Twins4Space: Enabling Modular Space Systems Using a SpaceWire Network

Stefan Lindörfer[1], Sven Wente, Julian Rothe, Sergio Montenegro

[1]Julius-Maximilians Universität Würzburg
Sanderring 2, 97070 Würzburg, Germany
Phone: +49 931 31-87888, Mail: stefan.lindoerfer@uni-wuerzburg.de

Abstract: The processing needs of space missions are increasing quickly. Current space-qualified on-board computers (OBC) are limited in their performance and flexibility. Future applications, such as autonomous spacecraft missions or image processing using AI, will exceed the capabilities of this hardware. To deal with the growing processing demands a modular architecture utilizing cheap and powerful commercial-off-the-shelf (COTS) components is a common approach. The Twins4Space project develops a new architecture of soft- and hardware components including a distributed runtime environment. The RODOS middleware is being adapted to create a modular, industry 4.0 inspired, solution. All nodes are connected by a SpaceWire network with a meshed topology that provides redundant data links to obtain fault tolerance. Available components on the market can be easily integrated into the system. The needed routing functionalities are implemented in VHDL and run in an FPGA. Additionally, the Twins4Space approach is able to complete a dynamic reconfiguration of the whole system during runtime. This enables adding new nodes to the system as well as removing faulty nodes. This paper is to explain the details of the Twins4Space concept and focuses on the implementation of the project's communication layer.

1. INTRODUCTION

Complex systems and missions play an increasingly important role in modern space missions. Modular systems are an adequate approach to meet these increased requirements. By connecting standardized modules, a network is created with different payload functionalities and integrated software applications, enabling the design of spacecraft with a wide range of capabilities. However, modular space systems are challenging to develop due to the complexity in routing and networking and require a deep understanding of the technical requirements.

The overarching goal of the Twins4Space project is to develop and demonstrate such a standardized platform, which should be able to form the basis of a modular space system by providing an easy-to-integrate software and hardware interface for different payloads and modules. This paper focuses on the presentation of the communication layer of the modular platform, describes the configuration and integration of the operating system and shows the implementation using real, commercially available hardware in a network topology.

2. METHODOLOGY

The fundamental component of this platform is a communication layer that not only enables data transfer and routing between connected modules but can also

react dynamically to changes in the network and perform reconfiguration. For this purpose, SpaceWire, a communication protocol specially developed for and tailored to space travel, is used.[1] The layer on top of this is provided with an easy-to-use interface so that the developed functionalities can be used in an uncomplicated way.

On top of the communication layer, an embedded Linux derivative is used as the operating system, which is characterized above all by its high flexibility and accessibility with regard to software development. In addition, the real-time operating system RODOS[2] was integrated via the POSIX interface in order to provide real-time requirements. Furthermore, RODOS takes over important tasks of the network communication using its built-in mechanisms.
The developed and customized framework of hardware and software is illustrated in Figure 1 and is implemented on a system-on-a-chip (SoC) from the manufacturer Xilinx, in which an FPGA is also integrated, and covers the lowest three layers of the OSI reference model up to the network layer.

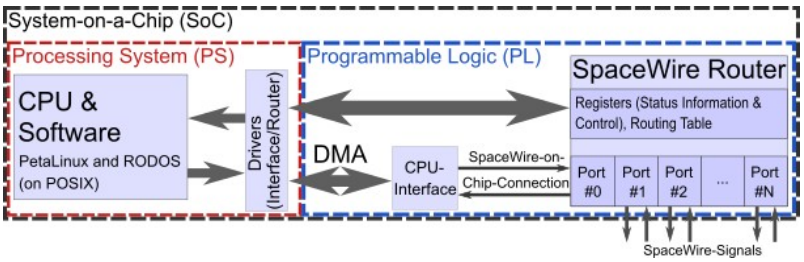

Fig. 1: Schematic Structure of the System-on-a-Chip

2.1 Communication Layer

By outsourcing communication technology applications to the FPGA, two main advantages can be realized. Firstly, there is a higher execution speed compared to relying on the CPU. Secondly, this approach frees the CPU of certain tasks, allowing more computational time on other important processes.

SpaceWire is used as the communication protocol due to its special suitability for space projects, as already mentioned. Its transmission interface standard LVDS (Low Voltage Differential Signaling) makes it less susceptible to noise and radiation. In addition, it offers useful mechanisms such as flow control and is connection-oriented to ensure reliable network traffic.
To realize the routing functionality at the network layer, a SpaceWire router based on the open-source IP "SpaceWire Light IP"[3] has been developed, which can be synthesized with a generic number of physical ports and is SpaceWire standard compliant. The router uses a crossbar architecture and is thus able to handle multiple independent data transfers simultaneously. A connection to the ARM-AMBA/AXI4-Lite bus allows the router to be controlled via software. In addition, the states of the individual ports can be read out to enable the control software to

react to various network events. In general, the router can also be operated without software. Additionally supported is the receiving and sending of SpaceWire Time-Codes over the network, which can also be generated periodically by the router on instruction.
To enable the software to send and receive data, a CPU interface was developed to provide this functionality. In the back-end this device is connected to the ARM-AMBA/AXI4 bus of the CPU and in the front-end to port 0 of the SpaceWire router. This port is specifically used for controlling the nodes and configuring the router. Certain signals, such as the reception of SpaceWire Time-Codes, are also communicated to the CPU via interrupts to allow faster response to events. The CPU interface was furthermore designed to be Direct Memory Access (DMA)-capable so data transfers can be carried out in a resource-saving manner without involving the CPU.

2.2 Software Architecture

Building on the hardware in the FPGA, a customized version of a PetaLinux[4] instance is run, which serves as the operating system. PetaLinux is a derivative of the Yocto-Project[5] developed by Xilinx and provides extensive support for the Xilinx board used, including kernel, drivers and Board Support Package (BSP). Various packages have also been included to facilitate the development of applications on this embedded platform. A custom-developed driver acts as a link between the software and the devices integrated in the FPGA, providing control and data transfer in both directions.
The customized RODOS middleware runs via POSIX on PetaLinux and provides data transfer concepts such as publish-subscribe methods and request-response mechanisms for network communication. Publish-subscribe allows nodes to transmit data over the network regardless of the number of recipients, since recipients are only subscribed to specific types of data. The request-response mechanism, on the other hand, allows a node to specifically request data from another node and wait for a response.

2.3 Routing Functionality

Each node in the network has its own identification number to enable unique addressing. Using a routing algorithm, each node independently detects the network topology and generates the routing table. This process is executed in regular, short intervals in order to be able to react quickly and dynamically to changes in the network. This allows for the addition or removal of nodes in a seamless way, without losing messages.

3. RESULTS

In order to test and verify the devices developed using a hardware description language, extensive testbenches were developed during the design phase. In

addition, after implementation, the SpaceWire compliance of all developed devices was successfully verified and validated using a STAR-Dundee SpaceWire Link Analyzer Mk3[6].

The hardware used for a demonstrator, the Xilinx Zynq Board ZC706[7] does not have SpaceWire connectors, therefore a separate board, shown in Figure 2, had to be developed first to provide this functionality. The board plugs into the FMC connector of the Xilinx board and also provides further GPIO pins for optional connection of additional devices. Required pins are routed from the FPGA to the board and back.

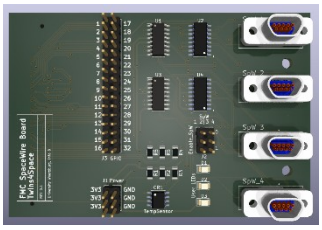

Fig.2: Developed FMC SpaceWire Board

Last but not least, a UART-SpaceWire adapter was designed to be integrated into the FPGA for easy access to the network from outside. It is controlled by commands via the UART interface and offers the possibility to send and receive data packets as well as to query status information.

4. CONCLUSION

In summary, this paper presents the basic architecture of a modular platform that is being developed as part of the Twins4Space project. The article outlined the structure of data communication over the network and explained the concepts and methods utilized for this purpose.
The presented structure will be tested in the next stages of the project by using multiple Xilinx Zynq boards. Through load and stress tests, the robustness and reliability of the structure will be demonstrated on real hardware.

5. REFERENCES

[1] SpaceWire Standard, ECSS-E-ST-50-12C, ESA, Noordwijk (Netherlands) (2019)
[2] S. Montenegro and F. Dannemann, RODOS – real time kernel design for dependability, DASIA (2009)
[3] J. van Rantwijk. SpaceWire Light IP. Available under https://opencores.org/projects/spacewire_light (Accessed on March 3, 2023)
[4] PetaLinux Tools. Available under https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html (Accessed on March 3, 2023)
[5] Yocto-Project. Available under https://www.yoctoproject.org/ (Accessed on March 3, 2023)
[6] STAR-DUNDEE SpaceWire Link Analyzer Mk3. Available under https://www.star-dundee.com/products/spacewire-link-analyser-mk3/#product_features (Accessed on March 3, 2023)
[7] AMD Zynq 7000 SoC ZC706. Available under https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html (Accessed on March 3, 2023)