Implementation of Different Encoding and Decoding Schemes

Md.Abu Bakar Siddique, Roll: **47**

1 Introduction

Encoding and decoding are fundamental processes in digital communication systems, converting data between formats suitable for transmission and interpretation. This report shows six key schemes:

- **NRZ-L (Non-Return-to-Zero Level)**: Maintains voltage level throughout bit duration.
- **NRZ-I (Non-Return-to-Zero Inverted)**: Inverts voltage on '1' bits.
- **Manchester**: Uses mid-bit transitions for clock synchronization.
- **AMI (Alternate Mark Inversion)**: Alternates pulse polarity for '1's.
- **Pseudo-ternary**: Encodes '0's with alternating pulses.
- **MLT-3 (Multi-Level Transmission-3)**: Cycles through three voltage levels.

These schemes enable reliable data transmission across various media while optimizing bandwidth and power efficiency.

2 Objectives

1. Implement six digital encoding and decoding schemes.
2. Analyze signal characteristics and synchronization mechanisms.
3. Validate through input-output waveform analysis.
4. Identify practical implementation challenges.

3 Algorithms / Pseudocode

NRZ-L Encoding

- Begin with an empty list for storing the output signal.
- Traverse each bit in the binary input:

  - Add a low signal for '0'.
  - Add a high signal for '1'.
- Output the resulting encoded signal.

NRZ-L Decoding

- Initialize an empty list for the decoded binary values.
- Go through each level in the encoded signal:
  - If it's a low level, append '0'.
  - If it's a high level, append '1'.
- Return the reconstructed binary sequence.

NRZ-I Encoding

- Start with an empty list for the encoded result and assume the initial signal level is high.
- Loop through the input bits:
  - On encountering a '1', toggle the signal and append the new level.
  - On a '0', maintain and append the current level.
- Output the full encoded sequence.

NRZ-I Decoding

- Create an empty list for storing decoded values.
- Assume an initial signal level of high.
- Iterate over the encoded values:
  - If the level has changed compared to the previous, append '1'.
  - If there's no change, append '0'.
- Provide the decoded output.

Manchester Encoding

- Initialize an empty list to hold the encoded waveform.
- Process each input bit:
  - For '0', generate a transition from low to high.
  - For '1', create a high to low transition.
- Output the encoded Manchester signal.

Manchester Decoding

- Begin with an empty list for decoding.
- For each encoded pair or bit transition:
  - If the transition is from low to high midway, it's a '0'.
  - If it switches from high to low, it's a '1'.
- Return the decoded binary stream.

AMI Encoding

- Set up an empty list for the encoded result and a counter initialized to zero.
- For every bit in the input:
  - Add '0' for a binary '0'.
  - For '1', alternate between +1 and -1 depending on the counter's parity, then increment the counter.
- Provide the final encoded signal.

AMI Decoding

- Start with an empty list to store the decoded data.
- Read each level from the encoded stream:
  - Append '1' for either +1 or -1.
  - Append '0' for zero level.
- Output the decoded binary string.

Pseudoternary Encoding

- Create an empty list for the encoded signal and initialize a counter to zero.
- For every bit in the binary input:
  - Add a zero for '1'.
  - For '0', alternate between +1 and -1 depending on whether the counter is even or odd, then increase the counter.
- Output the completed encoded list.

Pseudoternary Decoding

- Start with an empty list for decoded output.
- Go through each encoded signal level:
  - If the value is +1 or -1, it's a '0'.
  - If the level is zero, it represents a '1'.
- Return the decoded sequence.

MLT-3 Encoding

- Prepare an empty list to store the signal.
- Initialize `prev_state` to 0 and `level` to 1.
- For each input bit:
  - If it's '0', keep `prev_state` as is.
  - If it's '1', transition through a cycle: $0 \rightarrow +1 \rightarrow 0 \rightarrow -1 \rightarrow 0$, adjusting `level` and updating `prev_state`.
  - Append the `prev_state` to the encoded output.
- Return the full MLT-3 signal.

MLT-3 Decoding

- Initialize a list for the decoded bits with a starting value of '1'.
- Traverse through the encoded signal:
  - If there's no level change compared to the previous bit, append '0'.
  - If there is a transition, append '1'.
- Return the recovered binary stream.

4 Sample Input/Output

NRZ-L Encoding/Decoding

- **Input Stream**: [1, 0, 1, 1, 1, 0, 0, 1]
- **Encoded Input Stream**: [1, 0, 1, 1, 1, 0, 0, 1]
- **Decoded Stream**: [1, 0, 1, 1, 1, 0, 0, 1]

NRZ-I Encoding/Decoding

- **Input Stream**: [1, 0, 1, 1, 1, 0, 0, 1]
- **The previous state is assumed to be 1.**
- **Encoded Input Stream**: [0, 0, 1, 0, 1, 1, 1, 0]
- **Decoded Stream**: [1, 0, 1, 1, 1, 0, 0, 1]

Manchester Encoding/Decoding

- **Input Stream**: [1, 0, 1, 1, 1, 0, 0, 1]
- **Encoded Input Stream**: [0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0]
- **Decoded Stream**: [1, 0, 1, 1, 1, 0, 0, 1]

