American International University- Bangladesh

Course Name: Data Communication Laboratory

Experiment No: 05

Experiment title: Study of Digital to Digital Conversion (Line Coding) Using MATLAB

Section: D

Semester: Summer 23-24

Name: Tutul Majumder

ID:23-51364-1

<u>Title:</u> Study of Digital to Digital Conversion (Line Coding) Using MATLAB

Abstract:

This experiment is designed to-

1.To understand the use of MATLAB for solving communication engineering problems.
2.To develop understanding of Digital to Digital Conversion (Line Coding) using MATLAB.

Introduction:

Line Coding: Line coding is the process of converting digital data to digital signals. We assume that data, in the form of text, numbers, graphical images, audio, or video, are stored in computer memory as sequences of bits. Line coding converts a sequence of bits to a digital signal. At the sender, digital data are encoded into a digital signal; at the receiver, the digital data are recreated by decoding the digital signal. Figure 1 shows the process.

Performance Task for Lab Report: (my ID = 23-51364-1)

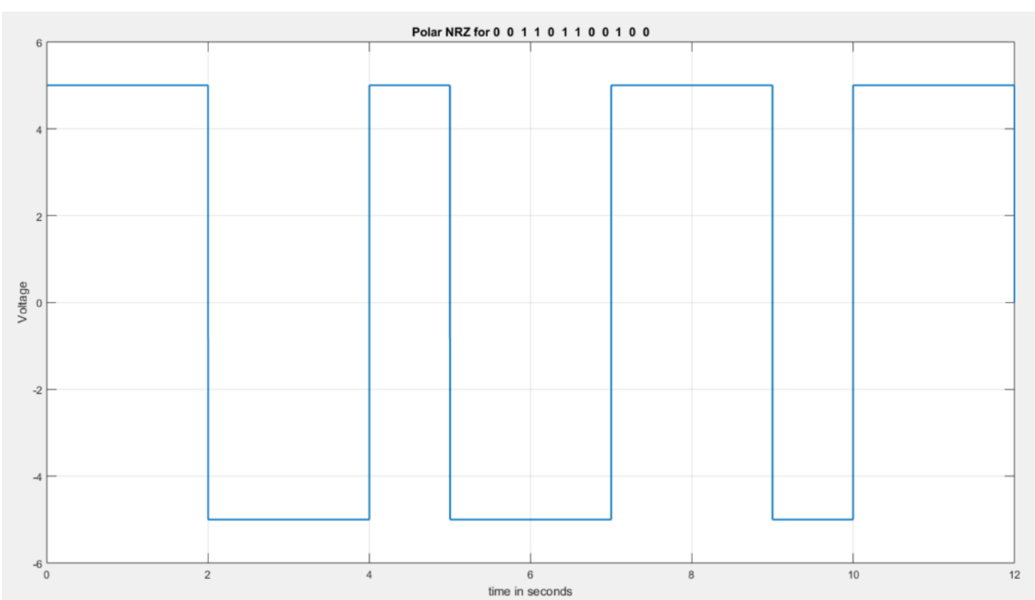
Assume your ID is **AB-CDEFG-H**, and then convert 'E', 'F' and 'G' to 4-bit binary to have a bit stream of 12 bits. Convert this bit stream to digital signal using the following methods:

1. Polar NRZ-L assuming bit rate is 4 kbps.
2. Manchester assuming bit rate is 2 kbps.
3. AMI assuming bit rate is 5 kbps.
4. MLT-3 assuming bit rate is 10 kbps.

