A Framework for Implementing Hardware Functions within a Fault-Tolerant Computer System

Tristan Running Crane*, Hezekiah Austin*, Chris Major*, Garrett Perkins†, Kris Allick*, Ben Macht*
Kieth Mecham‡, Alia Long‡, Clemente Izurieta†, and Brock LaMeres*
*School of Electrical & Computer Engineering, Montana State University, Bozeman, MT, USA
†Gianforte School of Computing, Montana State University, Bozeman, MT, USA
‡Idaho National Laboratory, Idaho Falls, ID, USA

***Abstract*—Space-based LIDAR systems require increased on-board signal processing to meet future mission requirements. While terrestrial computers can meet the performance demands of LIDAR processing, they do not work in space due to the harsh radiation environment. Ionizing radiation causes logical faults in terrestrial computers that prevent them from being used in critical space applications. While "hardening" techniques have been developed by NASA to allow electronics to work reliably in space, these techniques cause the computer systems to be cost prohibitive for most commercial satellite LIDAR systems. This paper presents a novel computing technology called "RadPC" that provides fault-tolerant computing in space using less-expensive commercial parts. The RadPC computer is implemented on a commercial Field Programmable Gate Array (FPGA). A unique feature of RadPC is that it allows the unused FPGA hardware resources to be bundled into hardware functions that are memory-mapped into the computer's architecture. This approach provides hardware-based signal processing functions on the same FPGA as a fault- tolerant controller that keeps the overall system operational in the presence of radiation. This paper will present the RadPC + hardware function architecture along with some examples of how this approach can support fault-tolerant signal processing in commercial satellite LIDAR systems.**

***Index Terms*—Small satellites, Logic gates, Field programmable gate arrays, Hardware acceleration, Resource Usage, Earth Orbit, Low Earth Orbit, Radiation Tolerance**

I. Introduction

Space-based computer systems that utilize on-board signal processing to meet mission requirements face challenges such as computational performance, power consumption, physical size of the system, and, above all, radiation tolerance. Terrestrial based computers can meet the signal processing demand but lack the other areas mentioned. Computers can be radiation hardened, but depending on the needs of the signal processing necessary to the system, the cost of this hardening process may be cost prohibitive for most missions. For that reason, there is a need for capable and affordable space-based computer systems. To address this inflexibility of space-based computer systems, this paper presents RadPC; a novel cost-effective fault-tolerant computer, with a framework for implementing hardware functions. The key idea is to take advantage of RadPC's unique implementation on a Field Programmable Gate Array (FPGA) and utilize unused FPGA resources as hardware functions that allow hardware-based signal processing. The main contribution of this paper was the development of the framework that enables these hardware functions to be memory mapped to RadPCs architecture, enabling a radiation tolerant computer system to be flexible in the signal processing that it provides. The organization of the remainder of the paper is as follows. Section II gives a background on RadPC technology and previous work on hardware functions. Section III presents the approach used to create the framework that is the basis of this paper. Section IV describes the evaluation of the framework. Section V summarizes the results and discusses further research. [1]

Identify applicable funding agency here. If none, delete this.

II. Background

A. RadPC

RadPC is a softcore processor that is designed in VHSIC Hardware Description Language (VHDL) and implemented as a logic design on a FPGA. RadPC's architecture is based on the RISC-V 32-bit integer instruction set and is RV32I compliant. RadPC's method of fault mitigation resides in quad modular redundancy (QMR). RadPC implements QMR with the RISC-V softcore as the central processing unit (CPU). RadPC's QMR when combined with a voter provides significant fault mitigation. RadPC takes further steps to ensure reliability by identifying which individual CPU faults and attempts to recover said individual CPUs with varying levels of severity. For low severity faults the CPU's internal registers are loaded with the correct values based on the non-faulted CPUs. For more severe faults within individual CPU's, RadPC uses partial reconfiguration (PR) of the FPGA to fully recover affected CPUs. The affected CPUs are then resynchronized, so the QMR system is operating in lockstep. In addition RadPC employs the use of memory scrubbers to facilitate a radiation tolerant computer. [2]

