American International University-Bangladesh (AIUB)

Department of Computer Engineering

COE 3201: Data Communication Laboratory

Lab Report 8

Title:Study of Digital to Digital Conversion (Line Coding) Using MATLAB.

Supervised By

SADMAN SHAHRIAR ALAM

Submitted By

Name	ID
MD. JOBAER HOSSAIN	22-47116-1

Group Members

Name	ID
RIFAH SANZIDA	22-47154-1
MD SAMIN YEASAR	22-47139-1
SHAYAN ABRAR	22-47156-1

Performance Task:

My id :22-47116-1

Here,

E=1

F=1

G=6

Convert E,F,G to 4 bits binary:

E=0001;

F=0001;

G=0110;

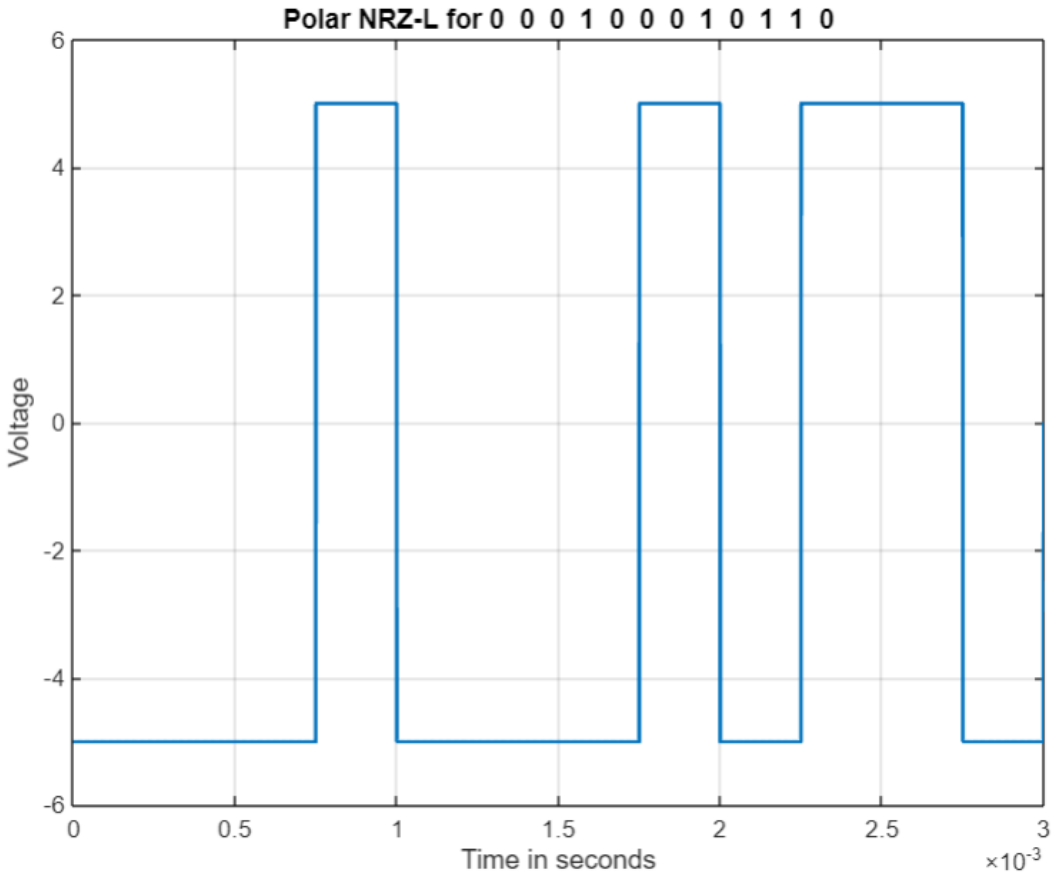
So,the 12 bits bit stream is:

Bit Stream=000100010110;