AMI Encoding/Decoding

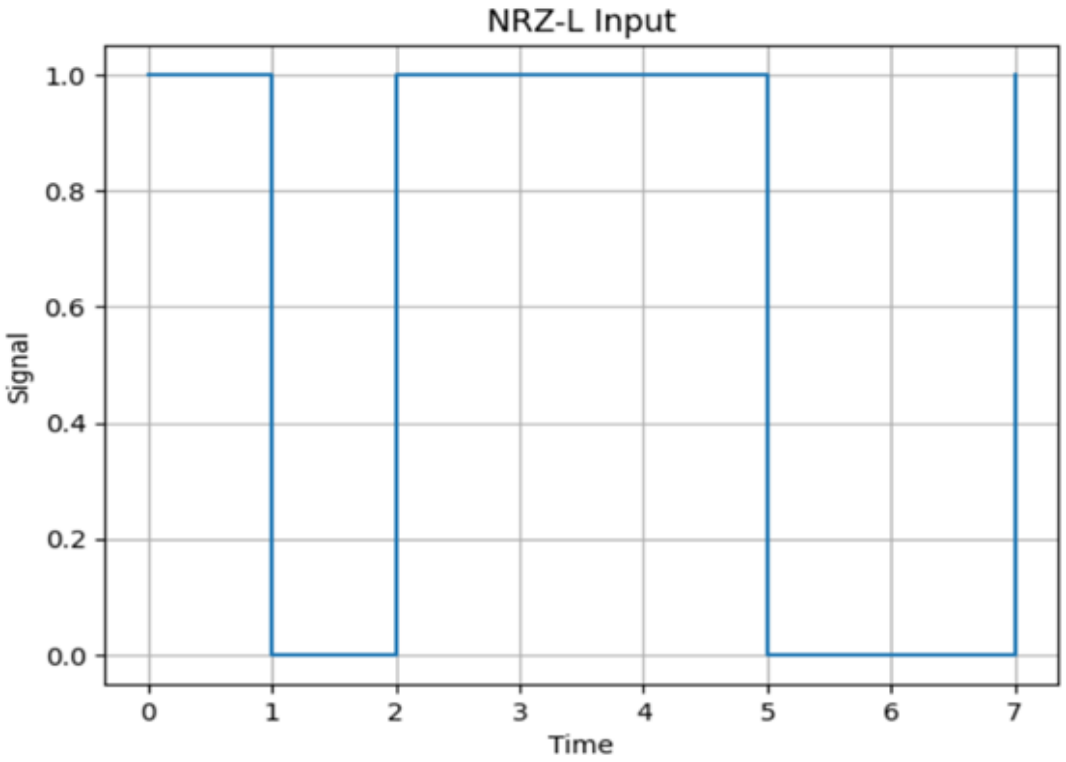
- **Bits**: 10100111001
- **Input Stream**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1]
- **Encoded Input Stream**: [1, 0, -1, 0, 0, 1, -1, 1, 0, 0, -1]
- **Decoded Stream**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1]

