FPGA Architecture of Reliable High-Throughput Error Correction for UAV FSO Communication System

Irzal Zaini, Ida Bagus Krisna Yoga Utama, Fadhila Ahmad, Muhammad Fairuz Mummtaz, Yeong Min Jang
Department of Electronics Engineering
Kookmin University
Seoul, South Korea
irzal.zaini00@gmail.com; ibkyutama@ieee.org; ahmad.fadhilaid@gmail.com; muhammad.fairuz91@ui.ac.id; yjang@kookmin.ac.kr

***Abstract*—Unmanned Aerial Vehicle (UAV) Free-Space Optical (FSO) is the next generation technology in wireless communication systems. The high bandwidth potential of FSO communication is not without flaw, the optical wave is easily distorted by channel conditions and must have accurate tracking. To improve reliability, error correction must be applied. In this paper, FPGA-based error correction architecture is proposed to comply with the UAV FSO system. The design is based on Reed-Solomon (RS) code and Line Product Code (LPC). The proposed system utilizes the parallel processing and pipelining of Euclidean Processor in decoding unit, which increases data throughput significantly. Our design also offers low resource usage on the FPGA board, effectively reducing power consumption. Performance test shows BER of 10^{-6}, data rate at 8.921 Gbps, and power consumption at 1.6 W.**

***Index Terms*—Error Correction, FPGA, RS, LPC, Euclidean Processor, parallel processing, UAV, FSO**

I. Introduction

UAV FSO communication system is a promising wireless technology that offers the users greater data bandwidth, higher security, and unlicensed frequency spectrum. Compared to the radio frequency (RF) systems, the FSO has more spectrum efficiency and lesser interference to neighboring channel frequency. Currently, the RF spectrum is highly congested due to the increasing number of wireless devices in network [1] [2]. The internet infrastructure must follow the ever-growing demands of user connections which results in higher requirement of data bandwidth in core network. The fastest current technology in RF-based system is the 6G radio that utilizes the mmWave frequency. However, in the near future to fulfill this requirement will be very difficult due to network congestion and the already limited RF spectrum [3] [4].

The UAV FSO communication system can be a solution we need to provide alternative of RF-based network. The FSO is based on optical wave medium for transmitting data on the air. Main advantage of this media is the longer range coverage due to lightwave property and its capability to bring more data on transmitting medium. One great example of current FSO company is the SpaceX with its Starlink product. The Starlink is a constellation of FSO network in the Low Earth Orbit (LEO) satellite communication. It utilizes the FSO as the main backhaul network link in LEO, in order to cover all of the earth surface user connections. The SpaceX focuses the FSO technology in the satellite-to-satellite and also the space-to-ground communication network [5].

One of the downsides of FSO communication is the optical wave susceptibility to distortion caused by the channel condition in the air. The atmospheric turbulence (AT) is a major issue in FSO that causes unpredictable changes in channel characteristics due to the nature of environmental effect such as wind gusts, temperature, humidity, sun rays, smoke debris, and others. The AT is a random event that can be describe in Probabilistic Distribution Function (PDF), therefore the designer can at least do some mitigation technique to reduce the data corruption while in transmission by predicting it from the PDF analysis [6] [7].

The writer proposes an architecture of error correction (EC) technique to be implemented on the UAV FSO communication system. The EC is designed and implemented using Field-Programmable Gate Array (FPGA) platform, so we can exploit the low level hardware programming and the low latency processing due the nature of FPGAs. The EC design is based on the Reed-Solomon (RS) code and Line Product Code (LPC) that are combined together to enhance the reliability of FSO system. RS code is known to have capability to correct burst error that occur frequently in the case of FSO communication.

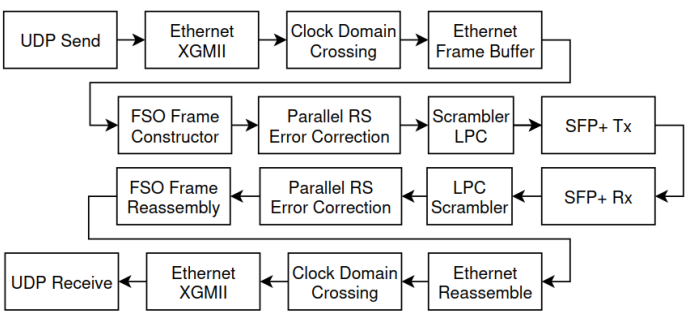

Fig. 1. FPGA FSO error correction main architecture.

