FPGA-based Tx and Rx for Communication using LEDs / LDs

A Project Report submitted by
Kalbhavi Vadhi Raj
Pranav Chakravarthy

in partial fulfillment of the requirements for the award of the degree of
Bachelor of Technology

Indian Institute of Technology Jodhpur
Electrical Engineering Engineering
November 2024

Declaration

I hereby declare that the work presented in this Project Report titled FPGA-based Tx and Rx for Communication using LEDs/LDs submitted to the Indian Institute of Technology Jodhpur in partial fulfillment of the requirements for the award of the degree of B.Tech., is a bonafide record of the research work carried out under the supervision of Dr. Nitin Bhatia. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature
Kalbhavi Vadhi Raj (B21EE030)
Pranav Chakravarthy (B21EE050)

Certificate

This is to certify that the Project Report titled FPGA-based Tx and Rx for Communication using LEDs/LDs, submitted by Kalbhavi Vadhi Raj (B21EE030), Pranav Chakravarthy (B21EE050) to the Indian Institute of Technology Jodhpur for the award of the degree of B.Tech., is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature
Dr. Nitin Bhatia

ACKNOWLEDGEMENTS

We want to express our heartfelt gratitude and appreciation to all the individuals who have contributed to the successful completion of this project.

First and foremost, we would also like to thank our supervisor, Dr. Nitin Bhatia, for his guidance and support. His expertise and mentorship have been instrumental in shaping our project and pushing us toward excellence.

We want to thank our fellow group members for their hard work, dedication, and cooperation throughout the project. Each team member played a crucial role, bringing unique skills and perspectives, greatly enriching our work.

We would also like to thank Mr. Champalal Lalani for helping us in setting up the experiments in the lab and also the pointers he gave along the way.

We are truly grateful for the collaboration, guidance, support, and resources provided. Thank you all for making this project a fulfilling and rewarding experience.

ABSTRACT

This report presents the design and implementation of a Low Density Parity Check (LDPC) coded transceiver system for audio and text transmission using Xilinx PYNQ-Z2. The implemented transceiver system leverages a forward error correction approach to improve transmission reliability in both direct and optical communication channels. We investigate the performance of the LDPC decoder using Python simulations to demonstrate the error correction capability of the system, with a particular emphasis on handling bit flipping errors typical in optical communication channels. The transceiver includes key components such as an encoder, decoder, finite state machine (FSM), and pulse-width modulation (PWM) for audio signal processing. The system's functionality is validated through experimental results, where we tested both direct transmission and transmission through an optical channel using the PYNQ-Z2 FPGA board. Initial experiments with direct transmission of audio and text signals confirmed the system's capability for real-time signal processing. However, challenges were encountered when transmitting through an optical channel, including signal attenuation and distortion, which necessitated further optimization.

CONTENTS

1. INTRODUCTION

1.1 Forward Error Correction (FEC)

Forward error correction (FEC), often referred to as error correction, involves the deliberate addition of redundancy to a data sequence at the transmitter. This redundancy is structured to allow the receiver to detect and correct a limited number of bit errors introduced by a noisy channel without needing feedback from the receiver to the transmitter. FEC is especially beneficial in applications where high data rates are critical, such as audio and video transmission. Including feedback mechanisms in these cases would require extra bandwidth and processing time, reducing the effective data rate and disrupting the real-time nature of the communication.

The error-correction capability of a code is primarily determined by its Hamming distance (d_H), a measure of the difference between two code words. The Hamming distance d_H between two binary sequences is defined as the number of bit positions in which they differ. The greater the minimum Hamming distance d_H of a code, the greater the number of bit errors it can correct. For instance, a code with a minimum d_H of three can correct single-bit errors, while codes with higher d_H values can correct multiple-bit errors.

The core idea behind FEC is to map the original message bits into a higher-dimensional space with additional bits, ensuring both uniqueness and separability. This encoding process maximizes the distance d_H between any two valid code words. Consequently, even if noise in the channel causes some bits to flip, the receiver can still reliably decode the transmitted message by identifying and correcting these errors, thereby preserving data integrity. This approach is highly effective in optical communication and high-speed data networks, where transmission conditions introduce various types of noise, and minimizing retransmissions is critical for maintaining throughput.

Most FEC codes today fall into two main categories:

1. Block codes
2. Convolutional codes

1.1.1 Block Code

In block coding, a message block of k information symbols is uniquely mapped to n coded channel symbols, which make up a codeword. A value of particular importance is the code rate, r which is defined as:

$$r = k/n$$

In block coding, codewords are generated by multiplying the message block by a generator matrix, **G**. Let the message block be represented by $\mathbf{c} = [c_0, c_1, \ldots, c_{k-1}]$, where each m_i is a binary element in the message sequence of length **k**. The resulting codeword $\mathbf{m} = [c_0, c_1, \ldots, c_{n-1}]$ is then defined as:

$$\mathbf{m} = \mathbf{cG}$$

Where **G** is a kxn generator matrix that maps the k - dimensional message space onto the n - dimensional codeword space. The encoding process performs a linear transformation by projecting the message vector **c** into the higher-dimensional codeword space through matrix multiplication. This transformation guarantees that each unique message vector **c** corresponds to a unique codeword **m**, thereby facilitating error detection and correction during transmission. Since the mapping process is linear, the resulting codes are classified as linear block codes. Linear block codes are particularly advantageous in communication systems due to their structured redundancy, which allows systematic error correction without requiring retransmission, thereby improving transmission efficiency and reliability in noisy channels.

Furthermore, a linear block code is said to be systematic if the information symbols, **c**, appear explicitly in the codeword. In particular, if the information symbols appear at the beginning of the codeword, then the generator matrix takes the form

$$\mathbf{G} = [\, I_k \;\; P]$$

where P is a k x (n - k) matrix called a parity matrix, and I_k is the k x k identity matrix.

To facilitate error detection in a received codeword, we define the parity-check matrix H, which has dimensions n x (n-k). The matrix H is constructed such that

$$H = [P^T \; I_{(n-k)}]$$

where P is a matrix of appropriate dimensions, and $I_{(n-k)}$ is the (n-k) x (n-k) identity matrix. This structure ensures that H serves as the parity-check matrix of code **c**.
For any valid codeword **m**, the multiplication of **m** with the parity-check matrix H results in the all-zero vector. This property can be expressed as:

$$\mathbf{m}H^T = 0$$

Where 0 represents the all-zero vector. This relationship is fundamental to ensuring that codewords adhere to the constraints imposed by the parity-check matrix, allowing for efficient error detection.

Decoding of linear block codes commonly utilizes minimum Hamming distance decoding. The Hamming distance d_H, between two binary vectors **x** and **y**, represents the number of positions where **x** and **y** differ. Consider the case where the n -dimensional received vector y is corrupted by noise during transmission. The decoder estimates the transmitted codeword **c** by selecting the codeword **m** closest to **y** in terms of Hamming distance. As long as the number of errors is low enough that **y** remains closest to the transmitted codeword **m**, the decoded message **u** will match the original transmitted message with no errors.

This concept can be visualized by representing each possible received vector **y** as a point within an n-dimensional vector space. The codewords of C form a distinct subset of these points, and around each valid codeword lies a sphere within which no two spheres overlap. This separation is achieved if the radius of each sphere is one less than half of the minimum Hamming distance between any two codewords in the code. All points (i.e., received vectors) within a sphere are thus decoded as the codeword at its center. A reduced-dimension visualization of this concept is provided in Figure 1.1.

The error correction capability of a linear block code is directly tied to its minimum distance d_{min}, defined as the smallest Hamming distance between any two distinct codewords in the code. A code with minimum distance d_{min} can correct any pattern of t errors as long as

$$d_{min} \geq 2t + 1$$

If more than t errors occur, the received vector may be displaced outside the intended codeword's sphere and into another, resulting in a decoding error.