B. Prior Hardware Function Work

Utilizing unused FPGA resources for the purposes of hardware functions has been explored previously for use as a coprocessor integrated within the RadPC computer. This coprocessor design was a dedicated floating point unit meant to operate in parallel with RadPC while also taking advantage of RadPCs fault tolerant systems. The coprocessor was placed within a PR region of a Artix-7 100T FPGA, this region was

memory mapped to RadPCs architecture. This allowed for the coprocessor to swap its functionality through the use of PR and in this case between nine different floating point operations. The results of this prior work served as a proof of concept for a dynamically reconfigurable coprocessor within RadPCs architecture. A major limitation of this implementation was the hardware setup which restricted size of the systems design, the number of floating point operations that could be swapped with PR and the rate of coprocessor PR data transfer. This prior work has served as the foundation for this research and these limitations are what this paper will address to achieve a framework for implementing hardware function. [3]

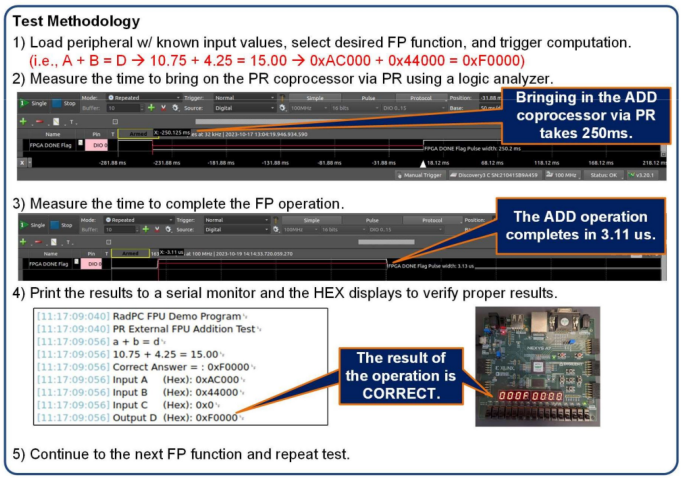

Fig. 1. Partial Reconfiguration Test Methodology

III. Approach

A. Hardware

The hardware that this framework was developed on was the RadPC Single-Board Computer (SBC) development board. The SBC development board contained hardware components that support the framework's implementation, this includes the following: Xilinx Artix-7 200T FPGA, TI MSP430 Microcontroller (MCU), and 4MB off chip non-volatile storage. The Artix-7 200T FPGA is what RadPC's logic design is implemented on, FPGA resources unused by RadPC will be utilized for the hardware functions implemented by this framework. The TI MSP430 MCU has many functions on the RadPC SBC that are vital to its fault mitigation system. The list of functions includes RadPC bootloading, partial and full reconfiguration of the FPGA, and monitoring of voltage and current measurements of the FPGA's 1.0V, 1.8V and 3.3V power rails through the use of its analog-to-digital (ADC) converter; it's these same functions that were leveraged to achieve this framework.

B. Software

The software written for the implementation of this framework was written in the C programming language for both the MSP430 MCU and the RadPC softcore processor. The RadPC's software development pipeline uses the open source RISC-V tools along with a proprietary RadPC plugin to allow for C programs to be written in a general purpose environment, in this case Microsoft's Visual Studio Code, then are adapted for RadPC then bootloaded with the MSP430 MCU. This RadPC C program is also what controls the partial reconfiguration of the hardware function that is memory mapped as a RadPC peripheral; this process will be described in a later section. The MSP430 program facilitates the partial reconfiguration when a command is sent from RadPC. Once a command is received the MSP430 retrieves a hardware function configuration that is stored in the off chip storage then initiates a partial reconfiguration of the FPGA via JTAG protocol. The MSP430's ADC peripheral is used to monitor power consumption on numerous voltage rails and in this case is used to further verify that the hardware function is in operation, these measurements of ADC data are sent to a secondary monitoring computer using the serial peripheral interface (SPI) communication protocol.