LPC is additional layer of the design to improve the bit error using the basis of bit disparity. The main system architecture is shown in Figure 1.

The Error Correction main architecture is divided into several parts: Ethernet Interface, FSO Frame Constructor, Parallel RS Channel Coding, LPC, and SFP+ Transceiver. The FPGA is compiled in Verilog language with modular hierarchy, increasing the system programmability and scalability in the future. The challenges to design our architecture is the clock domain management between each module to be able to work together flawlessly and the buffer manipulation to control the data and prevent overwrites or data mistiming. Our proposed design is the parallel processing Euclidean Calculator Unit to evaluate the Greatest Common Divisor (GCD) value. The proposed design introduces ping-pong memory buffer technique to the GCD calculator to increase throughput. The GCD calculator is constructed on a hardware wired parallel logics that increases the data throughput per clock cycle.

II. Methodology

A. System Overview

The FPGA FSO error correction implementation is conducted on AMD Xilinx Kintex Ultrascale XCKU060 development board. In the testbed experiment we utilize the GTH Transceiver that can support up to 16 Gbps data rate on the physical SFP+ interface port. The FPGA acts as the EC and FSO modem device that receives data from Ethernet based interface user terminals. The experiment is done using pair of FPGA boards to emulate communication scenario between 2 FSO terminals. The SFP+ port of the FPGA supports 10 Gbps data rate, so the Ethernet interface on the user terminals must comply the speed by installing XGMII supporting Ethernet card or simply PCIe 10 Gbps NIC. Test environment on the terminals is controlled and monitored by Python code which mainly send and receive traffic of UDP Packets. The SFP+ modules used are industry standard 10 Gbps speed module, 1270/1330 nm bidirectional pair wavelength, single mode fiber, and supports long range up to 10 km [8] [9]. The experimental testbed setup is shown in Figure 2.

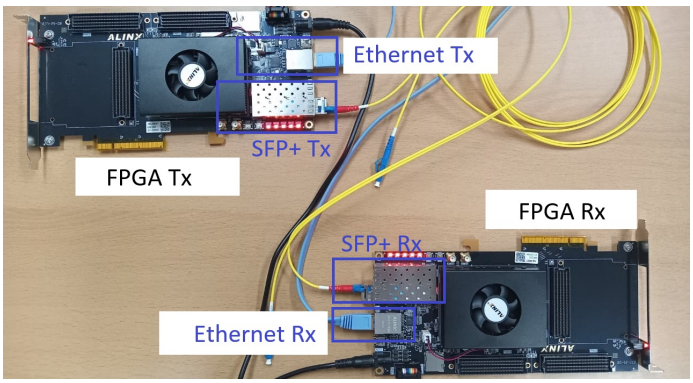

Fig. 2. FPGA FSO error correction testbed experimental setup.

B. Main Architecture

The design consists of FPGA Tx and Rx parts. The FPGA Tx main function is to encode the data before transmission to the FSO channel, in this experimental setup is fiber optic connection link. The encoding process include Ethernet data parsing, clock domain crossing, adaptive memory management, data checksum, parallel pipelined RS code error correction, interleaving, bit scrambling, LPC, and line coding. The user can use Ethernet connection (preferably SFP+ NIC) from their terminal to the FPGA system. The Ethernet data parsing process manages the data from terminal Ethernet packets to be converted into FSO frames. In this part, there is clock domain crossing to ensure clock synchronization and bit-width adjustments to match the data size. The checksum adds additional protection to data with error detection. We implement CRC based checksum for this design. The channel coding is based on high density parallel RS code error correction. The encoded data will produce parity symbols that is used as decoding component at the Rx side. The interleaver and scrambler preprocess the encoded data before LPC and line coding. The line coding process is based on Line Product Code (LPC), to ensure the clock synchronization at Rx side and bit disparity.

C. Reed-Solomon Error Correction Code

Reed-Solomon (RS) code is the main error correcting code that is used in the FPGA system. This allows the UAV FSO system to locate and correct any errors occurring at the receiver side. The RS code processes the original message data from terminal unit and results in adding protective element to the data array known as the parity symbols. The parity symbols calculation is derived mainly from the Galois Field (GF) numeration and polynomial arithmetic [10] [11]. The RS block diagram is shown in Figure 2.

