Design and Evaluation of a RISC-V based SoC for Satellite on-board Networking

Ánder Núñez, Armando Astarloa and Jesús Lázaro
Departamento de Tecnología Electrónica
University of the Basque Country (UPV/EHU)
Bilbao, Spain
armando.astarloa@ehu.eus

Mikel Rodriguez and David Modroño
System-on-Chip *engineering*
Ribera de Axpe 50 Erandio, Spain
mikel.rodriguez@soc-e.com

***Abstract*—SpaceWire is a communication protocol that has become widely used in spacecraft for connecting instruments to data processors, mass-memory, and control processors. Field-Programmable Gate Arrays (FPGAs) have been a popular choice for implementing SpaceWire nodes due to their flexibility to meet unique requirements of each program or product. This paper presents a comparative study of two implementations of SpaceWire nodes, based on two different FPGA technologies, AMD-Xilinx SRAM-based and Microchip (Microsemi) FLASH-based. The study compares the resource requirements and estimated power consumption of both implementations, using the same HDL SpaceWire IP core, with the SRAM-based one incorporating a 32-bit Microblaze soft-CPU, and the FLASH-based one using a 32-bit RISC-V CPU. The obtained results are compared, and the paper concludes that FLASH-based FPGAs are more suitable for applications that require high reliability, tamper resistance, and fast, reliable restarts. In contrast, SRAM-based FPGAs are preferred in applications that require high performance and reconfigurability. The study shows that both FPGA technologies are capable of implementing SpaceWire nodes effectively and efficiently, and designers can choose the technology that best suits the specific requirements of each project.**

***Index Terms*—SpaceWire, Spacecraft, On-Board, RISC-V, SRAM FPGA, FLASH FPGA,SoC**

I. Introduction

SpaceWire [1] is a widely-used on-board data-handling network for spacecraft, connecting instruments to data processors, mass-memory and control processors, based on the IEEE 1355 communication standard. FPGAs are a popular choice for implementing SpaceWire nodes due to their flexibility to meet unique requirements of each program or product. However, designers of such systems must consider power consumption and reliability, particularly in terms of resistance to Single Event Upsets (SEUs).

This research compares two implementations of SpaceWire nodes in terms of FPGA resource requirements and estimated power consumption. The first implementation is based on AMD-Xilinx SRAM-based technology, while the second one uses Microchip (Microsemi) FLASH-based technology. Both implementations use the same HDL SpaceWire IP core, with the SRAM-based one incorporating a 32-bit Microblaze soft-CPU, and the FLASH-based one using a 32-bit RISC-V CPU.

The remainder of this paper is organized as follows: Section II introduces the SpaceWire protocol and the two FPGA technologies used for the implementations. Section III presents the high-level block diagram of a generic SpaceWire node. Section IV provides details of the two implementations developed, with the first using a 32-bit Microblaze CPU on an AMD-Xilinx SRAM-based FPGA, and the second using a 32-bit RISC-V CPU on a Microchip (Microsemi) FLASH-based FPGA. Section V compares the obtained results in terms of FPGA resources and power consumption, and the paper concludes with the Conclusions section.

This work has been supported, within the fund for research groups of the Basque university system IT1440-22, by the Department of Education and, within SOC4CRIS KK-2023/00015 and COMMUTE ZE-2021/00931 projects, by the Elkartek and Hazitek programs, both of the Basque Government; the latter also by the Ministerio de Ciencia e Innovación of Spain through the Centro para el Desarrollo Tecnológico Industrial (CDTI) within the project IDI-20220543, and through the Fondo Europeo de Desarrollo Regional 2014-2020 (FEDER funds).

II. State-of-the-art

A. SpaceWire standard

SpaceWire is already in orbit or is being designed into more than 100 spacecraft. It provides high-speed (2 to 200 Mbit/s), bi-directional, full-duplex, data links which connect together SpaceWire enabled equipment.

The SpaceWire standard has several purposes, including facilitating the construction of high-performance on-board data handling systems, reducing system integration costs, promoting compatibility between data handling equipment and subsystems, and encouraging the re-use of data handling equipment across multiple missions.

By implementing the SpaceWire standard, compatibility is ensured at both the component and subsystem levels. This means that instruments, processing units, mass-memory units, and telemetry systems developed for one mission can be used in any other mission, resulting in cost and time savings, improved reliability, and increased scientific output within a limited budget.

