile has full access to a segment of main memory, which is mapped to the same address range on all tiles (the MMU component in the figures). All tiles can access the main memory read-only to simplify state synchronization and IPC.

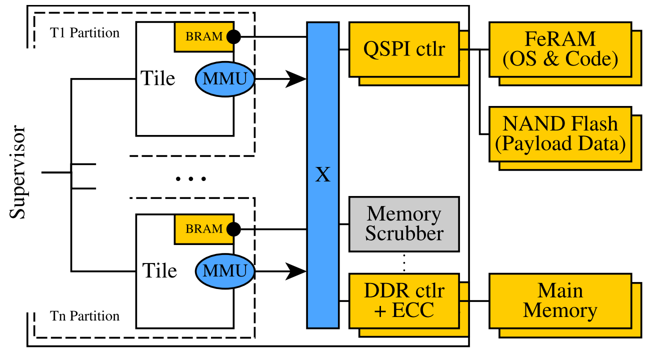

Fig. 3: Simplified topology of our tiled MPSoC design. Each tile exists in its own reconfiguration partition and therefore also clock domain, simplifying routing.

For nanosatellite missions to LEO, often only SECDED ECC support is required and readily available in library IP already, while basic error scrubbing can be facilitated in software. For critical, deep-space, and long-term missions, block coding should be used instead to compensate for the increased impact of SEEs and higher likelihood of MBUs in highly-density SDRAM. Reed-Solomon ECC as well as error scrubbers are available commercially, or can be assembled from open-source IP.

To safeguard main memory, FeRAM [19], and mass memory from SEFIs as well as permanent failure, these memories and their controllers are implemented redundantly to enable fail-over. This also enables further protective measures [20], and allows load distribution for timing critical main memory through segment interleaving, thereby avoiding performance limitations.

VII. I/O Sanitation

A fault resolved in Stage 1 may cause incorrect data to be emitted through I/O interfaces, an inherent limitation to coarse-grain lockstep [21] but acceptable for less critical nanosatellites. Larger spacecraft already utilize interface replications or even voting to protect I/O consistency, usually requiring considerable effort in hardware or logic to facilitate this replication. Our MPSoC architecture inherently provides interface replications by design, and requiring extra measures to be taken.

Further safeguards are necessary for very small CubeSats where interface replication is often undesirable, e.g., due to PCB-space constraints. Most embedded interfaces like I2C and SPI allow a simple majority decision per I/O line, which can be implemented on-chip through FIFO-buffers, as these have low pin count and run at relatively low clock frequencies. For packet-based interfaces such as Spacewire, AFDX, CAN, or Ethernet, no logic-side solution is necessary, as data duplication can be managed more efficiently at OSI layer 2+ [22].

VIII. Applications

To our knowledge, the MPSoC design presented in this paper is the first practical, non-proprietary, affordable architecture suitable for FT general-purpose computing aboard nanosatellites. The design presented in this paper was implemented on a Xilinx XCKU5P FPGA with modest resource utilization (28% LUTs, 33% BRAMs, 16%FFs, 5% DSPs) and 1.92W total power consumption, and we validated the effectiveness of our approach through fault-injection into RTEMS on ARM Cortex-A.

The relaxed cost, energy, and size constraints aboard microsatellites and larger spacecraft allow an implementation of our MPSoC spanning multiple FPGAs. A multi-FPGA MPSoC variant offers better scalability due to easier routing, can tolerate chip-level defects, and SEFIs to the globally shared memory controllers, these can be distributed to different FPGAs. Thread replicas can then be distributed across FPGAs, allowing non-stop operation even during full reconfiguration.

This approach and architecture could very well be implemented on ASIC without reconfiguration and Stage 2, and we see this as a "big-space" variant of our approach. An ASIC implementation offers lower energy consumption, and allows higher clock rates due to reduced timing and shorter paths. If manufactured in an inherently radiation hard technology such as FD-SoI [23], it would be less susceptible to transients and more robust to permanent faults. Due to the increased development and manufacturing costs, the resulting OBC would not be viable for most miniaturized satellite applications.

IX. Conclusions

The MPSoC design described in this summary was developed for miniaturized satellite use, as ideal platform for the software-side fault tolerance approach described in [18]. It utilizes fault tolerance measures across the embedded stack, and combines topological with software-side functionality to achieve the high level of reliability required to enable the use of nanosatellites in critical space missions. The architecture enables an on-board computer to adapt to varying performance requirements at run-time, allowing processing capacity, energy consumption or fault-coverage to be maximized.