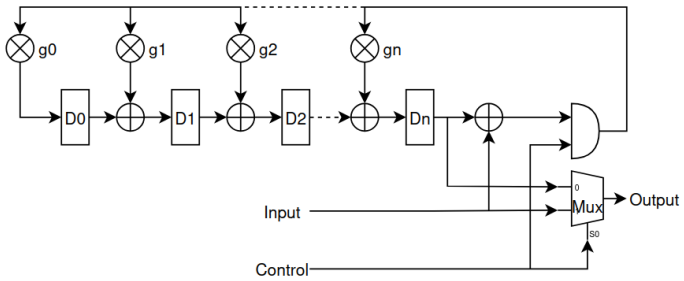

Fig. 3. FPGA FSO error correction Reed-Solomon encoder block diagram.

The parity symbols are essential in the decoding process of RS code. It is used as decoding element where the original message and the parity is combined in the Syndrome Calculator Unit to find out the Syndrome. The Syndrome is an evaluated polynomial function, usually in a polynomial of 16 term, that shows the characteristic of error of the original signal. However, if the value of Syndrome is zero then it means the original data is not corrupted at all.

The Euclidean Processing Unit is GCD calculator taking Syndrome as input and produces Delta and Omega functions

as output. Delta function is description of error locator value which later is processed in the Chien Search module. Omega function is the indicator of error magnitude or error value to revert the error on the received message. Omega function is processed at the Forney module in the decoder. The Euclidean Processor performs the GCD calculation which involve several multiplication and division steps in order to reduce the term of the polynomial. In case of our system, the GCD calculation requires 16 steps due to the 16 term polynomial of the Syndrome.

Chien Search and Forney module is parallel module that search the error location from received message index and also precisely provide the correct magnitude to revert the error symbol. The Chien-Forney algorithm utilize the multiplication and division module in the FPGA fabric logic.

D. Cyclic Redundancy Check

Cyclic Redundancy Check (CRC) is widely used checksum method to detect error on the received data packets. The CRC is important module in our system to make sure the decoder can find out error by sending flag signal to the user. The implementation of CRC on the FPGA FSO Error Correction system uses the short CRC-16 for the Header data and CRC-32 for the long stream of Payload data. To comply with the requirement of multi-gigabit data, the implementation of CRC is in highly parallel arrangement and using the high-speed transceiver clock.

E. Line Product Code

Line Product Code (LPC) is additional layer of error correction method to increase system reliability. The LPC is based on the Hamming code with combination of data matrix instead of just bit operations. LPC adds protective bits and maps the data bits into new matrix that later be sent to the communication channel. The effect of LPC is increased bit disparity which results in better coding gain and DC balance on the physical layer. Commonly, the LPC is used in conjunction with Scrambler module. The scrambler module is implemented using the Pseudo-Random Binary Sequence (PRBS). This will add the disparity bits by scrambling the common data which are usually ASCII based text that have repeating characters.

F. Clock Domain Crossing

The clock domain crossing is important module in the FPGA fabric to handle the data transfer from each modules. The variants of different clock frequency in the system requires this feature to ensure program stability. The implementation includes asynchronous asymmetric First-In First-Out (FIFO) buffer and Finite State Machine (FSM). The async FIFO captures the data asynchronously and sort out the output following the requested reading clock. The FSM is the trigger control for the clock domain crossing. It ensures the correct state sequence and reduces trigger confusion for the FPGA fabric logic.

G. Adaptive Buffer Memory

The adaptive buffer memory is used in the system to support the pipeline architecture that we implemented. Since the data is always continuous, the adaptive buffer will queue the data and process the ready packet to the decoder. On top of that, we implemented double layer FIFO for the adaptive buffer to increase the efficiency. The double layer provides continue runtime which activate the FIFO alternately in the processing. This method is also known as the ping-pong FIFO.

H. Proposed Decoder Design

At the Rx part, the process is inversion of encoding algorithm however more complexity in the channel coding function. The decoding process requires four-step unit process consists of Syndrome calculation unit, Euclidean processing unit, Chien search, and Forney algorithm. The syndrome calculation extract all the syndrome values out of the received data. The syndrome values are required for the Euclidean processing. Our design proposed a new approach for the Euclidean Processing by implementing highly parallel component architectures and modification on the concurrent process, so that the calculation is faster and has more throughput. The Euclidean processor unit is GCD algorithm that can be implemented to RS code. The proposed decoder architecture is shown in Algorithm 1.

Algorithm 1: Proposed parallel processing Euclidean Unit Decoder.

Initiate:
$A \leftarrow constant$;
$B \leftarrow syndrome$;
$C \leftarrow 0$;
$D \leftarrow 1$;
if $B > t$ **then**
  GF Division:
  $Q \leftarrow A/B$;
  Update Registers:
  $T_1 \leftarrow A - (Q * B)$;
  $T_2 \leftarrow C - (Q * D)$;
  Register Transfer:
  $A \leftarrow B$;
  $B \leftarrow T_1$;
  $C \leftarrow D$;
  $D \leftarrow T_2$;
else
  Output Results:
  $delta \leftarrow B$;
  $omega \leftarrow D$;
end
return 0;

III. Test Results

The performance of the UAV FSO FPGA error correction system is evaluated by several metrics. The data rate, Bit Error Rate (BER), power consumption, and resource usage.

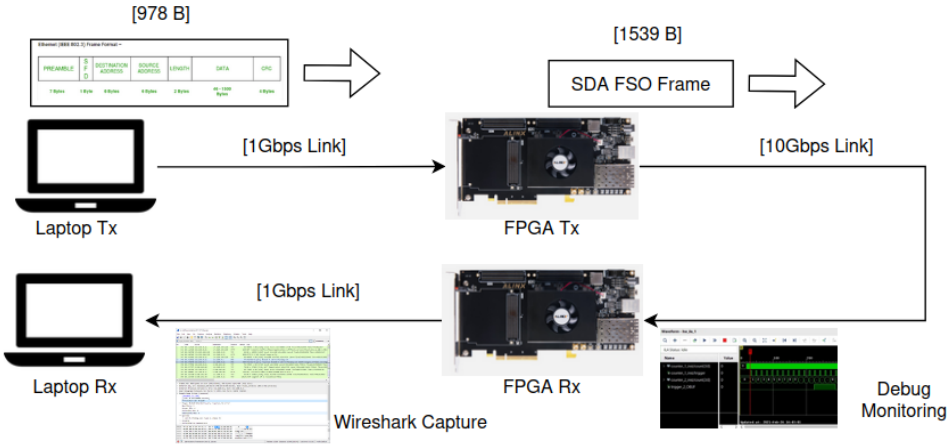

Fig. 4. FPGA FSO error correction testbed scenario.

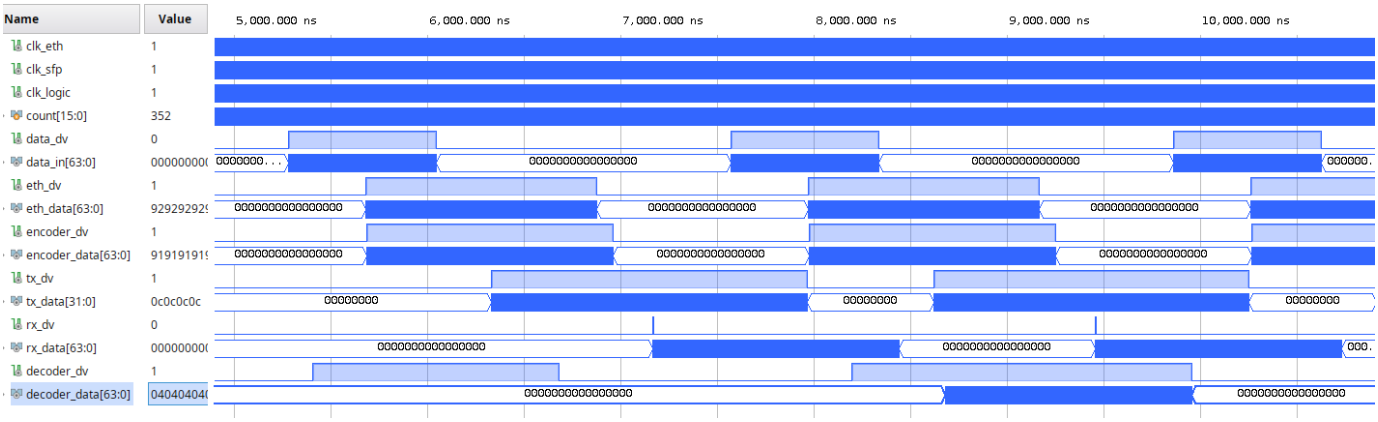

Fig. 5. FPGA FSO waveform signal analysis.

