DEVELOPMENT OF A TM/TC FRONT-END FOR SATELLITE COMMUNICATION WITH REAL-TIME DATA PROCESSING AS AN OPEN SOURCE SOLUTION

S. Schnöge*, H. Ketelhodt*, L. Siffrin*, F. Kunz*, J. Schlemar*, E. Fenske*, F. Leitner-Fischer*, S. Wertheimer†, S. Wanninger†

* Baden-Wuerttemberg Cooperative State University (DHBW) Ravensburg, Faculty of Engineering, Campus Friedrichshafen, Fallenbrunnen 2, 88045 Friedrichshafen, Germany

† SeeSat e.V., Fallenbrunnen 14, 88045 Friedrichshafen, Germany

Abstract

As part of the ERWIN project, SeeSat e.V. in cooperation with students from the DHBW Ravensburg is developing a CubeSat along with its corresponding ground station located at the Friedrichshafen campus. The main objective is to establish a fully operational satellite communication infrastructure directly at the university site. A core element of this infrastructure is the design of a telemetry and telecommand front-end, which is intended to process satellite data in real time in compliance with the international Consultative Committee for Space Data Systems (CCSDS) standard. On the receiving side, these front-end extracts the payload data from digitized radio signals, while on the transmitting side, it prepares data streams according to defined communication protocols.

The system is based on an AMD Zynq 7000 system-on-chip, combining an ARM processor with field-programmable gate array (FPGA) technology to achieve powerful real-time processing capabilities. It is designed as an modular component with defined network interfaces. Received data is sequentially decoded using convolutional decoding followed by Reed-Solomon decoding. Furthermore, received data is archived at all protocol levels using a RAID 5 storage system, ensuring reliability and traceability.

Future developments include the integration of a graphical user interface to provide intuitive monitoring and analysis of satellite data. The overall aim is to make both the software and hardware design openly available, thereby lowering the entry barrier for research institutions to communicate with CCSDS-compliant satellites and supporting further innovation in satellite communication.

Keywords

TM/TC front-end; telemetry processing; CCSDS standards; open-source satellite infrastructure

NOMENCLATURE

AMD	Advanced Micro Devices
ARM	Advanced RISC Machines
ASM	Attached Sync Marker
AXI	Advanced eXtensible Interface
CADU	Channel Access Data Unit
CCSDS	Consultative Committee for Space Data Systems
CPU	Central Processing Unit
DHBW	Baden-Wuerttemberg Cooperative State University (Germany)
ERWIN	Emission of Radiation based Wildfire Investigation
FECF	Frame Error Control Field
FPGA	field-programmable gate array
GUI	graphical user interface
HLS	high-level synthesis
LDPC	Low-Density Parity-Check
MAC	Media Access Controller
NAS	network-attached storage
PL	programmable logic
PS	processing system
SoC	system-on-chip
TC	telecommand
TEMAC	Tri-Mode Ethernet MAC
TM/TC	telemetry and telecommand
TM	telemetry
UHF	ultra-high frequency
VHDL	VHSIC Hardware Description Language

doi: 10.25967/650393

1. INTRODUCTION

The Emission of Radiation based Wildfire Investigation (ERWIN) CubeSat project, developed by SeeSat e.V. in collaboration with the Baden-Wuerttemberg Cooperative State University (DHBW) Ravensburg, is a student-led satellite mission with the objective of detecting wildfires from orbit, supporting early warning and environmental monitoring systems [1]. In support of this mission, the project aims to establish a modular and scalable ground segment for satellite communication. A key subsystem within this ground segment is the telemetry and telecommand (TM/TC) front-end, responsible for decoding incoming satellite data and encoding outbound control commands in compliance with Consultative Committee for Space Data Systems (CCSDS) standards [2]. This paper presents the design of a standalone, CCSDS-compliant TM/TC front-end based on the Advanced Micro Devices (AMD) Zynq 7000 system-on-chip (SoC) platform. The signal processing and protocol stack are developed using a hybrid design methodology. Initially, core components are modeled in Python and subsequently synthesized into VHSIC Hardware Description Language (VHDL) via high-level synthesis (HLS). This approach enables rapid prototyping while leveraging the performance benefits of the field-programmable gate array (FPGA)-based hardware acceleration. To ensure reliable storage of received satellite data, the system includes a network-attached storage (NAS) solution configured with RAID 5.
The scope of this project includes requirements engineering, system architecture design, hardware/software co-design, interface specification, and a comprehensive verification and validation process. This paper presents the architectural approach, development methodology, and key implementation results of the TM/TC front-end system.

