DEVELOPMENT OF AN OPEN-SOURCE FPGA CCSDS DATA LINK IMPLEMENTATION FOR SATELLITE TELEMETRY

M. D. Fuchs*, R. Eilers*, H. L. Lindner*, L. Reil*, N. Tunkowski*, F. Leitner-Fischer *, S. Wertheimer†
* Duale Hochschule Baden-Württemberg (DHBW) Ravensburg, Faculty of Engineering, Fallenbrunnen 2, 88045 Friedrichshafen, Germany
† SeeSat e.V., Fallenbrunnen 14, 88045 Friedrichshafen, Germany

Abstract

Every satellite project requires communication between the ground and the spacecraft. Telecommand and telemetry packets are used to control the spacecraft and send payload and housekeeping data back to Earth. The CCSDS offers standards, which cover many different communication and data handling protocols, from the physical layer to the application layer, solving a plethora of problems these channels face.
This work focuses on the data-link layer for telemetry packets, which is particularly defined by CCSDS 132.0-B-3 and CCSDS 131.0-B-6. The former defines the channel and data flow information of the packets, while the latter defines the synchronization and error correction measures. When implementing these standards for a specific mission, teams may either use open-source software-based solutions or commercial FPGA-based IP cores. Currently, no FPGA-based open-source IP cores are available. This forces small teams of students or hobbyists, such as those working on the CubeSat mission ERWIN, developed by SeeSat e.V., to use the software-based solution, since the commercial IP cores would exceed the budget of most projects of this scale. The goal of this work is to design and develop an FPGA-based open source solution of both standards. The system focuses on the concatenated coding path consisting of Reed Solomon and Convolutional Codes. During the initial development stage, the proposed encoder and decoder will be implemented on an AMD Zynq 7020 SoC. The final target system for ERWIN will be an AMD UltraScale SoC. Both the space and ground segments are implemented in VHDL and tested, resulting in: $42.370\,81\,\mathrm{Mbit/s}$ user data throughput, and a bit error rate improvement versus the uncoded case of 1 to 100 times. The full encoder requires 3700 LUT, and the decoder 11800 LUT.

Keywords
TM/TC; FPGA; CCSDS

1. INTRODUCTION

Almost all satellite projects require a communication link between the ground and the spacecraft, enabling the development and operations team to control and monitor the system and its behaviour. Messages sent via this link are commonly referred to as Telecommand (TC) and Telemetry (TM) packets.
TC messages are sent to the satellite, containing commands and instructions for the system to follow, while TM messages are sent by the satellite, containing information about the system's state or payload data, for example.
The goal of this work is to propose an open-source VHDL implementation of a Consultative Committee for Space Data Systems (CCSDS) compliant data link layer that can be used as such a TM/TC link. The followed specifications are: **CCSDS 132.0-B-3: TM Space Data Link Protocol** [1], as well as **CCSDS 131.0-B-6: TM Synchronization and Channel Coding** [2].
The implementation will focus on a legacy path that was proposed in previous work [3]. This data path consists of the packaging and data flow information defined in the CCSDS 132.0-B-3 standard, as well as the concatenated coding as defined in the CCSDS 131.0-B-6, as Reed–Solomon (R/S) and convolutional codes.

2. STRUCTURE OF THIS WORK

This work starts by giving an overview of the project context, before evaluating related work regarding other implementations of the referenced standards. Next, these standards are explained with respect to the implemented features, before the system architecture is subsequently explained. The 7th chapter goes further into detail about the implementation of the system components, before providing a detailed evaluation and testing of the system. Lastly, the results of this project are discussed and future work is identified.

CC BY 4.0

doi: 10.25967/680183

3. CONTEXT

This project was carried out in the scope of a Student Project at DHBW Ravensburg, in cooperation with the SeeSat e.V., a student and hobbyist space team. The system described in this work shall be integrated into SeeSat's ERWIN mission - a Cube Satellite with the goal to detect wildfires from a sun synchronous orbit using an infrared imaging sensor.

