Name : Aditya Udhav Suryawanshi

Div: TY\_CS\_D

Roll No : 49

Prn\_No:12111453

Subject : Computer Networks

Java Code :

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

public class CNASS1 extends JFrame implements ActionListener {

public int x1 = 300, x2 = 300, y1 = 100, y2 = 100;

int[] baud;

JTextField npData = new JTextField(12);

JComboBox<String> techniques = null;

JPanel drawPanel = new JPanel();

public CNASS1() {

setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);

setSize(700, 400);

setTitle("Line Techniques");

initComponents();

}

private void initComponents() {

JPanel topPanel = new JPanel();

topPanel.add(new JLabel("Input Data:"));

topPanel.add(npData);

String[] list = {

"Unipolar NRZ",

"Unipolar RZ",

"Polar NRZ",

"Polar RZ",

"Bipolar NRZ",

"Bipolar RZ",

"Manchester [Thomas]",

"Manchester [IEEE]",

"Differential Manchester"

};

techniques = new JComboBox<String>(list);

topPanel.add(new JLabel("Technique: "));

JButton encodeBtn = new JButton("Encode");

JButton clrBtn = new JButton("Clear");

topPanel.add(techniques);

topPanel.add(encodeBtn);

encodeBtn.addActionListener(this);

topPanel.add(clrBtn);

clrBtn.addActionListener(this);

drawPanel.setBackground(Color.*WHITE*);

add(drawPanel);

add(topPanel, BorderLayout.*NORTH*);

}

public static void main(String[] args) {

SwingUtilities.*invokeLater*(() -> new CNASS1().setVisible(true));

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("Encode"))

encode();

else if (e.getActionCommand().equals("Clear"))

repaint();

}

private void encode() {

drawPanel.getGraphics().drawString(techniques.getSelectedItem() + " Encoding", 400, 40);

char[] data = npData.getText().toCharArray();

int loop = 1;

boolean flag = true;

switch (techniques.getSelectedItem().toString()) {

case "Unipolar NRZ":

baud = new int[data.length + 2];

baud[0] = 0;

baud[baud.length - 1] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1')

baud[loop] = 50;

else

baud[loop] = 0;

loop++;

}

draw(baud);

break;

case "Unipolar RZ":

baud = new int[(data.length \* 2) + 1];

baud[0] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

baud[loop] = 50;

baud[loop + 1] = 0;

loop += 2;

} else {

baud[loop] = 0;

baud[loop + 1] = 0;

loop += 2;

}

}

draw(baud);

break;

case "Bipolar NRZ": // Added "Bipolar NRZ" case

baud = new int[data.length + 2];

baud[0] = 0;

baud[baud.length - 1] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

if (flag) {

baud[loop] = 50;

flag = false;

} else {

baud[loop] = -50;

flag = true;

}

} else {

baud[loop] = 0;

}

loop++;

}

draw(baud);

break;

case "Bipolar RZ": // Added "Bipolar RZ" case

baud = new int[(data.length \* 2) + 1];

baud[0] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

if (flag) {

baud[loop] = 50;

baud[loop + 1] = 0;

flag = false;

} else {

baud[loop] = -50;

baud[loop + 1] = 0;

flag = true;

}

loop += 2;

} else {

baud[loop] = 0;

baud[loop + 1] = 0;

loop += 2;

}

}

draw(baud);

break;

case "Polar NRZ":

baud = new int[data.length + 2];

baud[0] = 0;

baud[baud.length - 1] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1')

baud[loop] = 50;

else

baud[loop] = -50;

loop++;

}

draw(baud);

break;

case "Polar RZ":

baud = new int[(data.length \* 2) + 1];

baud[0] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

baud[loop] = 50;

baud[loop + 1] = 0;

loop += 2;

} else {

baud[loop] = -50;

baud[loop + 1] = 0;

loop += 2;