2. FOUNDATIONS

To contextualize the design choices for the TM/TC front-end, this section outlines the ERWIN CubeSat mission, its payload, the ground station architecture, and the strategies employed for data handling and storage, providing the basis for understanding the system design.

2.1. Satellite Overview

The ERWIN spacecraft will adhere to the 2U CubeSat standard ($200 \times 100 \times 100$ mm) and will be operated in a sun-synchronous orbit at an altitude of approximately 600 km. Its mission objective is the detection of wildfires, for which it will carry a bolometer as the primary payload to measure thermal radiation. This will be complemented by an optical camera for georeferencing. The satellite will generate approximately 15 W of electrical power and will employ ultra-high frequency (UHF) and S-Band links for telemetry, telecommand, and payload data transmission [1].

2.2. Ground Station Architecture

The ERWIN mission will be operated from a dedicated ground station located in Friedrichshafen. A key element of this infrastructure will be the TM/TC front-end, which is conceived as a standalone, modular and open-source system to allow a flexible and budget-friendly integration into various ground station configurations. This choice balances hardware complexity, ease of future upgrades, and performance requirements.
The design specifies implementation on an AMD Zynq-7000 SoC, which integrates reconfigurable FPGA logic with an embedded Advanced RISC Machines (ARM) processor. The selection of this SoC reflects a trade-off between the need for real-time hardware acceleration on the FPGA and ease of software development on the embedded processor. The FPGA fabric will be dedicated to real-time tasks, specifically the encoding and decoding of telemetry and telecommand data streams in compliance with CCSDS standards [2]. By directly implementing the protocol stack in hardware, the system will achieve the performance required to reliably process satellite communication data rates. Data inputs to the FPGA will originate either from the satellite, received via the antenna system, or from the operator workstation, which will provide a user interface and connect to the front-end via Ethernet as seen in Figure 1.

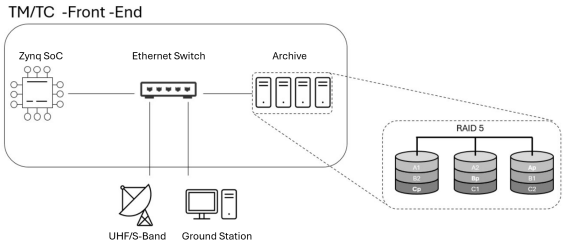

FIG 1. System overview of front-end

The ARM processor is intended for non-real-time operations, including data management, storage coordination, and system-level communication. This functional division ensures that the FPGA resources remain fully available for high-speed data processing, while the processor handles supervisory and archival tasks.
Currently, the default Zynq-Z2 board configuration routes the Ethernet interface exclusively to the processing system (PS) side. As a result, direct data transfer into the programmable logic (PL) is not possible, limiting the real-time performance of FPGA-based processing. This data-flow bottleneck constrains the system's ability to fully exploit parallelism in the FPGA fabric, and is illustrated in Figure 2. This illustration is a screenshot from the Vivado board configuration.

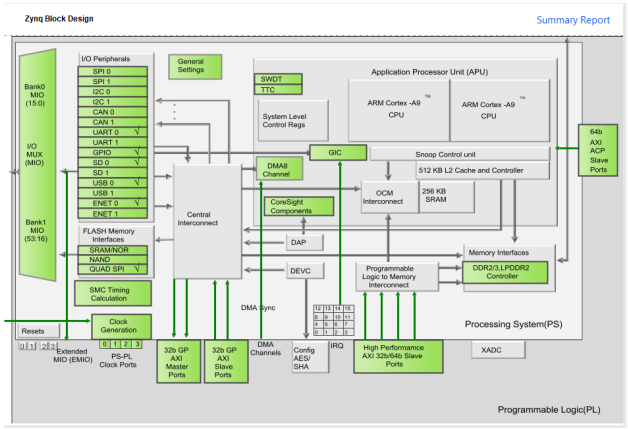

FIG 2. Data-flow bottleneck in the Zynq-7000 SoC due to Ethernet routing exclusively to the PS, limiting direct PL access.