Since the SpaceWire standard was written and published by the University of Dundee in 2003, it has been adopted by ESA, NASA, JAXA and RosCosmos for several missions, as it has been used for science, Earth observation and different spacecraft. SpaceWire has been used in missions like Gaia, ExoMars rover, GOES-R, Astro-H, etc. SpaceWire is a computer

network designed to connect together high data-rate sensors, processing units, memory devices and telemetry/telecommand sub-systems onboard spacecraft. It is based on two standards: IEEE 1355-1995 [2] and ANSI/TIA/EIA-644 [3]:

- **IEEE 1355-1995**: IEEE 1355-1995 is a data communications standard for Heterogeneous Interconnect (HIC).The protocol was designed for a simple, low cost switched network made of point-to-point links. This network sends variable length data packets reliably at high speed.
- **ANSI/TIA/EIA-644**: TIA/EIA-644, otherwise known as LVDS, is a signaling method used for high-speed, low-power transmission of binary data over copper. This signaling technique uses lower output-voltage levels than the 5-V differential standards (such as TIA/EIA-422) to reduce power consumption, increase switching speed, and allow operation with a 3.3-V supply rail. The LVDS current-mode drivers create a differential voltage (247 mV to 454 mV) across a 100-Ω load.

SpaceWire standard involves wires and physical switches, electric propierties and logic protocols, i.e. Transport Layer. Levels are defined:

- **Physical layer:** SpaceWire connectors, wires, integrated circuits, etc.
- **Signal layer:** Signal codification, voltage levels, noise margins and data signal ratios.
- **Character layer:** Data and control characters are used to lead the data flow through SpaceWire link. Each data character (also called "normal data" or "N-Chars") are formed by 8 bits, whilst control characters are formed by 4 bits.
- **Exchange layer:** Link initialization protocols, flow control, error detection and error recovery.
- **Packet layer:** Definition of how the data is split in different packets to send them through the SpaceWire link.

B. FPGA Technology

An important benefit of utilizing HDL IPs to implement the SoCe SpaceWire Core is the flexibility to leverage different FPGA technologies to meet the unique requirements of each product or program.

SRAM-based FPGAs and Flash-based FPGAs [4], [5] are two different types of FPGA architectures. Both FLASH-based and SRAM-based FPGAs have important applications in the aerospace and defense sectors. FLASH-based FPGAs are preferred in aerospace and defense systems that require a high degree of security, reliability, and resistance to tampering. This is because FLASH-based FPGAs have non-volatile memory, which makes it more difficult for attackers to tamper with the device's configuration or extract sensitive data. FLASH-based FPGAs are also suitable for applications where the device needs to be restarted quickly and reliably, such as in safety-critical systems.

On the other hand, SRAM-based FPGAs are preferred in aerospace and defense systems that require high performance and reconfigurability. SRAM-based FPGAs can be reconfigured on-the-fly, allowing for real-time changes to the device's functionality. This makes them suitable for applications such as software-defined radios, where the device's configuration needs to be adjusted frequently to accommodate changing communication protocols.

Some key differences between these two types of FPGAs are summarized below:

- **Configuration:** SRAM-based FPGAs are configured every time they power up, while Flash-based FPGAs retain their configuration across power cycles. This means that Flash-based FPGAs can be used in applications where the device needs to be restarted quickly and reliably, such as in safety-critical systems.
- **Performance:** SRAM-based FPGAs generally have higher performance and lower latency than Flash-based FPGAs. SRAM-based FPGAs can be reconfigured on-the-fly, while Flash-based FPGAs need to be reprogrammed each time they are reconfigured.
- **Power consumption:** Flash-based FPGAs consume less power than SRAM-based FPGAs. Flash-based FPGAs do not need to be reconfigured every time they are powered up, and they do not require a battery or other power source to retain their configuration.

III. SpaceWire node SoC

SpaceWire IP core [6], can be implemented on both SRAM-based FPGAs (such as AMD-Xilinx) and FLASH-based FPGAs (such as the Microchip Polarfire Family).

To illustrate these differences, an SpaceWire reference design was created for both technologies, as shown in the block diagram in Figure 1.

The reference design includes basically a soft processor and the SpaceWire IP core. The use of a soft processor comes from the need to use a RISC-V processor in Microsemi's technology. For this reason, the design developed in Xilinx technology must be done using a Microblaze processor, to be able to compare both designs properly.