}

}

draw(baud);

break;

case "Manchester [Thomas]":

baud = new int[(data.length \* 2) + 2];

baud[0] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

baud[loop] = 50;

baud[loop + 1] = -50;

} else {

baud[loop] = -50;

baud[loop + 1] = 50;

}

loop += 2;

}

draw(baud);

break;

case "Manchester [IEEE]":

baud = new int[(data.length \* 2) + 2];

baud[0] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

baud[loop] = -50;

baud[loop + 1] = 50;

} else {

baud[loop] = 50;

baud[loop + 1] = -50;

}

loop += 2;

}

draw(baud);

break;

case "Differential Manchester":

baud = new int[(data.length \* 2) + 2];

baud[0] = 0;

for (int i = 0; i < data.length; i++) {

if (data[i] == '1') {

if (flag && baud[loop - 1] != -50) {

baud[loop] = 50;

baud[loop + 1] = -50;

flag = false;

} else {

baud[loop] = -50;

baud[loop + 1] = 50;

flag = true;

}

} else {

baud[loop] = 50;

baud[loop + 1] = -50;

}

loop += 2;

}

draw(baud);

break;

}

}

private void draw(int[] baud) {

for (int i = 0; i < baud.length; i++) {

if (baud[i] == 100 - y1) {

y2 = y1;

x2 = x1 + 30;

drawPanel.getGraphics().drawLine(x1, y1, x2, y2);

} else {

x2 = x1;

y2 = 100 - baud[i];

drawPanel.getGraphics().drawLine(x1, y1, x2, y2);

}

y1 = y2;

x2 = x1 + 30;

drawPanel.getGraphics().drawLine(x1, y1, x2, y2);

x1 = x2;

y1 = y2;

}

x1 = 300;

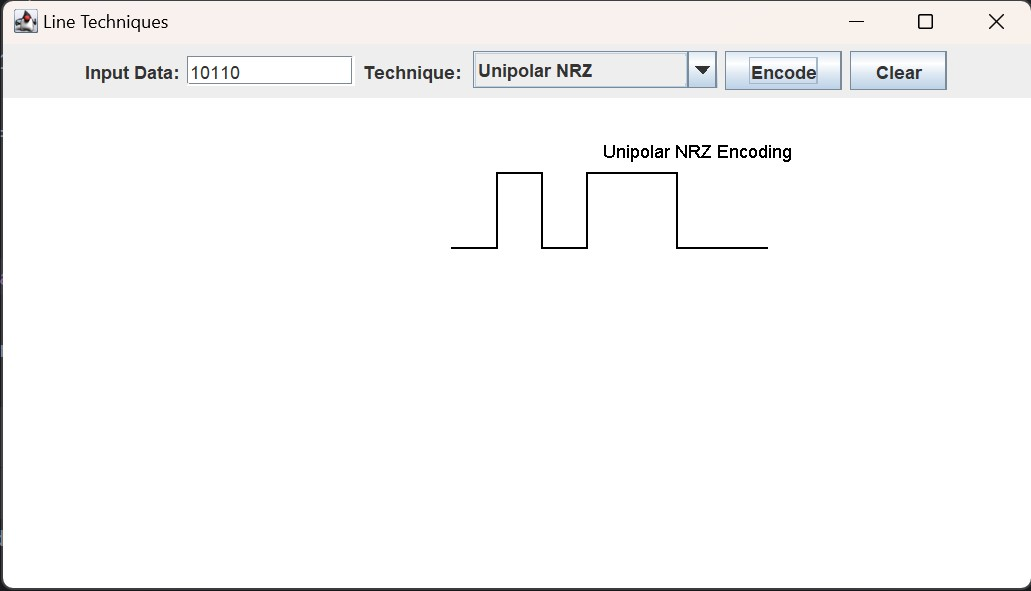
y1 = 100;