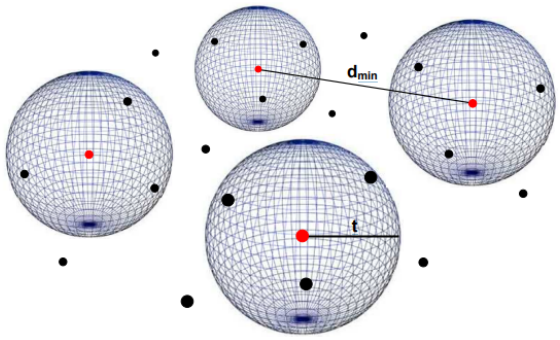


Figure 1.1: Reduced-dimensional visualization of linear block code decoding.

1.1.2 Convolutional codes

Convolutional codes represent a class of error-correction codes that operate fundamentally differently from block codes. While convolutional encoders also produce n output bits for every

k input bits, they employ a sliding window of data rather than processing data in fixed-size blocks. In typical applications, k and n are small integers, with $k < n$ to achieve redundancy. The encoded output at any time depends not only on the current input message of k symbols but also on the m previous input messages, each k symbols long. This dependence on past input symbols makes convolutional codes particularly useful in asynchronous transmission scenarios where information packets can vary in size.

Unlike block codes, which generally perform encoding through combinatorial operations that add negligible processing delay, convolutional codes require memory to store past symbols, introducing additional overhead in the encoding process. Consequently, this added complexity has made convolutional codes less favorable for designs that prioritize rapid encoding and low latency.

1.2 Low-Density Parity-Check Codes

Low-Density Parity-Check (LDPC) codes belong to a family of linear block codes distinguished by their sparse parity-check matrix, H, which contains a low proportion of 1's relative to 0's. Similar to other block codes, LDPC codes are uniquely characterized by their parity-check matrix H, which governs their encoding and decoding properties.

Due to the sparsity of H in LDPC codes, the associated generator matrix G is typically dense, leading to increased computational demands in encoding. Specifically, direct calculation of the G matrix has a time complexity that scales as $O(n^3)$, while the encoding operation itself scales as $O(n^2)$, where n represents the block length. Given that practical LDPC codes often involve block lengths on the order of 10^4 bits, specific structures are typically imposed on the H matrix to streamline code definition and reduce the computational complexity of the encoding process.

To visualize LDPC codes, a Tanner graph is often employed, as seen in Figure 1.2. In a Tanner graph, message bits are represented as "message nodes" and parity bits as "parity nodes." The connections between message and parity nodes are dictated by the parity-check matrix H; specifically, each parity node p_i is determined by the XOR of all the message bits to which it is connected. This graphical representation provides a clearer understanding of the decoding process by illustrating how information propagates between nodes in the decoding algorithm.

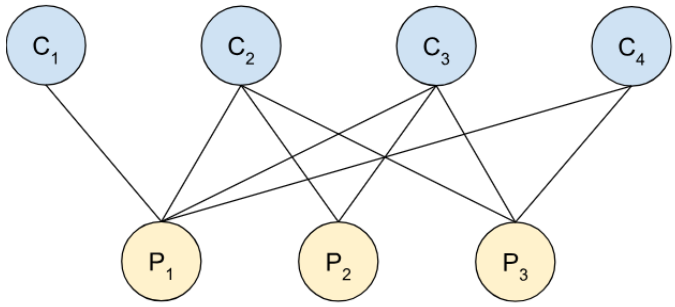


Figure 1.2: Tanner graph representation of an LDPC code.

1.3 Transceiver

A transceiver is an apparatus capable of both transmitting and receiving signals via a communications channel. An optical transceiver is responsible for the transmission and reception of light within optical fiber systems. Field-Programmable Gate Arrays (FPGAs) have become crucial elements in contemporary transceivers, markedly improving performance, adaptability, and efficiency in optical communication systems. Transceivers play a vital role in the conversion and processing of high-speed data within networks, while FPGAs offer robust programmable logic functionalities that facilitate real-time customization and optimization of transceiver operations. Their capacity for adaptability allows for the effective management of intricate signal processing tasks, as well as data serialization and deserialization. This capability is essential in addressing the growing requirements for elevated data rates, reduced latency, and resilient error correction across various communication contexts.

The design of the Transceiver is situated within the operational framework of the Data Link Layer in the Open Standards International communications model. The Data Link Layer plays a crucial role in facilitating data transfer between two directly connected nodes. It additionally identifies and rectifies errors that may arise within the physical layer. Errors typically arise in the physical layer as a result of noise present in the communication channel. For instance, in Free Space Optical (FSO) communication, the channel utilized is the atmosphere.

1.4 Finite State Machine

The design of a Finite State Machine (FSM) is essential for programming Field-Programmable Gate Arrays (FPGAs) in transceiver applications, as it offers a systematic and effective approach to handling the intricate sequences and control logic necessary for data transmission and reception. Finite State Machines (FSMs) are extensively utilized in digital design to establish accurate and predictable control flows, rendering them especially significant for transceivers that necessitate synchronized and dependable data processing to maintain signal integrity and facilitate high-speed operation.

Within the framework of transceivers, a finite state machine embedded in the FPGA governs a range of functions, including channel initialization, data encoding and decoding, error detection and correction, as well as protocol management. Utilizing a finite state machine allows designers

to establish clearly defined states that correspond to each phase of the transceiver's operation, thereby facilitating the management of the complex timing and sequencing necessary for efficient communication. This framework facilitates straightforward debugging and modification, as states are clearly delineated and transitions are governed by particular inputs or conditions.

An FSM-based design holds significant relevance for transceivers that accommodate multiple protocols or adjust to fluctuating data rates, as it facilitates the FPGA's ability to transition between operational modes with ease. Furthermore, finite state machines enable low-latency response times and real-time adjustments, which are crucial for attaining high-speed performance in field-programmable gate arrays. In summary, the application of finite state machines in the programming of field-programmable gate arrays within transceivers results in a communication system that is more robust, adaptable, and reliable, thereby addressing the stringent requirements of contemporary networking applications.

1.5 Analog-to-Digital Converter

The ADC transforms the input analogue signal into a digital format. Analogue signals are continuous in time and amplitude, while digital signals are discontinuous in both dimensions.

Digitizing analogue sources has several advantages, as outlined below [10]:

1) Digital systems exhibit greater resilience to noise compared to analogue systems. For extended transmission distances, the signal can be regenerated with minimal errors at various intervals along the route, allowing the original signal to be conveyed over the remaining distance.

2) Digital systems facilitate the integration of diverse services, such as video and its corresponding audio, under a singular transmission framework.

3) The transmission mechanism may operate independently of the source. A digital transmission technique capable of transmitting voice at 10 kbps may similarly send computer data at the same rate of 10 kbps.

4) Digital signal circuitry is more easily replicated and exhibits reduced sensitivity to physical influences such as vibration and temperature.

5) Digital signals are easier to define and generally lack the amplitude range and variability characteristic of analogue signals. This simplifies the design of the accompanying hardware.

6) Techniques exist for incorporating controlled redundancy into digital transmissions, enabling error correction at the receiver without supplementary information. These techniques are classified as channel coding, and we have employed the LDPC channel coding method in our work.

The ADC accomplishes this conversion through a two-step method comprising sampling and subsequent quantisation.

Sampling refers to the process of converting a continuous-time signal into a discrete-time signal. A prevalent illustration is the transformation of a sound wave into a series of "samples". A sample represents the value of a signal at a specific moment in time.

Amplitude **quantisation** refers to the process of converting the sample amplitude of a message signal into a discrete amplitude selected from a finite set of potential amplitudes.

The Sampling Theorem states that a band-limited signal with finite energy, possessing frequency components below W Hz, may be fully reconstructed from samples obtained at a rate of 2W samples per second. This rate is referred to as the Nyquist Rate.

In reality, signals are not entirely band-limited, resulting in an inherent degree of aliasing. Aliasing denotes the occurrence when a high-frequency component in a signal's spectrum seems to be represented as a lower frequency in the spectrum of its sampled version.

