BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 318 (July 2023)

Communication Systems I Laboratory

Final Project Report

Section: A1 Group: 04

PCM Line Code Generation

Course Instructors:

Shafin Bin Hamid, Lecturer
N, Part-Time Lecturer

Signature of Instructor: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name:** **Student ID: 1906xyz**	**Signature:** _____ **Full Name:** **Student ID:**
Signature: _____ **Full Name:** **Student ID:**	**Signature:** _____ **Full Name:** **Student ID:**

PCM Line Code Generation

Submitted By -

2006010
2006016
2006022
2006023
2006024
2006026
2006028

Table of Contents

List of Figures

List of Tables

1 Abstract

This report explores PCM Line Code generation, a widely employed digital data transfer method in communication systems. The construction process of a circuit with four blocks for PCM Line Code generation is detailed. The circuits are simulated using specialized software and subsequently implemented in hardware, featuring LED bulbs for enhanced visualization. The report addresses challenges associated with implementing such hardware systems for PCM and introduces a distinctive approach to implementing the IC for encoding, a noteworthy aspect of the project.

2 Introduction

One popular technique for digitally encoding analog signals is pulse code modulation (PCM), which is frequently used in audio and telecommunications applications. One of the most important components of PCM systems is PCM Line Code Generation, which is the process of converting digital signals into a particular line code for communication channels.

The primary objectives of PCM Line Code Generation include achieving efficient bandwidth utilization, ensuring synchronization between the transmitter and receiver, and providing mechanisms for error detection and correction.

Various line coding schemes, such as Unipolar, Polar, Bipolar, and Manchester encoding, are employed to represent the digital data in a way that is suitable for the characteristics of the communication channel. Analog signals are quantized and sampled in PCM in order to capture the signal's amplitude at discrete times. It is necessary to efficiently encode the resultant digital signal, which is a series of binary digits, into a format that can be transmitted over communication channels. PCM Line Code Generation is useful in this situation. The two logic values that make up PCM's signal representations are 1 (high) and 0 (low).

The final line code scheme can be of several types. NRZ-L type line coding is one of them. This process is fundamental to the integrity of digital communication systems, influencing factors like data rate, signal quality, and overall system reliability. PCM Line Code Generation plays a pivotal role in ensuring that the digital information is accurately and reliably transmitted over communication channels, making it a crucial aspect of telecommunications and other applications where digital data transmission is essential.

3 Design

3.1 Problem Formulation

The task is to design a basic PCM (Pulse Code Modulation) line coder that can convert an analog signal into a digital signal for transmission over a communication channel. The design must account for the following key requirements:

1. Sampling: The line coder must be able to sample the analog signal at a rate high enough to accurately capture the original signal without introducing distortion or aliasing.

2. Quantization levels: The line coder must quantize the sampled signal into a discrete set of levels to represent the analog signal digitally. The number of quantization levels must be chosen carefully to balance between signal fidelity and bandwidth efficiency.

3. Encoding scheme: The line coder must utilize an appropriate encoding scheme to efficiently represent the quantized signal in binary format for transmission. This scheme must be robust against noise and other channel impairments.

4. Output format: The line coder must produce a digital output signal that is compatible with the communication channel and any subsequent processing or decoding steps.

3.2 Design Method

To start off, we segment the problem into 4 parts, each of which is solved independently and implemented in an individual circuit block. This modularity allows for greater flexibility and adaptability, easier error detection, ease of customization and overall, faster implementation. Largely, the project problem consists of the following parts:

3.2.1 The Sampler Block

The process of PCM line code generation begins with sampling the message signal. Sampling refers to the process of converting a continuous-time signal into a discrete-time signal by capturing its values at discrete time intervals. This technique is essential for converting analog signals into digital signals for further processing and analysis. There are various methods of sampling, with natural sampling and flat-top sampling being two commonly used techniques.

Natural sampling involves sampling a continuous signal at regular intervals using a sample-and-hold circuit. This method captures the instantaneous value of the signal at specific points in time, which allows for accurate representation of the original signal. This means that the sampling rate is determined by the frequency of the signal itself. Natural sampling is often used when the signal is periodic and the sampling rate can be easily determined based on the frequency of the signal. This method is simple and easy to implement, but it may not always provide the most accurate representation of the signal, especially if the signal contains high frequency components.

On the other hand, flat-top sampling involves sampling a signal by holding a constant value over a specified time interval before sampling the next value. This method is often used in applications where the signal has a high frequency or where the sampling rate is limited.

One of the main advantages of natural sampling is its simplicity and ease of implementation. Since the sampling rate is determined by the frequency of the signal, there is no need to calculate or set a specific sampling rate. This can be especially useful when dealing with periodic signals, as the sampling rate can be easily determined based on the signal's frequency. Flat-top sampling, while providing better accuracy, has much more complex circuits which are rather tough to get properly working. This is why, in this project, we opt for natural sampling.

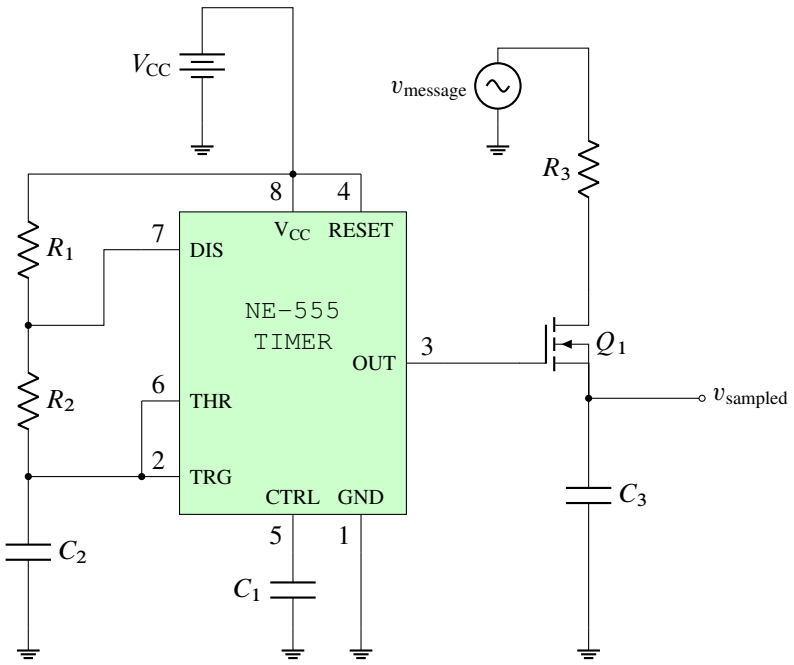

Figure 1: Sampling circuit

3.2.2 The Quantizer Block

This is where the sampled data is uniformly quantized in amplitude. Quantization simplifies the representation of analog data by reducing the number of possible values that can be taken on by the signal. This process involves rounding off the sampled analog values to the nearest quantization level.

