A Serial High-Speed Satellite Communication CODEC: Design and Implementation of a SpaceFibre Interface

Pietro Nannipieri[a,*], Gianmarco Dinelli,[a], Antonino Marino[a], Luca Dello Sterpaio[a], Alessandro Leoni[a], Luca Fanucci[a], Daniele Davalle[b]

[a] *Department of Information Engineering, University of Pisa, Pisa, Italy*
[b] *IngeniArs S.r.l, Pisa, Italy*

Abstract

In the last few years, satellite on-board data handling bandwidth requirements grew significantly, as well as production volume of these systems. A series of different protocols currently try to answer this need. In particular, the European Space Agency developed an open protocol solution: SpaceFibre. The SpaceFibre protocol can sustain a line rate of 6.25 Gb/s per lane (up to 16 lanes). It offers advanced and flexible Quality-of-Service features, as well as Fault Detection Isolation and Recovery services. The protocol structure has been developed so that full hardware implementation of its core layers is straightforward, granting high performances at low price in terms of complexity and power consumption, one of the most stringent requirements in space applications. In this paper, a FPGA implementation on both rad-hardened (RTAX2000, RTG4, Virtex-5) and commercial (ZYNQ 7000) devices of the SpaceFibre CODEC is presented together with its verification environment and a hardware validation set-up. Particular attention is given to the trade-off between resources utilisation, power consumption and CODEC configurations, in order to enable future system adopters to efficiently explore the design space.

Keywords: SpaceFibre, Communication, Network, Interface, CODEC,

*Corresponding author
Email addresses: pietro.nannipieri@ing.unipi.it (Pietro Nannipieri), gianmarco.dinelli@ing.unipi.it (Gianmarco Dinelli,), antonino.marino@ing.unipi.it (Antonino Marino), luca.dello.sterpaio@ing.unipi.it (Luca Dello Sterpaio), alessandro.leoni@ing.unipi.it (Alessandro Leoni), luca.fanucci@unipi.it (Luca Fanucci), daniele.davalle@ingeniars.com (Daniele Davalle)

Data-Handling, FPGA, OBDH, Satellite, Space, SpaceFibre, SpaceWire.

1. Introduction

On-board data handling systems for spacecraft deeply changed during their history [1]; in particular, data-rate requirement grew dramatically in the last few years, mainly due to the presence of high resolution instruments such as Synthetic Aperture Radars (SAR) and hyper-spectral imagers. This led different institutes, agencies and companies to start working on new communication protocols in order to meet this requirement. The number of spacecraft missions together with the need of high reliability led to the standardization of the SpaceWire (SpW) [2], [3] communication protocol in 2003. SpW can theoretically reach speeds of 400 Mb/s (200 Mb/s in practice), and it has already been adopted in nearly all European Space Agency (ESA) missions for payload data handling. At its first application, it was used as a point-to-point link (Sentinel-1 mission), while newer missions such as BepiColombo, SolarOrbiter, MetOP-SG and JUICE already make use of a large SpW network, connecting instruments and mass memory trough a routing switch function [4]. In these missions, SpW is the only communication link between control unit and instruments, therefore it must be able to carry both housekeeping data and control commands. The maximum available SpW bandwidth unfortunately is inadequate for high data-rate instruments already necessary in several missions (i.e. the Multi Spectral Imager (MSI) on-board Sentinel-2 [5]). This problem was solved with two different approaches: using several SpW links in parallel or adopting other solutions such as WizardLink or Channel Link. All these solutions share the lack of standardisation: this led ESA to start working on a new standard, which underwent public review in 2018, and it has been published by the European Cooperation for Space Standardisation (ECSS): SpaceFibre (SpFi) [6], [7].

SpaceFibre is a key milestone in ESA roadmap. Indeed, ESA is defining a system reference architecture for space avionics, in order to promote the standardisation of these systems, both at hardware and software level. The name

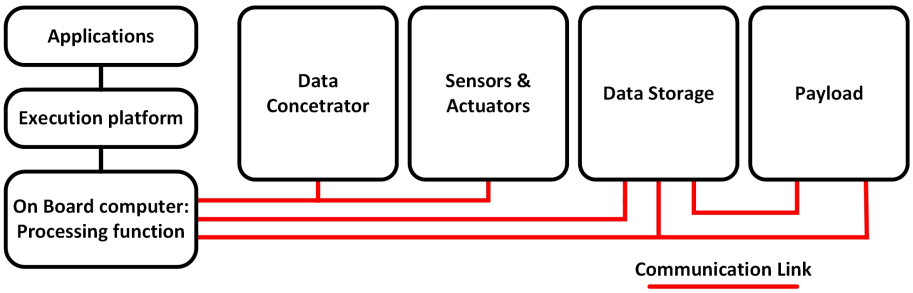

Figure 1: Simplified SAVOIR avionics system reference architecture

of the initiative is Space Avionics Open Interface Architecture (SAVOIR)[8]. SAVOIR main objective is to reduce risks and production costs while providing standardised interfaces to promote block re-usability. A simplified reference architecture is depicted in Figure 1. In the lower part of the figure, communication links between different blocks are identified: On-board computer [9], data concentrator, sensor/actuators, data storage and the payload shall be almost completely interconnected. There are several requirements involved in different types of communication link: high bandwidth, low latency and reliability. SpaceFibre achieves data-rates above the Gb/s, representing the ideal solution to connect high throughput payload to the data storage unit, and it can also exploit prioritized low latency communication. Potentially, a SpaceFibre link can be used for all the communication links presented in Figure 1. This paper presents the Field Programmable Gate Array (FPGA) implementation of a Hardware Description Language (HDL) single lane SpaceFibre CODEC fully compliant to the ECSS SpaceFibre standard [6], and it illustrates the design space exploration study carried out in the design process. Particular attention has been given to logical resources estimation and power consumption, two of the most critical parameters for a satellite system. SpaceFibre is a novel protocol for high-speed on-board communication, with the corresponding standard released in May 2019 [6]. Unfortunately, literature lacks detailed studies on SpaceFibre implementations and presents only fragmented contributions, all reported in this work. No comparison within different existing solutions is present

yet, and precise indications on system power consumption lacks. Consequently, this work consolidates and enriches the SpaceFibre state-of-the-art, providing a fair comparison between the proposed design and the other latest available SpaceFibre implementations. Moreover, an investigation on how power consumption is affected by the number of Virtual Channels (VCs) is carried out, providing to the space community such a critical information which will help to evaluate the novel SpaceFibre technology performance. Future system integrators will benefit from the output of this work, which will give an indication of the logical resources and power consumption of a SpaceFibre endpoint, varying the number of VCs employed. The work has been carried out on FPGA technology as it is far more strategic for current space applications: state-of-the-art rad-hardened FPGAs solutions (e.g. Microsemi RTG4) embeds quite large numbers of logical resources, together with dedicated hardware such as Gigabit Serialiser-Deserialiser (SerDes). Employing such solution instead of expensive and complex rad-hardened ASIC could boost scientific and technical capabilities of next generation space missions. The main contributions of this work cover all the SpaceFibre CODEC design steps:

- System architecture description and RTL design.
- RTL Verification approach with a dedicated verification environment.
- FPGA resource and power consumption analysis. Comparison with data available in literature.
- Design space exploration for a SpaceFibre endpoint (VC number).
- Hardware system validator description.

The scope of this work is thus to provide future technology adopters a first indication of its key features. It will also provide a solid literature background for future developments. The paper is structured with the following scheme: in Section II, the architecture of the SpFi CODEC Intellectual Property (IP) core is presented, both at interface level and internally, with references to its

performance requirements. In section III, an overview on the verification environment designed to support the HDL SpFi CODEC development is given. In section IV, the implementation results on various FPGA targets are presented and briefly analysed. In section V, conclusions are drawn.

2. The SpaceFibre protocol

2.1. The SpaceFibre protocol

SpaceFibre is multi-Gb/s data link and network technology developed under ESA funding specifically for payload data processing applications on-board spacecraft. It has been conceived to support high data-rate payload data-handling: indeed, it is able to operate both on copper cables and optical fibre. SpFi is the successor of (and is backward compatible with) the very popular SpW protocol but utilizes, like many other modern network architectures, a SerDes. It supports data-rate up to 6.25 Gb/s per communication lane (up to 16 communication lanes). From the specifications point of view, it includes built-in Quality of Service (QoS) and Fault Detection, Isolation and Recovery techniques (FDIR), providing system level benefits without complex software components. The main layers of the SpFi protocol stack are shown in Figure 2.

- **Network layer** is responsible for packet transfer over the network. This layer is optional, and it is not necessary in point-to-point communication links. In literature, there are examples of SpaceFibre routing switch [10],[11] and also of high level network simulators [12], [13].
- **Data Link layer** is responsible for the Quality of Service, flow control and for resending information in case a fault occurs over the link. It frames data packet to be sent over the link and handles the exchange of short broadcast (BC) messages with the rest of the network.
- **Multi-lane layer** is responsible for running several SpFi lanes in parallel to provide higher data throughput. This layer is optional and can be bypassed. Dedicated activities on this topic can be found at [14], [15].

- **Lane layer** is responsible for lane initialisation error detection and re-initialisation. Symbols are encoded with 8B10B encoding, with AC coupling of data signals.

- **Physical layer** is responsible for serializing and de-serializing encoded symbols and transmit them over the physical link. It also recovers clock from the received data.

- **Management layer** (or Management Information Base) is responsible for the control and configuration of each layer.

Three different interfaces are provided to the user application:

- A **Packet level Interface**, used to send and receive data packets.

- A **Management Interface**, to configure and control the whole SpFi interface.

- A **Broadcast message Interface** used to broadcast short messages over the network.

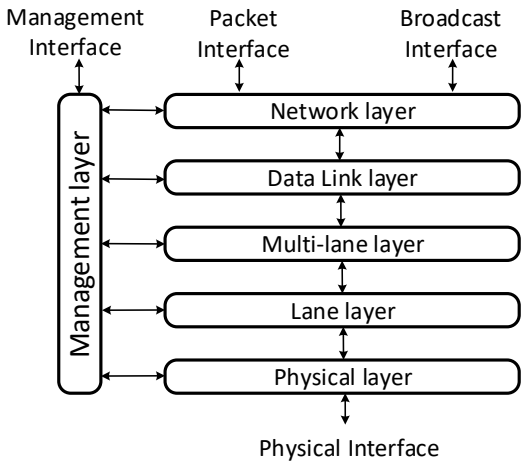

Figure 2: SpFi protocol stack

A single lane SpFi link is composed of two differential pairs, one for serial data transmission and one for serial data reception. The clock signal is not transmitted through a dedicated line, but it can be recovered from data thanks to 8B10B encoding, which provides enough bit transitions to recover the clock from the transmitted stream with a PLL.

3. SpaceFibre CODEC architecture

In this section, the SpFi CODEC architecture is presented. Being the standard open and well structured, its implementation is guided and the majority of the architecture is fixed. However, since the level of details presented in literature is not exhaustive, this work aims at clearly indicate the architecture of the system, at least at building blocks level. This to better understand the design choices made and their impact on resource utilisation and power consumption. The SpFi CODEC has different interfaces allowing host application to send and receive data over a SpFi communication link and to configure and monitor the status of each layer of the protocol. The standard allows up to 32 couples of input-output VCs in order to handle independent flows of information. The block diagram architecture of the SpFi CODEC is shown in Figure 3. In particular, two layers of the protocol stack are involved: the Data Link layer and the Lane layer. The CODEC presented in this work does not include the multi-lane layer, mainly for maturity reasons: being an optional layer, it has been added to the standard with consistent delay respect to the core layer and first implementations of the CODEC did not support it. Most probably, first SapceFibre-based systems to be adopted in space mission will rely on more mature single lane links, with the multi-lane layer to be introduced as the SpaceFibre protocol increases its heritage. The Output Virtual Channel (OUT VC) interface is responsible for transmitting data packets from the upper layers (Network/Application). Each OUT VC is equipped with a buffer, implemented as an asynchronous FIFO to solve potential clock domain crossing with higher layers which could work with a different clock with reference to the one used by

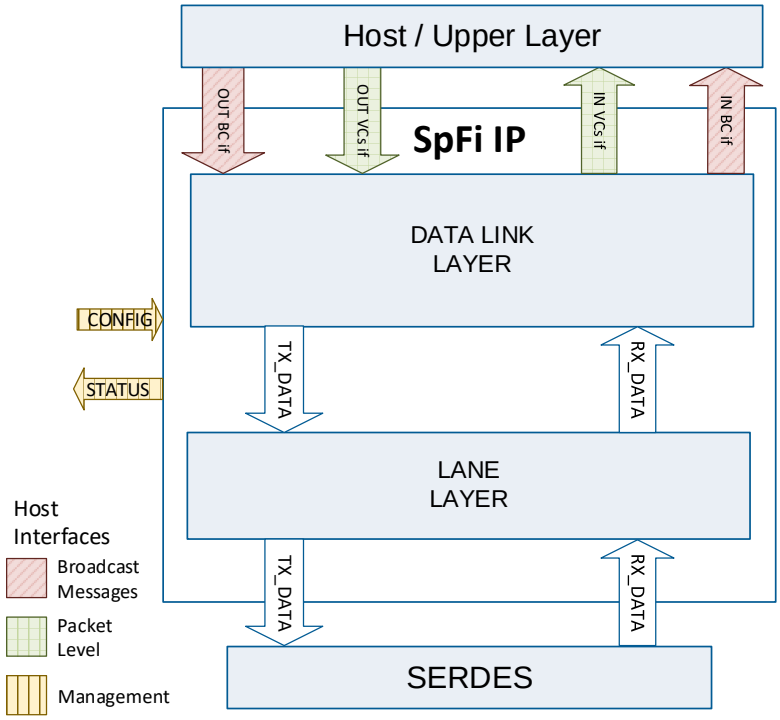

Figure 3: SpFi IP Block Diagram

the SpFi core.

The Input Virtual Channel (IN VC) interface is responsible for receiving data packets from the far-end of the link and for sending them to the upper layers. Each IN VC receives data from the far-end OUT VC with the same identification number, and it is equipped with a buffer, implemented as an asynchronous FIFO.

The Output Broadcast (OUT BC) interface is responsible for transmitting broadcast messages from the upper layer to the SpFi network. The OUT BC can be optionally equipped with a BC buffer, implemented an asynchronous FIFO.

The Input Broadcast (IN BC) interface is responsible for receiving broadcast messages from the SpFi networks, and for sending them to the upper layer.

The IN BC can be optionally equipped with a BC buffer, implemented an asynchronous FIFO.

The TX and RX Interface to-from the SerDes is a 20 or 40 bits configurable interface used to control the SerDes and for sending/receiving words from it. The lower section of the Lane layer is configurable so that some features such as the 8B10B encoding/decoding can be demanded to the SerDes.

Finally, the Management & Configuration Interface is used for configuring several parameters of the SpFi layers. Furthermore, it is used for reporting status information, such as error conditions and finite state machine conditions.

In the following paragraphs, a description of the architecture is given, focusing on the main features described in the SpFi standard.

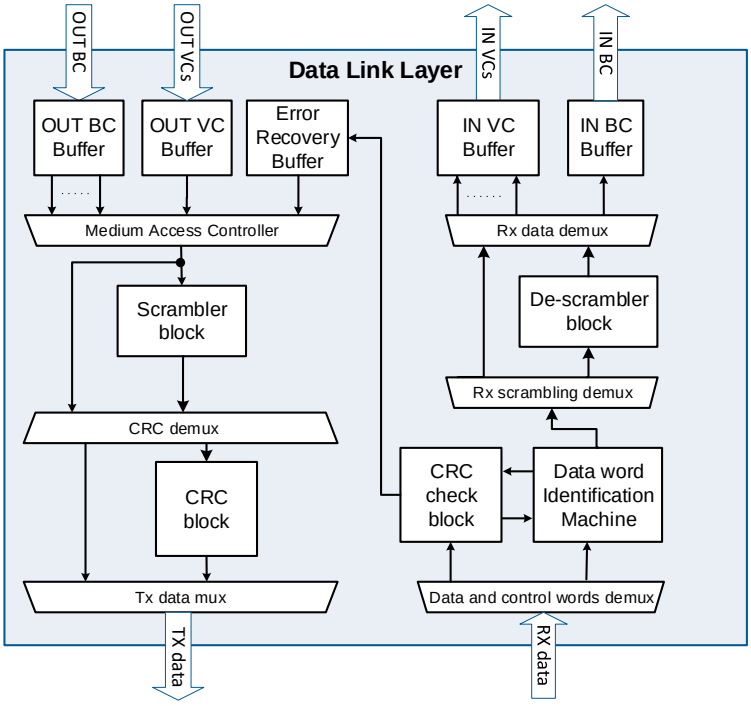

Figure 4: SpFi Data Link Layer Block Diagram

3.1. Data Link Layer

The architecture of the Data Link layer is shown in Figure 4. The transmission side of the Data Link includes the OUT VCs interface and the OUT BC interface. The Medium Access Controller (MAC) is responsible for VCs scheduling. A study on the possible algorithms is presented in literature [16]. The presented CODEC employs a priority and bandwidth reservation algorithm combined with a round robin approach to regulate which VC is going to send a data segment over the communication link. There are several levels of priority: a VC is allowed to send data only when no VC with higher priority is ready to send data. Furthermore, the MAC handles the Error Recovery buffer that allows to re-send corrupted data packets. The MAC thus selects one of the OUT VC, the OUT BC or the Error Recovery Buffer for transmitting data across the link. The MAC is also responsible for data framing (the first and the last word of a data frame or BC message are specific Data Link control words). Optional data scrambling may be included to reduce electromagnetic emissions and finally, the CRC block calculates the Cyclic Redundancy Check (CRC) code. It is an error-detecting code [17], which is used to detect accidental changes to raw data that are common in the space environment due to radiations. CRC is inserted at the end of each data frame, BC message and in specific control words.

The receiving side of the Data Link layer includes the IN VCs interface and the IN BC interface. The Data Word Identification State Machine is used to identify which type of frame a data belongs to. It is composed of five different state, which cover all the possible cases: idle, data, broadcast frames being received, mixed traffic (broadcast within a data frame) and no RX traffic at all. The exact behaviour of the state machine is carefully described in the standard [6]. The De-Scrambler block de-scrambles received data packets if optional data scrambling is active, which are then stored in the corresponding IN VC or IN BC. During data reception, CRC check block calculates the CRC code and compares it with the received one: if there is a mismatch between the two values, a retry request is generated and the corrupted data frame/broadcast message is

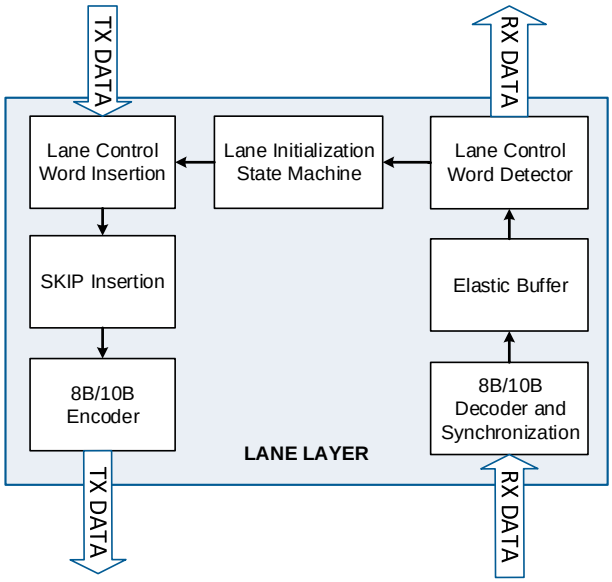

Figure 5: SpFi Lane Layer Block Diagram

dropped. For more details about Data Link layer please refer the SpFi standard [6].

3.2. Lane Layer

The architecture of the Lane layer is shown in Figure 5. The Lane layer is responsible for establishing the communication across a SpFi link through the Lane Initialization State Machine, which shall control the initialisation of a SpaceFibre lane: establishing the connection and responding to standby management requests. A detailed description of the state machine is presented in [6]. The synchronization between the two ends of the link is obtained by means of Lane layer control words. The Lane layer is also responsible for compensating differences that may arise between the data signalling rate of the two ends of the communication link. This may happen due to slight differences in the local clock of the two ends of the link. Therefore a receive Elastic Buffer is requested by the standard in order to accommodate small differences between the two clocks. The Elastic Buffer recognizes special SKIP word in the data

stream and, if the buffer is more than half-full, the SKIP word is effectively skipped. If, on the other hand, the buffer is less than half full, the SKIP word is not skipped. One SKIP word is transmitted every 5000 words. This mechanism allows absorbing 100 ppm difference between local oscillators at the two ends. Finally, the Lane layer is responsible for encoding data and control words into symbols through 8B10B [18] Encoder during data transmission, and for decoding received symbols into data or control words through the 8B10B Decoder. The adopted 8B10B encoder/decoder is the one presented in [18], which employs an innovative parallel approach so that is possible to save logical resource and power consumption.

4. Verification environment

The SpFi CODEC IP was verified with a SystemVerilog based test environment developed separately, starting from the SpFi standard [6]. This approach, based on the development of a golden reference (designed as high-level description by designers not involved in the CODEC HDL design), demonstrated to be very effective in order to identify possible misunderstandings and ambiguities in the standard specifications. The adopted methodology is compliant with the design flow described in [19]. The verification phase aimed at reaching 100% coverage score in: i) statements coverage; ii) branches coverage; iii) FSM coverage and iv) conditions coverage. The verification environment was built using the Universal Verification Methodology (UVM) [20] framework for SystemVerilog. It represents the state-of-the-art in digital design verification field. UVM consists in a set of built-in classes, macros and libraries allowing a rapid and robust development of the verification environment. One of the greatest advantages of UVM is the possibility to exploit the Transaction Level Modelling scheme: all the TestBench (TB) logic is developed at a transaction level instead of at a pin level. Note that in UVM a transaction is represented as a SystemVerilog class. This greatly simplifies the verification environment development and reduces test-bench and test-case design time. The verification environment was

built as a Bus Functional Model of a SpFi CODEC. This means that a SpFi module was developed in SystemVerilog according to the SpFi standard [6], and that model was connected to the RTL IP Core Device Under Test (DUT). In this way, the functionality of the DUT is tested from a high-level point of view. Figure 6 shows a generic Bus Functional Mode scheme. The main components of the verification environment are:

SystemVerilog Model. The SystemVerilog Model is the core of the verification environment. It is a SystemVerilog implementation of the SpFi standard and it is used to simulate the communication of the DUT with an external CODEC. The Model shares some configuration parameters with the DUT (for example, the number of Virtual Channels).

Interfaces. The DUT is connected to the rest of the testbench by means of SystemVerilog interfaces. They are used to group synchronous signals into and separate them depending on their functions. The interfaces are:

- Data Link layer: there is one network layer interface for each virtual

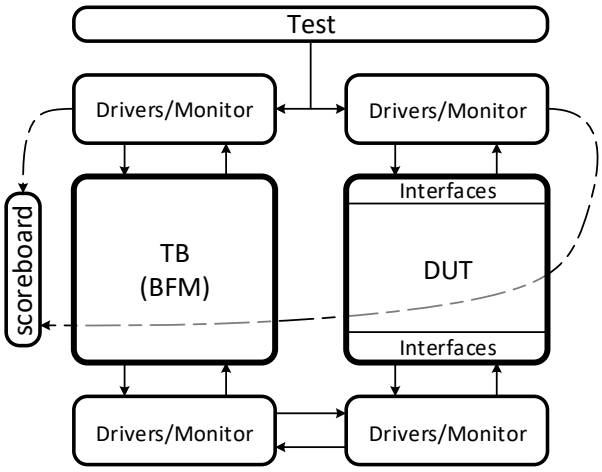

Figure 6: SpaceFibre verification environment scheme with bus functional model

channel

- Physical layer: there is one physical layer interface representing the serial bus for the connection with the SystemVerilog Model

- Management layer: there is one management layer interface for the status and control information. This interface is used also to send reset signal

Note that each interface has its own clock that can asynchronous with the others.

Drivers and Monitors. They are the components realizing the translation between the pin level world of the DUT and the transaction level world of the TB. The role of drivers is to take a transaction coming from the SystemVerilog Model and to drive the corresponding interface signals to send that transaction or to read a sequence of signals from an interface and build a transaction to send to the model. Monitors are passive elements that perform correctness checks on the interface signals. For each interface there are two drivers, one for the communication Model-DUT and one for the communication DUT-Model, and one monitor.

Scoreboard. This component keeps track of the number of errors occurring during the test. It is used also to register statistics such as the number of packets sent and arrived and their latency.

These components represent the architecture of the verification environment that was used to run tests. The test environment was developed in such a way that it is possible to run single test cases or aggregate tests suites. For each test case, a different test file is created. The verification environment produces two outputs for each test file: a coverage file, containing the coverage information generated by the simulation tool and a result file, containing PASS/FAIL information and in case the detailed description of the error occurred. The Verification Environment produces a coverage file for each executed test case and can merge separate test coverage files into a unique database. A verification plan describes the test cases used to verify the IP core. These tests have been

written varying both configuration parameters and creating different situations at simulation time. In each test, the following parameters are defined:

- N VCs: the number of Virtual Channels used for the test. This value must be equal between the TB and the DUT. Allowed values are [1, 32].
- Timeslot: the timeslots used by each Virtual Channel. The number of timeslots is always set equal to the number of Virtual Channels. A Virtual Channel can transmit in a certain timeslot (1) or not (0).
- Priority: the priority level of each Virtual Channel. Allowed values are [1, 15].
- Expected Bandwidth: the expected bandwidth for each Virtual Channel. Allowed values are [0, 100].
- Expected BC Bandwidth: the expected bandwidth for the broadcast messages. Allowed values are [1,100].

Let us define also the concept of Actual Bandwidth:

- Actual Bandwidth: this value represents the rate at which data are written into the Virtual Channel by the application. Depending on the situation and the QoS used, the Virtual Channel transmission buffer can become full. Allowed values are [1, 100].
- Actual BC Bandwidth: this value represents the rate at which data are written into the Broadcast buffer by the application. Depending on the situation, the Broadcast transmission buffer can become full. Allowed values are [1,100].

The verification plan is built incrementally, in order to test from the basic functions of SpFi (simple communication) to the most complex ones (corner cases, mixed QoS). The main functions under test are:

- Basic communication: this set of tests investigates a simple communication between the Model and the DUT, also using broadcast messages.

- Error Injection: this set of tests stimulates the FDIR mechanism of SpFi, injecting errors on the link.
- Broadcast Burst: this set of tests aims to stress the DUT with burst of broadcast, also injecting errors.
- Quality of Service – Basic: this set of tests assesses each QoS mechanism of SpFi (priority, bandwidth reservation and time scheduling) separately.
- Quality of Service – Mixed: this set of tests assesses the three QoS mechanisms mixed together.
- Reset: this set of tests stimulates the reset signals during the normal CODEC operations.
- Clock domains: this set of tests involves the asynchronous interfaces of the DUT.

All the tests in the test plan were successfully executed on the designed SpFi CODEC IP core. A coverage of 100% was reached for Statements, Branches, finite state machines states and finite state machines transitions, fully verifying the RTL description of the SpFi CODEC according to [19].

5. FPGA implementation

In this section, implementation results carried out on state-of-the-art space-grade FPGAs are presented and compared with other available implementations.

5.1. SpaceFibre CODEC Resource utilisation

SpFi CODEC IP hardware resources are presented in terms of combinatorial elements, sequential elements and RAM blocks of the target device. The percentage of used resources out of the total available is also indicated. RTAX2000 is an antifuse-based FPGA from Microsemi. Antifuses are normally open circuits and form a passive and low-impedance connections during the programming phase. RTAX2000 combinatorial resources are the C-cells, based on 8-input

Table 1: SpFi CODEC IP implementation on a Microsemi RTAX2000

VC	8Mux	8Mux%	DFF	DFF%	RAM [kb]	RAM %
1	6152	28.61%	2815	26.18%	54	18.75%
2	7605	35.37%	3593	33.42%	72	25.00%
4	9852	45.81%	4808	44.72%	108	37.50%

Table 2: SpFi CODEC IP implementation on a Microsemi RTG4

VC	4LUT	4LUT%	DFF	DFF%	RAM [kb]	RAM %
1	4689	3.11%	2291	1.50%	134	2.52%
2	5713	3.76%	2888	1.90%	152	2.85%
4	7073	4.65%	3835	2.52%	188	3.53%
8	10065	6.62%	5723	3.76%	260	4.88%

multiplexers, while the R-cells are its sequential elements and can be used as standalone flip-flops. RTAX2000 has 64 4.5 kbits RAM blocks. RTG4 is a flash-based FPGA from Microsemi, considered to be the next generation space-grade device. Its combinatorial elements are based on a 4-input Look-Up-Table (LUT). It has 209 24.5 kbit of LSRAM and 210 blocks of 1.5 kbits uSRAM. Finally, Virtex 5 is a SRAM-based FPGA from Xilinx. Combinatorial functions are realized through 6 input LUTs, and it has 36-kbits RAM blocks. Table 1, Table 2 and Table 3 show the result obtained respectively for Microsemi RTAX2000 [21], Microsemi RTG4 [22] and Xilinx Virtex 5 [23]. SpFi has been also implemented on a Xilinx Zynq 7000 XC7Z045 [24], which is not specifically designed for space application but is potentially employable in Electrical Ground Segment Equipment (EGSE) systems. It has been initially used for validation, with results presented in Table 4. Its combinatorial logic is based on 6-input LUTs, and it has 36 kbit RAM blocks. Data refers to different configurations of the CODEC implemented with 1, 2 and 4 256-words virtual channels. Optional data scrambling is included, as well as all the required lane layer features (i.e. 8B10B encoding/decoding). The target clock frequency is 62.5 MHz, necessary

Table 3: SpFi CODEC IP implementation on a Xilinx Virtex 5

VC	6LUT	6LUT%	DFF	DFF%	RAM [kb]	RAM %
1	2919	3.56%	1702	2.08%	252	2.35%
2	3657	4.46%	2199	2.68%	432	4.03%
4	5037	6.15%	3003	3.67%	576	5.37%

Table 4: SpFi CODEC IP implementation on a Xilinx Zynq-7000

VC	6LUT	6LUT%	DFF	DFF%	RAM [kb]	RAM %
1	2346	1.07%	1665	0.38%	216	1.10%
2	2824	1.29%	2245	0.51%	216	1.10%
4	4110	1.88%	3203	0.73%	432	2.20%

to achieve a data-rate of 2.5 Gb/s.

The Microsemi RTG4 is equipped with SerDes operating up to 3.125 Gbps. If we considered that the SerDes has to process 40 bits encoded data words (e.g. 4 8B10B encoded symbols), the operating frequency required to substitute this data rate is fixed to be at $3.125/40 = 78.125MHz$. The proposed design successfully operates at a frequency of 80 MHz, thus it completely exploits the SerDes bandwidth capability. The SpaceFibre CODEC is able to run up to 80 MHz, which is higher that the maximum operating frequency of the SerDes, fixed at 3.125 Gbps (with 40-bit encoded data words).

5.2. Power consumption

Power consumption has been measured on the Microsemi RTG4 board by means of post-implementation simulations. The RTG4 has been targeted as it represents next generation of space qualified FPGAs. Results have been obtained with 200 μs simulations corresponding to the transmission of about 520 kBit data divided in 36 bit data words, interleaved with 1 broadcast message each 1K data words. The operating frequency is set to 62.5MHz, and SerDes power consumption is not included. The computed dynamic power is to be

considered a good estimation with two assumptions:

- The power model accuracy, provided directly by the vendor (Microsemi), estimates the FPGA dynamic power consumption correctly. Neither the vendor, nor literature, provide information about model accuracy. Moreover, due to the high cost of the FPGA itself, no commercial boards are available or easy to design for direct measurement of power consumption.
- The Testbench stimulates the device under test in its maximum power dissipation scenario.

However, even if we assume that the real value could deviate up to 10% from the computed one, we need to consider that it represents anyway just a fraction of the overall power consumption, which is mostly determined by the static and the SerDes power consumption. Results are presented in Table 5. This values do not take into account SerDes power consumption, which can be estimated to be 285 mW (value obtained with the official Microsemi RTG4 power estimator tool). Unfortunately, literature lacks of detailed power analysis for a SpaceFibre interface and competing protocols do not provide such information for the single IP core: solutions such as TTEthernet comes embedded in complete hardware product, whose characteristics are given at system level rather than at single IP level. A rough comparison on the power consumption with a similiar protocol can consider the TLK2711 chip from Texas Instrument, which implements the WizardLink protocol, a subset of the SpaceFibre lane layer. The power consumption of the chip is estimated to be 500 mW [25].

Table 5: SpFi CODEC IP power consumption on RTG4

VC	**Static Pwr [mW]**	**Dynamic Pwr [mW]**	**Total Pwr [mW]**
1	183.3	148.4	331.7
2	183.3	166.0	349.3
4	183.3	186.5	369.8
8	183.3	241.2	424.5

The proposed interface, in its smallest configuration (1 VC), consumes about $SpFiIP + Serdes = 331.7 + 285 = 616[mW]$. It provides advanced lane layer and full data link layer features, which the TLK2711 does not provide at the price a about 100 mW. If we consider also the fact that our results are based on FPGA technology, which is by definition less optimised than ASIC, we also acknowledge the fact that we would be able to push the power consumption value even lower in a future ASIC SpaceFibre implementation. It is also interesting to measure the power consumption on a faulty link, e.g. with a Bit Error Rate (BER) greater than zero. We injected errors on the previous setup, in the single VC configuration. The BER level was quite high, due do the fact that the simulation time has to be minimized: being the design quite complex, too long simulation time would require long computational time and it would create heavy files without particular advantages in terms of measurement accuracy. The obtained results is shown in Table 6.

We can observe that the difference is almost negligible. Obviously, the static power in unchanged, being the circuit unaltered. The dynamic power increases of 1.3 mW, a value under 1%. This results perfectly meet our expectations. Considered the BER fixed at 10^{-5} (the worst value tolerated in the SpFi standard), the portion of the circuit dedicated to the retry mechanism is stimulated only one bit out of 10^5, which means roughly that 1 word out of little less than 3000 is corrupted, triggering the activation of the retry mechanism. Moreover, when the retry mechanism is activated, other parts of the circuit are de-activated (e.g the VCs); therefore, dynamic power is saved somewhere else. To be fair, this obviously does not take in account that, depending on the BER, the time necessary to deliver a packet is higher; therefore, also the total energy necessary

Table 6: SpFi CODEC IP 1 VC power consumption on RTG4 with BER = 10^5

BER	**Static Pwr [mW]**	**Dynamic Pwr [mW]**	**Total Pwr [mW]**
0	183.3	148.4	331.7
10^5	183.3	149.7	333

to deliver each single bit will be higher. However, this values is not directly affected be the increased power consumption of the circuit, but only by the longer transmission time required.

5.3. Results analysis and design space exploration

Two other SpaceFibre implementations have been identified in literature: STAR-Dundee, the company responsible for the standard publication, implemented its own version of the SpaceFibre IP onto various FPGA and partially documented it in [26]. Also ESA owns its own IP within its IP core portfolio [27] [28]. However, it is not possible to directly compare the number presented in this work with the one available in literature since:

- The standard has been developed in the last years, so all the IPs available in literature are based on previous versions of it. The SpaceFibre CODEC presented in this work is based on the final version of the standard available for public review, just before the publication of it.
- In the referenced IP cores some of the key CODEC configuration parameters, such as the number of VCs or depth of the buffers are not completely specified.
- It is not clear whether the referenced IP includes low Lane Layer Features such as 8B10B encoding and decoding, symbol synchronisation, comma detection, etc.

However, in Table 7 a summary with all the information available in literature about the RTG4 implementation of a SpaceFibre CODEC is presented. 4 VCs implementation results are presented for comparison, as it is the most common referenced configuration. From the numbers available, we can observe similar resource utilisation in all the FPGAs, with small variations that can be linked with different optimisations.

We can observe that our proposed implementation reaches the maximum available bandwidth. Compared to the others, it employs a similar number of

Table 7: SpFi CODEC IP on RTG4, 4 VCs - Comparison with [28],[26]

SpFi Impl.	**4LUT**	**DFF**	**Total Logic Elements**	**Total Memory [kb]**	**Bandwidth [Gbps]**	**Power Consumpt. [mW]**
ESA[28]	2429	5769	8198	257,8	3,125	na
STAR-Dundee[26]	3967	7219	11186	440,9	3,125	na
This work	7073	3835	10908	192,2	3,125	369,8

logic elements (even if the ratio between LUT and DFF is inverted), but manages to reduce the total memory usage by 25% respect to the ESA implementation and 56% respect to the STAR-Dundee one. Moreover, only our implementation is characterised by the power consumption, both static and dynamic.

This analysis is particularly interesting, especially for future adopters of the technology, to understand how IP configuration parameters affect both complexity and power consumption. Combining all the results obtained in Section V, we can observe that as expected, the total complexity increases consistently with VC number due to the large number of buffers and multiplexing logic involved. Also the power increases but at a lower rate: only one VC is allowed to transmit data within a timeslot, thus only the increased multiplexing logic contributes to dynamic power dissipation. A conceptual visualisation of these data is shown in Figure 7.

5.4. Hardware validator

The final step in the SpFi CODEC development, after its HDL design and its verification through appropriate simulations, is to map it onto real hardware and to test a circuit prototype. The FPGA target for the circuit prototype test is the Xilinx ZYNQ 7000 (XC7Z045-2FFG900) mounted on the ZC706 Development board. An FMC expansion board (XM104) has been used to provide appropriate eSATA high-speed connectivity for the SpFi ports. The SpFi CODEC has been embedded in a more complex system, based on [29], in order to get control and

visibility out of it. The block diagram of the system is shown in Figure 8.

The SpFi IP core validation system is based on a ARM Cortex A9 microprocessor. The user, interacting with a Graphical user Interface (GUI) on the Host PC, generates commands that are forwarded through the host interface to the validation system microprocessor.

The system used for the validation is partitioned between hardware (SpFi CODEC, error injection, word analyser) and software (services to handle communication between the Host-PC and the hardware peripheral cores). The main tasks of the software part are to write configuration registers, read status registers, write data to be processed by the hardware block to the interface FIFO buffer and read data processed by the hardware block from the interface FIFO buffer.

Thanks to the hardware/software partitioning, the system is able to handle the data stream, configure and control the SpFi CODEC, monitor it and generate/consume packets. All these features enable the detailed verification of the SpFi CODEC circuit implemented in HDL.

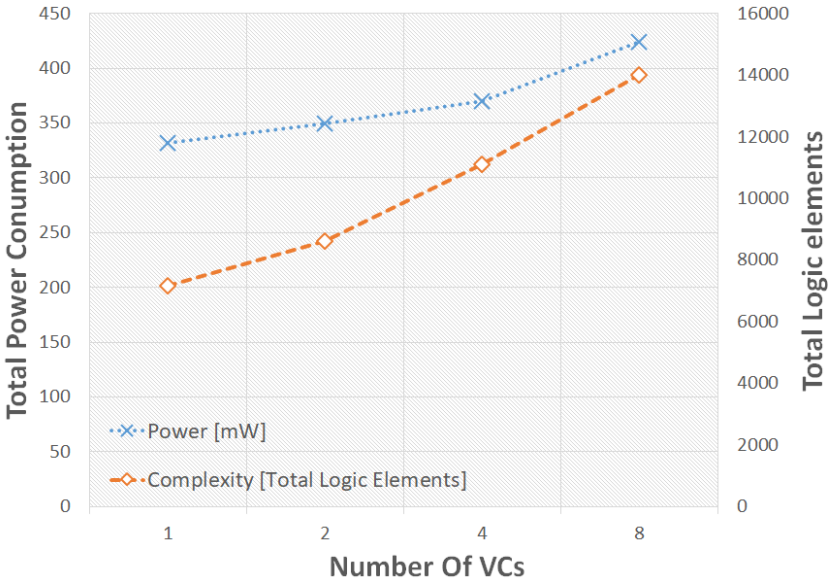

Figure 7: SpFi CODEC Power consumption & Complexity Vs Number of VCs

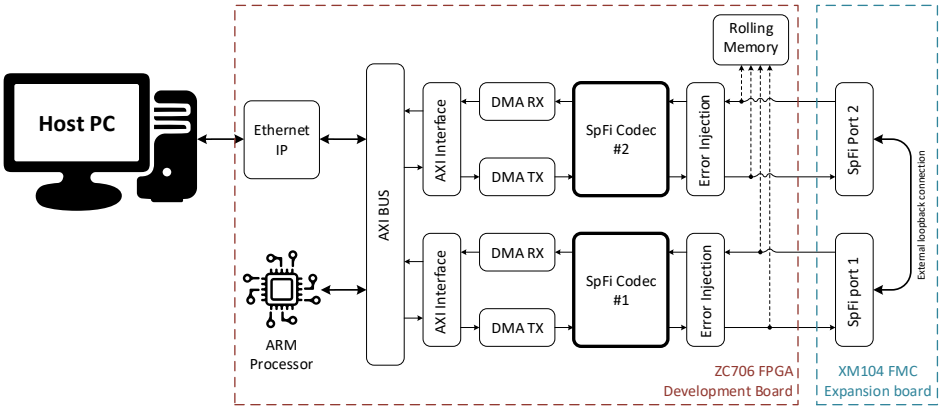

Figure 8: SpFi CODEC Validation System

The microprocessor is responsible for the configuration and the control of all the peripherals of the validation system. Each subsystem of a SpFi CODEC and of the validation system in general is highly configurable by the user at run time using the GUI, thanks to the specific system architecture adopted.
In the following, all the IP cores used are listed and briefly explained.

Microprocessor. The microprocessor used is dual ARM Cortex-A9 hard-core. It is an embedded core processor integrated on Xilinx ZYNQ FPGAs.

AXI Communication Bus. The AXI bus is part of the wide ARM AMBA, a family of microprocessor buses. The AXI bus interconnects IP contains AXI-compliant master and slave interfaces. It can be used to handle the transactions between one or more AXI masters and slaves. Data can move in both directions between masters and slaves simultaneously, and its sizes can vary.

Ethernet controller. The Tri-Mode Ethernet Media Access Controller (TEMAC) core is a fully verified embedded hard-IP resource. It is a customizable core that enables the designers to implement a wide range of integrated Ethernet design, from 10/100/1000 Mb/s to higher performance Gigabit ports.

SpaceFibre CODEC. The SpFi CODEC incorporates the core of the SpFi communication. It includes the SpFi IP core detailed in previous section, plus it is equipped with a high-performance AXI interface and a DMA engine interfacing with the AXI system, which allows efficient data movement on the system bus connecting the system memory to the SpFi ports. Also hardware packet generator and consumer are added to the CODEC in such a way they do not interfere with the other peripherals in the test equipment. They can saturate the SpFi link bandwidth without using bus bandwidth.

Rolling Memory. The rolling memory module has the role to implement the features of low-level analyser. In particular, it allows to directly monitor which data words are transmitted/received just before 8B10B coding and the Physical layer, in both the SpFi CODEC. This enables the user to investigate the behaviour of the SpFi IP Core, observing its input and output words. The main feature of the Rolling Memory is having a snapshot of the data-stream flowing on the SpFi link when a specific word is received or transmitted. In reception and in transmission, there is a dedicated memory that stores up to 8192 36-bit words. As the name suggests, once the rolling memory is filled, the memory controller overwrites the oldest location in order to continuously save the most recent window of data on the link.

The Rolling Memory controller can be set to trigger on a specific word, and when the word passes through the data-stream, the Rolling Memory continues to store other 4096 words and sends an interrupt to the microprocessor providing a snapshot of the traffic passed through.

Error Injection. This hardware module has the role to enable error injection on the SpFi link. The error injection module acts at the lowest possible word-level, allowing to accurately insert very specific errors in the data stream in order to stimulate complex scenarios. The errors can be independently injected in the two SpFi CODECs, both in transmission and reception sides. In Figure 9, the test setup is shown, with the board connected to the host PC through Ethernet and the SpFi ports connected in loopback with an eSATA cable. Many

specific tests were carried out to verify the correct behaviour of all the subsystems. Hardware packet generators and consumers were intensively tested varying packet size and bandwidth. The Error Injection module was used to greatly stress the SpFi CODEC, simulating different error situations and assessing error recovery capabilities. Both Software and Hardware packet generators and consumers were intensively used to stimulate properly the CODECs. This tool generates a stream of fixed-step incremental data that is sent to a certain VC of the SpFi ports. By using the loopback connection, this stream is received by the other SpFi port and sent back to the Host PC by the SpFi test

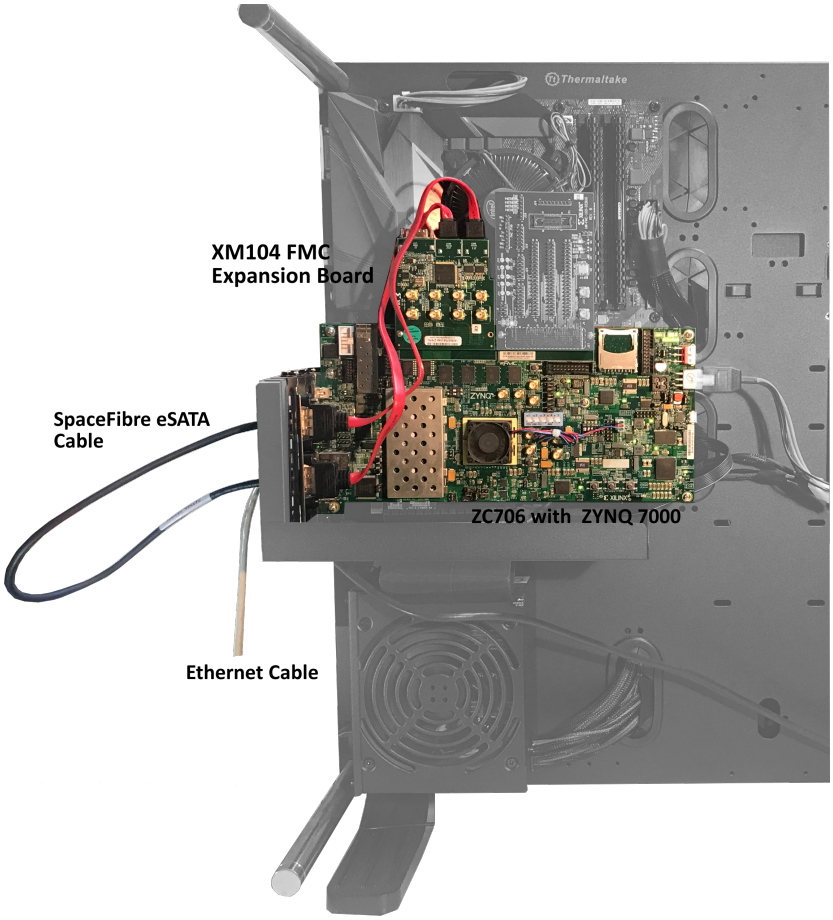

Figure 9: SpFi Validator Testbed.

equipment unit. The software Packet Consumer also checks the correctness of received data.

Once the design demonstrated to be stable, it was successfully tested also against the only commercial product available on the market, the STARFire from STAR-Dundee [26]. It demonstrated to be fully compatible also with independently developed and already qualified products. To properly validate the interface, the test plan described in [30] was followed, in order to cover all the possible SpaceFibre standard requirements. In terms of performance, we achieved 2.5 Gbps on the target technologies. If we take into account a link efficiency of 95 % and 20 % efficiency loss for the 8B10B, the actual bandwidth fully available for data transfer is around 1.9 Gbps. The system was successfully tested, as stated in the test plan [30], with BER up to 10^{-5}, demonstrating to work properly as required in the standard, even though the achieved bandwidth decreases drastically due to the high number of retries.

6. Conclusion

In this paper, the design and implementation of a SpaceFibre CODEC has been presented. The CODEC was developed as a solution for the recent breaking request of above-Gb/s communication for on-board of satellite systems. After a brief introduction to the problem, where the architecture of a satellite data-handling system is presented, an overview of the SpaceFibre protocol is given. The block diagram architecture of the circuit has been presented, first focusing on its interface and main components and then detailing the structure and functionalities of each layer. Once the architecture of circuit has been fixed and the circuit has been described in HDL, the CODEC has been fully verified by means of a SystemVerilog verification environment, covering 100% of the code with different test cases. Finally, the SpaceFibre CODEC IP has been synthesised on various relevant FPGA technologies, both rad-hardened (RTAX2000, RTG4, Virtex 5) and commercial (ZYNQ-7000). A comparison with state-of-the-art solutions is shown, demonstrating that the system is in line with other

devices in terms of performances, and it is able to reduce consistently memory resources used. Indication on power consumption are given, representing an innovative contribution to the state-of-the-art. The presented circuit has been carefully described so that it could serve as a reference point for future developments and investigations. A system validator has been set up, using a commercial ZYNQ-7000 board (ZC706) to validate the system on real hardware. The proposed circuit demonstrated to be fully compliant to the protocol.

Acknowledgment

IngeniArs SpaceFibre technologies have been developed in the framework of the project SIMPLE (Spacefibre IMPLementation design & test Equipment). This project has received funding from the European Unions Horizon 2020 research and innovation programme under Grant Agreement No 757038.

References

References

[1] B. Evans, P. Thompson, G. Corazza, A. Vanelli-Coralli, E. Candreva, 1945-2010: 65 years of satellite history from early visions to latest missions, Proceedings of the IEEE 99 (11) (2011) 1840–1857. doi:10.1109/JPROC.2011.2159467.

[2] ECSS, Spacewire links nodes routers and networks, ECSS-E-ST-50-12C (no. 1).

[3] S. Parkes, P. Armbruster, Spacewire: Spacecraft onboard data-handling network, Acta Astronautica 66 (1) (2010) 88 – 95. doi:https://doi.org/10.1016/j.actaastro.2009.05.016.

[4] S. Saponara, L. Fanucci, M. Tonarelli, E. Petri, Radiation tolerant space wire router for satellite on-board networking, IEEE Aerospace and Electronic Systems Magazine 22 (5) (2007) 3–12. doi:10.1109/MAES.2007.365328.

[5] P. Martimort, V. Fernandez, V. Kirschner, C. Isola, A. Meygret, Sentinel-2 multispectral imager (msi) and calibration/validation, International Geoscience and Remote Sensing Symposium (IGARSS) (2012) 6999–7002doi:10.1109/IGARSS.2012.6351960.

[6] ECSS, Space engineering – spacefibre – very high-speed serial link, ECSS-E-ST-50-11C.

[7] S. Parkes, C. McClements, D. McLaren, A. Florit, A. Villafranca, Spacefibre: A multi-gigabit/s interconnect for spacecraft onboard data handling, IEEE Aerospace Conference Proceedings 2015-June. doi:10.1109/AERO.2015.7119317.

[8] J.-L. TERRAILLON, Savoir: Reusing specifications to favour product lines, in: 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016), TOULOUSE, France, 2016.

[9] M. Gorbunov, Design of fault-tolerant microprocessors for space applications, Acta Astronauticadoi:https://doi.org/10.1016/j.actaastro.2019.04.029.

[10] A. Leoni, P. Nannipieri, L. Fanucci, Vhdl design of a spacefibre routing switch, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E102A (5) (2019) 729–731. doi:10.1587/transfun.E102.A.729.

[11] S. Parkes, A. Florit, A. Villafranca, C. McClements, D. McLaren, Spacefibre network and routing switch, IEEE Aerospace Conference Proceedingsdoi:10.1109/AERO.2017.7943805.

[12] A. Leoni, P. Nannipieri, D. Davalle, L. Fanucci, D. Jameux, Shine: Simulator for satellite on-board high-speed networks featuring spacefibre and spacewire protocols, Aerospace 6 (4). doi:10.3390/aerospace6040043.

[13] R. Xiong, L. Li, X. Yi, C. Zhang, G. Yang, H. Fang, The multilevel dynamic bandwidth allocation and performance analysis of spaceborne net-

work based on spacefibre, Proceedings of the International Astronautical Congress, IAC 2018-October.

[14] P. Nannipieri, G. Dinelli, D. Davalle, L. Fanucci, A spacefibre multi lane codec system on a chip: Enabling technology for low cost satellite egse, PRIME 2018 - 14th Conference on Ph.D. Research in Microelectronics and Electronics (2018) 173–176doi:10.1109/PRIME.2018.8430317.

[15] P. Nannipieri, G. Dinelli, L. Fanucci, A configurable hardware word reordering block for multi-lane communication protocols: Design and use case, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E102A (5) (2019) 747–749. doi:10.1587/transfun.E102.A.747.

[16] I. Korobkov, Algorithm for schedule-table's designing for spacefiber network technology, 2018 Wave Electronics and its Application in Information and Telecommunication Systems, WECONF 2018doi:10.1109/WECONF.2018.8604337.

[17] P. Koopman, T. Chakravarty, Cyclic redundancy code (crc) polynomial selection for embedded networks, Proceedings of the International Conference on Dependable Systems and Networks (2004) 145–154.

[18] P. Nannipieri, D. Davalle, L. Fanucci, A novel parallel 8b/10b encoder: Architecture and comparison with classical solution, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E101A (7) (2018) 1120–1122. doi:10.1587/transfun.E101.A.1120.

[19] ECSS, Space product assurance – asic and fpga development, ECSS-E-ST-60-02C.

[20] R. Madan, N. Kumar, S. Deb, Pragmatic approaches to implement self-checking mechanism in uvm based testbench, Conference Proceeding - 2015 International Conference on Advances in Computer Engineering and

Applications, ICACEA 2015 (2015) 632–636doi:10.1109/ICACEA.2015.7164768.

[21] Microsemi, Microsemi rtax2000 datasheet.
URL https://www.microsemi.com/product-directory/rad-tolerant-fpgas/1694-rtax-s-sl#documents

[22] Microsemi, Microsemi rtg4 datasheet.
URL https://www.microsemi.com/product-directory/rad-tolerant-fpgas/3576-rtg4#documents

[23] Xilinx, Xilinx virtex 5 datasheet.
URL https://www.xilinx.com/support/documentation-navigation/silicon-devices/mature-products/virtex-5.html

[24] Xilinx, Xilinx zynq 7000 datasheet.
URL https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[25] T. Instrument, Tlk2711-sp 1.6-gbps to 2.5-gbps class v transceiver datasheet.
URL http://www.ti.com/lit/ds/sgls307p/sgls307p.pdf

[26] S. Parkes, C. McClements, D. McLaren, B. Youssef, M. Ali, A. Florit, A. Villafranca, Spacewire and spacefibre on the microsemi rtg4 fpga, IEEE Aerospace Conference Proceedings 2016-June. doi:10.1109/AERO.2016.7500644.

[27] F. Siegle, S. Habinc, J. Both, Spacefibre port ip core (grspfi): Spacefibre, poster paper, Proceedings of the 2016 7th International SpaceWire Conference, SpaceWire 2016doi:10.1109/SpaceWire.2016.7771632.

[28] ESA, Esa hdl ip cores portfolio overview.
URL https://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/ESA_HDL_IP_Cores_Portfolio_Overview

[29] D. Davalle, A. Leoni, L. Dello Sterpaio, L. Fanucci, Design and implementation of test equipment for spacefibre links: Spacefibre, short paper, Proceedings of the 2016 7th International SpaceWire Conference, SpaceWire 2016doi:10.1109/SpaceWire.2016.7771618.
URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85010295314&doi=10.1109%2fSpaceWire.2016.7771618&partnerID=40&md5=132c9b8ec5961c364c355c5f44036404

[30] P. Nannipieri, L. Fanucci, F. Siegle, An investigation on spacefibre protocol maturity: Interoperability tests, Proceedings of the DASIA 2019 conference.