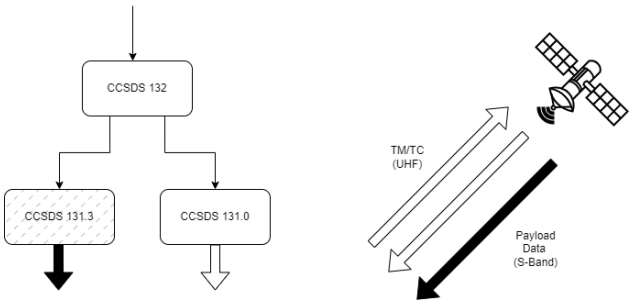

FIG 1. SeeSat Mission Concept

Figure 1 shows a high level view of the satellite's communication system. For communication purposes, it is equipped with an S-Band antenna that will be exclusively used for payload data, as well as an UHF antenna for communication of TM/TC information.
Although the payload link is not of further relevance for this paper, it is important to note that it is modulated in accordance with [4] [5] and therefore needs to pass through the CCSDS 132.0-B-3 as well.

4. RELATED WORK

Although many space teams face the issue of needing a TM/TC link, there are rarely open-source implementations of the specifications referred to in this work.
Available open-source software implementations include: [6], which is an implementation of the CCSDS 132.0-B-3 specification in C that was published under the Apache 2.0 License. For C++, there is [7], which was published under the MIT license. As for the CCSDS 131.0-B-5, there are implementations like the GNU Radio module, [8], that implements Reed Solomon coding and pseudo-randomization, but is missing the elements for the convolutional code.
Software implementations are, generally speaking, less complex to implement and more flexible, but especially when designing high data rate systems, implementing the data link layer in hardware increases its overall performance. Although there are many hardware implementations of single elements of the described standards, e.g. [9], [10], there was no suitable open-source implementation of the entire standard available at the time of development.

5. STANDARDS

The CCSDS data link layer is covered by two separate specifications, one describing packet formatting and padding [1], and the other one describing error correction and synchronization [2].

5.1. CCSDS 132.0-B-3

The CCSDS 132.0-B layer is responsible for formatting and padding the incoming space packets into the predefined structure of TM transfer frames. This is done to provide a continuous, fixed length data stream to the following elements, which is important, as space packets can vary in length. When no space packets are available to be processed, the layer shall generate dummy frames to guarantee a continuous transmission.
The space data link specification defines how space packets from different sources are packed into their respective TM Transfer Frame. These sources are called virtual and master channels, as it is not allowed for space packets of different sources to be mixed in a TM Transfer frame. This layer manages this flow control. The goal here is to multiplex/demultiplex different channels onto a singular physical channel. It also defines that space packets need to be merged or split depending on their size and the available space in a TM Transfer frame.
Additionally, to the data field, defined, the standard also defines header and trailer information that is added to the frame and contains information that is required for the correct identification and decoding of TM Frame on the receiving end of the chain, and operational control field (OCF) and frame error control field (FECF) may be implemented.

5.2. CCSDS 131.0-B-6

The CCSDS 131.0-B standard describes the synchronization and channel coding mechanisms that shall be used in the TM Space Data Link Protocol. Its content can further be subdivided into the mechanisms used for synchronization and the ones used for error detection and correction.
The synchronization layer describes how synchronization is achieved by adding a fixed 32 Bit synchronization marker between TM frames. This, so called Attached Synchronous Marker (ASM), is detected on the receiving end and used for identifying frame borders, which is essential for the error correction mechanisms to work.
This error correction mechanism, that is also referred to as coding layer, defines a list of algorithms to be used for error correction and error avoidance. As this work mainly focuses on the legacy path where a concatenated coding scheme is used. This consists of a Reed Solomon code as the outer coding layer, and the inner coding being defined as convolutional codes.
To increase the bit transition density, a pseudo randomization is added between the Reed Solomon and convolutional code before the attachment of ASMs.

CC BY 4.0

An overview of how the processing chain is structured following the concatenated coding scheme can be seen in Figure 2.



FIG 2. CCSDS 131 overview

The standard itself also defines the use of turbo codes or LDPC codes for an even larger coding gain, which is not discussed further in the scope of this work. It should be noted that the architecture described in this work was built using the previous version of the CCSDS TM Synchronization and Channel Coding Standard, CCSDS 131.0-B-5. In April of 2026, the CCSDS released a new version of the standard, CCSDS 131.0-B-6. To the knowledge of the authors, no significant change was introduced affecting the results of this work [2].

6. CONCEPT

The general architecture of the proposed system closely follows the design flow of the CCSDS Standards [1] [2]. A strict separation of concerns is used between both sub layers and their corresponding modules. This loose coupling allows interchange of modules, or even whole layers. In the example of ERWIN, the CCSDS 132.0-B-3, in addition to the ASMs implemented by the CCSDS 131.0-B-6, may also be used as a data-link layer for the high bandwidth S-Band, using DVB-S2 [11] [5].