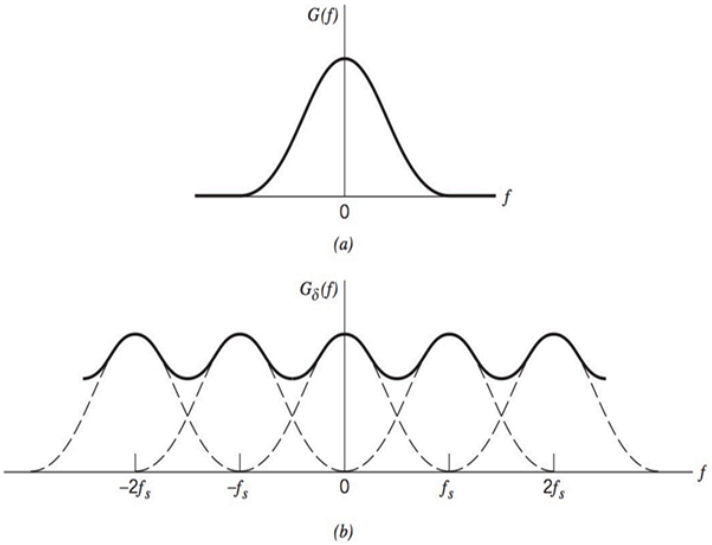


Figure 1.3: Aliasing Phenomenon. (a) Original Signal Spectrum (b) Spectrum of Sampled Signal at rate f_s

To limit this aliasing phenomenon, 2 methods exist:

1) Anti Aliasing Filter : An anti-aliasing filter can be employed before the sampling process is undertaken to strictly band limit the incoming signal.
2) Employing a sampling rate much higher than the Nyquist Rate.

The human hearing range extends from frequencies of 2 Hz up to 20 kHz. Hence, using a 40 kHz sampling rate in the ADC must suffice however, this requires that at the input, a very strict low pass filter needs to be employed as an anti-aliasing filter. Hence, in practice, audio signals are usually sampled at 44.1 kHz rate which allows for audio at frequencies up to 22.05 kHz.

2. METHODOLOGY

2.1 Low Density Parity Check

Low-Density Parity-Check (LDPC) codes are favored over other error correction codes for two principal reasons.

Firstly, the sparsity of the parity-check matrix H in LDPC codes, which contains relatively few non-zero entries, leads to a corresponding dense generator matrix G. This structure enables the use of efficient, low-complexity iterative decoding algorithms that converge rapidly to an error-free codeword. In contrast, codes with a non-sparse H necessitate more computationally intensive decoding methods, which can hinder performance.

Secondly, when compared to convolutional codes, LDPC codes exhibit significantly reduced encoding overhead. Convolutional codes rely on memory elements to generate parity bits, meaning each parity bit depends not only on the current input bits but also on previous bits, which must be stored and processed across multiple clock cycles. This reliance introduces substantial overhead in terms of both memory and computational load, particularly in asynchronous transmission scenarios where data packets may vary in size. On the other hand, LDPC encoding is inherently combinatorial, requiring no memory or clock cycles. This allows for streamlined encoding without additional timing constraints, making LDPC codes particularly well-suited for high-throughput applications, such as optical communication systems, where computational efficiency and simplicity are crucial.

Figure 2.1 illustrates the frame structure designed for each communication mode—text, audio, and video—in our optical transmission system. In our design, Low-Density Parity-Check (LDPC) codes are implemented to encode each mode of communication, using a unified 36-bit frame structure for all transmission types.

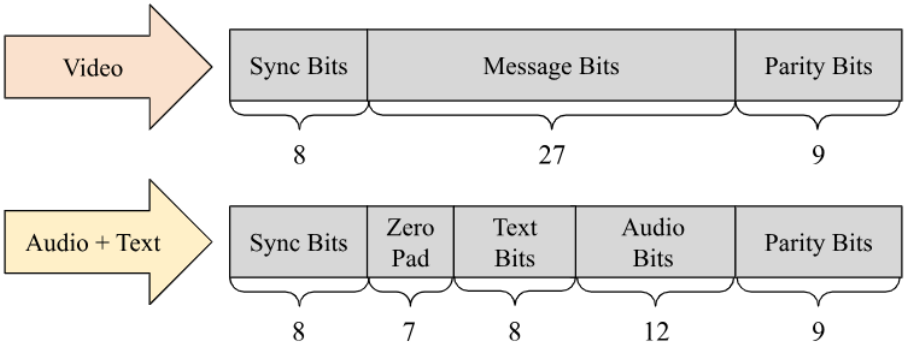


Figure 2.1: Frame structure for each communication mode: text, audio, and video.

Figure 2.2 presents the parity-check matrix H utilized in our design, while Figure 2.3 shows the corresponding generator matrix G. From these figures, it is evident that the parity-check matrix H is sparse, whereas the generator matrix G is comparatively dense, aligning with theoretical expectations. The H matrix was generated using a custom Python script and subsequently converted into systematic form via Gaussian elimination. The generator matrix G was then constructed based on the obtained parity matrix P, ensuring that the design adheres to the desired coding rate and structural properties.

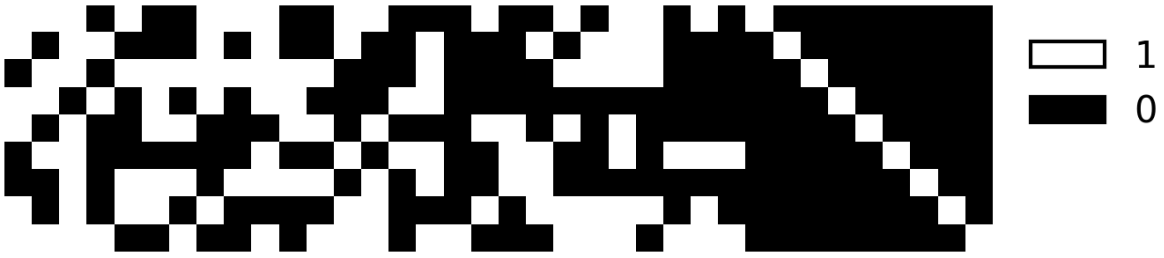


Figure 2.2: Parity-check matrix H used in the LDPC code design.

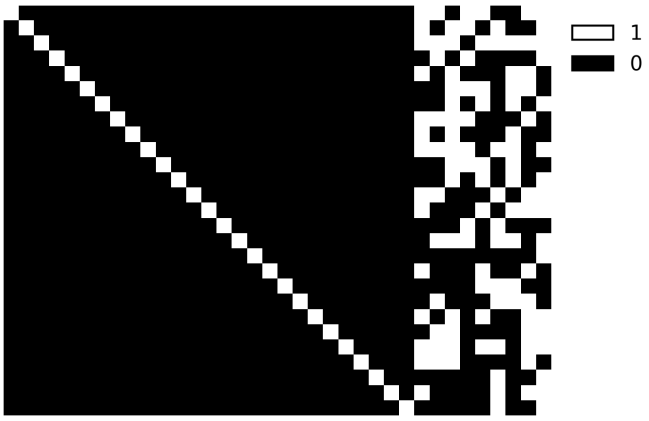


Figure 2.3: Corresponding generator matrix G derived from the systematic form of H.

2.1.1 Encoder

The encoder module was implemented in Verilog using assign statements for the generation of parity bits. Figure 2.4 illustrates the process for one of the parity bits. This procedure was systematically applied to generate all required parity bits. Due to the impracticality of manually writing assigned statements for each parity bit, we developed a Python script that takes the parity check matrix H as input. This script automatically generates the necessary code for both the encoder and decoder, which is subsequently integrated into our design.