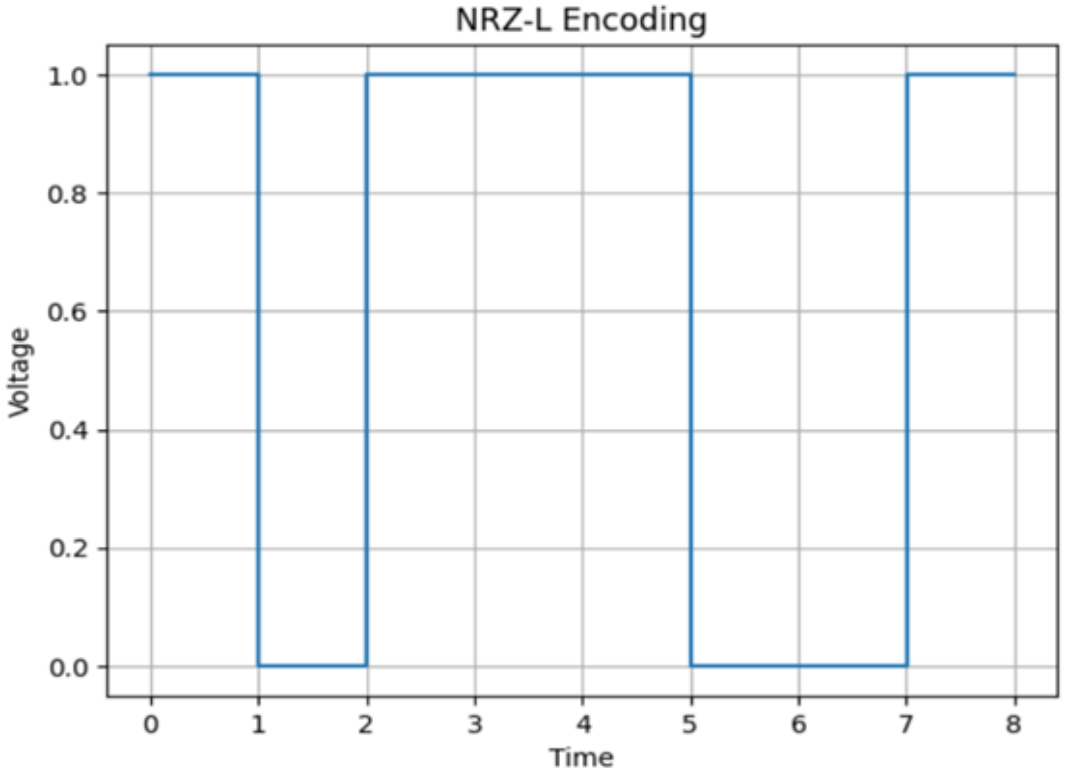
Pseudoternary Encoding/Decoding

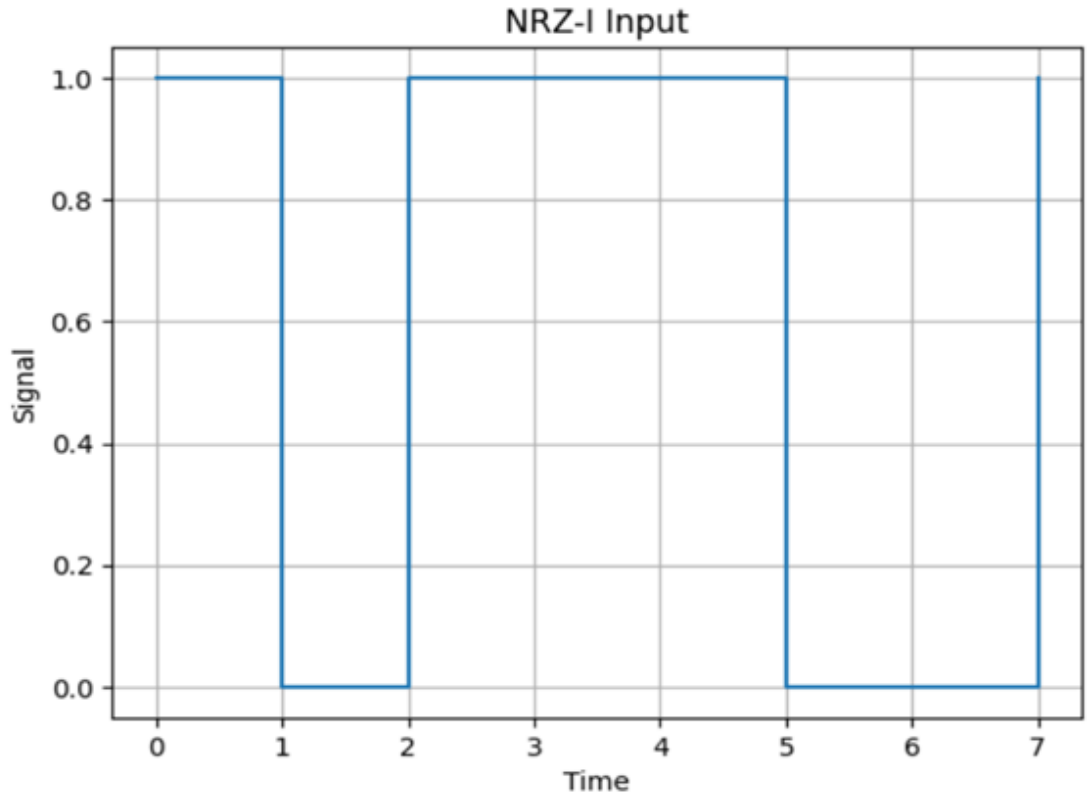
- **Bits**: 10100111001
- **Input Stream**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1]
- **Encoded Input Stream**: [0, -1, 0, 1, -1, 0, 0, 0, 1, -1, 0]
- **Decoded Stream**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1]