The quantization process can be uniform or non-uniform, depending on the spacing of the quantization levels. In uniform quantization, the levels are evenly spaced, while non-uniform quantization allows for variable spacing to better match the statistical properties of the signal. Non-uniform quantization can help reduce quantization error and improve the overall noise performance of the digital signal. Specifically in our case, the quantization is uniform.

We use comparators to assign one of the quantized voltage levels for the sampled input signal. Referring to the circuit diagram, the upper and lower peaks of the message signal is used as input to a voltage divider, which provides us with the eight possible values, all equally spaced, for the quantized signal. Furthermore, there are eight op-amps functioning as comparators. The sampled signal value is sent to the non-inverting inputs of the op-amps, and the eight quantized voltage levels are connected to the inverting channels of the eight op-amps. For a sampled value, the op-amps with a non-inverting supply greater than the sampled value will return a positive output and the rest negative, thus identifying the quantization level.

This is a truncation type quantizer, which truncates a voltage value to a higher reconstruction level. For example, a v_{sampled} very close to V_{min} will cause only the last op-amp (marked H) to have a positive output. If the sampled value is at the third decision level, the op-amps marked F, G, H will output positive value. We can configure the op-amps to operate between 0 and a positive value, corresponding to logic values of 0 and 1.

Overall, denoting the lowest quantization level as 1 and the highest as 8, we have the following table (refer to figure 2):

Quantization Level	Op-amp Outputs							
	A	B	C	D	E	F	G	H
8	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1
6	0	0	1	1	1	1	1	1
5	0	0	0	1	1	1	1	1
4	0	0	0	0	1	1	1	1
3	0	0	0	0	0	1	1	1
2	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1

Table 1: Logic outputs from the quantizer circuit

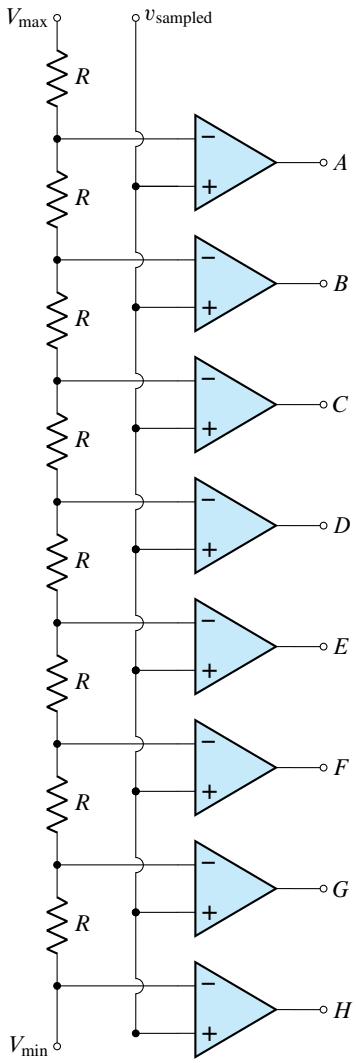

Figure 2: Quantizer circuit

Instead of the eight op-amps, we use a couple of `LM324` quad op-amps, each consisting of four independent op-amps in a single 14-pin DIP package.

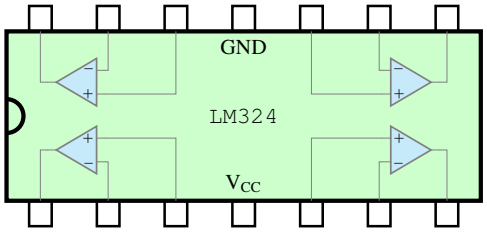

Figure 3: Quad op-amp IC for quantization

3.2.3 The Encoder Block

At this stage, the quantized data must be encoded to efficiently represent it using a limited number of symbols or bits. Encoding quantized data involves mapping the quantization levels to specific symbols or codes that can be easily transmitted or stored. There are several popular techniques which are used for encoding digital data, PCM being one of them. In PCM, each quantization level is represented by a specific binary code word. The number of bits in each code word determines the number of

quantization levels that can be represented. Initially, we decided to use the `IC 74148` CMOS 8 to 3 line priority encoder. But this is where we hit a little snag, as the truth table for the `74148` encoder is not an exact match for the data we get from quantizer. Extending Table 1, we need the encoder to function like the following:

Quantization Level	Op-amp Outputs								Encoded Value		
	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	M	MM	L
8	1	1	1	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1	1	1	0
6	0	0	1	1	1	1	1	1	1	0	1
5	0	0	0	1	1	1	1	1	1	0	0
4	0	0	0	0	1	1	1	1	0	1	1
3	0	0	0	0	0	1	1	1	0	1	0
2	0	0	0	0	0	0	1	1	0	0	1
1	0	0	0	0	0	0	0	1	0	0	0

Table 2: Final output of 3 bit encoder

However, the truth table for the `74148` encoder requires one zero input for the eighth quantization level (i.e. the output of 111), whereas we have none. We devised two methods to overcome this:

1. We could use the truth table to develop the necessary logical expressions and implement the encoder using logic gates. To follow this route, the three bits of the encoded data are extracted individually from the quantizer ouputs. We then use logic gates to implement this scheme, specifically, 74-series logic gates, which are a family of integrated circuits widely used in digital electronics. The 74-series has become a staple in the industry due to its reliability, versatility, and low cost. There are several types of logic gates in the 74-series family, including AND gates, OR gates, NOT gates, NAND gates, and NOR gates. Each gate performs a specific logical operation, such as AND gates outputting a high signal only when all input signals are high, or NOT gates inverting the input signal. A commonly used 74-series logic gate is the 7404 hex inverter. This gate contains six individual inverters on a single chip. he XOR gate, short for exclusive OR, is a digital logic gate that outputs true when the number of true inputs is odd. It is commonly used in computer arithmetic and digital communications to detect errors or produce parity bits. The 74 series XOR gates typically come in a 2-input configuration, with multiple gates packaged in a single IC chip. The AND gate is another fundamental logic gate that outputs true only when all of its inputs are true. It is used in a wide range of applications, including Boolean algebra, digital signal processing, and automatic control systems. The 74 series AND gates are available in various configurations, such as 2-input, 3-input, and 4-input gates. The OR gate is a logic gate that outputs true when at least one of its inputs is true. It is used in logical operations where multiple conditions need to be satisfied. The 74 series 2-input OR gates are most common, which is also the one we will use.From Table 2, the MSB for encoded data can be taken directly from the output of the D op-amp.