1.**Polar NRZ-L**

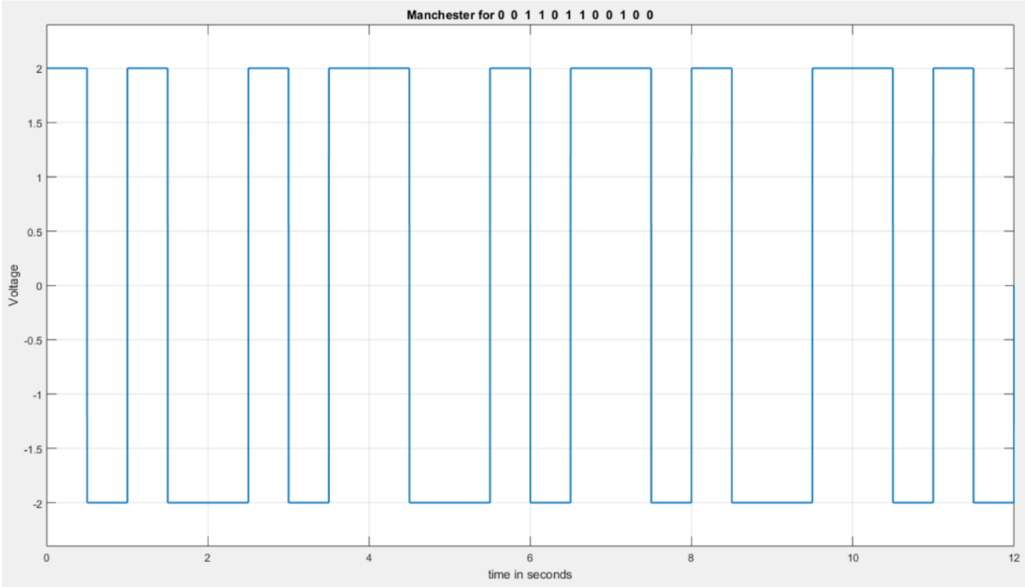
E(3)=0011
F(6)=0110
G(4)=0100

```
bit_stream=[0 0 1 1 0 1 1 0 0 1 0 0];
no_bits = length(bit_stream);
bit_rate = 4000; % 4 kbps
pulse_per_bit = 1; % for polar nrz
pulse_duration = (1/((pulse_per_bit)*(bit_rate)))*(bit_rate);
no_pulses = no_bits*pulse_per_bit;
samples_per_pulse = 2000;
fs = (samples_per_pulse)/(pulse_duration); %sampling frequency
% including pulse duration in sampling frequency
% ensures having enough samples in each pulse
t = 0:1/fs:(no_pulses)*(pulse_duration); % sampling interval
% total duration = (no_pulse)*(pulse_duration)
no_samples = length(t); % total number of samples
dig_sig = zeros(1,no_samples);
max_voltage = 5;
min_voltage = -5;
for i = 1:no_bits
if bit_stream(i) == 1
    dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=min_voltage*ones(1,samples_per_pulse);
else
    dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse)) =
max_voltage*ones(1,samples_per_pulse);
end
end
plot(t,dig_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('Voltage')
ylim([(min_voltage - (max_voltage)*0.2)
(max_voltage+max_voltage*0.2)])
title(['Polar NRZ for ',num2str(bit_stream),''])
```



2.Manchester

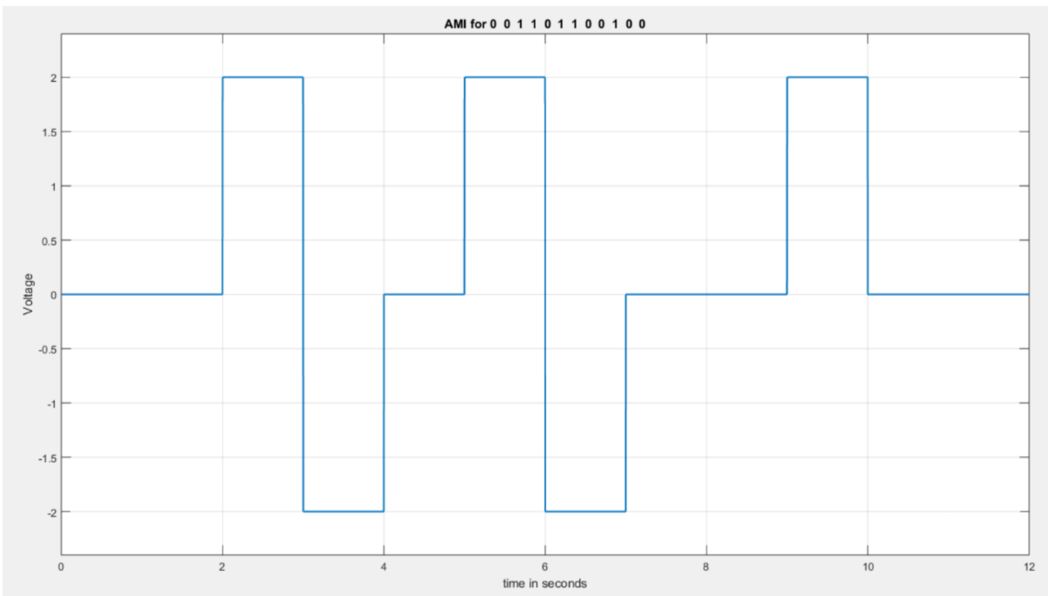
```
E(3)=0011
F(6)=0110
G(4)=0100

bit_stream=[0 0 1 1 0 1 1 0 0 1 0 0];

no_bits = length(bit_stream);
bit_rate = 2000; % 2 kbps
pulse_per_bit = 2; % for manchester
pulse_duration = (1/((pulse_per_bit)*(bit_rate)))*(bit_rate);
no_pulses = no_bits*pulse_per_bit;
samples_per_pulse = 500;
fs = (samples_per_pulse)/(pulse_duration); %sampling frequency
% including pulse duration in sampling frequency
% ensures having enough samples in each pulse
t = 0:1/fs:(no_pulses)*(pulse_duration); % sampling interval
% total duration = (no_pulse)*(pulse_duration)
no_samples = length(t); % total number of samples
dig_sig = zeros(1,no_samples);
max_voltage = +2;
min_voltage = -2;

for i = 1:no_bits
j = (i-1)*2;

if bit_stream(i) == 1
dig_sig((j*(samples_per_pulse)+1):(j+1)*(samples_per_pulse)) =
min_voltage*ones(1,samples_per_pulse);
dig_sig(((j+1)*(samples_per_pulse)+1):(j+2)*(samples_per_pulse)) =
max_voltage*ones(1,samples_per_pulse);

else
dig_sig((j*(samples_per_pulse)+1):(j+1)*(samples_per_pulse)) =
max_voltage*ones(1,samples_per_pulse);
dig_sig(((j+1)*(samples_per_pulse)+1):(j+2)*(samples_per_pulse)) =
min_voltage*ones(1,samples_per_pulse);
temp_cons = last_state; % temporary constant
end
end
figure
plot(t,dig_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('Voltage')
ylim([(min_voltage - (max_voltage)*0.2)
(max_voltage+max_voltage*0.2)])
title(['Manchester for ',num2str(bit_stream),''])
```