The SpaceWire IP core has different interfaces. The SpaceWire protocol determines a differential full duplex communication with strobe signaling. This means that it is necessary two differential lines for transmitting both the data and strobe signals, and two differential lines for receiving the data and strobe signals.

As it is said before, SpaceWire employs Data-Strobe (DS) encoding, which combines the transmission clock with the data through Data and Strobe lines. This allows for the clock to be easily retrieved by XORing the two lines. The data is transmitted as is, while the Strobe signal alters its state when the data remains unchanged from one data bit interval to the next. DS encoding is implemented to enhance skew tolerance, which can reach up to nearly 1-bit time. This is in contrast to the 0.5 bit time achieved with basic data and clock encoding.

The XORing recovery operation of this SpaceWire IP core is based on asynchronous oversampling of the incoming signals, instead of recovering the clock by combinational logic. The

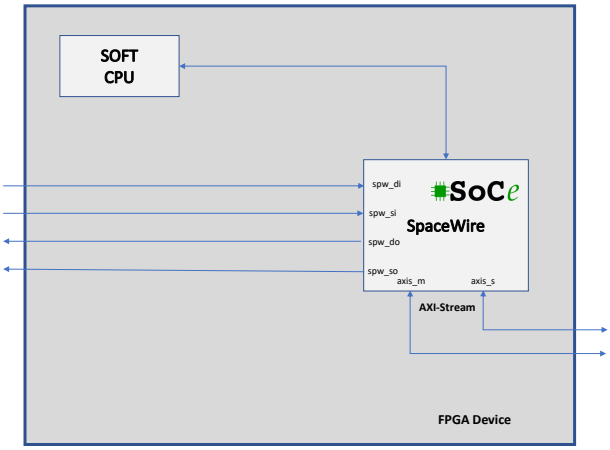

Figure 1. SpaceWire IP Core reference design.

data and strobe signals are sampled at a fixed rate, significantly faster than the link bitrate, so the bit transitions are detected by comparing the sampled signals to the previous ones.

On the other hand, there are the AXI4-Lite and AXI-Stream interfaces. The AXI-Stream interface is used to transmit and receive data packets to or from a device such as a measure instrument or a memory with AXI-Stram interface. In this case, both references designs have a FIFO memory core, which allows the write or read of data packets between the FIFO memory and the SpaceWire core through the AXI-Stream interface.

The AXI4-Lite interface is used to configure the registers of the SpaceWire IP core. These registers are:

- **System Clock frequency:** It specifies the system clock frequency used in the IP core.
- **Rx Implementation:** It defines if generic mode or fast mode is used in the receiver.
- **Rx Bitrate:** Only considered in fast mode implementation. It defines the received data bitrate.
- **Tx Implementation:** It defines if generic mode or fast mode is used in the transmitter.
- **Tx Clock frequency:** Only considered in fast mode implementation. Determines the maximum transmitted data bitrate.
- **Rx FIFO Depth:** It specifies the Rx FIFO Depth used in the IP core.
- **Tx FIFO Depth:** It specifies the Tx FIFO Depth used in the IP core.
- **AXI-S Data Width:** Determines the data width of the AXI-Stram interface.

The HDL code of the IP is highly customizable, allowing for wide configuration options, previously described. Designers have the flexibility to choose various parameters, such as system frequency, AXI-Stream data width, and Tx implementation, among others. A comprehensive list of all configurable features can be found in the SpaceWire datasheet [7].

For this experiment analysis, the following parameter configurations were used:

- **System Clock frequency:** 100 MHz
- **spw_rx_clk:** 200 MHz
- **spw_tx_clk:** 200 MHz
- **Rx Implementation:** Fast
- **Rx Bitrate:** Up to x4
- **Tx Implementation:** Fast
- **Tx Clock frequency:** 200 MHz
- **Rx FIFO Depth:** 4096 bytes
- **Tx FIFO Depth:** 4096 bytes

IV. Implementations

A. *Microblaze based implementation on SRAM-based FPGA*

Figure 2 shows the high-level design of the SpaceWire core implemented on a AMD-Xilinx Zynq Ultrascale+ MPSoC device. The evaluation board used is ZCU102 that includes a xczu9eg-ffvb1156-2-e SoC device and FMC expansion connectors are used to connect the different SpaceWire cores.