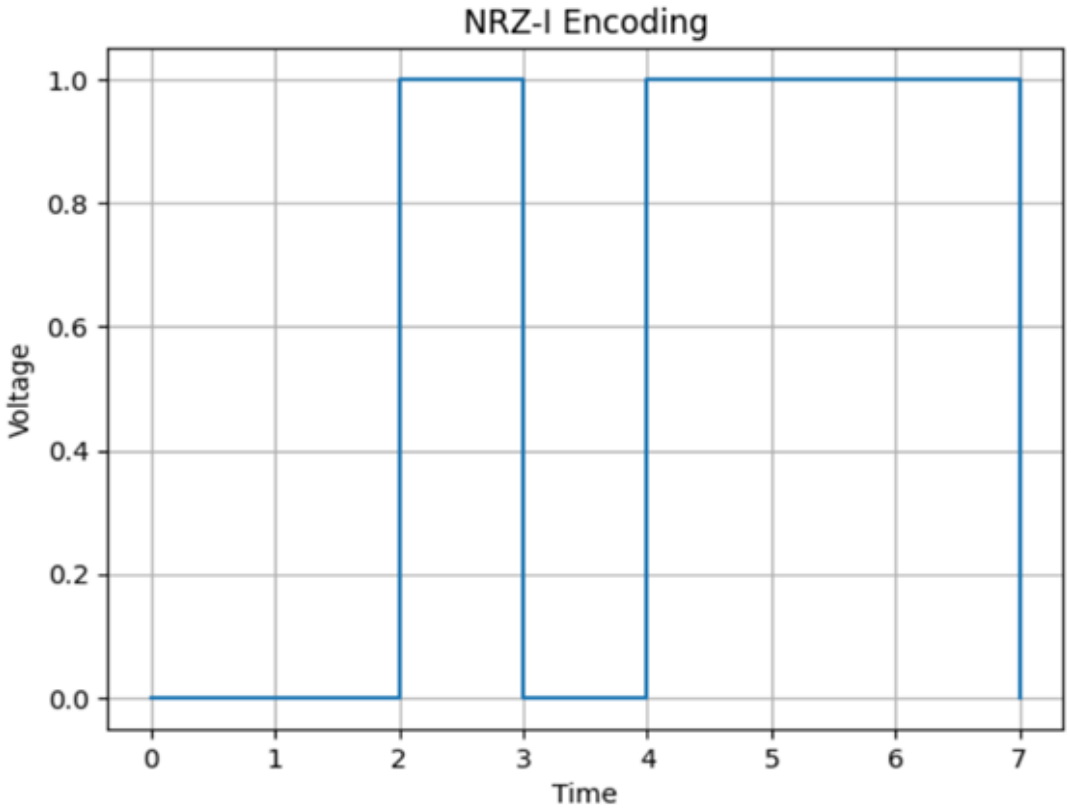
MLT-3 Encoding/Decoding

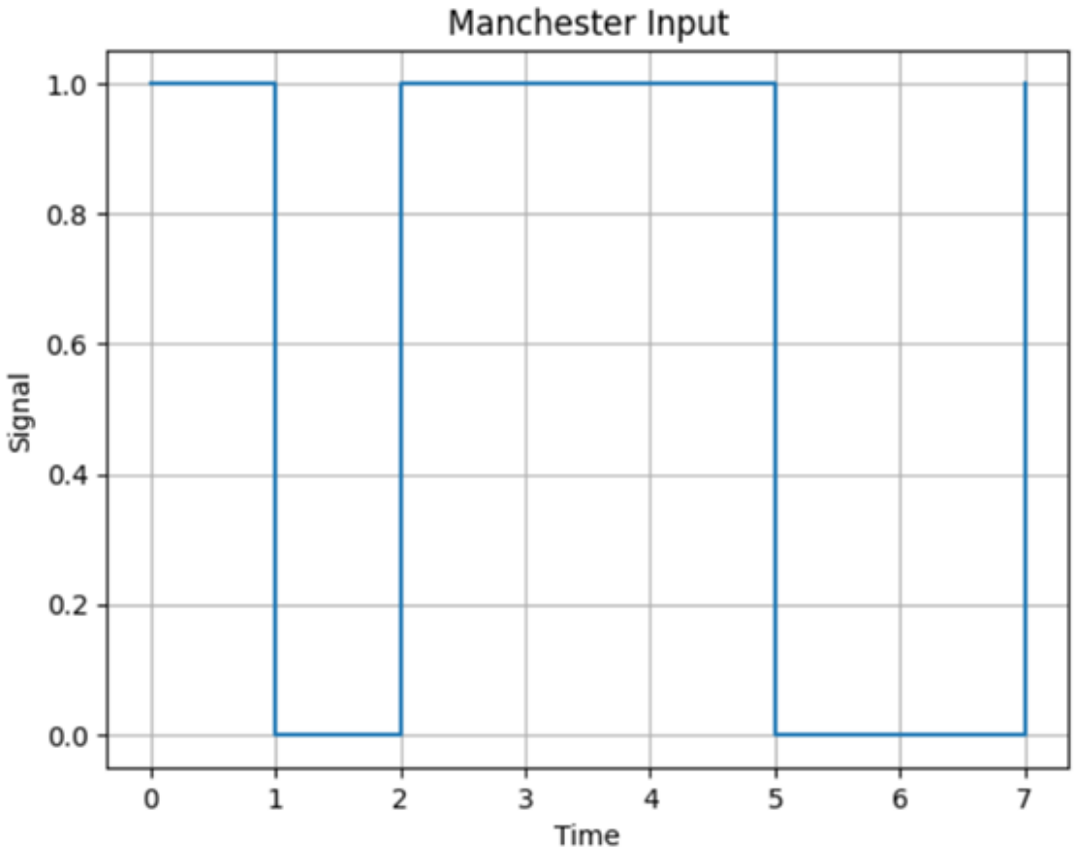
- **Bits**: 10100111001
- **Input Stream**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1]
- **Encoded Input Stream**: [1, 1, 0, 0, 0, -1, 0, 1, 1, 1, 0]
- **Decoded Stream**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1]