Figure 4: Extracting MSB from quantizer output

Extracting the MMSB is a little more complex, but it can be done by directing the outputs from the op-amps marked B, D, F through the logical expression $B \vee (\neg D \wedge F)$.

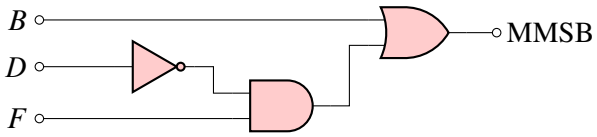

Figure 5: Extracting MMSB from quantizer output

The LSB can be extracted by routing the first seven outputs (the last one is immaterial since it never changes state) through an XOR. Since XOR is commutative and associative, we can XOR the values one by one.

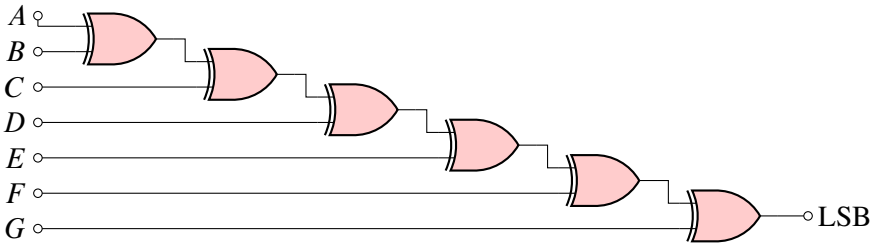

Figure 6: Extracting LSB from quantizer output

To summarize,

$$\begin{aligned} \mathrm{M} &= D \\ \mathrm{MM} &= B \vee (\neg D \wedge F)\,. \\ \mathrm{L} &= A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G \end{aligned}$$

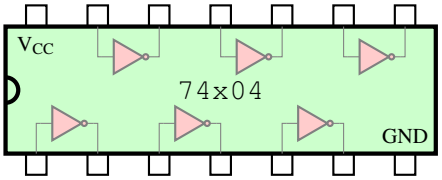

Figure 7: `7404` Hex inverter

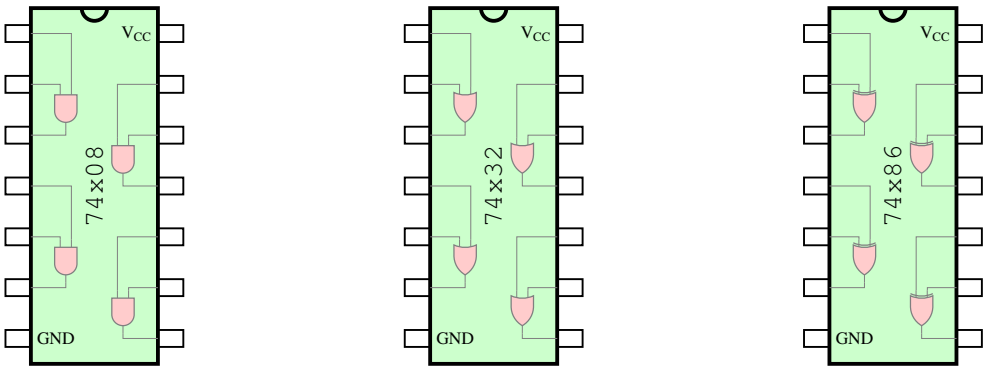

Figure 8: Quad 2-input logic gate ICs

2. Alternatively, we could modify the logical values before the input to the encoder IC.

Inputs									Outputs				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H[1]	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L[2]	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X[3]	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

[1] High, Logic level 1
[2] Low, logic level 0
[3] Doesn't matter

Table 3: Function table for `74148` priority encoder

The truth table above provides insight into how the encoder functions. Since this is a priority encoder, input number seven is given the highest priority. The highest priority input that is currently active is matched by the priority encoder's output. Thus, any other inputs with a lower priority will be ignored in the presence of an input with a higher priority. Using the first seven op-amp outputs as the last seven inputs to the encoder, and feeding an inverted version of the H output to the first input gives us the desired logic outputs, which can be verified from the above table.

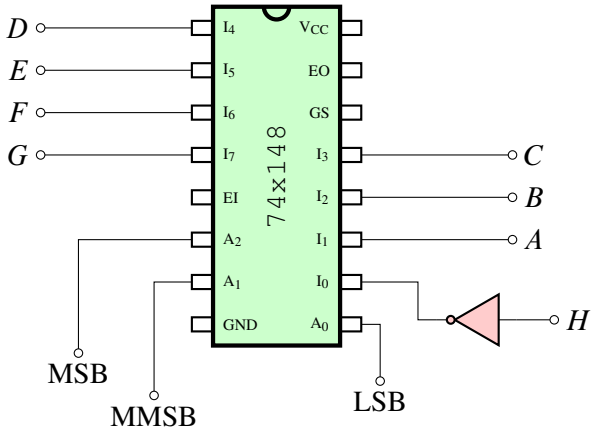

Figure 9: Modified connection for accurate encoding

3.2.4 The Serialization Block

We get three different outputs from the encoder block, which can be transmitted individually as respective bits. However, the requirement of sending three bits at a time can be relaxed by sending the bits one at a time. Serialization of parallel PCM code is an important technique in digital communication systems to reduce the complexity of data transmission and to improve overall efficiency. It is the process of converting parallel data streams into a single serial data stream, resulting in a reduction in

the number of transmission lines needed and simplifying the overall transmission process. Additionally, serialization simplifies the synchronization of data transmission, as all the parallel channels are synchronized into a single serial stream for transmission. By converting multiple parallel PCM signals into a single data stream, serialization improves the efficiency and reliability of data transmission in digital communication systems.