}

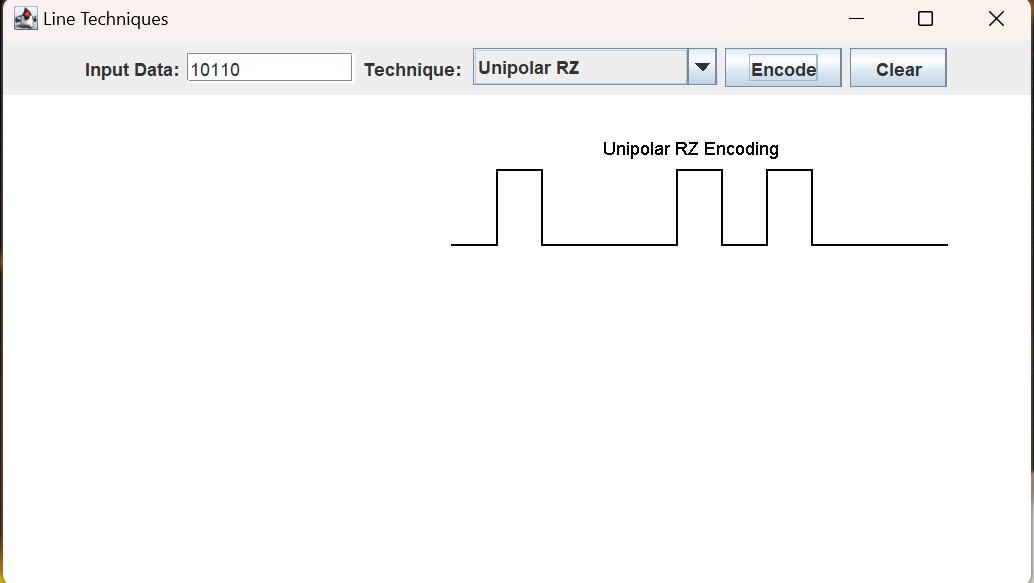
}

Output :