1.

```
clc
clear all
close all
bit_stream = [0 0 0 1 0 0 0 1 0 1 1 0];
no_bits = length(bit_stream);
bit_rate = 4000; % 4kbps
pulse_per_bit = 1; % for unipolar nrz
pulse_duration = 1 / (pulse_per_bit * bit_rate);
no_pulses = no_bits * pulse_per_bit;
samples_per_pulse = 500;
fs = samples_per_pulse / pulse_duration; %sampling frequency
t = 0:1/fs:(no_pulses) * (pulse_duration); % sampling interval
no_samples = length(t); % total number of samples
dig_sig = zeros(1, no_samples);
```

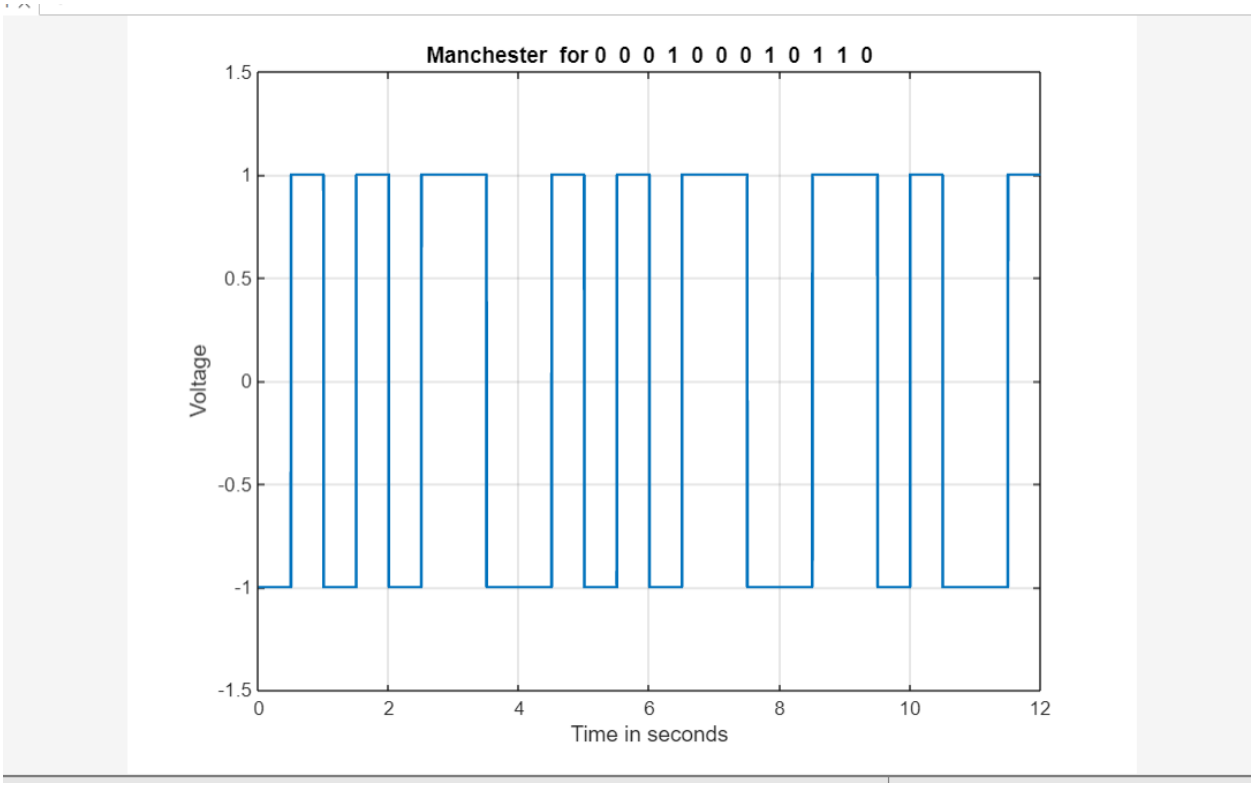
```
voltage_high = 5;
voltage_low = -5; % Polar NRZ-L, so we use positive and negative voltages
for i = 1:no_bits
if bit_stream(i) == 1
dig_sig(((i-1) * samples_per_pulse) + 1 : i * samples_per_pulse) = voltage_high;
else
dig_sig(((i-1) * samples_per_pulse) + 1 : i * samples_per_pulse) = voltage_low;
end
end
plot(t, dig_sig, 'linewidth', 1.5)
grid on
xlabel('Time in seconds')
ylabel('Voltage')
ylim([voltage_low - 1, voltage_high + 1]) % Adjusted ylim to show both voltage levels
title(['Polar NRZ-L for ', num2str(bit_stream), ''])
```



2.

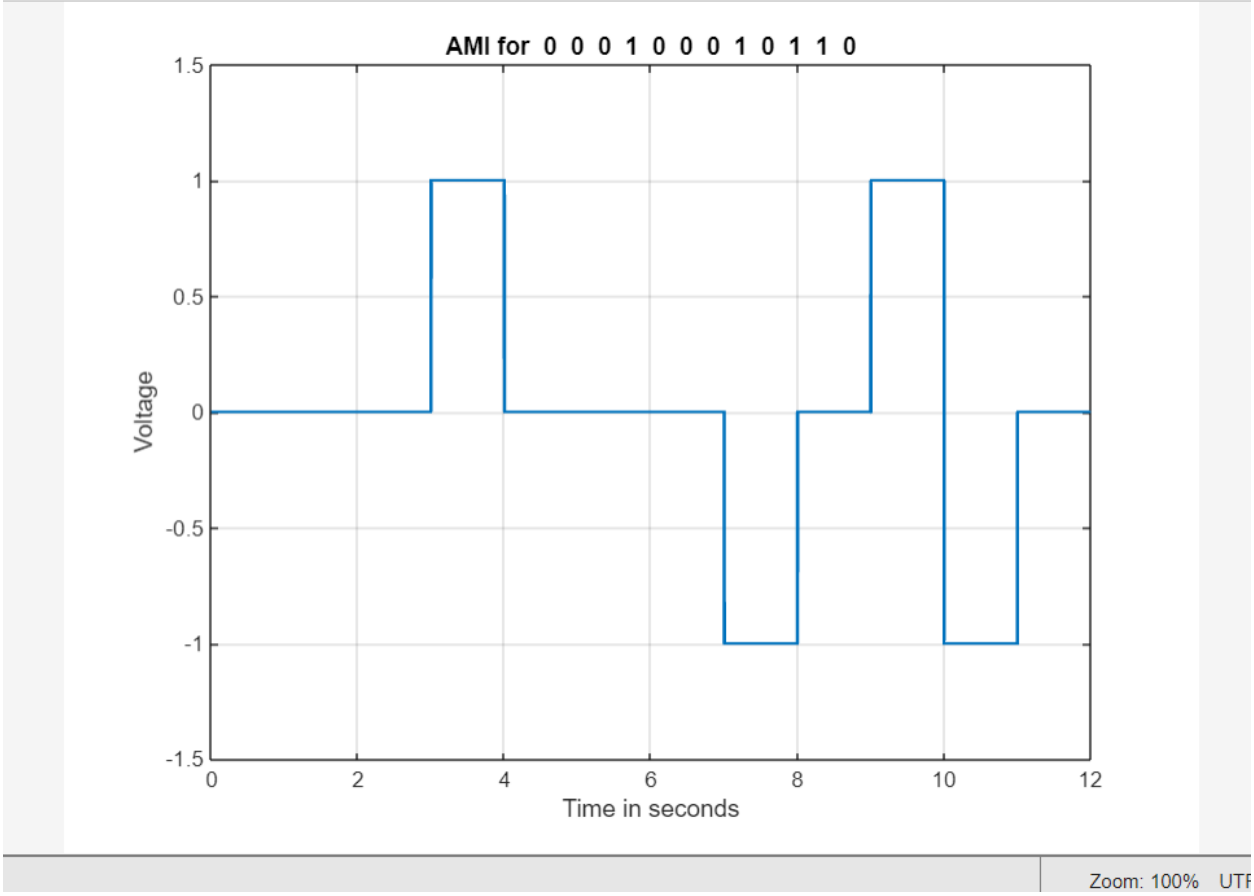
```
clc
```

```
clear all
close all
bit_stream = [0 0 0 1 0 0 0 0 1 0 0 0]; % Given bit stream
bit_rate = 2000; % 2 kbps
samples_per_bit = 500; % Number of samples per bit
no_bits = length(bit_stream);
total_samples = no_bits * samples_per_bit;
% Generate Manchester encoded signal
manchester_sig = zeros(1, total_samples);
for i = 1:no_bits
if bit_stream(i) == 1
manchester_sig((i-1)*samples_per_bit + 1 : (i-0.5)*samples_per_bit) = 1;
manchester_sig((i-0.5)*samples_per_bit + 1 : i*samples_per_bit) = -1;
else
manchester_sig((i-1)*samples_per_bit + 1 : (i-0.5)*samples_per_bit) = -1;
manchester_sig((i-0.5)*samples_per_bit + 1 : i*samples_per_bit) = 1;
end
end
% Time vector
t = linspace(0, no_bits, total_samples);% Plot Manchester encoded signal
plot(t, manchester_sig, 'linewidth', 1.5)
grid on
xlabel('Time in seconds')
ylabel('Voltage')
ylim([-1.5, 1.5]) % Adjusted ylim to show Manchester encoding
title(['Manchester  for ', num2str(bit_stream), ''])
```



3.

```
clc
clear all
close all
bit_stream = [0 0 0 1 0 0 0  1 0 1 1 0]; % Given bit stream
bit_rate = 5000; % 5 kbps
samples_per_bit = 500; % Number of samples per bit
no_bits = length(bit_stream);
total_samples = no_bits * samples_per_bit;
% Generate AMI encoded signal
ami_sig = zeros(1, total_samples);
voltage_level = 1; % Initial voltage level
for i = 1:no_bits
if bit_stream(i) == 1
ami_sig((i-1)*samples_per_bit + 1 : i*samples_per_bit) = voltage_level;
voltage_level = -voltage_level; % Invert voltage for each 1 bit
else
ami_sig((i-1)*samples_per_bit + 1 : i*samples_per_bit) = 0; % Zero voltage for 0 bit
end
end
% Time vector
t = linspace(0, no_bits, total_samples);% Plot AMI encoded signal
plot(t, ami_sig, 'linewidth', 1.5)
grid on
xlabel('Time in seconds')
ylabel('Voltage')
ylim([-1.5, 1.5]) % Adjusted ylim to show AMI encoding
title(['AMI for  ', num2str(bit_stream), ''])
```



4.

```
clc
clear all
close all

bit_stream = [0 0 0 1 0 0 0  1 0 1 1 0]; % Given bit stream
bit_rate = 10000; % 10 kbps
samples_per_bit = 500; % Number of samples per bit
% Generate LT-3 encoded signal
voltage_levels = [-1, 0, 1]; % 3 voltage levels for LT-3 encoding
lt3_sig = zeros(1, length(bit_stream) * samples_per_bit);
current_level = 1; % Initial voltage level, starting from the middle level
for i = 1:length(bit_stream)
if bit_stream(i) == 1
if current_level ~= 2
next_level = 2; % Transition to 0 if current level is not 0
else
last_nonzero = find(bit_stream(1:i-1), 1, 'last'); % Last non-zero bit
next_level = -voltage_levels(last_nonzero); % Transition to the opposite of the last
nonzero
```

```
level if current level is 0
end
else
next_level = current_level; % No transition for 0 bitend
% Update current level and assign voltage for the current bit duration
current_level = next_level;
lt3_sig((i-1)*samples_per_bit + 1 : i*samples_per_bit) =
voltage_levels(current_level);
end
end
% Time vector
t = linspace(0, length(bit_stream), length(lt3_sig));
% Plot LT-3 encoded signal
plot(t, lt3_sig, 'linewidth', 1.5)
grid on
xlabel('Time in seconds')
ylabel('Voltage')
ylim([-1.5, 1.5]) % Adjusted ylim to show LT-3 encoding
title(['MLT-3 for  ', num2str(bit_stream), ''])
```