To implement the serialization block, we use shift registers, which is is a type of digital circuit that can store and shift data in the form of binary bits. It consists of a series of flip-flops connected in a chain, with each flip-flop storing one bit of data. By sequentially clocking the shift register, the bits can be shifted from one flip-flop to the next, effectively converting the parallel data into serial form. Specifically, in our case, the parallel data is applied to the input of the shift register. The data is then clocked into the flip-flops one bit at a time, starting from the least significant bit (LSB) to the most significant bit (MSB).

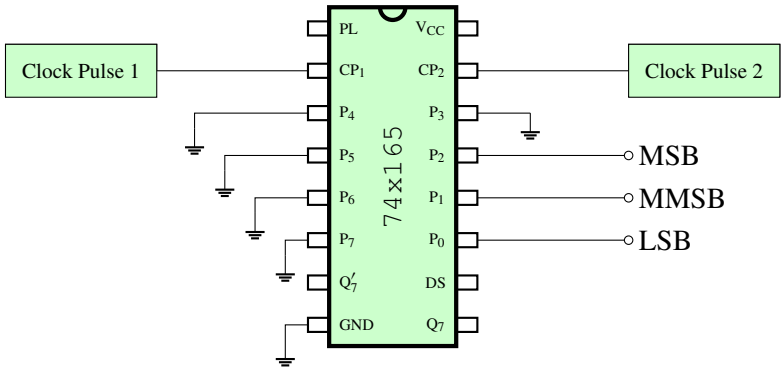

Figure 10: Connections to shift register

3.3 Software Simulation

We simulated the designed circuit in paspice:

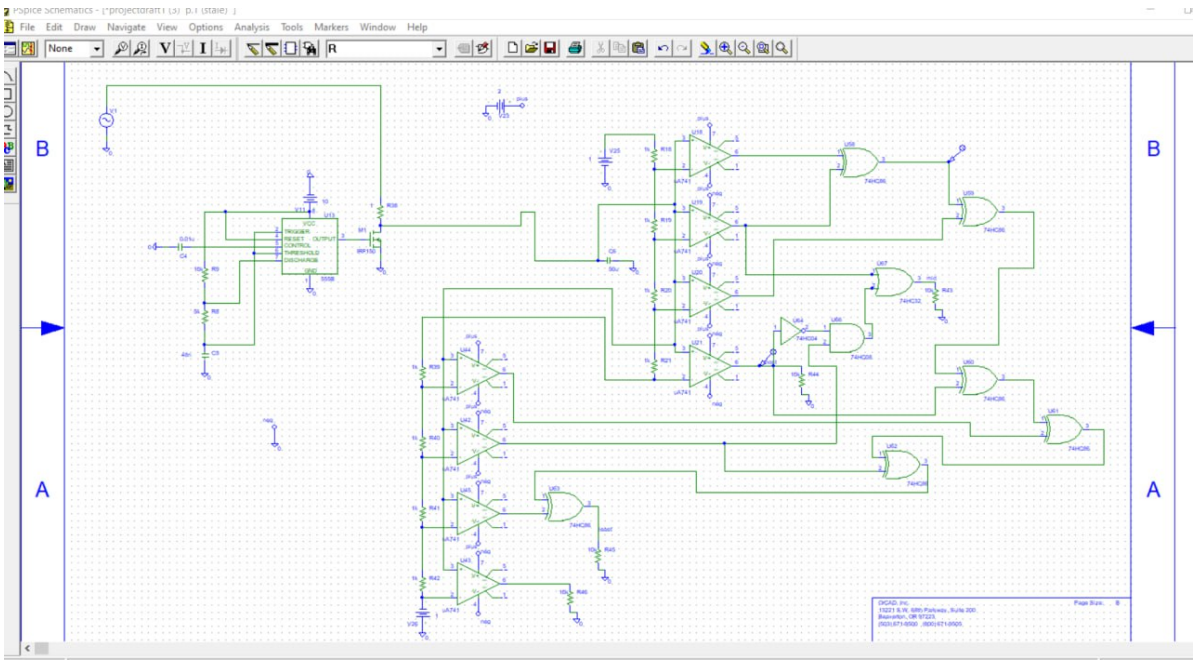

Figure 11: PSPICE Schematic

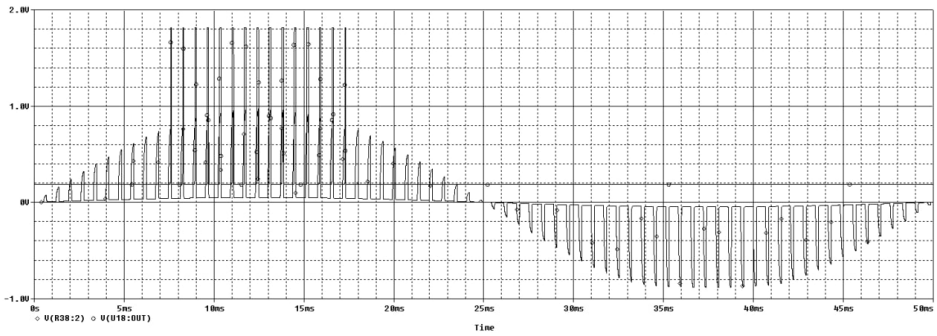

Figure 12: Output from Op-amp 1

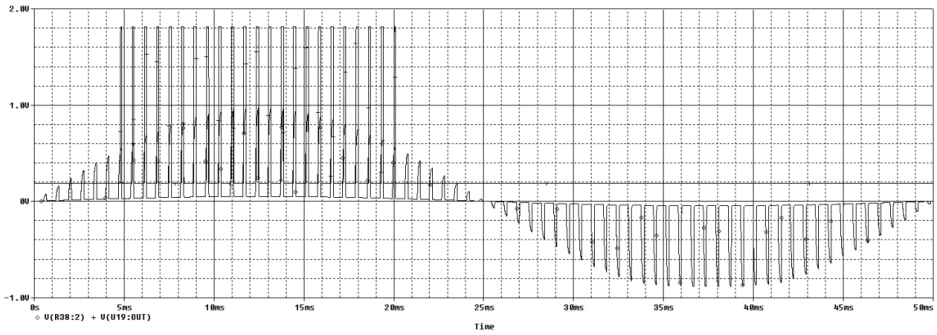

Figure 13: Output from Op-amp 2

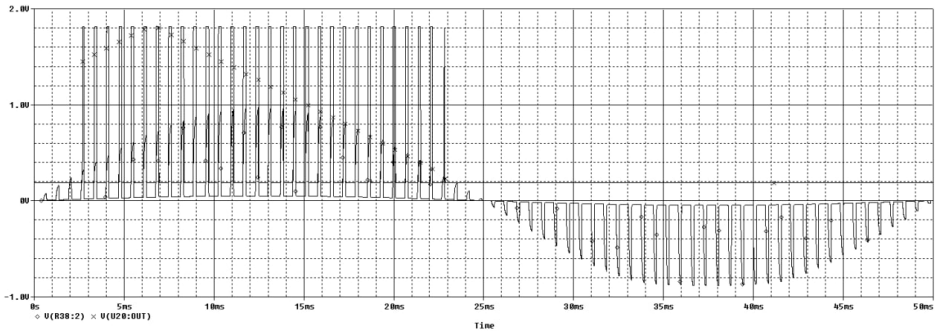

Figure 14: Output from Op-amp 3

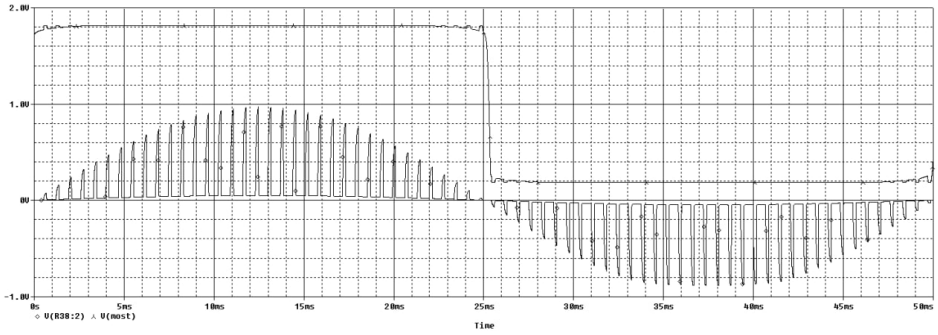

Figure 15: Output from Op-amp 4

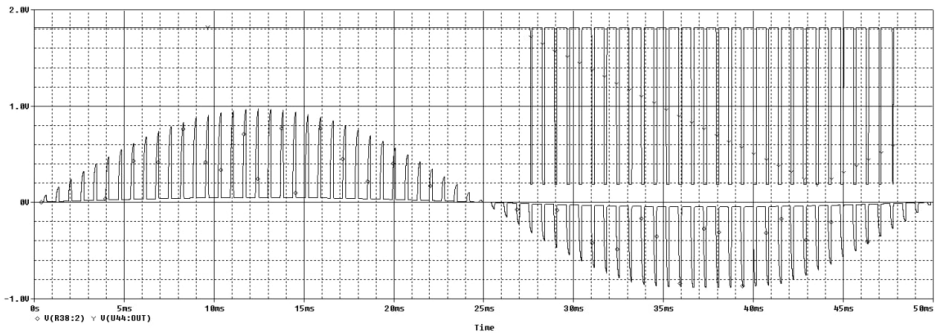

Figure 16: Output from Op-amp 5

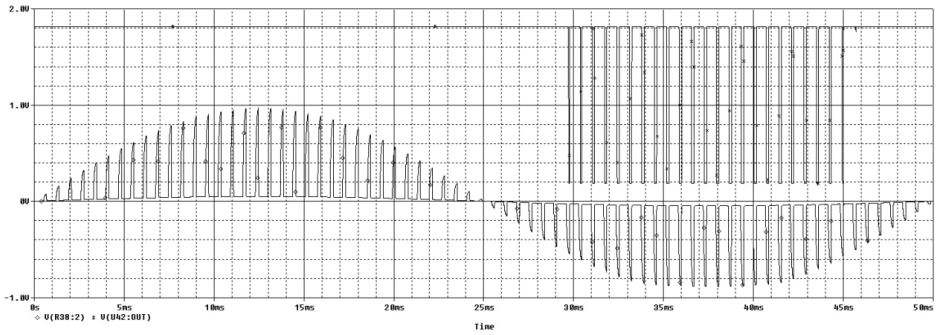

Figure 17: Output from Op-amp 6

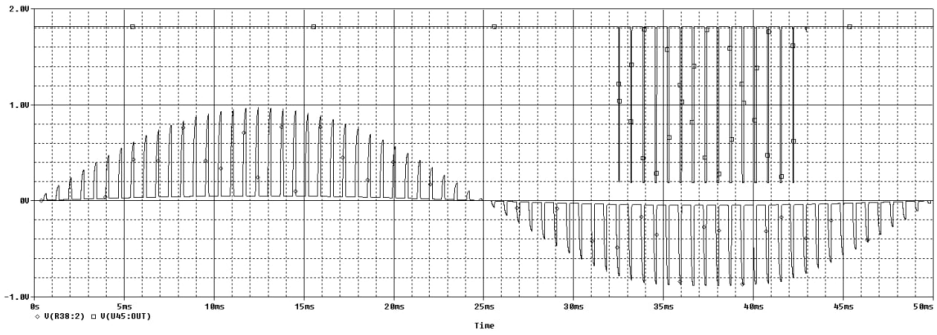

Figure 18: Output from Op-amp 7

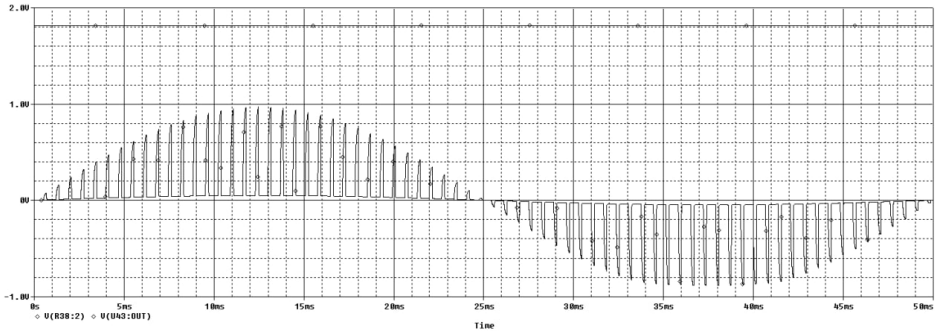

Figure 19: Output from Op-amp 8

As expected, we can see the quantization levels in action and the patterns correspond to the amount of time across a cycle that each op-amp remains in "on" state. The first op-amp is at logic value 1 only

when the message is at the highest level, and the last one is always in logic state 1. The intermediate outputs change accordingly.