C. Execution

The method of bringing all these pieces together was through a test program that runs on the RadPC softcore processor. This starts with compiling the test program using the RISC-V compiler to generate a binary file. The RadPC software then takes the binary file and uploads it to the MSP430 over the SPI protocol. The MSP430 writes the binary file to the non-volatile memory chips, which also store the logic design of both the RadPC softcore and the hardware functions. The MSP430 then bootloads RadPC with the test program. This test program writes to registers that are inputs to the memory mapped PR region of the FPGA then repeatedly reads from an output register that is an output of the PR region. The program then sends a command to the MSP430 to begin a PR for another hardware function to be implemented. Once the MSP430 completes the PR a signal is sent from the MSP430 to RadPC allowing inputs to be sent to the new hardware function. This process is repeated to test all eight preset hardware functions. The hardware functions are floating point arithmetic functions and are as follows : addition, subtraction, multiplication, division, modulus, remainder, and reciprocal. Throughout this process, the MSP430 ADC peripheral measures the power measurements for the four power rails that supply the FPGA. This MSP430 sends this measurement data to a host computer using the MSP430 SPI peripheral to the host computer running the same RadPC software to upload the binary file. The measurement data is then processed using Python scripts and graphed which will be shown in the evaluation section.

IV. Results and Evaluation

Figures 1-8 presented in this section is the result of 10 runs of the test program averaged and graphed showing the voltage, current, and the resulting power consumption of each rail separated into sections for each hardware function PR'd. These figures are also paired for example, figure 1 and figure

2 both show the measurements of the 1.0 V rail, with figure 1 showing the addition, subtraction, and multiplication, while figure 2 shows the division, modulus, remainder, and reciprocal measurements, this . The significance of these plots lies in the difference in power consumption between each section of the graphs as these indicates consecutive successful PR of each hardware function executed in the test program. The exact power consumption for each hardware function tested was not evaluated in this paper, however the previous work mentioned did examine the specifics of each hardware function tested in this paper. [CITATION] In addition to these plots the output results of computations, figure 9, show successful computations, this provides further evidence that the .