3.AMI

```
E(3)=0011
F(6)=0110
G(4)=0100

bit_stream=[0 0 1 1 0 1 1 0 0 1 0 0];

no_bits = length(bit_stream);
bit_rate = 5000; % 5 kbps
pulse_per_bit = 1; % for AMI
pulse_duration = (1/((pulse_per_bit)*(bit_rate)))*(bit_rate);
no_pulses = no_bits*pulse_per_bit;
samples_per_pulse = 500;
fs = (samples_per_pulse)/(pulse_duration); %sampling frequency
% including pulse duration in sampling frequency
% ensures having enough samples in each pulse
t = 0:1/fs:(no_pulses)*(pulse_duration); % sampling interval
% total duration = (no_pulse)*(pulse_duration)
no_samples = length(t); % total number of samples
dig_sig = zeros(1,no_samples);
max_voltage = +2;
avg_voltage=0;
min_voltage = -2;

inv_bit=1;


for i=1:no_bits
    if bit_stream(i)==1
        if inv_bit == 1
            dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=max_voltage*ones(1,samples_per_pulse);
            inv_bit=0;
```

```
        else
            dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=min_voltage*ones(1,samples_per_pulse);
            inv_bit=1;
        end
    else
        dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=avg_voltage*ones(1,samples_per_pulse);
    end
end
figure
plot(t,dig_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('Voltage')
ylim([(min_voltage - (max_voltage)*0.2)
(max_voltage+max_voltage*0.2)])
title(['AMI for ',num2str(bit_stream),''])
```



4. **MLT-3**

```
E(3)=0011
F(6)=0110
G(4)=0100

bit_stream=[0 0 1 1 0 1 1 0 0 1 0 0];

no_bits = length(bit_stream);
bit_rate = 10000; % 10 kbps
pulse_per_bit = 1; % for mlt3
pulse_duration = 1;
no_pulses = no_bits*pulse_per_bit;
samples_per_pulse = 500;
fs = (samples_per_pulse)/(pulse_duration); %sampling frequency
% including pulse duration in sampling frequency
% ensures having enough samples in each pulse
t = 0:1/fs:(no_pulses)*(pulse_duration); % sampling interval
% total duration = (no_pulse)*(pulse_duration)
no_samples = length(t); % total number of samples
dig_sig = zeros(1,no_samples);
max_voltage = +2;
neutral_volt=0;
min_voltage = -2;

last_state=neutral_volt;

prev_last_state=min_voltage;


for i=1:no_bits
    if bit_stream(i)==1
        if last_state==max_voltage

            dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=neutral_voltage*ones(1,samples_per_pulse);
            last_state=neutral_volt;
            prev_last_state=max_voltage;

        elseif last_state==min_voltage
            dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=neutral_voltage*ones(1,samples_per_pulse);
            last_state=neutral_volt;
            prev_last_state=min_voltage;

        else
            if prev_last_state==max_voltage
                dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=min_voltage*ones(1,samples_per_pulse);

                last_state=min_voltage;
                prev_last_state=neutral_voltage;

            else
```

```
                dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=max_voltage*ones(1,samples_per_pulse);
                last_state=max_voltage;
                prev_last_state=neutral_voltage;
            end
        end

    else
            dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse))
=last_state*ones(1,samples_per_pulse);

    end
end
figure
plot(t,dig_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('Voltage')
ylim([(min_voltage - (max_voltage)*0.2)
(max_voltage+max_voltage*0.2)])
title(['MLT-3 for ',num2str(bit_stream),''])
```