2.3. Data Archiving and Reliability

All telemetry and payload data will be persistently stored on a NAS system connected to the front-end via an Ethernet switch. To guarantee both availability and resilience against hardware failures, the NAS will be configured with a RAID 5 scheme. This configuration will distribute parity information across multiple disks, thereby enabling recovery from a single disk failure without data loss, while simultaneously providing efficient utilization of the available storage capacity. The reliable archiving of mission data is crucial for subsequent post-processing, long-term analysis, and comprehensive mission documentation, making this storage solution a central component of the system.

The NAS itself is implemented as a custom-built system based on a Raspberry Pi. It is operated with open-source software, representing a cost-effective alternative to commercial NAS solutions. This design minimizes costs while ensuring redundancy and fault tolerance, prioritizing reliable data archiving over maximum throughput, which aligns with the mission-critical requirements.

The storage system is designed to support a network connection of up to 10 Gigabit Ethernet and aims for low latency to enable timely archiving of received telemetry and telecommand data.

3. DESIGN OF THE TELEMETRY FRONTEND

Given the operational requirements of the ERWIN mission, the TM front-end is designed to efficiently process telemetry and telecommand streams in real-time.

The central problem begins with the reception of the raw data stream, provided by the antenna as a continuous bit flow. In this form, the data are unusable for subsequent application layers, such as mission control or scientific payload processing. The stream must first be synchronized, then corrected for transmission errors using channel coding schemes, and finally converted into standardized transfer frames. The objective of the presented concept is the development of a flexible and modular telemetry (TM) front-end that supports a variety of coding and synchronization schemes as defined by the CCSDS [2]. The key contribution is the realization of a multi-path architecture, enabling the system to dynamically adapt to the requirements of different transmission standards. This ensures high compatibility with missions across generations, from legacy satellites to state-of-the-art platforms.

3.1. Theoretical Background: CCSDS Standards for Telemetry

Interoperability in spaceflight is largely achieved through the standards defined by the CCSDS. These recommendations, published as so-called Blue Books, specify protocols and data structures to ensure seamless communication between systems of different agencies. For the presented TM front-end, the primary reference is CCSDS 131.0-B-5: TM Synchronization and Channel Coding [2]. Key concepts include:

Channel Coding: Forward Error Correction appends redundant information to the payload, enabling error detection and correction without retransmission. A common approach is concatenated coding: an inner convolutional code protects against random bit errors, while an outer Reed-Solomon error-correcting code addresses burst errors.

Transfer Frame: The fundamental data packet for synchronization and channel coding, carrying the payload. Although the length can vary, it is specified by the operator depending on the applied implementation of the CCSDS layer above.

Channel Access Data Unit (CADU): Generated after applying the outer code to a transfer frame and adding an Attached Sync Marker (ASM).

Synchronization: The ASM, a fixed bit sequence, is used to identify the start of each CADU in the continuous stream and ensure reliable frame synchronization.

3.2. Architecture of the TM Processing System

The TM front-end design employs a parallel pipeline architecture. The incoming stream of channel symbols is directed to one of multiple specialized processing chains, each designed to implement a particular CCSDS coding and synchronization scheme.

3.2.1. Multi-Path Design

The choice of a multi-path architecture is motivated by the need for flexibility and backward compatibility. Different missions employ different coding schemes depending on mission generation and requirements. While traditional missions rely on the concatenated convolutional and Reed-Solomon codes, modern missions increasingly use Turbo or Low-Density Parity-Check (LDPC) codes. Before the mission start, the operator can configure the system via a dedicated settings interface to select the appropriate processing path. Paths are classified as "mandatory/chosen" or "optional," reflecting the definition specified in the different CCSDS standards [3].

3.2.2. Detailed Description of an Exemplary Path

To illustrate the functionality, the classical processing path is described, which implements the widely established concatenation of convolutional and Reed-Solomon coding. This path has proven to be a robust and reliable standard architecture.

Convolutional Decoder: Using the Viterbi algorithm, the decoder reverses the convolutional encoding applied to the satellite, correcting high-frequency random bit errors. This significantly reduces the bit error rate, forming a reliable basis for subsequent error correction stages.

ASM Remover (Synchronization Unit): The system searches for the ASM in the stream. Once detected, the beginning of a CADU is identified and synchronization is established. The ASM is removed, as it is not required for further processing.

Reed-Solomon Codeblock Reconstructor and Decoder: This stage performs outer error correction, reconstructing and correcting code blocks affected by burst errors. Reed-Solomon codes can recover a predefined number of corrupted symbols, substantially improving transmission reliability.