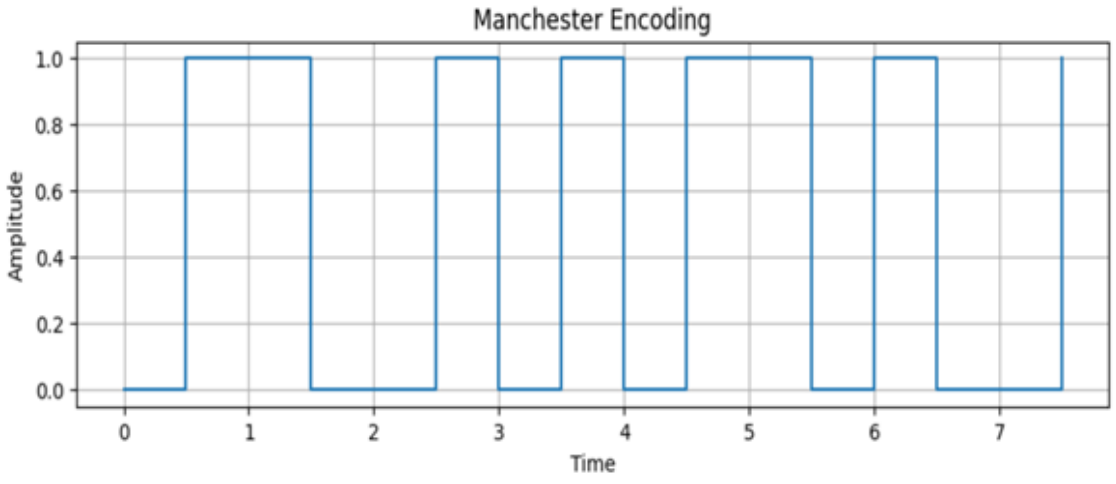

NRZ-L Input
Signal
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7

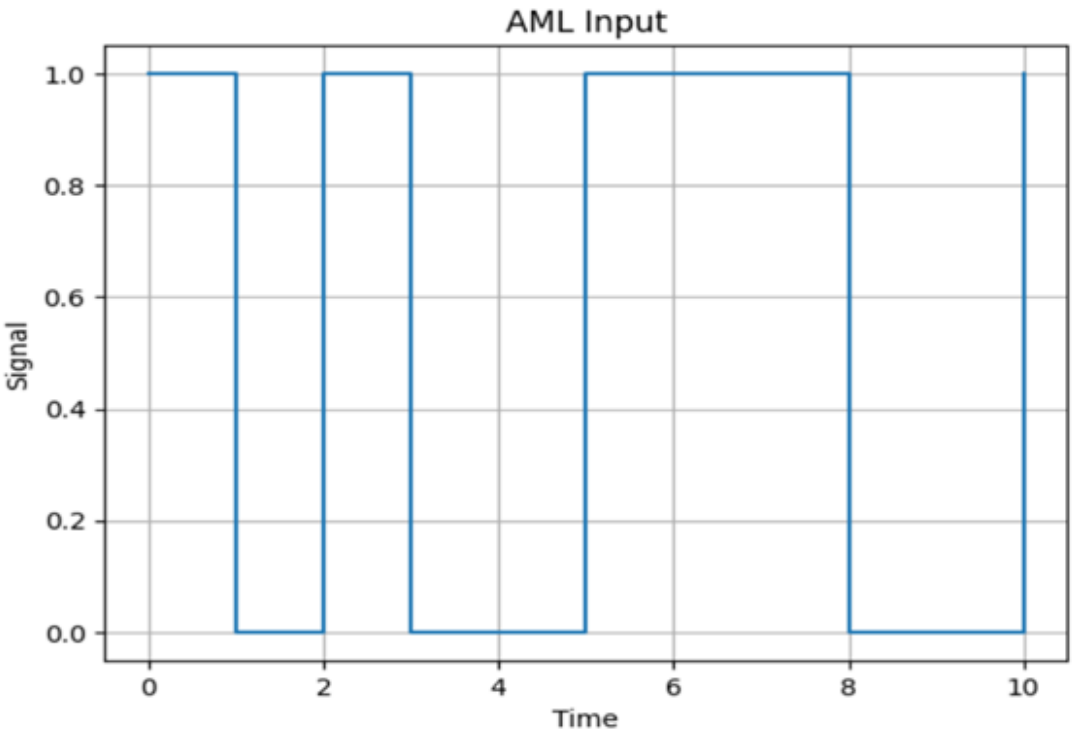

NRZ-L Encoding
Voltage
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
8

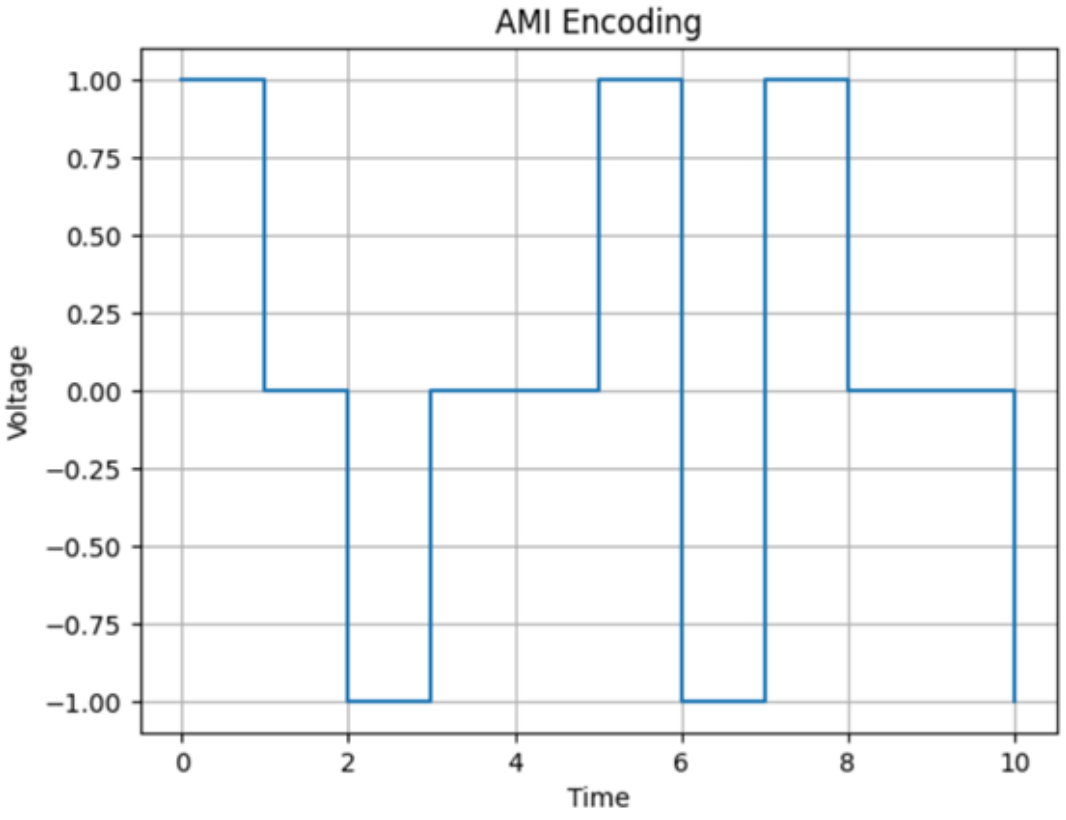

NRZ-I Input
Signal
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7

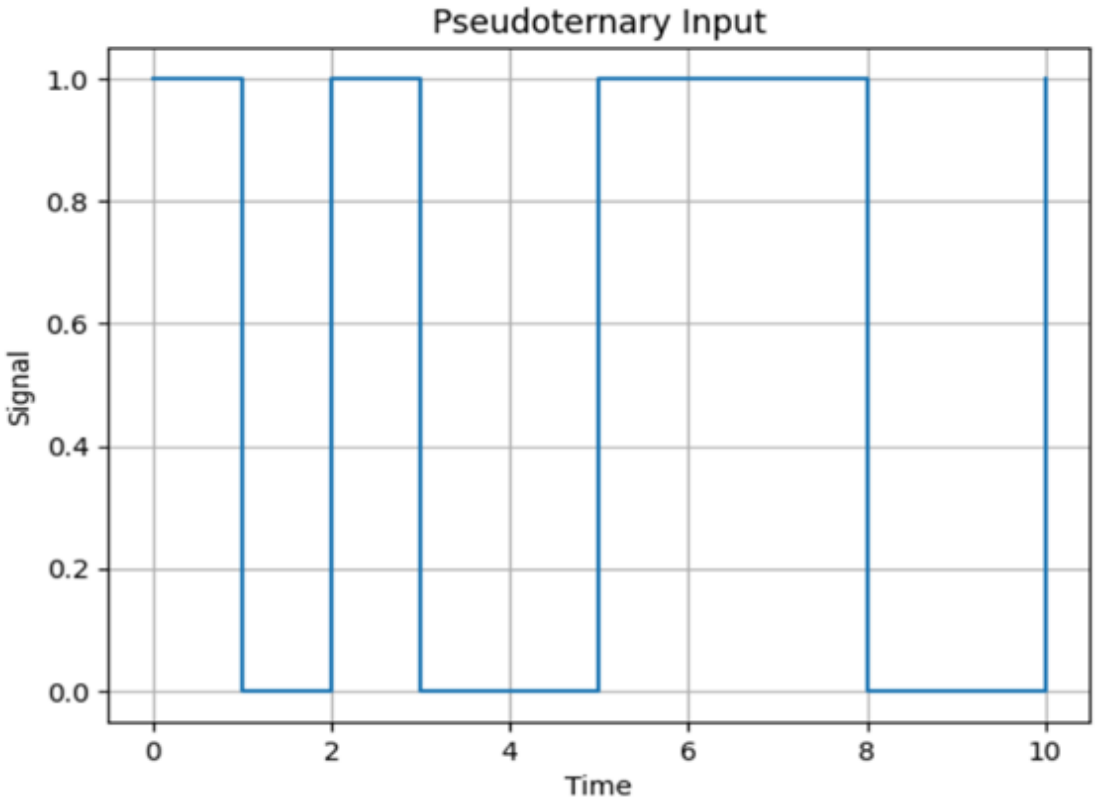

NRZ-I Encoding
Voltage
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7

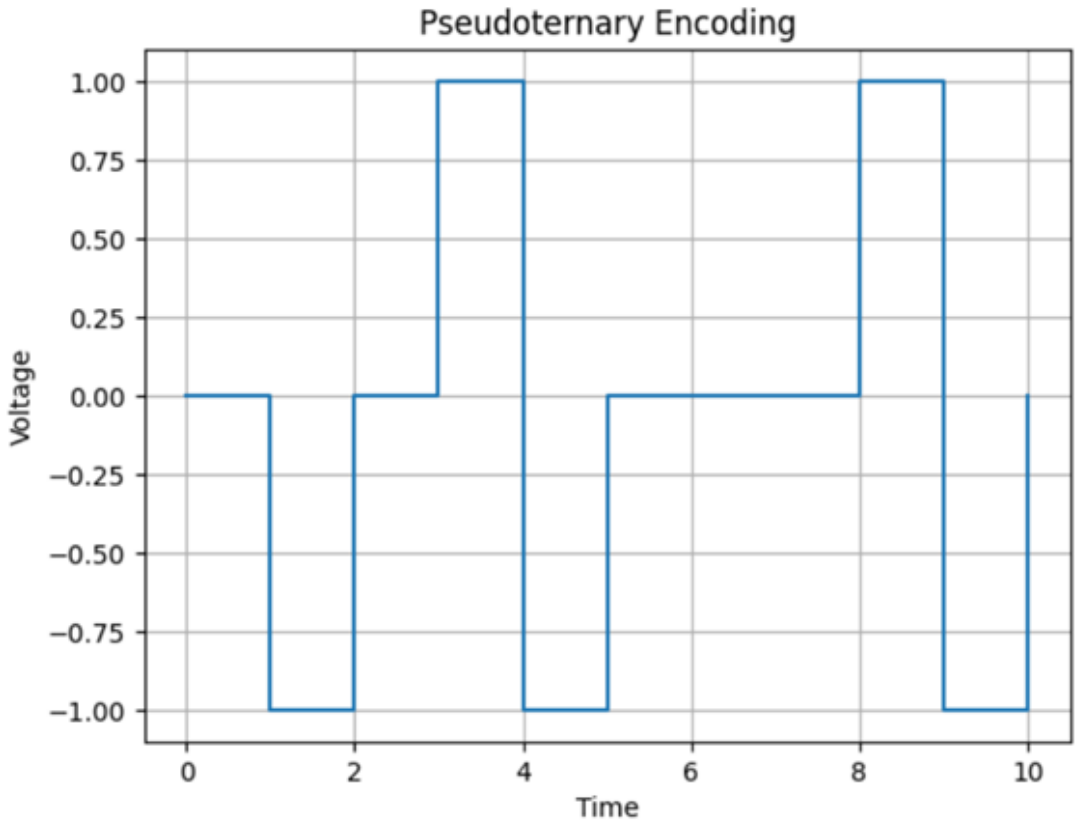

Manchester Input
Signal
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7