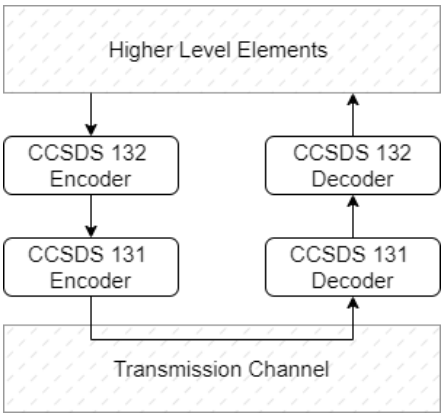

FIG 3. High Level overview of TM Space Data Link Protocol

The whole system is organised into four sub-modules, two for each layer, and two for the corresponding decoder and encoder layers. In addition, interface and control flow logic is defined between the modules.

6.1. Interfaces

The AXI4-Stream protocol [12], was chosen as the interface for all components within the implemented encoder and decoder, as well as for all external interfaces. This provides a safe and stable flow control mechanism, and allows for easy interfacing with the AXI4-Stream IP cores provided by Xilinx that can be used to interface with other components of the development board, for example [13].
Although AXI4-Stream only defines data widths that are clean multiples of a byte, the internal interfaces were implemented with a 1 bit wide TDATA signal. This is due to the fact that the pseudorandomizer, the ASM and the convolutional coding elements work with one bit per clock cycle. The modules that are able to run using higher bit widths, like the TM-frame encoder/decoder or the Reed Solomon encoder/decoder, run at a higher width of 8 Bit.
AXI4-Stream's TLAST flag is used to mark the endings of the frames/packets and is used to trigger the corresponding behaviour inside the modules. For example, in the TM Synchronization and Channel Coding layer, they are used to mark the end of a Reed-Solomon Code word to reset the pseudorandomizer element and add an ASM.

6.2. CCSDS 132.0-B-3: Encoder

The data-link encoder implements a state machine to model which part of the TM-Transfer-Frame shall be transmitted on its 1 Byte output bus. The module has internal counters to model the byte number to be outputted inside of a field; if the number of bytes reaches the size of a field, the next state transition triggers. Allowing for constant output to the synchronization and channel coding sub-layer. For each Virtual Channel, the five parameters are modeled, as shown in Table 1. These decide if an OFC or FEC field shall be generated. Currently, these fields are only implemented using dummy data.

TAB 1. Configuration for each Virtual Channel

Parameter	Value Type
VC ID	3-Bit Integer
TM-Frame version Number	2-Bit Integer
Spacecraft ID	10-Bit Integer
has OFC	Boolean
has FECF	Boolean

For each virtual channel, an AXI4-Stream FIFO is used as an input. The Virtual channel itself buffers the payload for a full TM-Frame before sending it to the TM-Frame encoder, ensuring only completely filled TM-Frames are sent. In addition to the external Virtual channels that allow for transmission of user data from the space craft, another virtual channel is defined, the only idle data (OID) channel. This channel ensures lower layers always have a constant data stream, since if no user data is available to send, the OID channel generates random numbers. Therefore, the priority of the virtual channels is the lowest virtual channel ID to the highest, with the lowest priority being the OID.

CC BY 4.0

6.3. CCSDS 132.0-B-3: Decoder

The Data Link Protocol Sub-Layer Decoder is responsible for separating the incoming TM transfer frames into transportable data units. The main focus is on the space packets, as they are containing the telemetry that is mainly transferred via the TM chain. Other data units include the data of the secondary header or the operational control field, which allows the use of the security protocol or retransmission control [1]. As the goal of this work is to implement a basic and modular hardware implementation, only the packet transfer will be described.