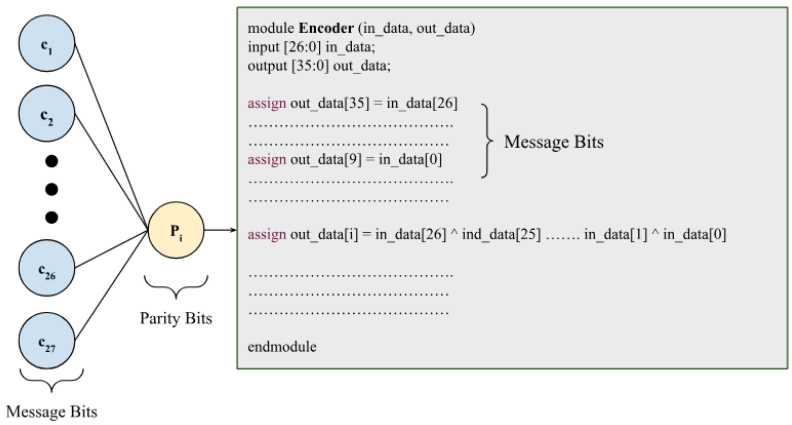


Figure 2.4: Schematic representation of the encoder's parity bit generation process.

`

2.1.2 Decoder

The decoder module, implemented in Verilog, utilizes the Sum-Product Algorithm, adapted for efficient hardware implementation. As depicted in Fig. 8, the decoding process begins by calculating the syndrome vector (**s**) using matrix multiplication of the received bit vector (**y**) with the transpose of the parity-check matrix ($\mathbf{H^T}$):

$$\mathbf{s = y.H^T}$$

The matrix multiplication was implemented using the XOR gates, as seen in the encoder module. Once the syndrome vector is generated, it serves as an indicator of transmission errors: a zero syndrome vector indicates no errors, while a non-zero vector prompts error correction.

In the subsequent steps, each message node is assigned a count register initialized to zero. For every bit in the syndrome vector with a value of 1, the corresponding count registers associated with that syndrome bit's connections are incremented. The message bit corresponding to the count register with the highest value is then flipped to correct potential errors.

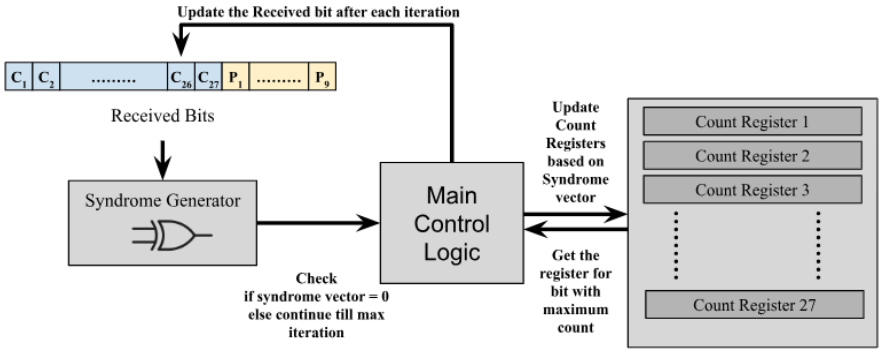


Figure 2.5: High-level design of the decoder module.

This iterative correction process is managed by a **Main Control Logic** block, as illustrated in Figure 2.5. After each iteration, the control logic updates the received bit and re-evaluates the syndrome vector. The iterations continue until the syndrome vector becomes zero or the maximum iteration limit is reached, ensuring error resolution within the bounds of the decoder's design constraints.

To accommodate larger message blocks required for higher data rates, the decoder's Verilog implementation is automated using Python scripts. This approach enables rapid scalability and efficient adaptation of the design without extensive manual code adjustments, streamlining future development for increased data throughput.

2.2 FSM for Transceiver

2.2.1 Analog-to-Digital Converter

In our design, we have used the XADC module present in the Pynq-Z2's Zynq-7020 SoC for sampling and quantization. The XADC is a 12-bit 1 MSPS ADC, i.e., it can quantize the input to 12 bits resolution, or in other words, 4096 discrete amplitude levels, and can sample at rates up to 1 Mega Samples Per Second. We configured it in the Vivado 2018.3 wizard to operate at the rate of 50 KSPS. It takes a differential voltage as an input in the range of 0-1 V.

The audio output coming from the microphone or 3.5 mm jack has an amplitude that ranges from -0.5 V to +0.5 V. Hence, we need to add a DC offset of 0.5 V to this signal before sending it in the V_P and V_N ports of the PYNQ Z2 board. This is accomplished by employing the following circuit given in Figure 2.6 utilizing the on-board 3.3 V and GND ports [11].

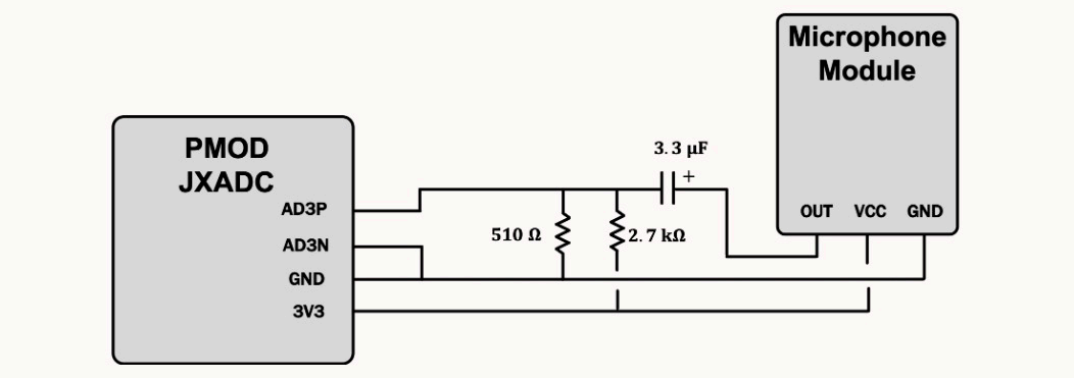


Figure 2.6: Connecting the V_P and V_N pins with the audio input

2.2.2 Data Frame

The 12-bit ADC data obtained from XADC needs to be converted into an appropriate format for transmitting. For timing recovery, we need to add synchronization bits at the start of the frame. There will also be trailing bits for LDPC channel coding. Here, we are using a 3/4 code rate LDPC code.

We first concatenate the 12-bit ADC data with the text data character that we want to send and pad this with 7 0 bits. After that, we perform the LDPC encoding on these 16 bits. The LDPC encoder is just a combinatorial circuit because the encoding involves performing the XOR operation on the available bits based on the given H matrix.

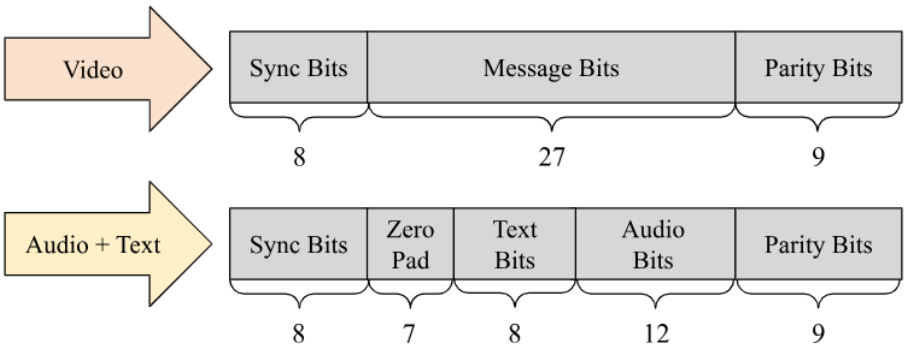


Figure 2.7: Frame Structure

2.2.3 Transmitter FSM

The transmitter FSM needs to be made such that it can wait for the ADC to complete conversion and only then start propagating the frame bits through the medium. It then needs to wait till the entire transmission of one frame is complete before waiting for the next frame to arrive. The transmitter initially starts in the IDLE state, and only after it is enabled does it go into the

TRANSMIT state. It then moves to the WAIT_ADC state after all 44 bits of the frame have been transmitted.