This MPSoC can be implemented using only COTS hardware and widely available, pre-existing library IP, requiring no proprietary logic or costly space-grade processor cores. Its architecture includes a high level of spatial isolation for each processor tile, utilizing architectural features originally conceived for ManyCore systems to increase performance. Each tile functions as an stand-alone processing compartment with dedicated I/O, existing in its own clock domain, thereby minimizing shared resources and reducing routing complexity. Tiles were purposefully designed to best support thread-level coarse-grain lockstep of weakly coupled cores, while allowing partial reconfiguration independent of the rest of the system. The architecture was implemented successfully, and tested on current generation Xilinx Zynq/Kintex and Virtex FPGAs with 4, 6 and 8 tiles, and validated through fault-injection.

X. Acknowledgements

This MPSoC was developed in a 4-year ESA project. We would like to thank Gianluca Furano, Giorgio Magistrati, Antonios Tavoularis, Tomasz Szewczyk at ESTEC/TEC-EDD for their support and feedback.

References

[1] M. Langer and J. Bouwmeester, "Reliability of cubesats-statistical data, developers' beliefs and the way forward," in *AIAA SmallSat*, 2016.
[2] F. Kastensmidt and P. Rech, *FPGAs and Parallel Architectures for Aerospace Applications: Soft Errors and Fault-Tolerant Design*. Springer, 2016.
[3] R. Carlson, K. Hand, and E. Ozer, "On the use of system-on-chip technology in next-generation instruments avionics for space exploration," in *IEEE VLSI-SoC, revised paper*. Springer, 2016.
[4] S. Gupta *et al.*, "SHAKTI-F: A fault tolerant microprocessor architecture," in *IEEE ATS*, 2015.
[5] M. Pigno *et al.*, "A testbench for validation of DST fault-tolerant architectures on PowerPC G4 COTS microprocessors," in *Eurospace DASIA*, 2011.
[6] A. S. Jackson, "Implementation of the configurable fault tolerant system experiment on NPSAT-1," Ph.D. dissertation, Naval Postgraduate School Monterey, 2016.
[7] K. Reick *et al.*, "FT design of the IBM Power6 microprocessor," *IEEE micro*, 2008.
[8] M. Wirthlin, "High-reliability FPGA-based systems: space, high-energy physics, and beyond," *Proceedings of the IEEE*, vol. 103, no. 3, 2015.
[9] L. Bozzoli and L. Sterpone, "Self rerouting of dynamically reconfigurable SRAM-based FPGAs," in *NASA/ESA AHS*. IEEE, 2017.
[10] M. D. Berg, K. A. LaBel, and J. Pellish, "Single event effects in FPGA devices 2014-2015," in *NASA NEPP/ETW*, 2015.
[11] L. A. Tambara *et al.*, "Heavy ions induced single event upsets testing of the 28 nm Xilinx Zynq-7000 all programmable SoC," in *IEEE REDW*, 2015.
[12] M. Ebrahimi *et al.*, "Low-cost multiple bit upset correction in SRAM-based FPGA configuration frames," *IEEE Transactions on VLSI Systems*, 2016.
[13] F. Rittner *et al.*, "Automated test procedure to detect permanent faults inside SRAM-based FPGAs," in *NASA/ESA AHS*. IEEE, 2017.
[14] A. Stoddard, A. Gruwell, P. Zabriskie, and M. J. Wirthlin, "A hybrid approach to FPGA configuration scrubbing," *IEEE Transactions on Nuclear Science*, 2017.
[15] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: survey of current and emerging trends," in *DAC*. ACM, 2013.
[16] P. Meloni *et al.*, "System adaptivity and fault-tolerance in NoC-based MPSoCs: the MADNESS project approach," in *IEEE DSD*, 2012.
[17] N. K. R. Beechu *et al.*, "Hardware implementation of fault tolerance NoC core mapping," *Springer Telecommunication Systems*, 2017.
[18] C. M. Fuchs *et al.*, "Bringing fault-tolerant gigahertz-computing to space," in *IEEE ATS*, 2017.
[19] Z. Zhang *et al.*, "Single event effects in COTS ferroelectric RAM technologies," in *REDW*. IEEE, 2015.
[20] C. M. Fuchs *et al.*, "A fault-tolerant radiation-robust mass storage concept for highly scaled flash memory," in *Eurospace DASIA*, 2015.
[21] B. Döbel, "Operating system support for redundant multithreading," Ph.D. dissertation, Dresden University, 2014.
[22] Aeronautical Radio, INC, *ARINC Specification 664: Avionics Full Duplex Switched Ethernet (AFDX)*, 2005.
[23] M. Kochiyama *et al.*, "Radiation effects in silicon-on-insulator transistors with back-gate control method fabricated with OKI semiconductor 0.20 μm FD-SOI technology," *Nuclear Instruments and Methods in Physics Research*, 2011.