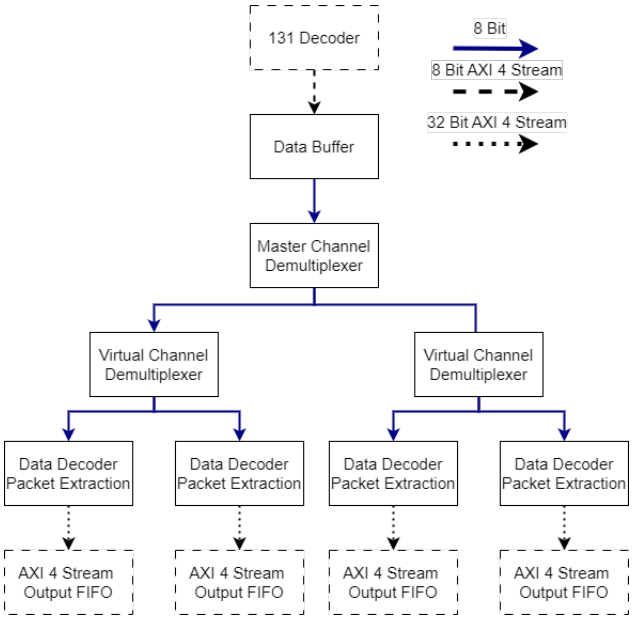

FIG 4. CCSDS 132 Decoder Architecture

The CCSDS 132.0.1 separates the data aggregation into virtual, master, and physical channels. The communication link (normally the antenna) is represented by the physical channel, holding multiple master channels, which can consist of multiple virtual channels. As described in subsection 6.2, the multiple user configured data sources are used to construct a fixed length TM transfer frame containing data from one virtual channel and/or idle data. The architecture of the decoder, as shown in Figure 4, allows the separation of every virtual channel packet data into its dedicated AXI4-Stream FIFO, to be extracted by the receiver. Similar to the encoder, the decoder also employs a state machine to model what part of the TM-transfer frame is expected and how the data shall be interpreted.

6.4. CCSDS 131.0-B-6: Encoder

The synchronization and channel encoder is subdivided into four elements. Starting from the interface layer between the Data-Link layer and finishing at the Physical layer interface, the first element is the Reed Solomon encoder. This is implemented as a $GF(2^8)$ polynomial divider. For this project, the larger of the two generator polynomials, defined by the CCSDS, is chosen, so up to 16 errors are correctable per R/S Code word of 255 Bytes in length (223 Bytes user data and 32 Bytes parity). No interleaving and no virtual fill is implemented [2]. As elaborated above, the R/S Encoder generates its own TLAST flag if a R/S Code word is finished.
The next element is the Pseudorandomization (RNG); this is used to archive a higher bit change rate. For this work, the longer random sequence of degree 17 and 131071-Bit period is chosen, defined by the CCSDS. After every RS Code word, the RNG sequence is reset to its initial condition; this means for every Code word, the same sequence is generated by the RNG. This is XORed with the incoming data stream from the R/S.
To keep different elements of the system in synchronization, the attached synchronization marker (ASM) is used; this element generates a predefined bit pattern between Reed-Solomon code words. This is also triggered by the R/S TLAST flag that is forwarded by the RNG. Both elements are kept separate to ensure high separation of concern, since the ASM may be used for other physical channels, like the DVB-S2 [11].
The convolutional encoder is directly implemented, like the CCSDS 131.0-B-6 recommends, using a shift register of length 6 and $GF(2)$ adders. The whole system is designed to allow for a constant data stream on the output of the convolutional encoder, this would allow a physical layer to just incorporate symbol mapping and modulation.

6.5. CCSDS 131.0-B-6: Decoder

Similar to the synchronization and channel encoder, the decoder is subdivided into four elements. Every one inverts the corresponding code generated by the encoder, and performs error correction or synchronization, depending on the element.
Starting with the Reed Solomon decoder, similar to many other elements, hardware complexity shall be low. In general, the R/S decoder is split into 3 units, the first one calculates the syndromes, the next one uses the Enhanced Parallel Inversionless Berlekamp-Massey Algorithm (ePIBMA) architecture proposed by [14] to find error locator and error magnitude polynomials, this algorithm is an improvement on the classical Berlekamp-Massey Algorithm used to decode R/S Codes [15] [16]. The last unit is the Enhanced Chien Search & Error Evaluation (eCSEE), also proposed by [14], this performs the error magnitude and position calculation, and finally outputs the correction values to the datastream, this data stream is delayed using a FIFO that has the same length as cycles required by the three units to produce a result [14]. The output of the FIFO is added with the error magnitudes provided by the decoder architecture, to correct up to 16 Byte errors in a codeword. This is the output of the CCSDS 131.0-B-6: Decoder. The whole R/S decoder architecture is shown in Figure 5.