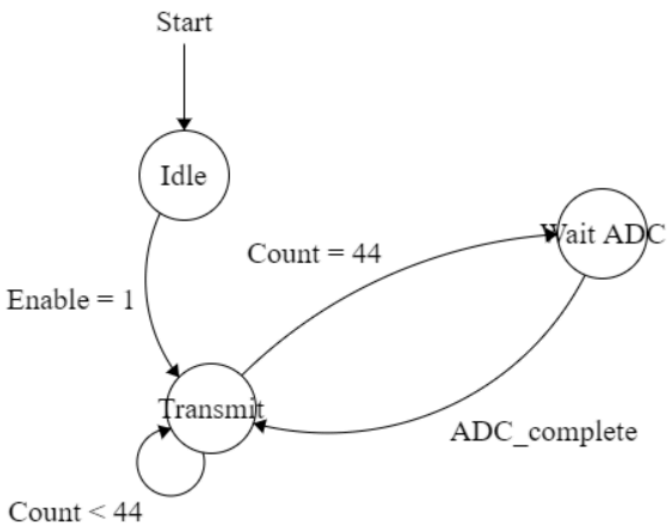


Figure 2.8: Transmitter FSM

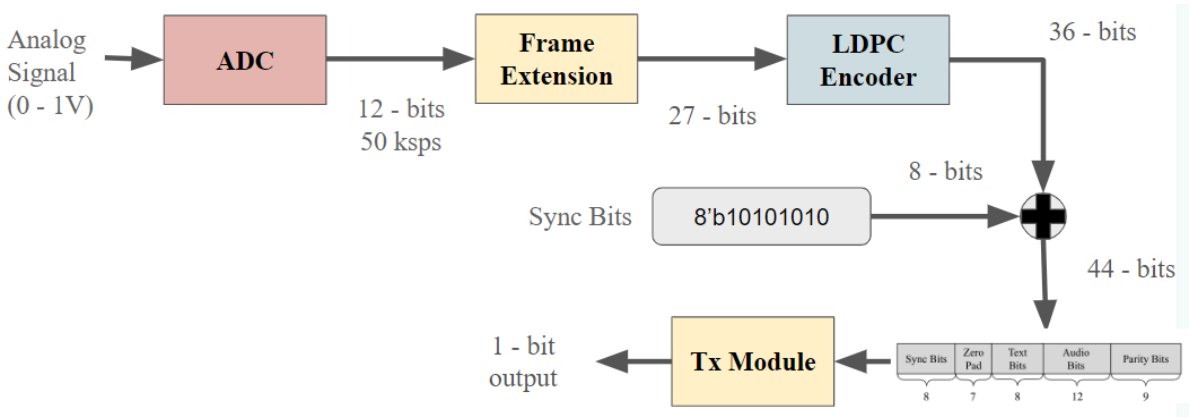


Figure 2.9: Transmitter Block Diagram

2.2.4 Receiver FSM

The receiver FSM needs to be made such that it can wait for the ADC to complete conversion and only then start propagating the frame bits through the medium. It then needs to wait till the entire transmission of one frame is complete before waiting for the next frame to arrive.

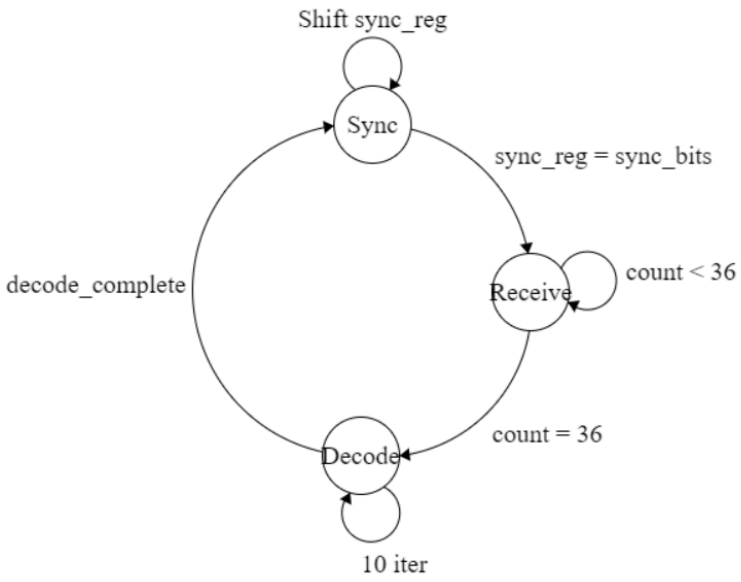


Figure 2.10: Receiver FSM

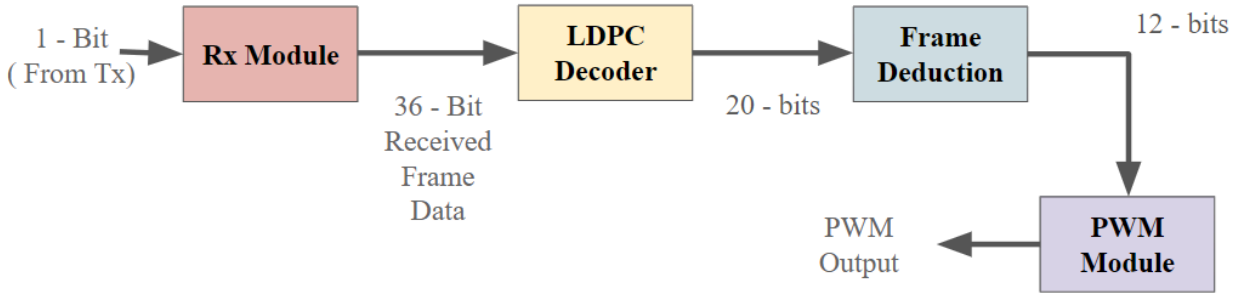


Figure 2.11: Receiver Block Diagram

2.2.5 PWM Audio

At the receiver output, instead of utilizing a DAC, a PWM (Pulse Width Modulation) module was employed for digital-to-analog conversion. The implemented scheme is a Digital PWM approach, designed to handle 12-bit quantized values as input and produce a PWM audio signal as output.

Figure 2.12 illustrates the flow diagram of the PWM module implemented in the system. The module operates as follows:

1. The 12-bit input value (`pwm_in`) is compared with an internal counter (`count[11:0]`), initialized to zero at the start of the process.
2. The counter increments continuously until it reaches its maximum value of 4095. If the counter exceeds this value, it resets to zero.

3. The comparison between the counter and the input determines the output state:
    - If `count > pwm_in`, the output is set to logic `0`.
    - Otherwise, the output is set to logic `1`.
4. This results in a PWM signal where the duty cycle is proportional to the input quantized value, effectively representing the analog signal.

This approach enables the system to efficiently generate an analog waveform corresponding to the input digital values, ensuring compatibility with the audio output requirements while minimizing hardware complexity.

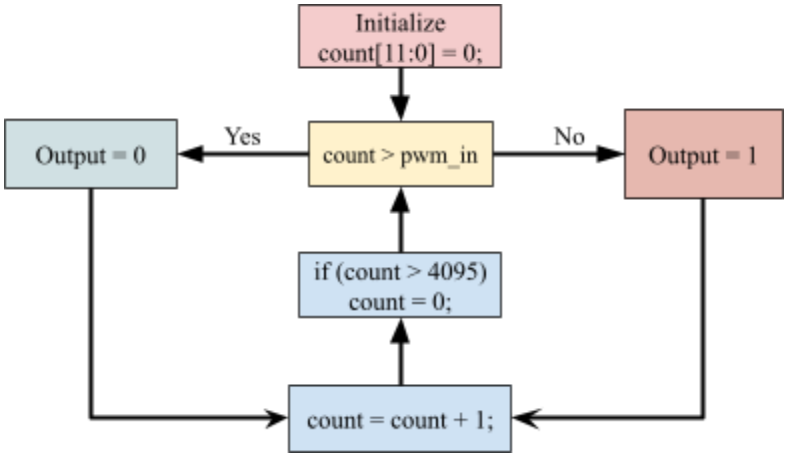


Figure 2.12: PWM Module flow chart

3. RESULTS

3.1 Low Density Parity Check

One of the critical hyperparameters in our LDPC decoder design is the maximum number of decoding iterations, which directly influences error correction performance. Increasing the maximum iterations generally improves error correction, as it provides the decoder with more opportunities to resolve ambiguities in the received signal. However, a higher iteration count comes with a trade-off: it increases the number of clock cycles required for decoding, leading to potential delays in the output signal. This trade-off is particularly significant in applications like video transmission, where minimizing latency is crucial.