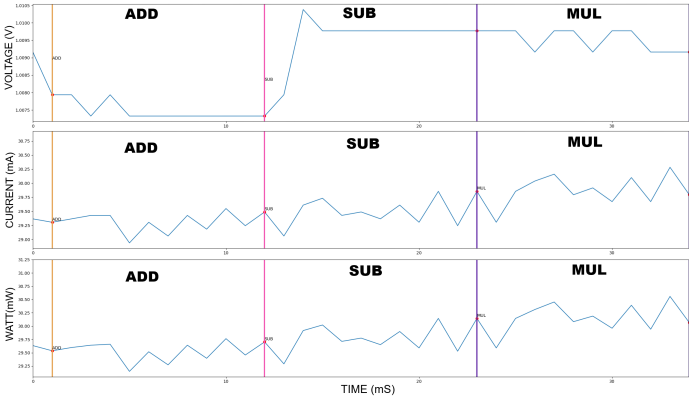

Fig. 2. 1.0 V ADC Measurements for ADD, SUB and MUL

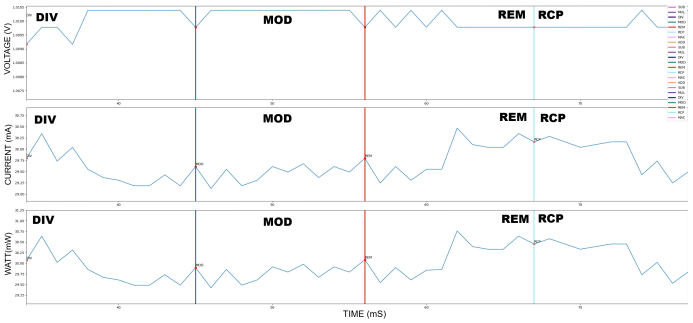

Fig. 3. 1.0 V ADC Measurements for DIV, MOD, REM and RCP

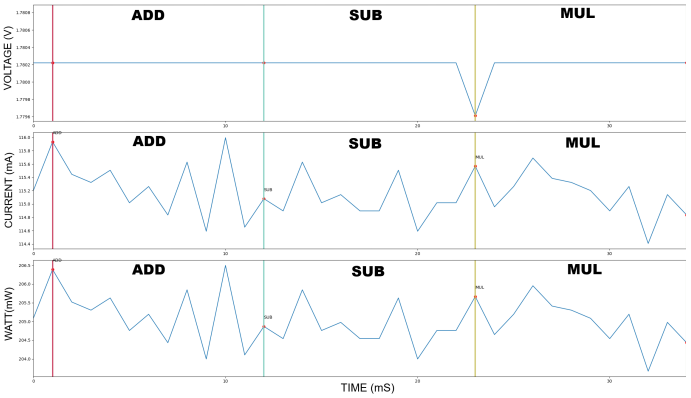

Fig. 4. 1.8 V ADC Measurements for ADD, SUB and MUL

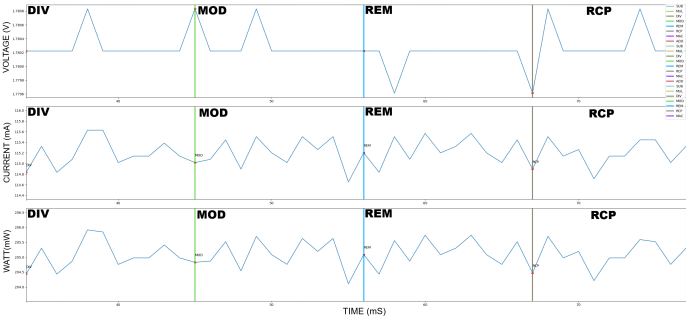

Fig. 5. 1.8 V ADC Measurements for DIV, MOD, REM and RCP

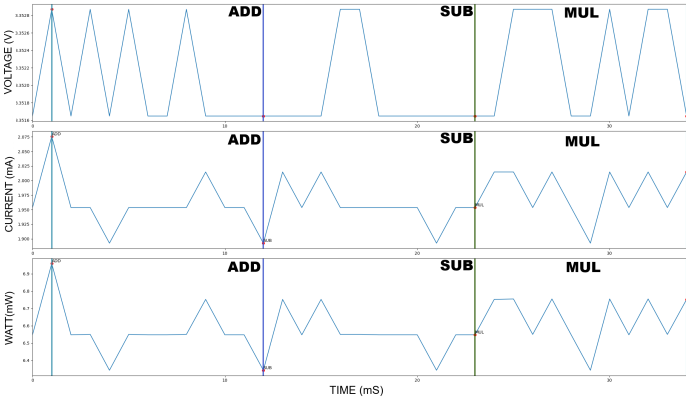

Fig. 6. 3.3(A) V ADC Measurements for ADD, SUB and MUL

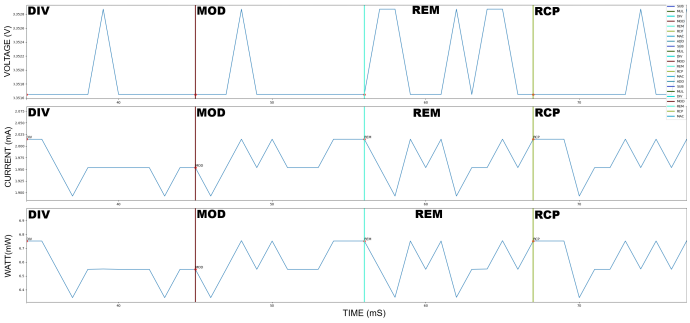

Fig. 7. 3.3(A) V ADC Measurements for DIV, MOD, REM and RCP

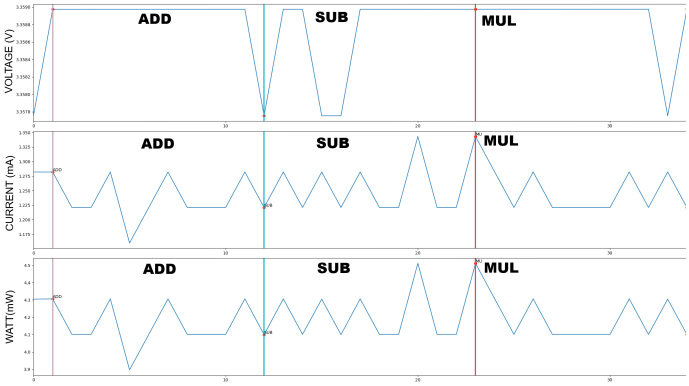

Fig. 8. 3.3(B) V ADC Measurements for ADD, SUB and MUL

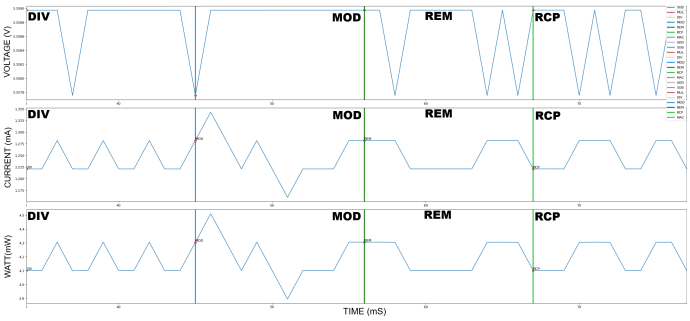

Fig. 9. 3.3(B) V ADC Measurements for DIV, MOD, REM and RCP

V. Conclusion

Based on the results presented in the previous section the goal of creating a framework for implementing hardware functions within a fault-tolerant computer systems was successfully demonstrated.

A. Application

This framework gives flexibility to space-based computer systems that face constraints by allowing for multiple hardware functions to be packaged with a fault-tolerant computer system without the added weight or volume required for dedicated hardware. The main advantage of this framework allows for hardware functions to be swapped in during the systems operation allowing for suitable mission critical signal processing to occur with underutilized hardware resources. A rather unique application for this framework technology lies in cybersecurity applications, specifically in disrupting cyberattacks known as side-channel attacks. [4]

B. Limitations

The current limitations with this framework are the current architectural implementations with the RadPC system are still limited. Currently the only way to read an output from a hardware function is through polling of it's output registers. The output register are memory mapped and can be interacted with by the RadPC softcore processor, but the framework is not tied to the RadPC interrupt system thus limiting the frameworks parallel processing potential.

C. Further Work