CC BY 4.0

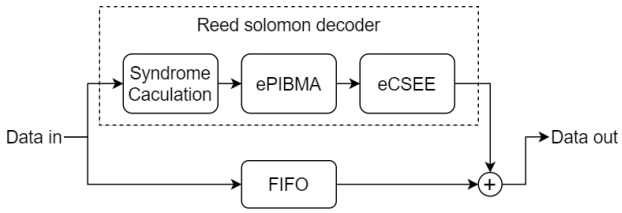

FIG 5. Top level overview of R/S Decoder

The next element is the De-RNG unit, since the RNG operation is invertible by running the same RNG pattern over the data word again, the same RNG unit can be used in the Decoder architecture as is used by the Encoder. This of course requires resynchronization to the Reed-Solomon code words separated by ASMs. The ASM decoder tries to archive this synchronization; it implements a shift register of the pattern length it tries to match to. Since an exact matching is prone to synchronization errors, especially in a high noise environment [17]. A Hamming matcher is used by calculating the Hamming distance between the pattern and the shift register, for this work 6 was chosen as a matching distance, meaning that 6 bits are allowed to flip in an ASM to still detect it as a correct ASM [18].
Since the decoding of the Convolutional Code is not a trivial task, involving a Viterbi Decoder [19] of high backtracking length and large state space, a suitable open source implementation is used. The open source Viterbi Decoder IP-Core, designed and developed by Creonic, was found to satisfy all requirements, both in terms of complexity and licensing [9]. As the IP-Core requires a soft decision input, but it was unknown at the design time if the Physical Layer used for the project is able to produce soft decisions [1], it was chosen to implement a small VHDL module that converts the binary 0 and 1s sent to the decoder to a Log Likelihood Ratio used as a soft decision. This does allow for later removal of this module, if soft decisions are provided by the physical layer.

7. IMPLEMENTATION

The proposed design architecture is implemented in VHDL 2008 and synthesised for the PYNQ Z2 SoC board. The archived frequency is 100 MHz, with a WNS of 0.2 ns [20]. The system's hardware requirements are shown in Figure 6 and Figure 7 for the encoder and decoder, respectively. They are split into different elements to show estimates if one chooses to interchange a singular element, or have more than one element if interleaving or multiple virtual channels are used. It should be noted that the shown hardware requirements do not include the buffer structures required to read out the data streams from the FPGA to the PS. One can assume about 750 LUT, 600 FF, and 2 BRAM for these buffers; as they are outside of the systems scope, they are not included in the hardware requirements [13].

[1] as this layer is still in early development

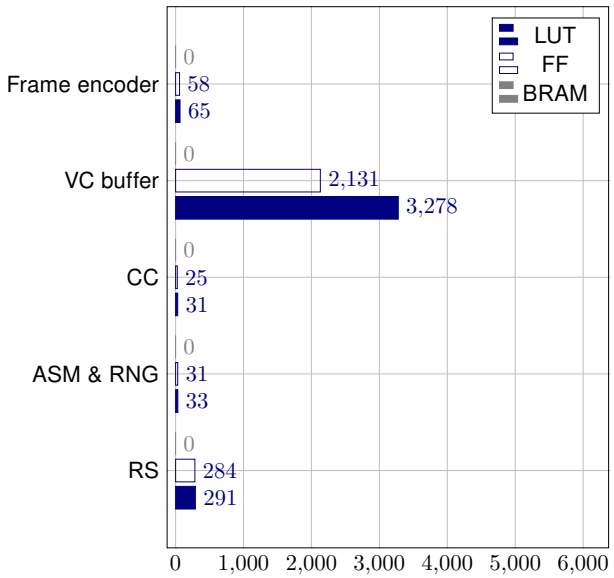

FIG 6. Hardware requirements of the encoder for the data link and synchronization and channel coding

As the encoder is synthesised on $100\,\mathrm{MHz}$ and outputs one bit per clock cycle to the physical layer a theoretical data throughput can be calculated:

$$R_{phy} = 1\,\mathrm{bit} \cdot 100\,\mathrm{MHz} = 100\,\mathrm{Mbit/s} \tag{1}$$

Here, R_{phy} is the rate outputted to the physical layer. This does include all parity check symbols, synchronization patterns, and framing included by the data-link layer. Using the coding rates of the R/S, CC, and ASM, the rate entering the synchronization and channel coding layer (R_{sync}) can be computed:

$$\begin{aligned} R_{sync} =& R_{phy} \cdot r_{CC} \cdot r_{ASM} \cdot r_{RS} = \\ & R_{phy} \cdot \frac{1}{2} \cdot \frac{255}{259} \cdot \frac{223}{255} = 43.0501\,\mathrm{Mbit/s} \end{aligned} \tag{2}$$