To balance these trade-offs, we selected an optimal maximum iteration count of 10, which achieves effective error correction while limiting delays in real-time transmission scenarios.

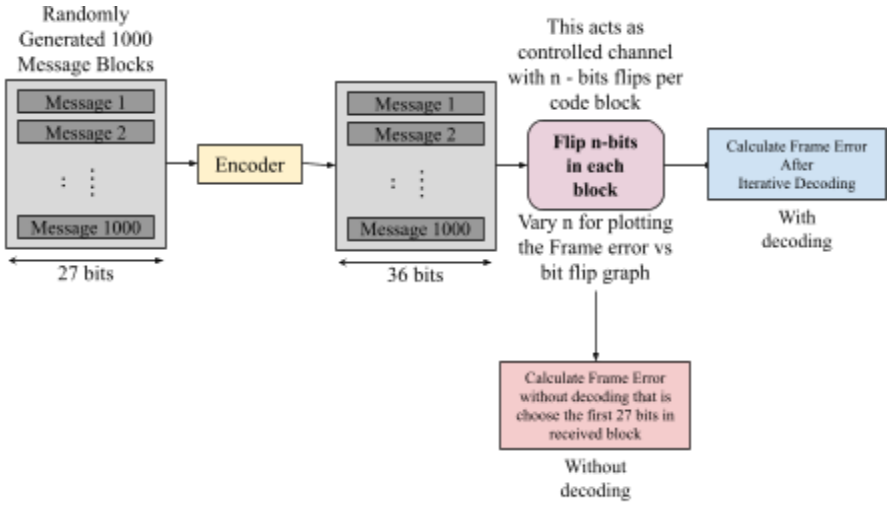


Figure 3.1: Flow Diagram of the simulation.

To further evaluate the error correction performance of our chosen LDPC code under this configuration, we conducted detailed simulations in Python, using the specified generator and parity-check matrices. The simulation setup and results are presented in the following section, providing a comprehensive analysis of the code's performance in the context of the defined iteration limit. Figure 3.1 illustrates the simulation methodology.

In the simulations, we generated 1,000 random message sequences, each comprising 27 bits. These sequences were encoded using the generator matrix to produce 1,000 codewords, each consisting of 36 bits. To simulate transmission errors, a specified number of bits in each codeword was randomly flipped. The corrupted sequences were then passed through the LDPC decoder, and performance was measured by calculating the frame error rate (FER), which represents the proportion of decoded code blocks that deviate from the original messages.

Figure 3.2 shows the relationship between the frame error rate (FER) and the number of bit flips per frame, with and without decoding. Key observations are as follows:

Single Bit Flip Case:

- Without decoding, the FER is approximately 0.75. This is because about 25% of message blocks contain errors in the parity bits, which do not directly affect the decoded message bits (first 27 bits). This aligns with theoretical expectations.
- With iterative decoding, the FER significantly reduces to around 0.2. This demonstrates the decoder's ability to correct errors, although the error rate could be further reduced by increasing the maximum iterations. However, as discussed earlier, we limit iterations to 10 to manage processing delays.

Error Correction for Optical Channels:

- The model is particularly effective for channels with a bit error rate (BER) of 0.027 or less (equivalent to 1/36). This aligns with the BER range typically encountered in optical and underwater communication channels, which is between 10^{-6} and 10^{-3}.

Theoretical Analysis: Average Bit Flips

The average number of bit flips in a 36-bit block can be estimated using the binomial distribution. Let p denote the bit flip probability. The probability of n bit flips in a block is given by:

$$P(n) = \binom{36}{n} p^n (1 - p)^{36-n}$$

The average number of bit flips is:

$$E[bit\ flips] = \sum_{i=0}^{36} i \cdot \binom{36}{n} p^n (1 - p)^{36-n}$$

For bit error rates (BER) in the range 10^{-6} to 10^{-3} , the average number of bit flips per block is approximately in the order of 10^{-5} to 10^{-2}. These values are well within the decoder's capabilities, confirming its suitability for such communication scenarios. These results confirm that for such low BERs, the LDPC decoder can effectively handle the errors, ensuring accurate error correction for communication channels operating within this range.

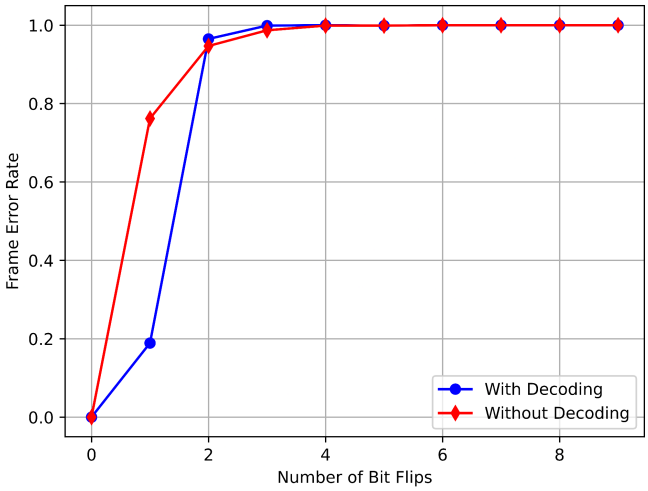


Figure 3.2: Error Performance of LDPC Code with Varying Bit Flips.

3.2 Finite State Machine

With the methodology given above, we have made an FPGA-based transceiver that can be used in a variety of scenarios for audio transmissions. Since it is a digital modulation scheme, the given transceiver can also be extended to applications involving other complicated data, like video or text, by changing the frame structure accordingly. All the simulations have been performed in Xilinx Vivado 2018.3.

Following is the simulation result for the Transmitter FSM operation:

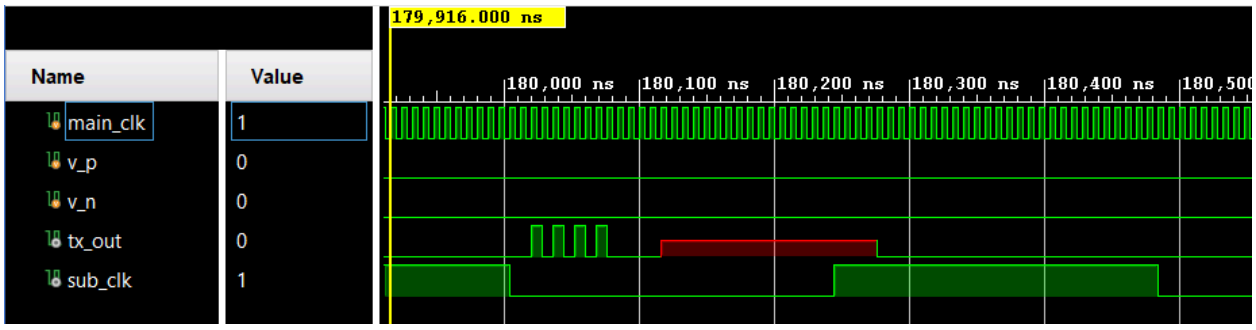


Figure 3.3: Transmitter Simulation

The waveform for tx_out is the one that is used to modulate the light source in the OOK modulation scheme[12]. The first 8 bits are for synchronization, followed by four zeros, after which come the data bits. Once the entire frame has been transmitted, the tx_out waveform returns to 0.