Pseudo Randomization Reconstructor (Derandomizer): The derandomizer reverses the bit scrambling applied on the satellite to ensure a balanced distribution of ones and zeros. This restores the original structure of the transfer frame.

FECF Check (Integrity Verification): Finally, the Frame Error Control Field (FECF) is evaluated. If the checksum does not match, the frame is flagged as corrupted. Only error-free and fully reconstructed frames are forwarded to subsequent ground segment systems.

This path combines complementary error correction mechanisms with robust synchronization and integrity verification. It represents the reference implementation against which modern alternatives, such as Turbo and LDPC coding, are benchmarked in terms of efficiency and error resilience [2–4].

3.2.3. Comparison of Telemetry Processing Paths

While the conventional path ensures maximum compatibility with legacy missions, the Turbo and LDPC paths provide significantly higher performance at low signal-to-noise ratios. By employing iterative decoding, they operate close to the theoretical Shannon limit, enabling reliable transmission even at very low signal strengths. LDPC paths are implemented in two variants:

- Processing CADUs with ASM-based synchronization.
- Processing continuous streams of Synchronous/Streaming Mission Telemetry Frames, where synchronization is achieved via Codeword Sync Markers integrated into the LDPC codewords.

The latter approach avoids ASM overhead and is particularly suitable for high-data-rate streaming applications [5]. Prototype results indicate feasibility of both approaches, though full implementation remains future work.

3.3. Interfaces and Data Flow

The TM front-end design includes well-defined interfaces for integration into a ground segment:

Input Interface: Raw channel symbols are supplied by a preceding receiver (e.g., S-band receiver) via a network interface, typically Ethernet.

Control and Monitoring: Operators configure the system via a user/settings interface by selecting the processing path and specifying mission parameters (e.g., data rate). At the same time, continuous status updates (e.g., frame lock, error correction statistics, data quality) are provided.

Output Interface: The final output consists of error-free, synchronized, and CCSDS-compliant transfer frames. These are delivered to subsequent systems such as mission control or scientific processing pipelines.

The front-end design targets real-time message processing, with the goal of decoding and forwarding each message within the interval of the next incoming message. The design aims to meet latency requirements of less than one millisecond, aligning with the mission-critical performance goals.

4. LESSONS LEARNED

The implementation of the project provided numerous opportunities to gain both technical and personal experience. This chapter offers a summary of the key insights gained during the developmental process, implementation, and collaborative efforts. These can be classified into two categories: technical insights and personal and team-related insights.

4.1. Technical Insights

During the project, it became evident that compatibility is a critical factor in hardware selection. The initially specified hardware presented significant challenges as the project progressed, particularly because the interfaces of the chosen components were insufficient to implement the planned tasks within the available timeframe. For future projects, a clearly defined decision-making and validation process should be established to systematically assess the suitability of hardware. In this context, it is particularly important to first specify the hardware requirements precisely in the planning phase before selecting a specific platform. One potential approach is to assign the creation of requirements and the selection of hardware to different individuals, who interact only minimally during this process. This strategy ensures a dual control principle, whereby the final selection is verified independently.

4.2. Personal Takeaways

In addition to technical insights, the project revealed important interpersonal aspects, emphasizing the necessity of maintaining a clear separation between personal relationships and professional responsibilities. Establishing this professional distance proved crucial for the progress of the project, particularly when addressing technical

challenges.
Furthermore, regular group meetings proved to be very effective. These meetings allowed tasks, problems, and solutions to be addressed directly, ensuring a continuous flow of information and promoting transparency between the different areas of work. The use of shared cloud services and a project dashboard that showed the current status of the individual work packages also contributed significantly to strengthening team dynamics. The awareness that each member was actively contributing to progress prevented resentment over unevenly distributed workloads, and at the same time promoted a willingness to take on new tasks flexibly.

5. CONCLUSION AND FUTURE WORK

The present work provides insights into the design and implementation of the TM front-end system, highlighting both the achievements and the challenges encountered. The most salient findings are presented in the conclusion, while the future work section identifies remaining tasks, potential improvements, directions for further development, particularly regarding FPGA-based acceleration, telecommand (TC) integration, and user interface design.

5.1. Conclusion

The work presented in this paper demonstrates the design and conceptual integration of a modular, CCSDS-compliant TM front-end for satellite communication. The system provides a flexible multi-path architecture that supports both legacy and modern coding schemes, ensuring compatibility across different mission generations. The proposed RAID 5 NAS design further guarantees reliable data archiving, supporting integrity and traceability of satellite data.
Technical and design challenges, such as interface constraints and the need for optimized FPGA-based processing, were identified and provide valuable guidance for subsequent implementation. Overall, the TM front-end establishes an extensible framework for ground segment infrastructure, forming a solid basis for further development and eventual operational deployment.

5.2. Future Work

Building on these results, several key components remain under development. Specifically, parts of the Python-based implementation are still incomplete. The implementation of VHDL code on the FPGA has not yet been completed, and a fully functional front-end with a graphical user interface and display has not yet been developed. These aspects highlight possibilities for future work. Key tasks include the completion of the TM system, the integration of the TC component to enable command transmission and system control, and the implementation of the processing logic in VHDL on the FPGA for hardware acceleration. Another focus will be the development of an intuitive front-end.
A significant technical challenge identified during the project pertains to the default configuration of the Zynq-Z2 board, wherein the Ethernet interface is exclusively connected to the PS side. This architecture prevents direct data transfer into the PL, eliminating the timing advantage of FPGA-based processing.
The reconfiguration of the SoC architecture in Vivado presents a promising avenue for future development. Establishing this connection requires linking the Ethernet interface directly to the PL through an additional Media Access Controller (MAC), such as a Tri-Mode Ethernet MAC (TEMAC) core, to the FPGA logic [6]. The revised configuration permits the FPGA to process incoming data streams independently, bypassing the Central Processing Unit (CPU) and thereby reducing latency. Implementing this solution requires reconfiguring the SoC in Vivado and adjusting the FPGA logic to integrate the additional MAC. The Advanced eXtensible Interface (AXI) data streaming interfaces also need to be configured and the timing constraints verified. This method provides full FPGA acceleration but also increases design complexity and necessitates careful verification.
An alternative approach is to use hardware that provides at least two Ethernet interfaces: one connected to the PS to handle standard networking and control, and a second one directly connected to the PL for high-speed data streaming. This configuration enables the direct transfer of data into the FPGA, thereby ensuring that the processing pipeline fully utilizes the FPGA's parallelism, without the need for routing through the PS. While this approach has the potential to simplify design modifications and enhance performance, it may incur higher hardware expenses and necessitate adjustments to the system architecture.
Both strategies have the potential to restore the intended performance benefits of FPGA acceleration. However, they differ in terms of design complexity, cost, and adaptability. Evaluating these trade-offs is essential to determine the most suitable solution for operational constraints.
Beyond the Ethernet architecture, further work is required to complete the overall system. This includes the implementation of a complete front-end with a graphical user interface for real-time data visualization, potentially complemented by a display for user interaction. In addition, systematic testing procedures for both the TC functionality and the FPGA implementation should be established to ensure correctness and reliability. Performance measurements, simulations, and subsequent optimizations will be valuable to assess system efficiency and guide future improvements.

Contact address:

sarah_schnoege@web.de
jeremyschlemar@yahoo.de
info@seesat.eu

References

[1] SeeSat e.V. Erwin – emission of radiation based wildfire investigation. https://seesat.eu/erwin/, 2025. Accessed: 2025-09-02.

[2] Consultative Committee for Space Data Systems (CCSDS). Tm synchronization and channel coding, blue book ccsds 131.0-b-5. Technical report, Consultative Committee for Space Data Systems (CCSDS), Washington, D.C., US, 2023.

[3] Stephen B. Wicker and Saejoon Kim. *Fundamentals of Codes, Graphs, and Iterative Decoding*. Springer NewYork,Ny, 2002. ISBN: 978-0-306-47794-2.

[4] M. S. Ryan and G. R. Nudd. *The Viterbi algorithm*. PhD thesis, University of Warwick. Department of Computer Science, 1993. Unpublished. https://wrap.warwick.ac.uk/id/eprint/60926/.

[5] P. Massoud Salehi and J. Proakis. *Digital Communications*. McGraw-Hill Education, 2007. ISBN: 9780072957167.

[6] AMD. Tri-Mode Ethernet Media Access Controller (TEMAC), 2025. https://www.amd.com/de/products/adaptive-socs-and-fpgas/intellectual-property/temac.html.