From that, the theoretical input data rate (R_{data}) into the data link layer can be computed. As currently only the primary header is implemented, 6 bytes of additional overhead per transfer frame are introduced:

$$\begin{aligned} R_{data} =& R_{sync} \cdot r_{data} = \\ & R_{sync} \cdot \frac{2040}{2046} = 42.9239\,\mathrm{Mbit/s} \end{aligned} \tag{3}$$

This data rate is only the input data rate into the data link layer; that means that still overhead due to space packets and higher layer protocols, like packet utilization standard (PUS) or the space packet protocol [21] [22]

Since the decoder is also synthesized on $100\,\mathrm{MHz}$ and also is able to read an input bit every clock cycle from the physical layer, the theoretical data throughput calculated above does also apply for the decoder chain.

CC BY 4.0

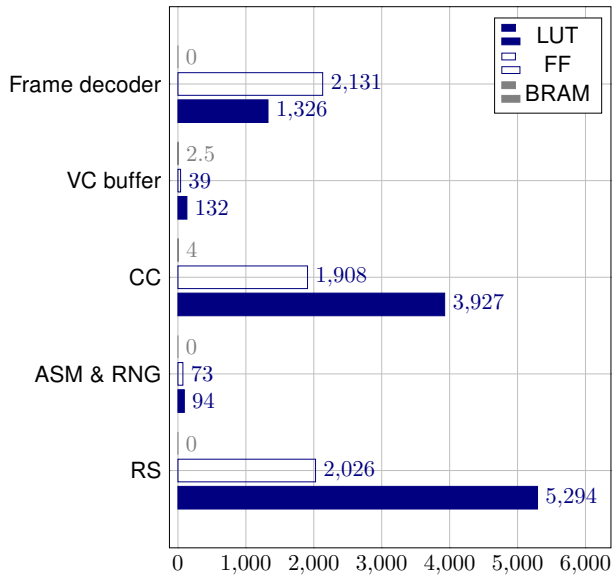

FIG 7. Hardware requirements of the decoder for the data link and synchronization and channel coding

8. EVALUATION

8.1. Testsetup

To test the developed system, two methodologies are employed: the first is unit and system level simulation; the second is testing on real hardware.
For simulation, each VHDL module is tested separately, ensuring that the outputs are as expected for the corresponding inputs. Additionally, every encoder module is tested with its decoder, for example, the R/S decoder is tested against the R/S encoder; this ensures that both elements, even if not developed by the same person, are compatible with one another. As a final step, the whole system is tested in simulation to ensure that the layers are compatible with each other and produce the same data as input. Simulation is both performed as a pure behaviour and as a post implementation simulation.
To evaluate the system on real hardware, both the encoder (satellite) and the decoder (ground station) are synthesized into a single FPGA. To ensure that data can be changed to model bit flips, the output data stream of the encoder is not directly fed into the input data stream of the decoder, but it is first fed into the PS, where modifications can be performed to model the losses of a HF transmission.
The FPGA used in this setup is the ZYNQ 7020 SoC, one of the A9 Cores is used to stimulate the system with input data, insert bit flips, and compare the input data with the resulting output data. The in build debugger and UART connection can be used to read the results for further analysis. This setup allows the testing of different bit error rates in real time on the real hardware. Every tested BER is calculated by sending 100 Mbit of user data through the encoder and decoder chain, and flipping bits with the given BER. Since no decoding element is currently implemented using soft decisions, a binary symmetric channel is used [23].
In addition, the theoretical throughput, calculated in Equation 3, is validated using a modified synthesis, where the output of the encoder is not fed back into the PS, but is mapped to external pins with a constant ready signal. This is to ensure that the system is able to run at its maximum speed without validation overhead in software. Only the encoder is tested, as the decoder has the same data rates, as elaborated above.

8.2. Results

Figure 8 shows the coding gain for a simulated physical layer upset rate ranging from 0.01 to 0.08. This range corresponds to the CCSDS 131's rationale for some of its codes [17]. In addition to the coded case showing the performance of the decoder, the uncoded case, where the output error rate equals the input error rate, is shown. The plot shows that the decoder has a better performance than the uncoded case, for bit error rates smaler than 0.07. For higher error rates, the performance is reduced, which is expected, as both the Reed Solomon and the convolutional code may worsen the BER when it is high to begin with [17].