The following table depicts the utilization of FPGA resources in the transmitter:

Name	Slice LUTs (53200)	Bonded IOB (125)	Bonded IPADs (2)	BUFGCTRL (32)	XADC (1)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)
transmitter	41	3	2	1	1	33	16	41
ADC (get_adc_data)	0	0	0	0	1		0	0
XADC_INST (audio_...	0	0	0	0	1		0	0
F (div_clk_temp)	41	0	0	0	0		16	41

Figure 3.4: Transmitter Utilization

Following is the simulation result for the receiver FSM operation:

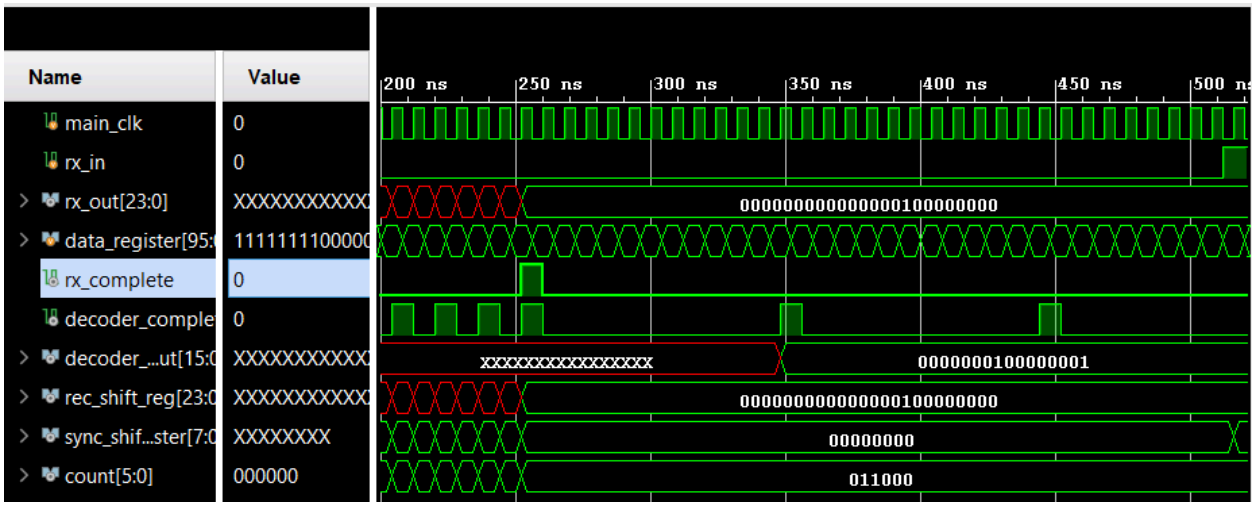


Figure 3.5: Receiver Simulation

The following table depicts the utilization of FPGA resources in the receiver:

Name	Slice LUTs (53200)	Bonded IOB (125)	BUFGCTRL (32)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)
receiver	40	8	2	103	37	39	1
DIV (div_clk)	10	0	0		14	10	0
PWM (gen_pwm8)	16	0	0		6	16	0
R (rx)	14	0	0		12	13	1

Figure 3.6: Receiver Utilization

The methodology used here can be extended to be used for any channel and application for digital audio transmission.

3.3 Experimental Results

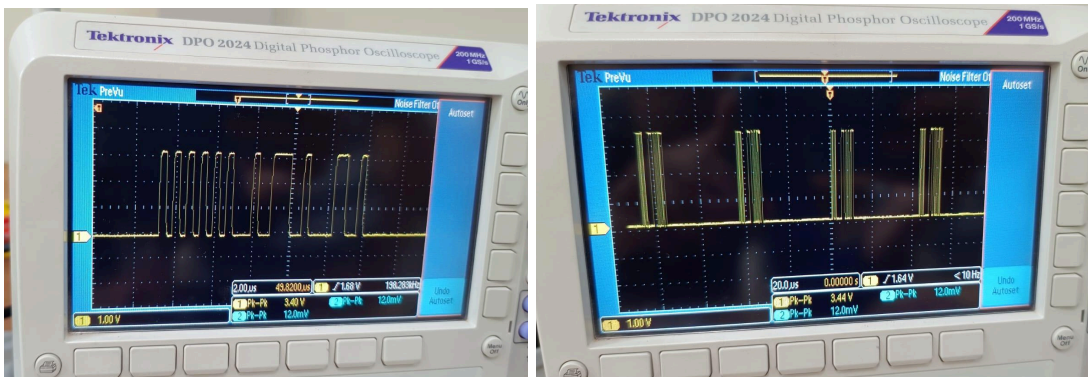
This section presents the results obtained from the experimental setup of the transmitter and receiver model on the PYNQ-Z2 FPGA board. To validate the functionality of the system, we

initially tested the transmission and reception of audio and text signals using a direct wired connection between the transmitter and receiver. Once the correct operation was confirmed, the same setup was replicated with the inclusion of an optical channel between the transmitter and receiver, aligning with the intended objective of the experiment.

3.3.1 Direct Transmission (Without Optical Channel)

To validate the functionality of the transmitter and receiver modules, we conducted an initial experiment involving direct transmission without an intermediate optical channel. In this setup, a 3.5 mm audio jack was used to obtain an analog audio signal from a laptop, which served as the input to the PYNQ-Z2 FPGA board through designated input pins. Additionally, we utilized the four onboard buttons of the PYNQ-Z2 board to simulate a text message, serving as a placeholder for testing purposes.

The audio signal was digitized and encoded, while the states of the buttons were binary-coded to represent text messages. Specifically, 8 bits were allocated for the text in each frame, with 2 bits assigned to each of the four buttons in this case. This configuration was implemented solely to test the capability of transmitting textual data alongside audio signals. The frame structure, including the lengths allocated for audio and text data, is detailed in Figure 2.4.



(a) (b)

Figure 3.7: Visualization of (a) single frame, (b) multiple frames transmitted by the transmitter, observed using an oscilloscope.

Figure 3.7.a illustrates a single frame transmitted by the transmitter, as visualized using an oscilloscope. In contrast, Figure 3.7.b displays multiple consecutive frames, transmitted at a bit rate of 3.125 Mbps and a frame rate of 50 K frames/second These visualizations provide insight into the structure and timing of the transmitted signal.

The transmitted signal was then connected directly to the receiver using wires for validation. The audio component of the frames was processed and visualized through a generated PWM

waveform. This waveform was subsequently fed into a speaker at the receiver end, confirming successful audio playback.

The outputs corresponding to the button states were represented by illuminating the four LEDs on the PYNQ-Z2 board, with each LED reflecting the status of one button. This setup effectively demonstrated the accurate decoding of text and audio signals at the receiver.

Figure 3.8 illustrates the PWM output generated by the receiver. This PWM signal was subsequently provided as input to a speaker, enabling the audio transmitted from the input at the transmitter end to be reproduced audibly. This successfully demonstrates the end-to-end functionality of the system for audio signal transmission and reception.

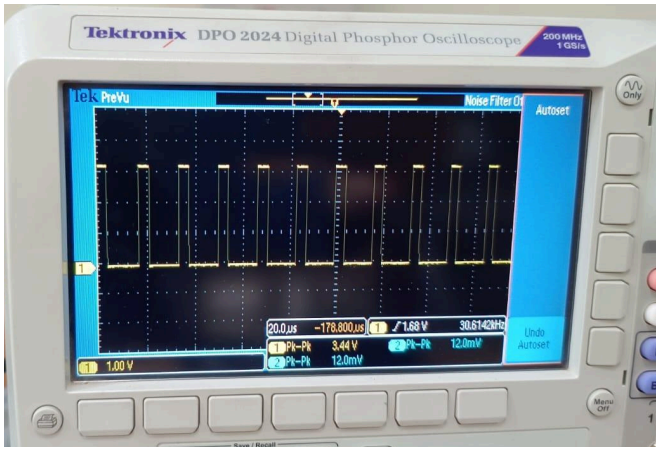


Figure 3.8: PWM output on Oscilloscope

3.3.2 Transmission with Optical Channel

In this experiment, instead of directly connecting the transmitter output to the receiver, the signal was passed through an optical modulation unit. The optical modulation unit performed two key functions: converting the electrical signal into an optical signal and modulating it. The modulated optical signal was then transmitted through a single-mode optical fiber and subsequently processed by a transducer unit at the receiver end. This transducer converted the optical signal back into an electrical signal for further processing.

