Challenges in FPGA Design for Complex, High Performance Space Applications

Chinh H. Le
LeWiz Communications, Inc.
Sunnyvale, CA
ChinhL@LeWiz.com

Lynn R. Miles
NASA Goddard Space Flight Center
Greenbelt, MD
Lynn.R.Miles@nasa.gov

2023 IEEE Space Computing Conference (SCC) | 979-8-3503-4143-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SCC57168.2023.00016

***Abstract*— Field Programmable Gate Array (FPGA) technology has been used extensively in space applications where the natural radiation environment presents major challenges to electronic parts. Commercial FPGA technology is trending to deep nano-meter silicon processes, which impacts the availability of radiation resilience FPGA chips. Space systems require long timeframes for development and launch, and often the electronics and code may become obsolete or require updating before the system can be launched. FPGA logic/fabric-size continues to grow dramatically which allows and practically requires more and more IP cores to be integrated within a chip. New IP cores and tools will be needed to enable space designs with commercial FPGA technology to withstand radiation. This paper discusses the challenges in designing FPGA-based space systems and potential open-source and commercial technologies that will be useful to space application developers. It also references an ongoing FPGA based space telescope spectrometer design to discuss different aspects of complex FPGA design with mixed analog and digital circuits.**

Keywords*— *Field programmable, FPGA, IP core, TMR tool, space electronics, space chip, spectrometer, space telescope, open source, mixed signal design

I. Introduction

As the Field programmable Gate Array (FPGA) industry moves forward in time, more functionality will be embedded into individual chip. Larger FPGAs can fit complex processing arrays, network-on-chip capabilities and peripherals. As more functionalities are implemented within an FPGA, more on-chip resources (FPGA look up table, memories, routing channels, etc.) are required. Space applications using these functionalities present additional challenges. Intellectual Property (IP) Cores intended for use in space applications require fault tolerance to handle radiation effects and upsets, making them more complex than commercially available IP Cores intended for terrestrial applications [1]. FPGA vendors provide some IP cores for common use, but these are controlled and licensed by the vendors. When modifications of the cores are required, they cannot be easily modified by the users to suit the application.

Commercial FPGA devices are continuously and rapidly evolving. A family of FPGA device may become obsolete or outdated sooner than a space system or telescope can be launched. So, designing firmware that depends on a vendor specific FPGA IP core library (or making use of specific FPGA's capability, e.g., hard core) may cause the overall design to become obsolete if the particular device becomes obsolete. By creating platform-independent firmware, different FPGAs can be targeted. Designing platform-agnostic Register Transfer Language (RTL) code costs more time/money/effort on the front-end but saves time/money/effort later on. Furthermore, developing firmware in a modular fashion would enable reuse or quicker adaptation for different applications or platforms.

There are IP core suppliers available, but these are generally targeted for the Application Specific Integrated Circuit (ASIC) development market as that offers better monetary return on the licensing than the FPGA development market. For high performance applications, complex ASIC IP cores are very difficult to re-adapt for FPGA usage. Routing channels and FPGA memory resources are more restricted on FPGA devices than ASICs, making timing more difficult to meet.

Open-source cores provide an interesting option, but useful ones for space applications would require more complete code release with test benches, test vectors, and documentation good enough for user to modify the core to suit space applications and be able to re-verify. At times, open-source domain supporting materials can be severely lacking, thus making their use difficult. This paper discusses the challenges of complex, high performance FPGA designs for space applications. It covers available open-source cores and uses a design example of a subsystem with digital signal processing, hardware-based accelerators, RISC-V CPU(s), and 100Gbps Ethernet for a space telescope application to illustrate. Before discussing the issues, we will provide some background about space applications and FPGAs used.