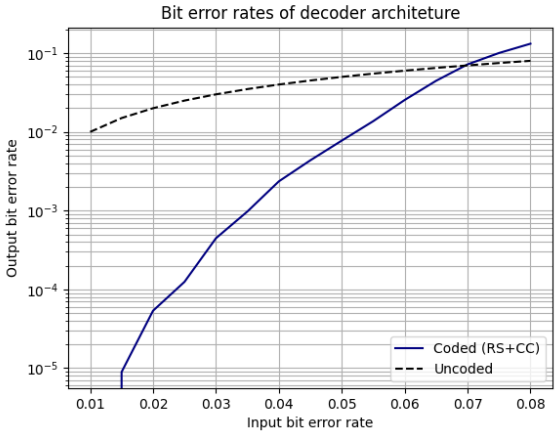

FIG 8. Bit Error rate of decoder and the uncoded case

The test data showed that not all frames are perfectly synchronised. This is due to an incorrect detection of the ASMs, which leads to the RNG sequence not resetting, resulting in high BERs for these frames. To analyse the ASM decoder's performance in high BER environments, the frame synchronization rate is shown in Figure 9, frame synchronization rate meaning the ratio of correctly synchronized frames vs the total number of transmitted frames. It can be seen that for a BER smaller than 0.03, the system did not lose synchronization. When increasing the BER, the frame synchronization rate decreases, reaching about 86% for a BER of 0.08. This shows that some improvements may be done to the ASM matching algorithm to increase this rate even further [17] [24], which may, however, lead to false positive detections.

CC BY 4.0

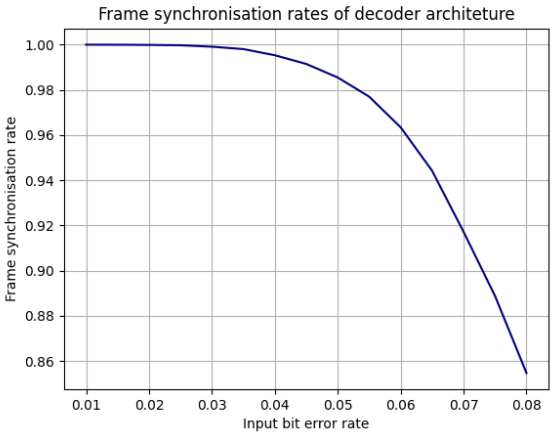

FIG 9. Synchronisation rate for the ASM decoder

Finally, an analysis of the throughput measurement shows that the minimum, maximum and average are close to one another. For this reason, they were not plotted in a histogram, but rather displayed in Table 2. It is apparent that the theoretically expected value and the average value differ by about 1%. This may be caused by the software's inability to saturate the system to 100%, since a bit of software overhead is required, even in this small setup. This overhead leads to the measured data rate not reaching the theoretical maximum value, but getting really close to it.

TAB 2. Messured throughput of the system

Average	$42.370\,71\,\mathrm{Mbit/s}$
Maximum	$42.370\,81\,\mathrm{Mbit/s}$
Minium	$42.370\,70\,\mathrm{Mbit/s}$
Std	$7.855\,72 \times 10^{-6}\,\mathrm{Mbit/s}$

9. CONCLUSION

In summary, a system for the CCSDS data link was proposed, implemented, and tested, which can reach up to $42.370\,81\,\mathrm{Mbit/s}$ user data throughput. The system is designed to be as modular as possible and allows for a positive coding gain for BERs less than 7%. The system was synthesised for the ZYNQ 7020 FPGA chip, requiring about 3700 LUT and 11800 LUT for the encoder and decoder, respectively. Testing was performed on the PYNQ Z2 SoC development board. Testing showed good synchronization performance of about 86% of frames at 8% BER.
In the future, the system needs to be tested in more complex scenarios, and maybe even tested using a real physical layer, outside the simulation model shown. In addition, both different coding options, like LDPC or Turbo codes, or the usage of interleaving or punctured convolutional codes, may be considered, to increase coding gain and flexibility of the system. Talking about the CCSDS 132.0-B-3, the FEC Field with its CRC and OCF Field may be added in future. Additionally, more of the defined services or the secondary header may be added if this is required.

Apart from adding more features to the system, existing parts have to be improved. One of the blocks requiring attention is the used Viterbi decoder, which may be configured to use soft decisions [9], which wouldn't just improve the coding gain of the convolutional code, but also improve the frame synchronization rate of the ASM decoder, since this may also profit from soft decisions [24]. This improvement would come with an increase in complexity of the system, as more resources are required to store and model the soft channels. These additional hardware requirements may be offset by a more efficient implementation of the data-link decoder and the Reed–Solomon decoder.
Adding to improvements in the TM Data Link layer, the TM Data Link layer must be integrated with a TC Data Link layer built by another group at SeeSat. To archive full duplex communication with the satellite [25] [26].
Finally, the system must be validated against existing implementations of the CCSDS Data-Link layer. This ensures the system is compatible and can be used in coexistence with other parts of the CCSDS stack.