As for the encoding, logic values at 3 bits have 3 outputs and an input for comparison at pspice output window, which are a bit tough to discern. For this reason, we provide the simulation results for a 2 bit scheme to verify that the encoding circuit does indeed work.

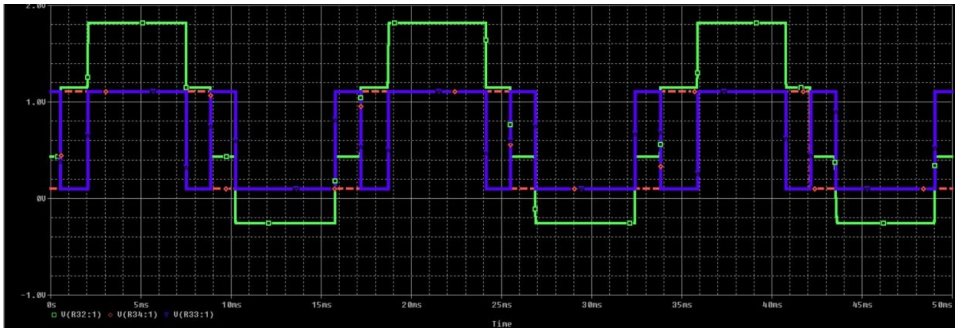

Figure 20: Encoding at 2 bits

Here, the solid green line is the quantized input signal, the solid blue line is the LSB and the dashed brown line is the MSB. It is quite clear from the figure that the bits correspond to the quatization level.

4 Implementation

4.1 Description

As with the design, we decided to build the entire circuit on four different breadboards, each breadboard serving as an independent block, and effortlessly connected to the neighboring stages. This also has the added benefit that we can conclusively test each block individually (without requiring input from the previous stage), and isolate components which do not behave expectantly, without a hitch.

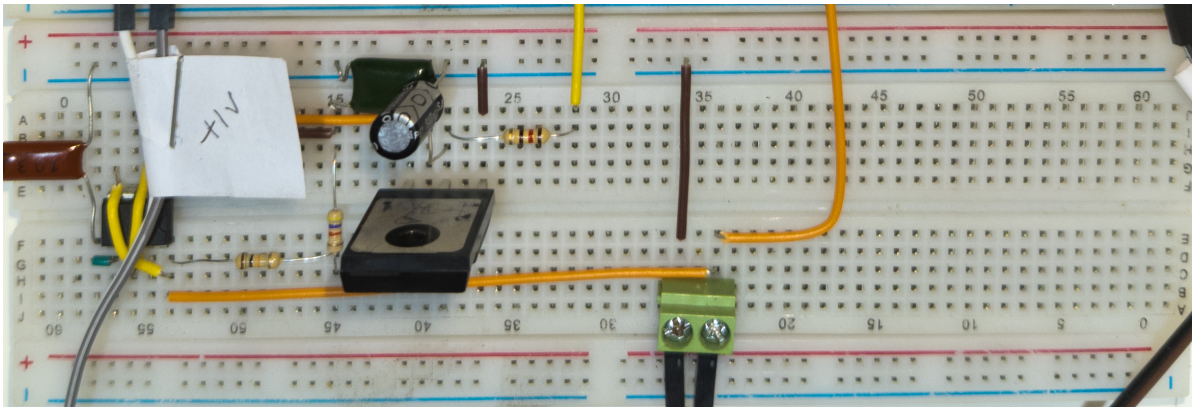

Figure 21: Sampling stage

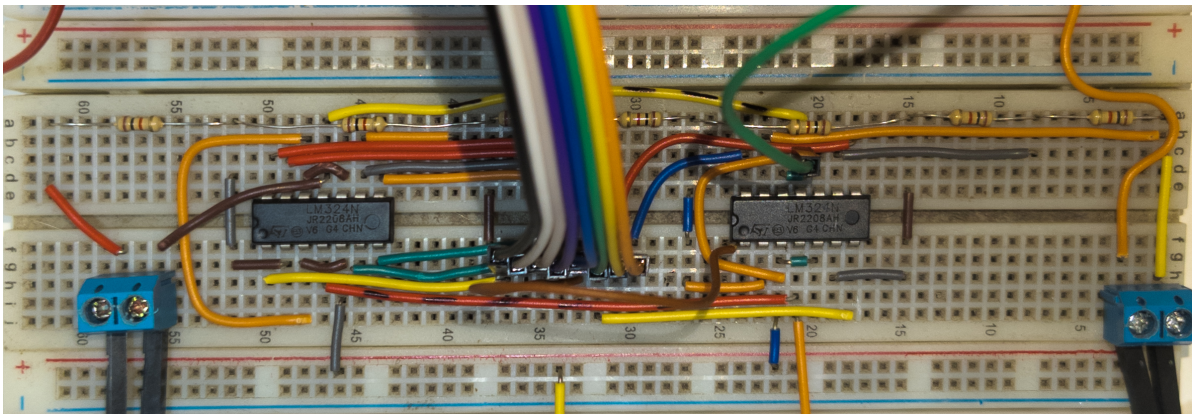

Figure 22: Quantization stage

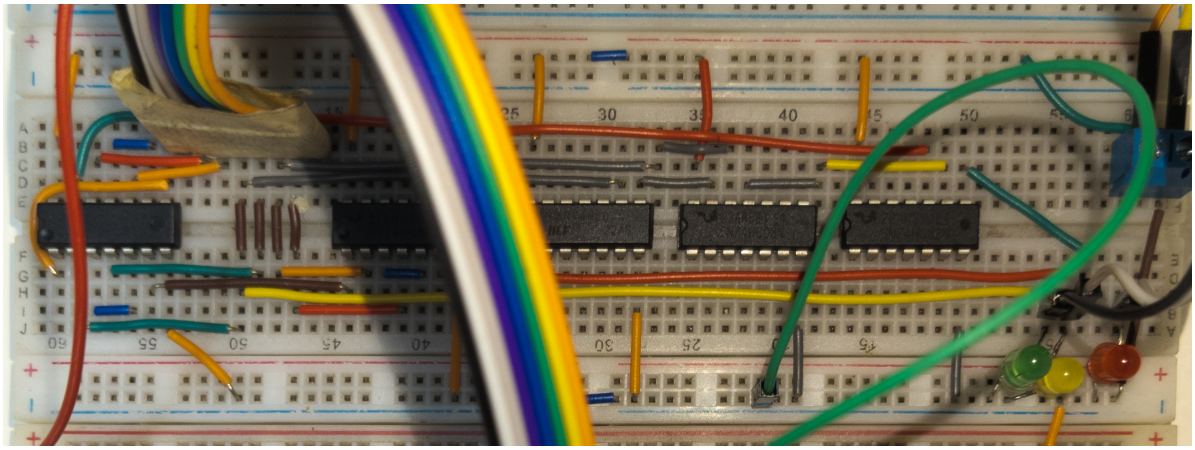

Figure 23: Encoding stage

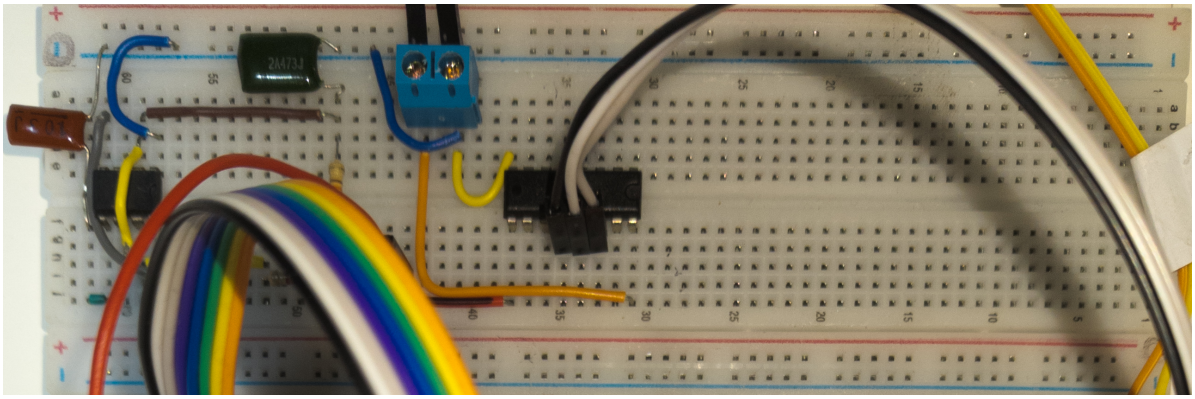

Figure 24: Serialization stage

4.2 Experiment and Data Collection

We tested each block individually. In this section, we collect the output wave-shapes and numerical data we obtained from out testing sessions conducted in the electronics laboratory.

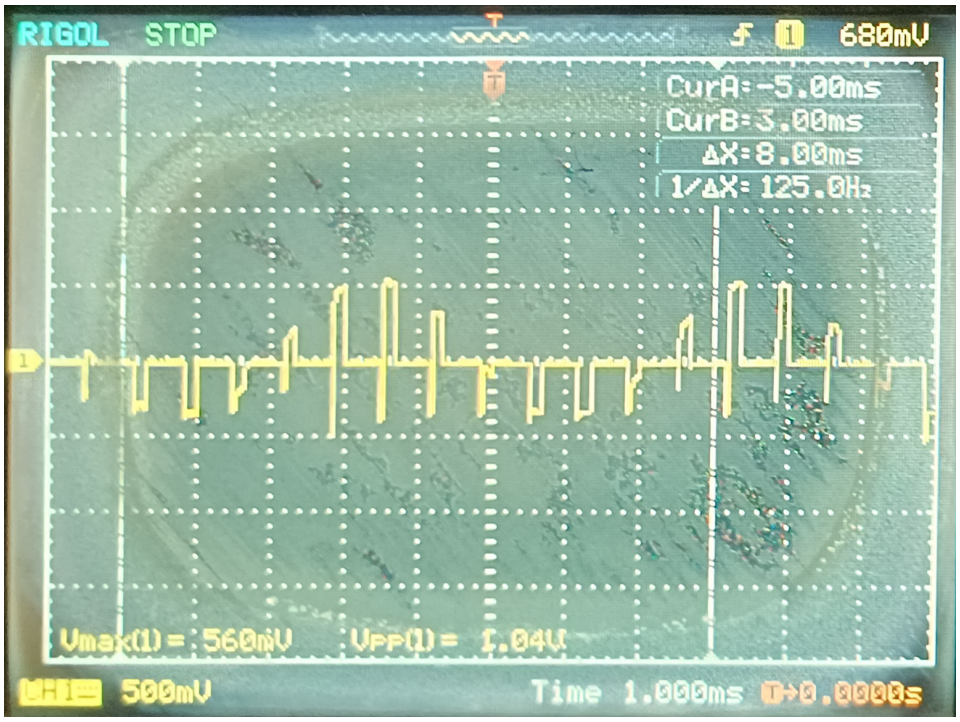

Figure 25: Output from the sampler

Testing the quantizer circuit with AC signal is rather difficult because of rapidly changing output. To overcome the challenge, we tested the circuit with DC values. We used fixed DC values as input (in place of v_{sampled}) and checked if the expected quantization level was indeed achieved. In fact, we could cascade the encoder along with the quantizer to check that the entire system works as expected. To read the encoded output, we connceted three LEDs to the three output bits. With higher logic values lighting up the LEDs, we could easily read out the PCM codes. To cross check, we used a multi-meter also, along with the above setup. Following is the data we recorded:

Applied Voltage (V)	Encoded Value	Expected Value
0	000	000
0.1	001	000
0.2	010	001
0.3	010	010
0.4	011	011
0.5	011	011
0.6	100	100
0.7	101	101
0.8	110	110
0.9	111	111

Table 4: Encoding data obtained during testing

From the table, we note that all but two values match the calculated result. Since we took utmost care while conducting the experiment, and both values surpass the expected value, this is most likely due to the power supply control being coarser than expected, and the quantization level changing before the display on the supply changed.

4.3 Results

As discussed above, the quantization and encoding scheme works properly and can output the encoded message via high and low voltage logic levels.