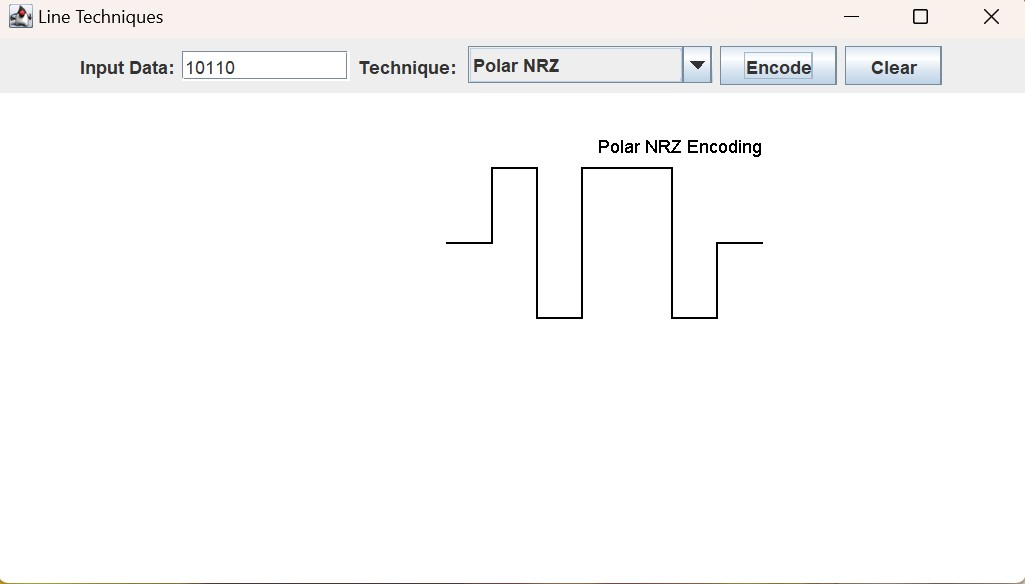
1. Unipolar NRZ



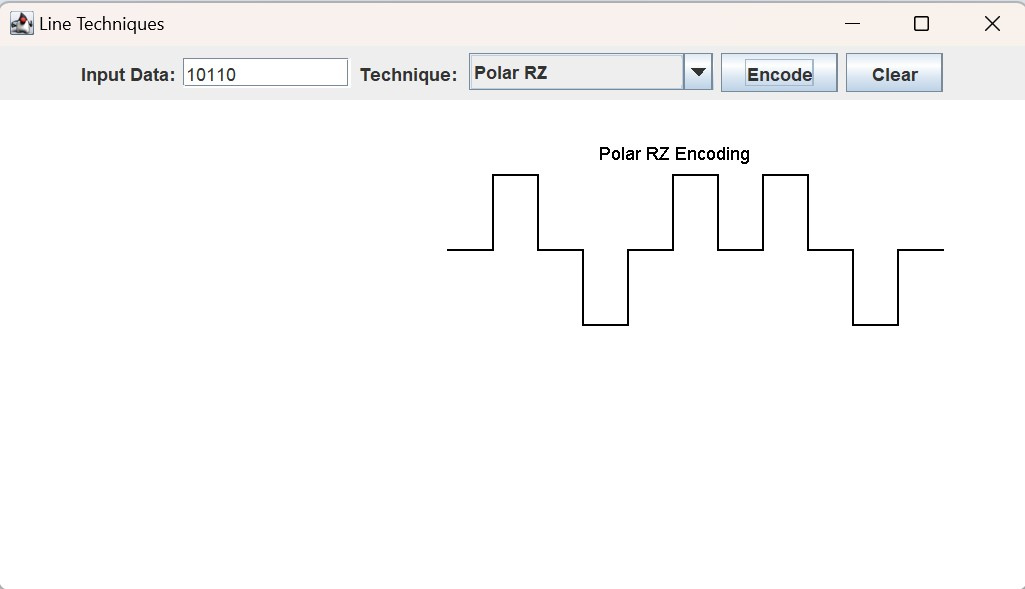
2.Unipolar RZ



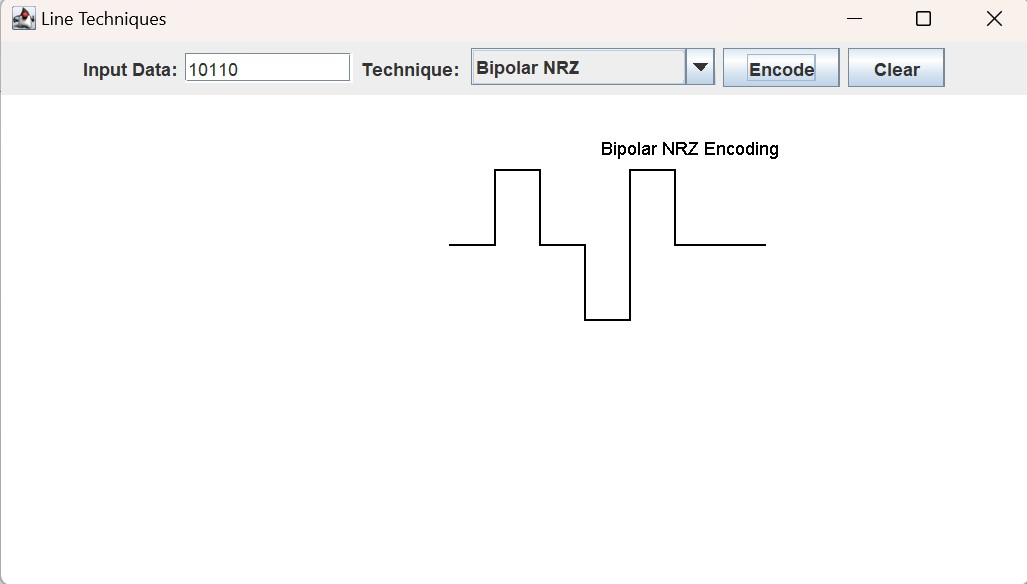
3. Polar NRZ



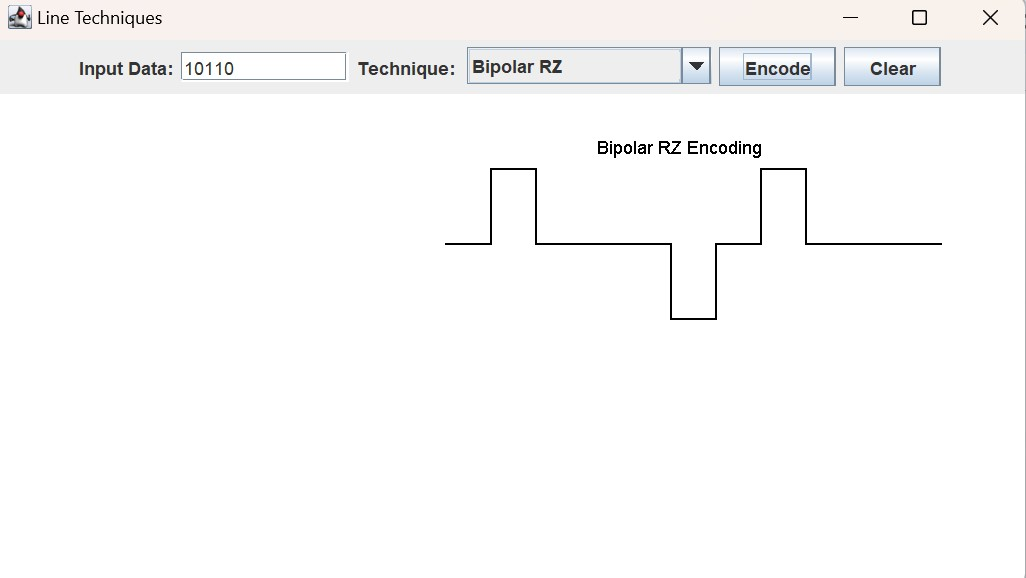
4. Polar RZ



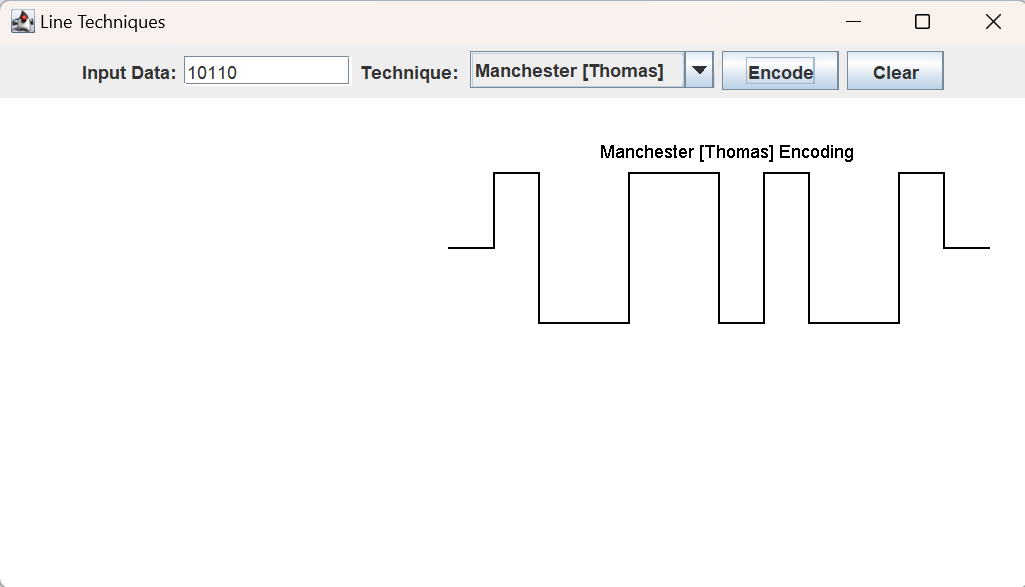
5.Bipolar NRZ



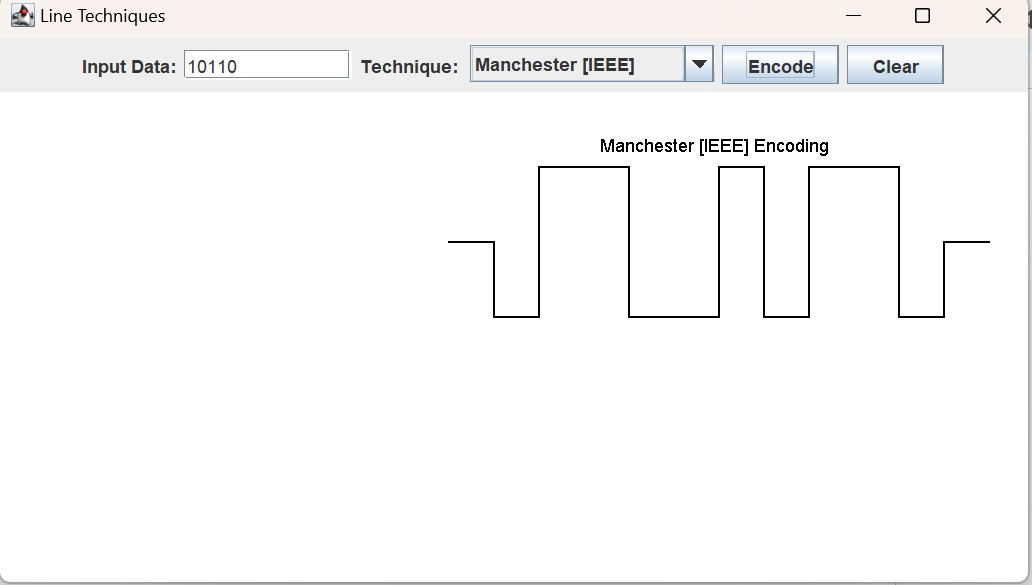
6. Bipolar RZ



7. Manchester Thomas Encoding



8. Machester IEEE



Javascript Code :

<!DOCTYPE html>

<html>

<head>

<title>Unipolar Encoding Square Wave</title>

<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

</head>

<body>

<h2>Unipolar Encoding Square Wave</h2>

<label for="inputBits">Enter the bit pattern (0s and 1s):</label>