The structure of the FSO Frame and Ethernet packets is also monitored in the testing. The monitoring is conducted by utilizing the debug monitor from the AMD Vivado based on JTAG protocol. Each monitoring capture indicates a window of captured data on 1 time cycle.

A. *Testbed Scenario*

The channel coding system performances are verified using 3 monitoring tools: Python, Wireshark, VIVADO Integrated Logic Analyzer (ILA). The VIVADO ILA works similarly as logic analyzer testbench device. It is used to observe the logic signals on the SFP and/or the Ethernet port interfaces. Wireshark captures the Ethernet base traffic in the form of TCP/IP data packets, in the testing case we use UDP packets. The timing and counter of the packets can be tracked using Wireshark. It tracks down the packets based on the related MAC addressed that can be filtered by the program. At the user end, Python is used to generate and receive UDP packets. The Python code will calculate BER and data rate using counter of received packets. The testbed experiment scenario shown in Figure 4.

The testbed scenario consists of pair of transmitter part and receiver part. Input to the transmitter is an Ethernet packet driven by user's laptop or PC terminal connected to the FPGA error correction system board. Preferably, the user terminal should support 10 Gbps speed by installing PCIe SFP+ NIC to the system. The user should install the specified UDP packet generator program in their PC and make the IP static to be able to connect to the FPGA board. To monitor the direct signal analysis on the FPGA board, the user terminal should connect to FPGA board using JTAG interface to run the debug monitoring. The debug monitor is a real-time analysis tool proprietary to the AMD Xilinx.

B. *Data flow Waveform Analysis*

The data flow can be divided into 4 major parts: Ethernet Frame, FSO Frame, RS Encoding, and LPC Coding. The data from the user terminal is defined by Ethernet Frame which contains the UDP testing packet from the Python. The XGMII protocol is used as Ethernet handler for the FPGA fabric logic and uses 64-bit width data stream. The Ethernet Frame must be converted into FSO Frame before encoding process. The clock domain crossing and adaptive buffer memory modules are used together to generate the FSO Frame from the Ethernet interface.

The Ethernet Frame comply the standard of TCP/IP protocol stack which starts with Preamble, followed by MAC header, IP, UDP, and the Payload itself. The FPGA system will strip out the Ethernet Preamble when first detect the data egress, and capture all the data starting the MAC header. In our design, the MAC header is treated differently with the rest of the packets. It is considered as essential header data, so the RS coding is separated and use different shorter RS(16,32) setup. This means the data path for the MAC header is conditioned to be higher parity density compared to the payload for stronger error correcting capability. The captured signal waveform is shown in Figure 5.

The process of encoding up until decoding requires the FPGA latency at 3.9 us. The Ethernet UDP Packets are sent at controlled interval from the Python UDP program. The FSO Framer module converts the data into channel coding. The minimum latency is required to ensure FIFO clearance before next cycle occur.

C. Resource Utilization

The Kintex Ultrascale XCKU060 development board resource utilization shows the optimized usage of logic gate transistors and other component modules. The summary is generated using the utilization reporting feature from the Vivado implementation results. The parallel pipelining euclidean processing error correcting system shows the summary of LUT usage of 30.03 %. The Flip-flop (FF) and Block Random Access Memory (BRAM) shows usage of 2.66 and 2.04 % respectively. The Mixed-mode Clock Manager (MMCM) is used as the clock management module for the entire FPGA fabric, it shows usage of 16.67 % from the Kintex Ultrascale. The resource utilization report data is shown in Table I.

TABLE I
FPGA RESOURCE UTILIZATION

Resource	Used	Available	Utilization %
LUT	99609	331680	30.03
LUTRAM	595	146880	0.41
FF	17642	663360	2.66
BRAM	22	1080	2.04
IO	3	520	0.58
MMCM	2	12	16.67

D. Power Consumption

The Kintex Ultrascale XCKU060 development board power consumption shows a total chip usage of 1.377 W. The device static power consumption shows 0.635 W with dynamic power consumption takes 0.743 W. The system Clocks consume 0.214 W is the 2nd biggest draw of power from overall system. The MMCM module is the largest module that draws the power from the FPGA. The proposed design can be concluded to be low power consumption and suitable for portable deployment. The 1.377 W power draw means roughly, using a 10,000 mAh battery can provide power up to 7 hours runtime. This is critical consideration especially the experiment scenario on flying UAV drone or UAV with FSO communication system. The FPGA power consumption is shown in Figure 6.

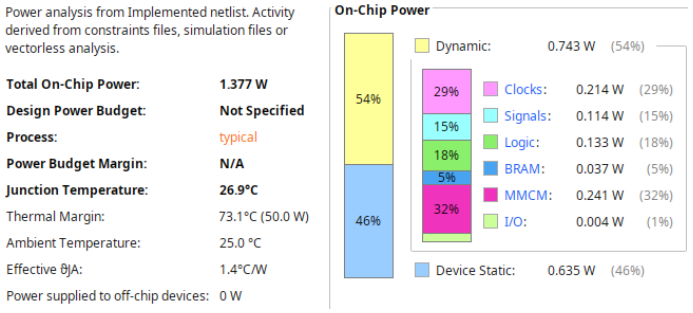

Fig. 6. FPGA FSO power consumption analysis from implementation report.

IV. CONCLUSIONS

FPGA implementation of UAV FSO error correction system can results in a robust and power efficient prototyping platform for research and development. FPGA is also considered cost effective solution for the area of high-speed communication prototyping compared to the CPU based platform or ASIC based device. Our implementation can show multi-gigabit throughput thanks to the FPGA innate ability to process data in concurrent and parallel pipelining program and hardware architecture. The FPGA also has an advantage in the reprogrammability, so any changes needed during the research can be applied easily. Our proposed decoder introduce a solution to complex hardware network of the Euclidean GCD Processor by implementing parallel processing structure that combine concurrent programming paradigm and smart ping-pong buffer technique. The FPGA resource utilization shows that low transistor usage in the FPGA fabric, so effectively decreasing the overall power consumption of the error correction system.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2022R1A2C1007884).

REFERENCES

[1] K. Matsuda, M. Binkai, S. Koshikawa, T. Yoshida, H. Sano, and Y. Konishi, "Field demonstration of real-time 14 tb/s 220 m fso transmission with class 1 eye-safe 9-aperture transmitter," Ieee.org, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9489830

[2] M. A. Fernandes, G. M. Fernandes, B. T. Brandão, M. M. Freitas, N. Kaai, A. Tomeeva, B. van Der Wielen, J. Reid, D. Raiteri, P. P. Monteiro, and F. P. Guiomar, "4 tbps+ fso field trial over 1.8 km with turbulence mitigation and fec optimization," *Journal of Lightwave Technology*, vol. 42, pp. 4060–4067, 06 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10414049

[3] R. F. Pamungkas, I. B. K. Y. Utama, K. Hindriyandhito, and Y. M. Jang, "A hybrid approach of convlstmbnn-dt and gpt-4 for real-time anomaly detection decision support in edge–cloud environments," *ICT Express*, vol. 10, pp. 1026–1033, 10 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959524000857

[4] M. M. Rahman, M. S. Nazim, M. I. Joha, and Y. M. Jang, "Real-time implementation of ofdm modulation for an occ system: Unet-based equalizer for signal denoising and ber optimization," *ICT Express*, vol. 11, pp. 728–733, 07 2025.

[5] K. D. Dang, H. D. Le, C. T. Nguyen, and A. T. Pham, "Resource allocation for hybrid fso/rf satellite-assisted multiple backhauled uavs over starlink networks," *IEICE Communications Express*, vol. 13, pp. 52–55, 01 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10400561

[6] "Space development agency optical intersatellite link (oisl) standard," Space Development Agency, 2022. [Online]. Available: https://www.sda.mil/wp-content/uploads/2023/12/SDA-OISL-Standard-v2.1.2.pdf

[7] C. K. P. Clarke, "Reed-solomon error correction," 2002. [Online]. Available: https://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP031.pdf

[8] C. I. Yeo, S. Park, Y. Heo, J. H. Ryu, and H. S. Kang, "Evaluation an analysis of 2.3 gbps full-duplex mobile optical wireless communication system for high-speed link between air and ground station," *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1643–1645, 10 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9952908

[9] T. Van Le and N. Van Dinh, "Implementation of a binary reed-solomon codec using the berlekamp-massey algorithm on fpga," *2024 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, pp. 85–89, 12 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10908798

[10] I. Zaini, I. Bagus, and Y. M. Jang, "Fpga implementation of channel codec for optical intersatellite link communication system," pp. 0200–0203, 02 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10920878

[11] Y. Heo, C. I. Yeo, S. Park, and H. S. Kang, "Demonstration and analysis of outdoor owc system using fpga-based 2.5gigabit ethernet including fec," *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 2338–2340, 10 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9952674