II. Background

Spacecraft and space-borne science instruments both commonly utilize FPGAs and IP cores. FPGA based onboard processing solutions are often favored because they offer high degrees of design flexibility to suit specific applications, power efficient solutions, large sets of input and output connections, and high capacity for parallel processing capabilities. FPGA chips are ubiquitous across all sizes and classes of spacecraft, and can be found in both commercial and government satellites, including everything from flagship missions to relatively low

979-8-3503-4143-0/23/$31.00 ©2023 IEEE
DOI 10.1109/SCC57168.2023.00016

cost cubesat applications. The advent and availability of FPGA IP cores has dramatically decreased non-recurring engineering costs and design cycle time, and has greatly improved the accessibility and capability of commercial space industry, research institutions, and government agencies to implement sophisticated and interoperable FPGA solutions on new technology FPGA devices, and space flight missions.

Often times, National Aeronautics and Space Administration (NASA) FPGA designers encounter the need to create their own IP cores for a particular application. These cores are then used in support of a NASA mission, and often made publicly available via the NASA Technology Transfer Program. In one such example the Magnetospheric Multiscale (MMS) mission had a requirement to use the Remote Memory Access Protocol (RMAP) over its SpaceWire network. At the time, a suitable commercially available core could not be found, and so designers on the MMS mission had to create their own application specific RMAP IP Core [2]. This and many other useful IP cores that can be found through the NASA Software Catalog at https://software.nasa.gov/

There are multiple examples and publications that describe the use of FPGAs in onboard processing systems. The NASA SpaceCube family of high-performance reconfigurable processor systems represents one prominent example. One of the latest iterations in this family, the SpaceCube v3.0 Mini features the 20nm Xilinx Kintex Ultrascale FPGA combined with a radiation hardened FPGA monitor to help handle the radiation effects of space [3]. The inclusion of these FPGA devices allows the system to be highly reconfigurable and adaptable for many different space applications, missions, and platforms, and has greatly contributed to its success. The team building the SpaceCube v3.0 Mini explored several Triple Modular Redundancy (TMR) tool options including a built Xilinx TMR solution with Soft Error Mitigation IP Core, and the BL-TMR (BYU-LANL TMR Tool).

While NASA has authored and published some IP cores, that list remains a small subset of all the IP cores that are useful and necessary for modern space flight FPGA designs. As we've discussed there are many different interface and protocol IP cores that could be pulled in to suit a specific application. TMR tools, fault tolerance, and error mitigation tools are available through both commercial entities and open-source repositories. And newer high speed device interfaces such as JESD204B require fairly complicated and timing sensitive IP cores that would be very challenging for users to create on their own. Fortunately, there are commercial and open source IP core solutions available, however it is left up to the users to intelligently select the cores that offer the level of performance and fault tolerance required for their application.

FPGA chips are produced mainly by AMD/Xilinx, Intel/Altera, Microchip/Microsemi, Lattice and others. In the future, eventually embedded FPGA within-a-chip devices will be used for space applications. Here, we will focus primarily on FPGA devices that are commercially available. Xilinx and Intel offer programmable logic based on SRAM technology. Other vendors also offer programmable logic based on non-volatile memory technology. FPGA devices primarily divide into 3 categories: high-end, mid-range and low-end. FPGA chips offer programmable logic, but also embedded memory blocks, Digital Signal Processing (DSP) acceleration, hard/soft processor(s) and peripherals on-chip. The high-end devices such as Xilinx Virtex Ultrascale+ or Intel Agilex has large FPGA fabric >4M look-up tables (Xilinx) or >2M logic elements (Intel) and high speed Serdes >50Gbps performance, and memory interface supporting various DDRx as well as High Bandwidth Memory (HBM) technologies. These can be used to design just about any complex system. The mid-range devices such as Kintex or Aria chips offer less fabric but at lower price. The low-end (such as Zynq) has built-in ARM processors, I/O peripherals as well as programmable logic at affordable cost.