Based on the current limitations of this framework future iterations with the RadPC architecture could provide more seamless parallel processing of the inputs and outputs of the hardware function that is in use. Further development of this framework includes development of specialized hardware functions to be used for disruption of power monitoring of the computer system during processing of sensitive data. Most notably the handling of key data used encryption algorithms such as AES-256. [5]

Acknowledgments

The authors thank the Idaho National Labs (INL) for funding this project.

The authors thank Montana State University - Bozeman (MSU) for providing funding, lab space, equipment, and personnel for this project.

The authors thank Resilient Computing for providing personnel, equipment, and advisors for this project through their partnership with MSU.

References

[1] B. Osterloh, H. Michalik, S. A. Habinc and B. Fi-ethe, "Dynamic Partial Reconfiguration in Space Applications," 2009 NASA/ESA Conference on Adaptive Hardware and Systems, San Francisco, CA, USA, 2009, pp. 336–343, doi: 10.1109/AHS.2009.13.

[2] C. M. Major, A. Bachman, C. Barney, S. Tamke, and B. J. LaMeres, "Radpc: A novel single-event upset mit-igation strategy for field programmable gate array-based space computing," Journal of Aerospace Information Systems, vol. 18, no. 5, pp. 280–288, 2021. https://doi.org/10.1109/AERO63441.2025.11068533

[3] Austin, H. A., Major, C., Becker, Z., Crane, T. R., Allick, K., LaMeres, B. J. (2025). Dynamically Reconfigurable Coprocessor for Floating-Point Arithmetic Capability in Small Satellites. Proceedings - IEEE Aerospace Conference, 1–8.

[4] Zajic, A., Prvulovic, M. (2023). Understanding Analog Side Channels Using Cryptography Algorithms (First edition.). Springer Nature Switzerland AG. https://doi.org/10.1007/978-3-031-38579-7

[5] Mushtaq, M., Bricq, J., Bhatti, M. K., Akram, A., Lapotre, V., Gogniat, G., Benoit, P. (2020). WHISPER: A Tool for Run-Time Detection of Side-Channel Attacks. IEEE Access, 8, 83871–83900. https://doi.org/10.1109/ACCESS.2020.2988370

[6] B. J. LaMeres, "Fpga-based radiation tolerant computing," Journal of Aerospace Information Systems, 2012.

[7] B. J. LaMeres and C. Gauer, "Dynamic reconfigurable computing architecture for aerospace applications," 2009 IEEE Aerospace Conference, 2009.