5 Design Analysis and Evaluation

5.1 Novelty

The most notable novelty factor in our execution of PCM line code generation is the implementation of the 8 line to 3 line encoder. Instead of simply slotting in an encoder, we implement the encoder ground up by developing truth expressions from scratch and implementing them via logic gates. Although we had a couple of solutions to the encoder problem, it is the novelty of the logic gate approach which inspired us to design the encoder. In fact, we can further refine the design and reduce the number of logic gate chips needed. As any combinations of logic gates can be expressed via an equivalent combination of NAND gates, we can modify our expression for the MMSB as:

$$B + \overline{D} \cdot F = \overline{\overline{B} \cdot \overline{\overline{D} \cdot F}}$$

The modified logic expression can be implemented via four NAND gates, i.e., only one quad 2-input NAND chip.

5.2 Limitations of Tools

One of the major constraints we faced was the inefficiency of using an 8-bit PISO (Parallel-In, Serial-Out) shift register to convert a 3-bit sequence from parallel to series. This inefficiency resulted in a waste of 5 bits, reducing the overall bandwidth efficiency of our system. This constraint not only affected the performance of our project but also limited the amount of data we could transmit effectively.

Another significant limitation we encountered was the issue of NOT gates getting burnt when we tried to change the voltage across them, even slight changes. This constraint caused setbacks in our project as we had to replace the burnt components and adjust the voltage levels to prevent further damage. This also added complexity to our project and required additional measures to be taken to ensure the proper functioning of the circuit.

One other notable problem arose in the form of loading effect. The message signal kept getting severely distorted upon connection to the sampling block, and the samples came out as a series of random impulses. We solved this by using a simple buffer made of `uA741`.

Furthermore, the malfunctioning of the 555 timers added another layer of complexity to our project. The unreliable performance of these components hindered the overall functionality of our circuit and limited our ability to achieve the desired outputs. Overcoming these constraints would have allowed for more reliable and consistent results.

Difficulties were also faced during the testing phase, due to inaccuracies in supply sources and measuring instruments like oscilloscope. The supply controls are substantially worn out from usage over time, and the same goes for the connectors and testing probes. Consequently, the power supply to our circuit was inconsistent and the wave-shapes we observed was distorted, which defeated parts of our purpose, since we failed to notice some of the finer points in the outputs.

6 Reflection on Individual and Teamwork

6.1 Individual Contributions of Each Member

Name	ID	Contributions in Project
Adib As-Ad Tonmoy	2006010	Sampling, Quantization, Serialization hardware implementation
Md Ahsan Islam	2006016	Simulation and serialization hardware implementation
Md Azim Sikder	2006022	Simulation and encoder hardware implementation
Abrar Jamil	2006023	Simulation and sampling hardware implementation
Adib Rahman	2006024	Qunantization, encoder hardware Implementation
Ashik Shahriar	2006026	Sampling and serialization hardware implementation
Kawsain Bin Salim	2006028	Simulation and Quantization hardware implementation

Table 5: Individual contributions of team members

6.2 Mode of Teamwork

Teamwork is an essential aspect of any successful project. However, teamwork can only be effective if it is done in an efficient manner. One key factor in efficient teamwork is clear communication. Team members must be able to effectively communicate with one another in order to share ideas, provide feedback, and coordinate their efforts. Without clear communication, team members may work at cross purposes and/or duplicate efforts.

In the initial stages of our project, progress was slow and we missed multiple goals we set ourselves as a team (this is apparent from the log book presented in the next subsection). This is when we realized that efficient teamwork requires a well-defined structure and process. A well-defined structure also helps to prevent confusion, duplication of efforts, and conflicts over roles and responsibilities.

We then started to establish clear procedures, timelines, and milestones so that we could stay organized, focused, and on track. We emphasized that each member understood their roles and responsibilities, as well as how their work fit into the overall project. This new outlook worked charm, seen in the progress log, as we finished building and testing the project in the span of a few days.

6.3 Log Book of Project Implementation

Notable Achievements	Date
Sampling Simulation and Circuit Construction	29/01/24
Full Circuit Simulation	15/02/24
Quantization Circuit Construction	18/02/24
Encoder Circuit Construction	20/02/24
Laboratory test of the stages above	23/02/24
PISO Block Construction	27/02/24

Table 6: Milestones

7 Communication

7.1 User Manual

User Manual

Welcome to the user manual for the PCM line code generator. First thing to know, the blocks are cascaded serially : sampling,Quantization, Encoder and PISO. Your message signal will be given as the input of the sampling Block. As natural sampling method has been used here, the expected sampling output will be seen across the resistance that has been used in series with the source and another terminal is connected with the drain of the MOSFET. Next the quantization block has been represented with 8 Op-amps. Each of the Op-amps will represent the quantization level of the sampled values. To see the levels of each sampled values,you need to use probs to the output terminals of the Op-amps. Here you will see 8bit Quantization Level. The Encoder block actually converted the 8 bit Quantization levels into 3 bits using gate combination. Here, the MSB, Mid bit and LSB bit terminals are highlighted. Using LED Bulb, you can visualise the 3bit encoded values. Lastly, the PISO block actually represents the line code generation in series. Using oscilloscope, you can see the generations of the digital data sequence of your message signal. Here as 8 bit PISO has been used,you will see 5 zero bits always and the remaining 3 bits will represent the bits of your message signal.

8 Project Management and Cost Analysis

8.1 Cost of Components

Component	Per Unit Cost	Amount	Cost
Resistors	10/=	30	300/=
Capacitor	15/=	8	120/=
555 Timer	18/=	5	90/=
N-mos (`IRF150`)	150/=	3	450/=
Op-amp (`uA741`)	80/=	3	240/=
Not Gate (`7404`)	25/=	2	50/=
AND Gate (`7408`)	25/=	1	25/=
OR Gate (`7432`)	25/=	1	25/=
XOR Gate (`7486`)	25/=	2	50/=
Connectors	3/=	100	300/=
Total			1650/=

Table 7: Component costs

8.2 Other/Miscellaneous Costs

Name	Cost
Breadboards (×4)	600/=
Transportation	220/=
Additional Testing Equipment	350/=
Total	1170/=

Table 8: Other costs

9 References

1. Lathi and Ding, Modern Digital and Analog Communication Systems
2. https://www.tutorialspoint.com/digital_communication/digital_communication_pulse_code_modulation.htm
3. https://www.electronics-tutorials.ws/opamp/op-amp-comparator.html