Zynq UltraScale+ MPSoC devices are 16 nm bulk FinFET technology.This family of products integrates a feature-rich 64-bit quad-core or dual-core ARM Cortex-A53 and dual-core ARM Cortex-R5F based processing system (PS) and Xilinx programmable logic (PL) UltraScale architecture in a single device. The basic building block of an Ultrascale FPGA is the Configurable Logic Block (CLB). The UltraScale architecture CLBs provide advanced, high-performance, low-power programmable logic with:

- Real 6-input look-up table (LUT) capability (configurable into 2 5-input LUT with separate output but common adress or logic inputs).
- Distributed memory and shift register logic (SRL) ability.
- Dedicated high-speed carry logic for arithmetic functions.
- Wide multiplexers for efficient utilization.

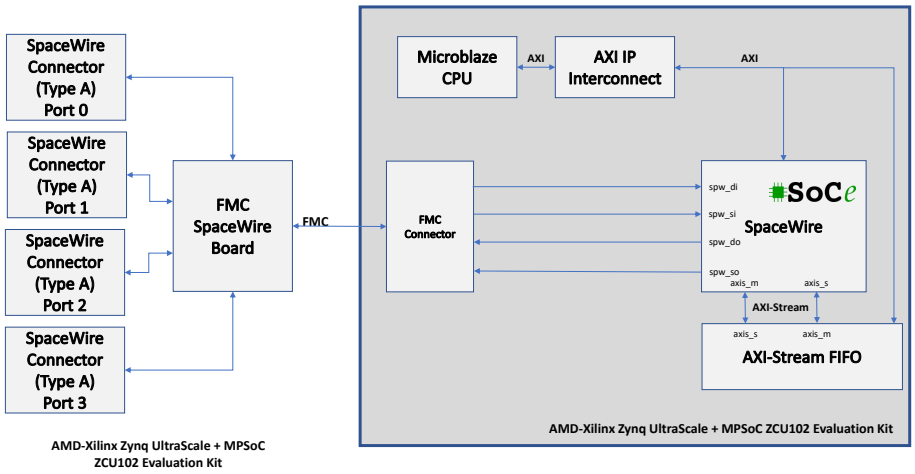

Figure 2. SpaceWire Core implemented on AMD-Xilinx Zynq Ultrascale+ SoC device (ZCU102 Evaluation Board).

- Dedicated storage elements that can be configured as flip-flops or latches with flexible control signals.

As shown in Figure 2, the design is complemented by a 32-bit soft Microblaze CPU [8] for executing the user configuration interface. The design also incorporates an AXI Interconnect IP, which enables access to the internal registers of the SpaceWire IP.

The design is equipped with auxiliary IPs that manage SoC reset and initialization, clock tree, and AXI on-chip bus infrastructure.

Table I summarizes the implementation results after Place&Route required to implement the whole design. The resources required to implement SpaceWire IP are detailed in Table II.

Resources	Used	Available	Percent
6-Input LUT (LUTRAM included)	4820	274080	1.76 %
LUTRAM(512b)	651	144000	0.45 %
Flip-Flop	4366	548160	0.80 %
Block RAMs(36Kb)	17	912	1.86 %

Table I
FPGA RESOURCES USED IN SRAM-BASED AMD-XILINX TECHNOLOGY TO IMPLEMENT THE WHOLE DESIGN.

Resources	Used	Available	Percent
6-Input LUT (LUTRAM included)	658	274080	0.24 %
LUTRAM(512b)	6	144000	0.01 %
Flip-Flop	756	548160	0.14 %
Block RAMs(36Kb)	3	912	0.33 %

Table II
FPGA RESOURCES USED IN SRAM-BASED AMD-XILINX TECHNOLOGY TO IMPLEMENT SPACEWIRE IP CORE.

A Power consumption estimation for this design is presented in Table III.

	Power (mW)	Percentage
Total Power	794	100.0 %
Static Power	621	78.2 %
Dynamic Power	173	21.8 %

Table III
POWER CONSUMPTION OF THE DESIGN IN SRAM-BASED FPGA TECHNOLOGY.

B. RISC-V based implementation on FLASH-based FPGA

Figure 3 shows the high-level design of the SpaceWire core implemented on a Microchip (Microsemi) Polarfire FPGA device. The evaluation board used for the test is an MPF300-EVAL-KIT that includes a FLASH-based MPF300TS-1FCG1152I Polarfire FPGA device [9] and an FMC extension port used to connect the SpaceWire board.

Polarfire are Non Volatile-based 28nm FPGAs devices. The Logic Cell, Logic Element as known for this technology, is composed of 4-input LUT and a D-type flip-flop can be used as a register or latch. This 4-input LUT with carry chain can be configured to implement any 4-input combinational logical function or arithmetic function. The carry chain is used to implement fast addition and subtraction. PolarFire FPGAs also include embedded memory blocks, which can be used to implement data storage and processing functions.The largest ones, named LSRAM, are 20Kbits each with 20-bit width and a depth of 1024.

As is shown in Figure 3, the design is completed with a soft 32-bit RISC-V CPU [10] to run the user configuration interface. The design also includes an AXI Interconnect IP, which provides access to the internal registers of the SpaceWire core.

The design is completed with auxiliary IPs to manage SoC reset and initialization, clock tree and on-chip bus infrastructure.

The post Place&Route implementation results for this design are summarized in Table IV. To aid in the identification

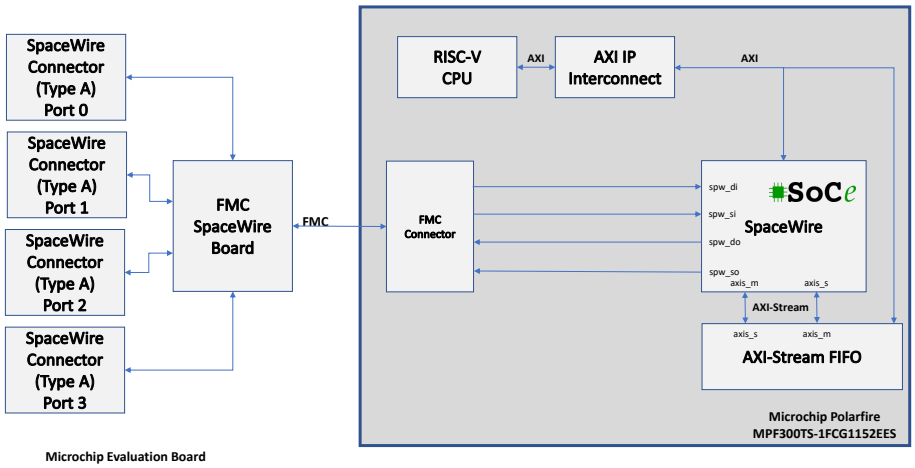

Figure 3. SpaceWire Core implemented on Polarfire MPF300TS-1FCG1152I device (MPF300-EVAL-KIT Evaluation Board).

of the elements required for the SpaceWire IP and the whole design, the are shown two different tables.

Table IV summarizes the implementation results after Place&Route required to implement the whole design. The resources required to implement SpaceWire IP are detailed in Table V.

Resources	Used	Available	Percent
4LUT	12356	299544	4.12 %
DFF	5687	299544	1.90 %
single I/O	4	512	0.78 %
differential I/O	4	256	1.56 %
uSRAM(1Kb)	28	2772	1.01 %
LSRAM(18Kb)	3	952	0.32 %

Table IV
FPGA RESOURCES USED IN FLASH-BASED MICROSEMI TECHNOLOGY IN THE WHOLE DESIGN

Resources	Used	Available	Percent
4-Input LUT	1000	299544	0.33 %
Flip-Flop	728	299544	0.24 %
uSRAM(1Kb)	1	2772	0.04 %
LSRAM(18Kb)	2	952	0.21 %

Table V
FPGA RESOURCES USED IN FLASH-BASED MICROSEMI TECHNOLOGY TO IMPLEMENT SPACEWIRE IP CORE

A Power consumption estimation for this design is presented in Table VI.

	Power (mW)	Percentage
Total Power	250	100.0 %
Static Power	192	76.8 %
Dynamic Power	58	23.2 %

Table VI
POWER CONSUMPTION OF THE DESIGN IN MICROSEMI TECHNOLOGY

V. RESULTS COMPARISON

When comparing the resources used in the SpaceWire IP core for both SRAM and FLASH-based FPGA technologies, they appear to be quite similar. However, the design developed for the Microchip PolarFire device seems to require slightly more resources than the one implemented on the Xilinx Ultra-Scale+ device. It is worth noting that UltraScale+ devices have 6-input LUTs, while PolarFire devices have 4-input LUTs, which has an impact on the resource usage.

The results presented in Table VII provide a comparison of LUT resources in the SpaceWire core for the two different FPGA technologies. To account for the difference between 6-Input LUTs used in Xilinx devices and 4-Input LUTs used in Microchip devices, a scaling factor of 2.1875, as suggested by AMD-Xilinx [11], was applied. To compare the SRAM memory resources used in both cases, we considered the total size of each block type. The AMD-Xilinx LUTRAMs offer up to 512 bits, while the BRAM size is 36 Kb. On the other hand, the LSRAM memory blocks of Microsemi are 18 Kb in size, while the uSRAM size is 1 Kb.