Code location

The VHDL design described in this project was published under the GPL 2.0 License and can be found at the following location:
https://github.com/rtbnb/CCSDS-TM

Contact address:

matthias_fuchs28@gmx.de

References

[1] Council of the Consultative Committee for Space Data Systems. Tm space data link protocol. Standard, Washington, DC, USA, 2021.

[2] Council of the Consultative Committee for Space Data Systems. Tm synchronization and channel coding. Standard, Washington, DC, USA, 2026.

[3] S. Schnöge, H. Ketelhodt, L. Siffrin, F. Kunz, J. Schlemar, E. Fenske, F. Leitner-Fischer, S. Wertheimer, and S. Wanninger. Development of a tm/tc front-end for satellite communication with real-time data processing as an open source solution. *Deutscher Luft- und Raumfahrtkongress*, 2025.

[4] Council of the Consultative Committee for Space Data Systems. Ccsds space link protocols over etsi dvb-s2 standard. Standard, Washington, DC, USA, 2022.

[5] European Telecommunications Standards Institute. Digital video broadcasting (dvb); second generation. Standard, FRANCE, 2014.

CC BY 4.0

[6] Michal Pogorzelec. Embeddedsdlp. https://github.com/OpenSpaceCode/EmbeddedSDLP, 2026.

[7] AcubeSAT. ccsds-space-data-link-protocols. https://github.com/AcubeSAT/ccsds-space-data-link-protocols, 2024.

[8] André Løfaldli, Jeppe Johansen, and Jon Adair. gr-ccsds. https://github.com/lofaldli/gr-ccsds, 2020.

[9] Cronic. Viterbi decoder user guide. Product guide, 2012.

[10] B P Swathi, Amulya K, Divya Shree K V, Suma M N, and Keerthi K. Design and implementation of ldpc decoder using vhdl for space application. In *2024 2nd International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, pages 1–8, 2024. DOI: 10.1109/ICNEWS60873.2024.10730918.

[11] Council of the Consultative Committee for Space Data Systems. Ccsds space link protocols over etsi dvb-s2 standard. Standard, Washington, DC, USA, 2022.

[12] Advanced RISC Machines (ARM). Amba® 4 axi4-stream protocol. Standard, 2010.

[13] Advanced Micro Devices (AMD). Axi4-stream fifo logicore ip product guide. Product guide, 2026.

[14] Yingquan Wu. New scalable decoder architectures for reed–solomon codes. *IEEE Transactions on Communications*, 63:2741–2761, 08 2015. DOI: 10.1109/TCOMM.2015.2445759.

[15] E. Berlekamp. Nonbinary bch decoding (abstr.). *IEEE Transactions on Information Theory*, 14(2):242–242, 1968. DOI: 10.1109/TIT.1968.1054109.

[16] J. Massey. Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969. DOI: 10.1109/TIT.1969.1054260.

[17] Council of the Consultative Committee for Space Data Systems. Tm synchronization and channel coding summary of concept and rationale. Standard, Washington, DC, USA, 2020.

[18] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[19] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. DOI: 10.1109/TIT.1967.1054010.

[20] Pynq-z2 reference manual v1.0. Product guide, 2018.

[21] European Cooperation for Space Standardization. Space engineering telemetry and telecommand packet utilization. Standard, 2016.

[22] Council of the Consultative Committee for Space Data Systems. Space packet protocol. Standard, Washington, DC, USA, 2020.

[23] Thomas M. Cover and Joy A. Thomas. *Channel Capacity*, chapter 7, pages 183–241. John Wiley Sons, Ltd. ISBN: 9780471748823. 2005. DOI: https://doi.org/10.1002/047174882X.ch7.

[24] J. Massey. Optimum frame synchronization. *IEEE Transactions on Communications*, 20(2):115–119, 1972. DOI: 10.1109/TCOM.1972.1091127.

[25] Council of the Consultative Committee for Space Data Systems. Tc synchronization and channel coding. Standard, Washington, DC, USA, 2026.

[26] Council of the Consultative Committee for Space Data Systems. Tc space data link protocol. Standard, Washington, DC, USA, 2021.

CC BY 4.0