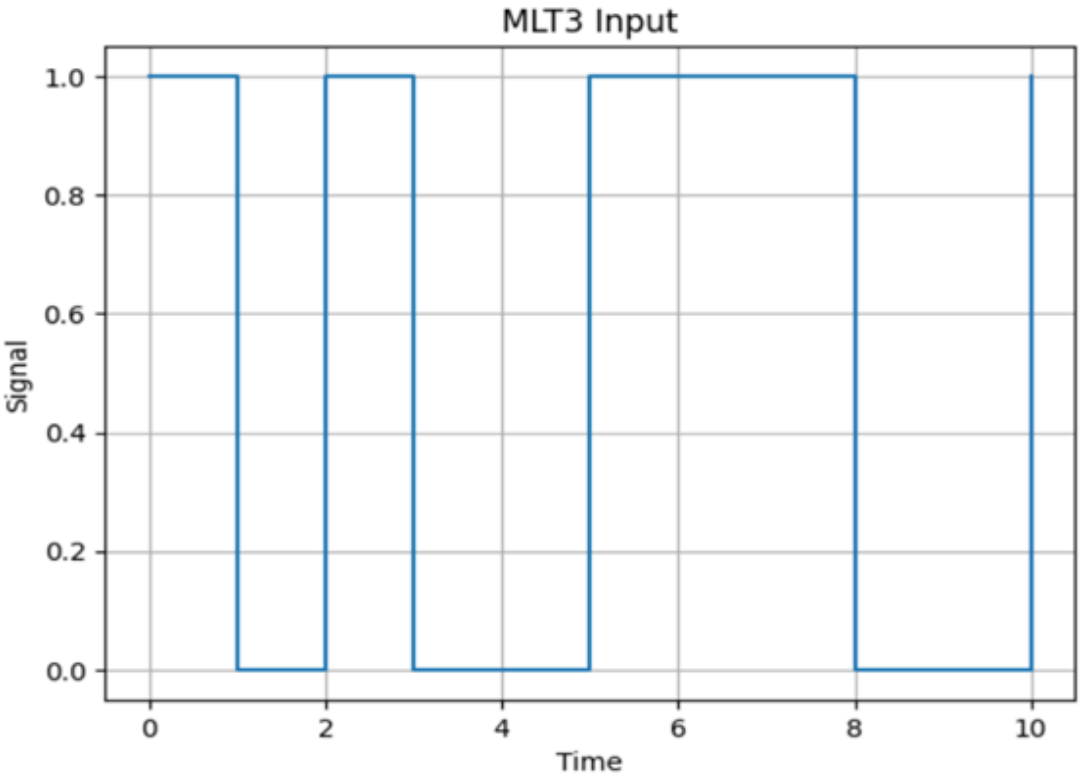

Manchester Encoding
Amplitude
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7

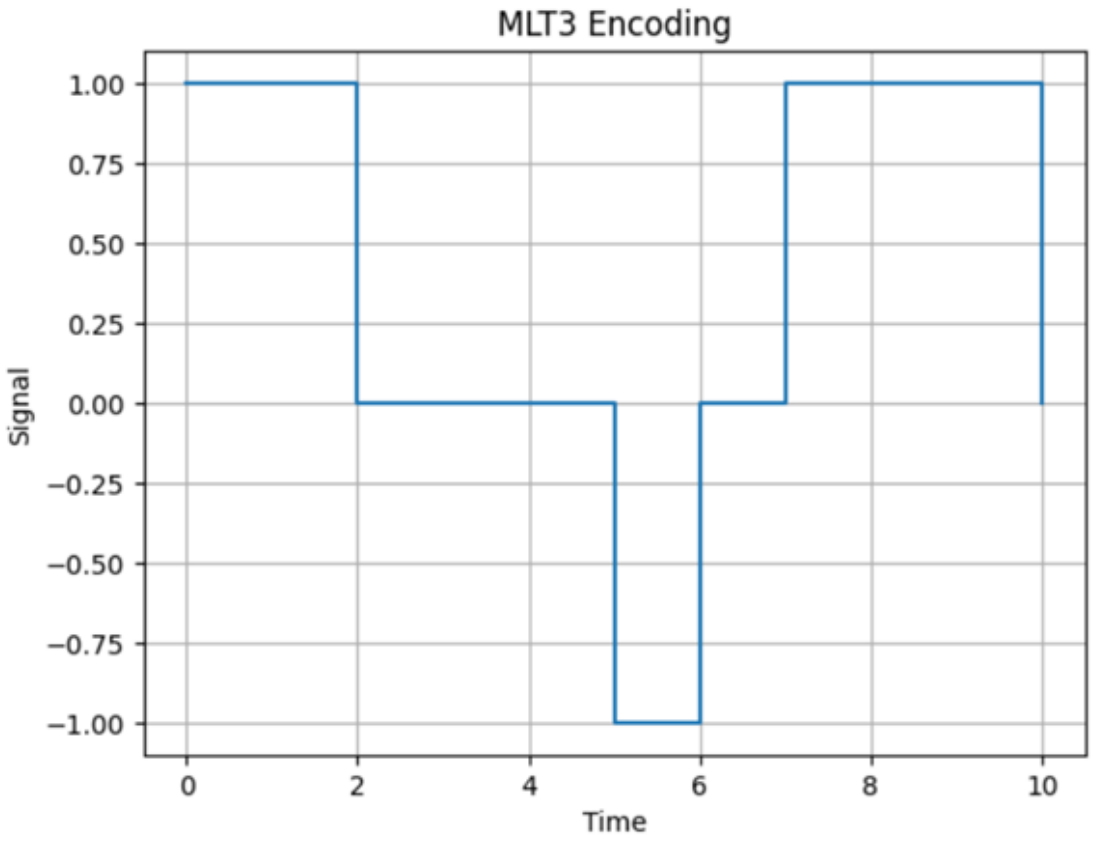

AML Input
Signal
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Time


AMI Encoding
Voltage
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
Time


Pseudoternary Input
Signal
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Time


Pseudoternary Encoding
Voltage
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
Time


MLT3 Input
Signal
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Time


MLT3 Encoding
Signal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
Time

5 Learning and Difficulties

- Ensuring proper state transitions in schemes like NRZ-I and MLT-3 is critical, as even minor logic mistakes can lead to incorrect decoding results.
- Interpreting signal transitions in Manchester encoding can be challenging, particularly when identifying the precise mid-bit transition point and distinguishing it from general signal levels.
- Managing polarity shifts in bipolar encoding methods such as AMI and Pseudoternary requires attention, as the alternating pattern must be maintained consistently to preserve decoding accuracy.
- A common challenge is resolving discrepancies between the original input and the decoded output; this necessitates thorough verification of the implemented encoding and decoding procedures.

6 Conclusion

This lab provided hands-on experience with various encoding and decoding techniques, essential in digital communication. Each encoding scheme has distinct properties that make it suitable for specific transmission environments. Understanding these schemes enables better system design and selection based on performance needs and channel characteristics.