Resources	SRAM-based FPGA	FLASH-based FPGA
4-Input LUT	1426 ((658-6)*2.1875)	1000
Flip-Flop	756	728
RAM(Kb)	111 (6*0.512+3*36)	37 (2*18+1*1)

Table VII
FPGA RESOURCES COMPARED FOR SRAM-BASED AND FLASH-BASED FPGA TECHNOLOGY TO IMPLEMENT SPACEWIRE IP CORE

In terms of power consumption, the total power of the design implemented on the Microchip device is significantly lower than that of the Xilinx device as it can be shown in Table VIII. This is mainly due to the use of FLASH-based technology. Unlike SRAM, FLASH memory is non-volatile and does not require power to maintain its contents, which results in significantly lower static power consumption.

	SRAM-based FPGA (mW)	FLASH-based FPGA (mW)	**Ratio (SRAM-b/FLASH-b)**
Total Power	794	250	**3.18**
Static Power	621	192	**3.23**
Dynamic Power	173	58	**2.98**

Table VIII
POWER CONSUMPTION OF THE DESIGN IN SRAM-BASED AND FLASH-BASED RECONFIGURABLE TECHNOLOGY.

VI. CONCLUSION

In conclusion, this research compared two implementations of SpaceWire nodes, one based on SRAM-based FPGA technology and the other based on FLASH-based FPGA technology, in terms of FPGA resource requirements and estimated power consumption. Both implementations used the same HDL SpaceWire IP core, but with different CPUs, and were analyzed for their performance and energy efficiency. Although the comparisons are made between two devices with some differences such as the technology nodes or performance, the results can give an idea of the differences between both FLASH and SRAM based technologies.

The results showed that the FLASH-based implementation required fewer FPGA resources and had lower power consumption than the SRAM-based implementation. The selection of each technology for a specific application not only depends on the factors discussed in this paper (there are other important features like reprogrammability, radiant resistance or volatility). Nevertheless, one important fact that can affect to the election of an specific techology for an application is the power consumption. Nowadays, the term "new-space" is becoming more popular and it refers to the tendency of sending to the space small spacecrafts (instead of big ones), so they need to reduce their power consumption. Because of this, as it has been discussed here, a FLASH based FPGA solution can achieve this goal.

The study also highlighted the benefits and drawbacks of each FPGA technology, making it easier for designers to choose the right technology for their specific application. This research provides valuable insights for the development of future spacecraft data-handling systems, contributing to the continued improvement of space exploration technology.

REFERENCES

[1] S. Parkes, *SpaceWire's User Guide*, STAR-Dundee Ltd., STAR House, 166 Nethergate, Dundee DD1 4EE, Scotland, UK, 2012. [Online]. Available: https://www.star-dundee.com/spacewire/getting-started/an-overview-of-the-spacewire-standard/

[2] "Ieee standard for heterogeneous interconnect (hic), (low-cost, low-latency scalable serial interconnect for parallel system construction)," *IEEE Std 1355-1995*, pp. 1–148, 1996.

[3] *Interface Circuits for TIA/EIA-644 (LVDS)*, Texas Instruments, Sep. 2002.

[4] M. Abusultan and S. P. Khatri, "Exploring static and dynamic flash-based fpga design topologies," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 2016, pp. 416–419.

[5] L. Sterpone, "Sram and flash-based fpga testing and mitigation solutions," in *Chairman's Invitation*, 2019, p. 24.

[6] "SpaceWire IP Core - SoC-e — soc-e.com," https://soc-e.com/spacewire-ip-core/.

[7] *SpaceWireIP Core Datasheet*, System-on-Chip engineering, 2019.

[8] "MicroBlaze Soft Processor Core — xilinx.com," https://www.xilinx.com/products/design-tools/microblaze.html.

[9] "PolarFire FPGA Evaluation Kit — microchip.com," https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html.

[10] "RISC-V CPUs — Microsemi — microsemi.com," https://www.microsemi.com/product-directory/mi-v-risc-v-ecosystem/4406-risc-v-cpus.

[11] Xilinx Corp., "Zynq UltraScale+ MPSoC product advantages," https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascalempsoc.html, 2018.