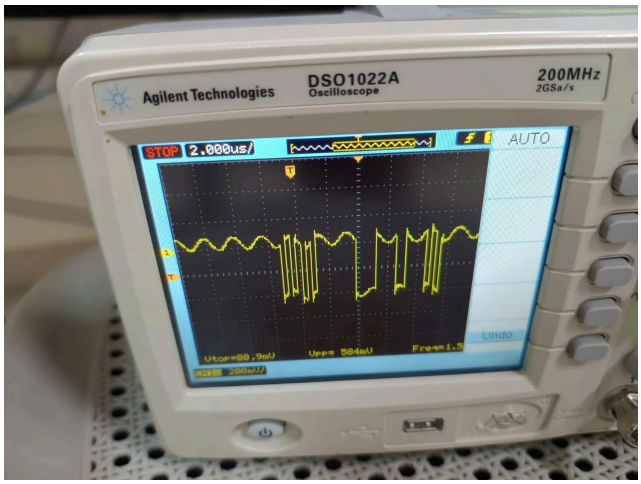


Figure 3.9: Output of Transducer

Figure 3.9 illustrates the output of the transducer unit at the receiver end. The received signal was found to be inverted and significantly attenuated. Since the PYNQ-Z2 board requires an input signal with a 3.3V range, the received signal was too weak for direct input. To address this, an RF amplifier was used to amplify the signal.

Figure 3.10 shows the output of the RF amplifier. However, the amplified signal exhibited distortion, with its baseline dropped below acceptable levels, and the overall signal range remained unsuitable for input to the receiver board. Due to these issues, we were unable to proceed further with the experiment in its current configuration and will need to identify solutions to resolve these challenges.

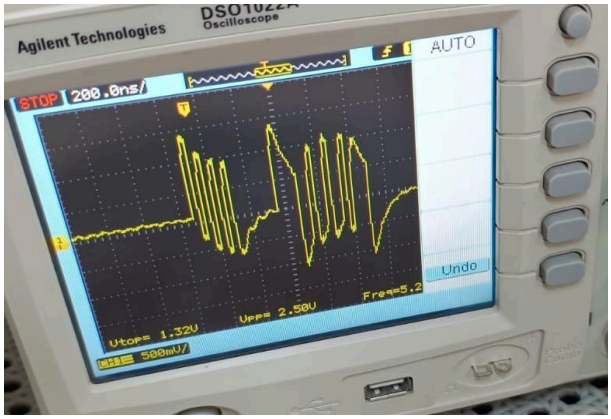


Figure 3.10: Output of RF Amplifier

4. FUTURE PLAN

The immediate focus of our work will be to address the challenges encountered in optical channel transmission, as highlighted in the results section. Resolving issues such as signal attenuation and distortion will be critical to ensuring reliable communication through this medium. Once the audio and text data can be successfully transmitted over the optical channel, we plan to expand the system's capabilities by incorporating video data transmission using the HDMI protocol.

Video transmission will require significant improvements in system performance, particularly in terms of clock speed. Currently, the PYNQ-Z2 FPGA board operates at a maximum clock frequency of 125 MHz, which may limit the video transmission capacity. To meet the higher data rate demands of video, we will explore potential optimizations, such as increasing the system clock or implementing more efficient data encoding schemes.

In parallel, we will focus on enhancing the system’s clock synchronization. At present, the clocks of the transmitter and receiver are manually synchronized, which introduces the potential for errors and inefficiencies. Moving forward, we aim to implement an automatic clock synchronization mechanism to ensure precise alignment between the transmitter and receiver clocks, thus improving the reliability and performance of the system.

Furthermore, we intend to target the IEEE RAPID Conference, showcasing our work on transmitting high-bandwidth data, including video, through the optical channel. By addressing the unique challenges of underwater communication, particularly the transmission of high-speed data via visible light, we aim to contribute to the development of efficient communication systems for underwater environments.

CONTRIBUTION

Kalbhavi Vadhi Raj (B21EE030):

- LDPC Encoder and Decoder Verilog code.
- Python Script for generating LDPC encoder and decoder Verilog Code from arbitrary H matrix.
- Implemented Cyclic Redundancy Check encoder and decoder, which was intended to be used before LDPC.
- Implemented module to control the frame rates to facilitate the smooth audio transmission.
- Submitted technical paper for application of LDPC in optical channels in technical hindi seminar.

Pranav Chakravarthy (B21EE050):

- Implemented the Transmitter and Receiver FSM Modules in Verilog.
- Implemented the ADC using XADC for Analog-toDigital conversion for audio input.
- Implemented low-pass filter for filtering of PWM output.
- Submitted technical paper for Transmitter and Receiver FSM in Technical Hindi Seminar 2024, IIT Jodhpur.

REFERENCES

[1] Le Bidan, R., Leroux, C., Jego, C., Adde, P., & Pyndiah, R. (2008). Reed-Solomon Turbo Product Codes for Optical Communications: From Code Optimization to Decoder Design. *EURASIP Journal on Wireless Communications and Networking*, 2008(1), Article 658042. https://doi.org/10.1155/2008/658042

[2] Sasaki, N., Shimada, H., Shimada, S., & Kobayashi, H. (2016). *Evaluation of transmission quality of visible light communication using bit error rate measurement*. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)* (pp. 1362–1365). IEEE. https://doi.org/10.1109/ICCAS.2016.7832488

[3] Ingale, M. A. (2003). *Error correcting codes in optical communication systems* (Master's thesis, Chalmers University of Technology, Gothenburg, Sweden). School of Electrical Engineering, Department of Signals and Systems.
Examiner: Erik Agrell, Technical Supervisor: Tume Römer, Administrative Supervisor: Dr. Björn Rudberg.
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=40103c5ea422b14dfcceb9c1b674262a9cd40640

[4] Everett, J. S. (2009). Forward-error correction coding for underwater free-space optical communication (Master's thesis, North Carolina State University, Raleigh, North Carolina). Graduate Faculty of North Carolina State University.
https://repository.lib.ncsu.edu/server/api/core/bitstreams/e1bebc66-da9a-4cae-bb43-1399111ced90/content

[5] S. H. Gupta, & B. Virmani. LDPC for Wi-Fi and WiMAX Technologies. 2009 International Conference on Emerging Trends in Electronic and Photonic Devices & Systems. Varanasi, India, 2009;262–265. https://doi.org/10.1109/ELECTRO.2009.5441120.

[6] Morello, A., & Mignone, V. DVB-S2: The Second Generation Standard for Satellite Broad-Band Services. Proceedings of the IEEE. 2006;94(1):210–227. https://doi.org/10.1109/JPROC.2005.861013.

[7] IEEE. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11™-2016 (Revision of IEEE Std 802.11-2012).

[8] IEEE. IEEE Std 802.11ad-2012: Enhancements for Very High Throughput Operation in Bands below 6 GHz. IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems.

[9] IEEE. IEEE Std 802.11ah™-2016: Sub 1 GHz License Exempt Operation. IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems.

[10] Simon Haykin, Michael Moher, Communication Systems, 2017

[11] MIT. (2019). Lab 5: Digital Communication Using On-Off Keying. Retrieved from https://web.mit.edu/6.111/volume2/www/f2019/handouts/labs/lab5_19a/index.html.

[12] Sarkar, T. S., Sinha, B., Mukherjee, S., Joardar, I., & Mazumdar, S. (2020). Development of an FPGA-based Indoor Free Space Optical (FSO) Communication System using 808 nm Infrared (IR) LASER Source. In 2020 IEEE Calcutta Conference (CALCON) (pp. 313-317). https://doi.org/10.1109/CALCON49167.2020.9106422.

[13] Otternes, R. FSM Designer. Retrieved from https://www.cs.unc.edu/~otternes/comp455/fsm_designer/.