Xilinx also offers space grade FPGA devices. Its Virtex 5 65nm products were designed for radiation hardened applications and have been used on Mars Rovers [4]. Newer Kintex Ultrascale 20nm products offer decent radiation tolerance up to 120krad of Total Ionizing Dose [5] but still requires additional mitigations to handle Single Event Effects. From the industry trend, it's increasingly likely that future space grade FPGA offerings will come from a subset of the commercially available offering. The volume of chips used for space is very small compare to the volume of commercial applications.

FPGA boards are also commercially available in various forms. Certain form factors tend to be favored for compatibility with space applications. These range from OpenVPX 3U/6U that are SOSA/VITA/FACE standard compliant to compact PCI form factor with different high-end, mid-range and low-end FPGA devices with different timing speeds available.

III. Open Source

There are many open-source cores available in Git-Hub and other places. These can be categorized as follows: university (or school) created, corporate created, and others. University created has 2 sub-types: those created by government grants and those created by the university for research and/or course work. Open-source created by grants are usually very useful and more complete. Examples are from Eth Zurich, Princeton, and Utah. Some are actually validated in silicon at 22nm (or other silicon node), while some are FPGA verified. Course work created open-source is generally not very useful, not optimized, and likely abandoned down the road. The draw-back of university created open-source is that the researchers, including the professor(s) managing the projects, may potentially move on to other jobs and positions. This created a brain drain and may be lacking in support especially when the funding ran out.

The Defense Advanced Research Projects Agency (DARPA) from US Department of Defense (DoD) in recent years created a couple programs to encourage open-source development. It funded several universities and companies (LeWiz Communications included, search for LeWiz on Github

[6]) to develop and released to open source with IP cores, tools with complete source code and support materials. (In Europe, European Space Agency and European government also funded similar efforts recently, e.g., European Processor Initiative).

The second category of open-source cores are created by corporations. These also have free open-source cores – but not as many. Some are released under not for commercial purpose, i.e., can be used for research but not for sale. The support for these varies, but generally requires licensing or payment for support or support materials. The third category of cores are created by individuals for the purpose of selling a small core or being published to gain consulting benefits. Some are released with missing files or missing information making their use limited without paying for consulting cost. Recently, organizations such as Open Hardware Group started efforts to create open hardware. But this is relatively in early stages.

Open-source code is vital for the progression of the programmable logic device community. Open forums are created specifically to address unique design issues. When developers are faced with design issues, they post questions in search of solutions. If multiple people stumble upon the same documented issue, eventually someone may provide the answer. As more eyes begin to focus on the subject at hand, and begin to discuss/communicate, the probability of finding a solution in open forums would be greater. For newer or complex devices, FPGA vendors provide limited technical support and documentation for their software/hardware chip and tool environments. This can be extremely frustrating when one has an aggressive deadline to meet on an already complicated design. FPGA classes and open-source help may be helpful to FPGA developers especially for large complex devices such as the AMD Xilinx RFSoC [7] with a mix of analog and digital peripherals on an FPGA device. The level of support available for a specific device tends to vary among part manufacturers and product lines, with some devices being better supported than others, especially for simpler FPGA devices. In one example, end users noted that when the Microchip RTG4 (flash-based FPGA chip) development board was first released, Field Application Engineers were immediately available to answer questions and assist with the hardware verification process.

With the vast availability of code in open source, developers are challenged to determine which cores are useful for their application, and which would require more effort to modify than being useful. A list of known useful cores would be very helpful to users, but such a useful list would require serious research and effort to create, validate and maintain. Maintaining such a list is not easy as the open-source world is continuously changing. There are a couple of lists that are known to space hardware developers to be helpful. Examples include: https://github.com/aolofsson/awesome-opensource-hardware which lists from accelerators to small cores and https://github.com/jimbrake/cpu_soft_cores/blob/main/uP_notables_221227.pdf which lists soft CPU cores of various types RISC, CISC CPUs. These are helpful, but they are the tip of the iceberg. It would require much more coordinated effort than these to categorize open-source cores to be useful to potential users of different applications.

IV. Space Design Methods for FPGAs

Newer generations of FPGA chips such as Intel Agilex [8], AMD Ultrascale+ have support for enhancing the reliability of the device under soft-error conditions. These devices use a soft-error mitigation controller [9] to detect and correct single bit error in its configuration memory. The latency for detection, however, can be very high and may not be useful for all applications. This latency is a function of the FPGA size and can be in the range of 9mS to 57mS long. For block Random Access Memories (RAMs), distributed RAMs, flip-flops, state machines, and other functional components, other mitigation design techniques are still required.

RAMs (or memory in general) can be designed with error detection and correction logic to detect and correct single bit errors. Scrubbing can further clean out the errors within the memory before the memory location is used. This is usually done with large memory blocks such as a Central Processing Unit (CPU) cache or external memory.

Flip-flops, state machines (or even combinational logic or buses), tri-modular redundancy (TMR) can be implemented to mitigate soft-errors [10]. TMR triplicates a function and uses voting logic to output the correct result. For single bit errors, the correct result is the matching of 2 out of 3 outputs, i.e., majority voting. TMR in combination with scrubbing offers very high reliability. Logic scrubbing may not always be possible with dynamic logic involving feedback paths, i.e., state machines. TMR offers continuous operation upon encountering errors but also incurs additional logic, routing space, and power. Thus, different space missions may use different level of TMR. As examples, for short duration mission in low Earth orbit, some designs may not implement TMR or only partial TMR in critical blocks or for flip-flops only. For long duration, deep space missions, higher level of TMR implementation may be required. The design decision is very application dependent.

TMR tools are available to assist in TMR implementation. One open-source example is the SpyDrNet-TMR https://github.com/byuccl/spydrnet-tmr. Another open-source example is the BYU-LANL EDIF tool, but it seems lacking in support. Example of commercial TMR tools include Precision Hi-Rel (Siemens/Mentor) [11] and Synplify Premier (Synopsys) [12]. (Xilinx used to offer a TMR Tool for its ISE design tool but it is no longer available in the newer Vivado Synthesis Design Suite.) As expected, the commercial tools offer more features and support more families of FPGA chips and manufacturers. The tools offer DO-254 design assurance for safety critical applications. For state machines, they provide support for 1-hot, binary, or gray code state encoding and safe finite state machine – where a state machine may be forced into a reset state or a user-defined error state for error handling. They also provide support for different TMR implementation levels: local, distributed or global, where local is for sequential elements only, distributed is for sequential and combinational elements, and global implements TMRs for global buffers,

sequential and combinatorial circuits [11]. In addition, the tools support error injection to enable validation of system response due to single bit error.

Next, we will use a spectrometer design case to discuss different aspects of complex FPGA design with a mix of analog and digital (RFSoC design).

V. Practical Usage

The figures below show the top-level system diagram (**Fig 1**) and the detailed algorithm of an RF digital spectrometer (Fig 2). It is a digital system that interfaces with an RF front-end electronics system and is used to compute the power spectral density of signals of interest for spectrum identification and classification of microwave phenomena. The front-end used in this example is a Microwave Kinetic Inductance Detector (MKID) Array. It has an array of resonators at specific frequencies that are excited when a frequency of that same value is transmitted across that array.

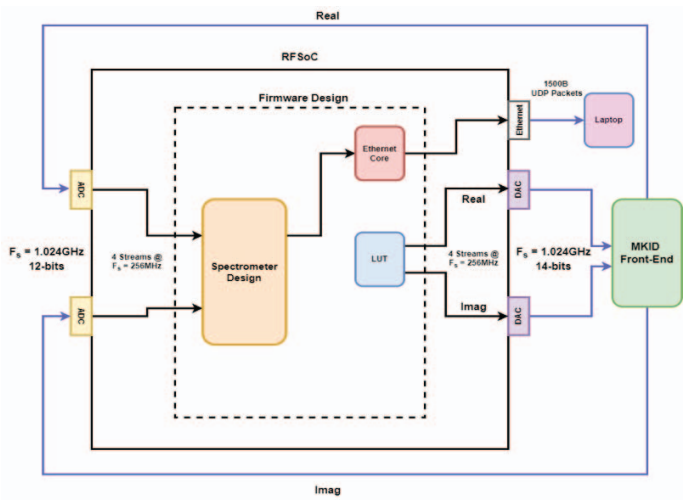

Fig 1: MKID Test Setup for top level system (LUT contains the RF front end real/imaginary stimuli)

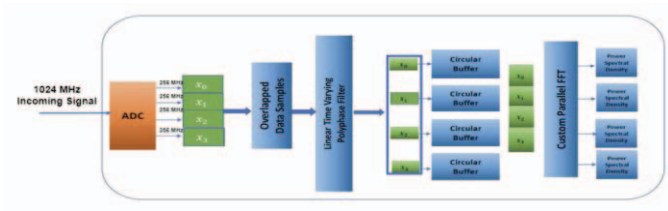

Fig 2: Detailed Spectrometer Block Diagram

At the beginning of any design is the proof of concept. This is where the design is proven to be conceptually achievable given all the necessary and required parameters. In this scenario, a 3-step verification process was used to verify the design (MATLAB Simulation → RTL Simulation → Hardware Validation).

MATLAB Simulation

First, a MATLAB model is designed for each VHDL module to verify functionality or complex algorithms. Verifying each module individually makes debugging easier since any potential bugs can be isolated. Next, a top-level design connecting each module is used to verify functionality collectively. The MATLAB model will act as a reference for the design.

RTL Simulation

After the MATLAB model is designed and verified, a VHDL algorithm is developed. The same input signal that was used for the MATLAB simulation should be used for RTL simulation to create an apples-to-apples comparison. The data needs to be quantized to the correct bit length and is imported to the testbench as a text file. The simulation tool (e.g., ModelSim, ISim) executes the testbench and exports the outputs to text files. MATLAB verifies functionality through post-processing. RTL simulation should match MATLAB model/hardware outputs within some tolerance. The difference, if any, should be from rounding and/or truncation. If the results differ greatly, then something else may be wrong. **Fig 3** illustrates this process.

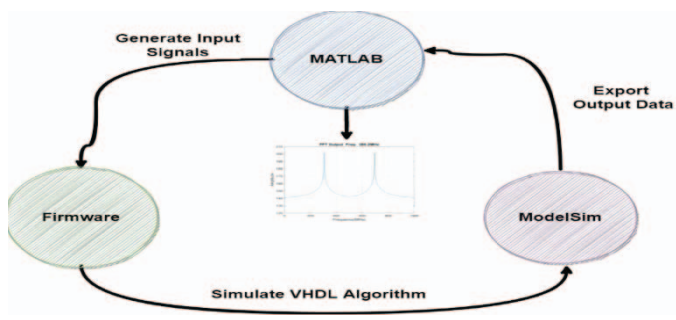

Fig 3: RTL Simulation

Hardware Validation

Once the RTL model is designed and verified through simulation, firmware is imported onto hardware (RFSoC FPGA). Analog input stimuli can be sent over Ethernet, read from a pre-loaded look-up-table (LUT) in memory, or digitized by an ADC from some front-end analog source (Arbitrary Waveform Generator (AWG) or Signal Generator). The same input signal/data that was used for the MATLAB should be used for sending via Ethernet or the pre-loaded LUT. Firmware is then loaded onto the FPGA hardware. Output data may be exported via Ethernet or text files using onboard logic analyzers. Hardware validation outputs should match MATLAB model/RTL simulation within some tolerance. **Fig 4** illustrates this process.

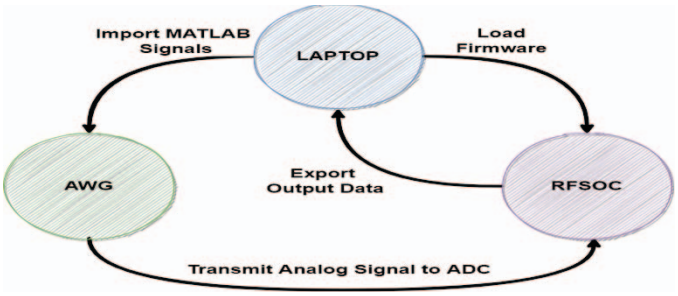

Fig 4: Hardware Verification

With the ever-growing science goals that require broader bandwidth and improved resolution, scientists are requesting digital back-end systems to provide high-resolution and wide-bandwidth for current and future science instruments. However, the hardware capabilities don't always agree with the science requirements/requests. One of the main design obstacles is how to efficiently sample, process, and analyze the data from end to end considering hardware limitations such as FPGA clock rate, FPGA resource utilization, and output data rate.

Current FPGA technology cannot run at very high clock rates. Most complex designs would peak at sampling clock rate (**Fs**) of ≤400MHz. Effectively processing ultra-wide bandwidth input analog signals (≥1 GHz) within an FPGA is a challenge. Modern ADCs use parallelization and require implementation of JESD204B standard interfaces and IP cores to meet the requirements of both high channel count and/or wide instantaneous signal bandwidth applications (**Fig 5**). Having N parallel streams lowers Fs by a factor of N or (**F_S/N**). Lower clock rates use less resources and make timing easier to meet with speed-limited FPGA devices.

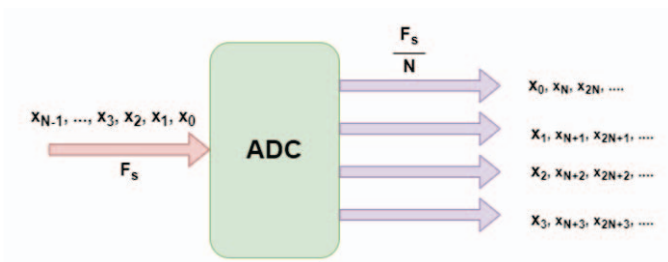

Fig 5: ADC Channelization

Oversampling will save FPGA resources. Using multiple clock rates allows resource sharing. A design can use up to N times (N is the clock factor) as less resources (multipliers, adders, memory) sampling off the faster clock and cycling through like a commutator (**Fig 6**). A commutator can

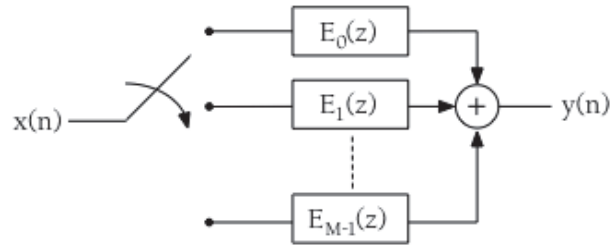

Fig 6: Commutator

be thought of like a card dealer. If there are four players in a card game, each player (**E_0 –E_{M-1}**) would receive the cards at a rate of $\frac{F_S}{4}$, whereas the dealer (**x(n)**) deals the cards at a rate of **Fs**. Using **Fig 7** as an example, the fast clock is twice the speed of the slow clock (Fast = 200MHz, Slow = 100MHz). This would allow one multiplier/adder/logic element to be used twice in reference to the slower clock and not miss a sample, ultimately saving resources.

This design combines different techniques and methodologies to overcome certain hardware design limitations/challenges while still meeting the increasing demands of digital back-end readout systems to create a working end-to-end product. There are parallel streams at both the ADC/DAC peripherals that take the higher sampled data stream and generate N smaller data streams at a lower rate. There's a LUT that acts as an input stimulus to the RF Front-End. An Ethernet IP core is required to transmit data out to a processing system for MATLAB post-processing. The MKID generates an analog signal that the ADC will digitize, and the digital spectrometer will process. Open-source code was used in the Vivado Software Development Kit environment so that the RFSoC ADC/DAC cores could be used.

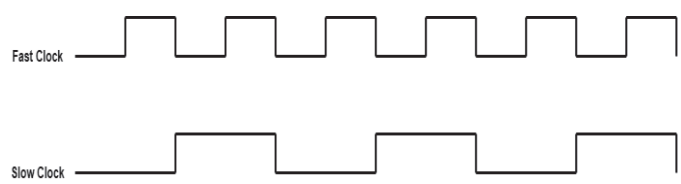

Fig 6: Multiple Clock Rates (2-to1 Ratio)

VI. Conclusion

FPGA technology will continue to play an important role in space environment applications for aerospace and defense. As deep nano-meter silicon technology advances, commercial FPGAs will be the dominant type of devices available for the majority of users. We must find ways to use commercial technology for space exploration. New IP cores with enhanced capability and functionality will be required to support tolerance in the radiation environment of space. New, lower cost tools will be required for FPGA code developers to add support functions that can withstand radiation effects and be certified for safety critical applications. As FPGA chips become more complex and integrated, advanced IP cores will be needed to support digital and analog mixed signal designs within a

single FPGA chip. The role of open-source cores and tools will also continue to be critical to the space industry. We must find ways to enable organizations to sustain an open-source business model.

VII. Acknowledgment

We are very grateful for Christopher Green's (NASA: GSFC) contributions to the applications section of this paper. We also would like to thank Christopher Wilson (NASA: GSFC) for providing valuable inputs to the TMR tool section.

VIII. References

[1] M. Berg and K. LaBel, "Challenges Regarding IP Core Functional Reliability," Microelectronics Reliability and Qualification Working Meeting 2017.

[2] O. Haddad "Remote Memory Access Protocol Target Node Intellectual Property," NASA Tech Briefs, August 2013

[3] C. Brewer, N. Franconi, R. Ripley, A. Geist, T. Wise, S. Sabogal, G. Crum, S. Heyward, C. Wilson, "NASA SpaceCube Intelligent Multi-Purpose System for Enabling Remote Sensing, Communication, and Navigation in Mission Architectures," 34th Annual Small Satellite Conference, 2020

[4] A. E. Johnson, S. Aaron, H. Ansari, C. Bergh, H. Bourdu, J. Butler, J. Chang, R. Cheng, Y. Cheng, K. Clark, D. Clouse, R. Donnelly, K. Gostelow, W. Jay, M. Jordan, S. Mohan, J. F. Montgomery, J. Morrison, S. Schroeder, B. Shenker, G. Sun, N. Trawny, C. Umsted, G. Vaughan, M. Ravine, J. Schaffner, J. M. Shamah, J. Zheng, "Mars 2020 Lander Vision System Flight Performance," American Institute of Aeronautics and Astronautics SciTech 2022 Forum

[5] P. Maillard, J. Barton, M. J. Hart, Y. P. Chen, M. L. Voogel, "Total Ionizing Dose and Single-Events characterization of Xilinx 20nm Kintex UltraScale™," 2019 19th European Conference on Radiation and Its Effects on Components and Systems (RADECS)

[6] LeWiz's open source releases, https://github.com/lewiz-support, LeWiz Communications, Inc., 2/2023

[7] Zynq UltraScale+ RFSoC Data Sheet: Overview (DS889 v1.13), AMD/Xilinx, January 7, 2022

[8] Intel® Agilex™ SEU Mitigation User Guide (UG-20253), Intel, 2022

[9] LogiCORE IP Soft Error Mitigation Controller v3.1 Product Specification (DS796), AMD/Xilinx, October 19, 2011

[10] Precision® Hi-Rel Advanced FPGA Synthesis Datasheet, Siemens/Mentor Graphic, 2018

[11] Wirthlin, M., "High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond," Proceedings of the IEEE, vol. 103, no. 3, Mar. 2015, pp. 379-389.

[12] FPGA Design Solution for High-Reliability Applications (Brochure), Synopsys, Inc. 2015