<input type="text" id="inputBits" placeholder="e.g., 101010">

<button onclick="convertToUnipolar()">Convert & Plot</button>

<h3>Unipolar Encoding Result:</h3>

<p id="unipolarResult"></p>

<div id="graphContainer" style="width: 800px; height: 300px;"></div>

<script>

function convertToUnipolar() {

const inputBits = document.getElementById('inputBits').value;

const unipolarResult = document.getElementById('unipolarResult');

const graphContainer = document.getElementById('graphContainer');

const unipolarEncoding = inputBits

.split('')

.map(*bit* => bit === '1' ? 1 : 0)

unipolarResult.textContent = `Unipolar Encoding: ${unipolarEncoding.join('')}`;

const squareWaveData = [];

for (let i = 0; i < unipolarEncoding.length; i++) {

squareWaveData.push(unipolarEncoding[i]);

squareWaveData.push(unipolarEncoding[i]);

}

const trace = {

x: Array.from({ length: squareWaveData.length }, (*\_*, *i*) => i),

y: squareWaveData,

type: 'scatter',

mode: 'lines',

line: {

shape: 'hv',

color: 'slateblue'

}

};

const layout = {

title: 'Unipolar Encoding Square Wave',

xaxis: {

title: 'Bit Index'

},

yaxis: {

title: 'Voltage Level',

range: [0, 1]

},

width: 800,

height: 300

};

Plotly.newPlot('graphContainer', [trace], layout);

}

</script>

</body>

</html>

OutPut:

