AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF ENGINEERING

Course name: Data Communication

Course code: COE 3201

Section: H

Semester: Spring 2023-24

Name: MD. ABU TOWSIF

ID: 22-47019-1

Instructor name: Dr. Muhammad Morshed Alam

Experiment no: 05

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

Submission date: March 08th, 2024

Performance Task for Lab Report: 5

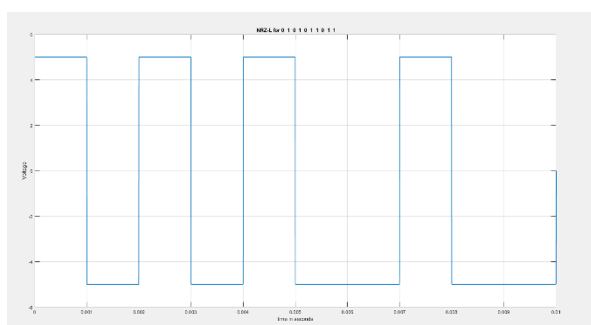
1. For the bit_stream = [0 1 0 1 0 1 1 0 1 1], write a MATLAB code to generate NRZ-L line coded signal and plot it over a suitable time domain.
2. For the bit_stream = [0 1 0 1 0 0 1 1], write a MATLAB code to generate Manchester line coded signal and plot it over a suitable time domain.
3. For the bit_stream = [0 1 0 1 0 0 1 1], write a MATLAB code to generate Differential Manchester line coded signal and plot it over a suitable time domain.

SOLUTION:

1. For the bit_stream = [0 1 0 1 0 1 1 0 1 1], write a MATLAB code to generate NRZ-L line coded signal and plot it over a suitable time domain.

<table>
<tr><th>MATLAB Code</th><th>Output Figure</th></tr>
<tr><td><pre>
clc;
clear all;

bit_stream = [0 1 0 1 0 1 1 0 1 1];
no_bits = length(bit_stream);
%no_bit= 10
bit_rate = 1000; % 1 kbps
pulse_per_bit = 1; % for unipolar
nrz
pulse_duration =
1/((pulse_per_bit)*(bit_rate));
%pulse_duration=1/1*1000=1x10-3
(second)=1 ms
no_pulses = no_bits*pulse_per_bit;
%no_pulses=10*1=10

samples_per_pulse = 500;
fs =
(samples_per_pulse)/(pulse_duration)
; %sampling frequency
t =
0:1/fs:(no_pulses)*(pulse_duration);
% sampling interval, total duration
= (no_pulse)*(pulse_duration)
no_samples = length(t); %
no_samples=4001
</pre></td><td>
</td></tr>
</table>

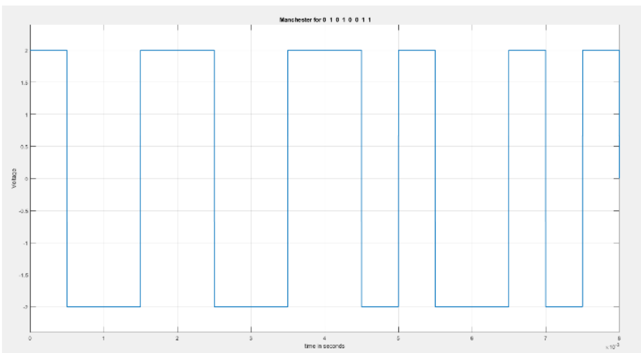
`` dig_sig = zeros(1,no_samples); % intial digital signal (line coded signal according to unipolor nrz) max_voltage = 5; min_voltage = -5; for i = 1:10 if bit_stream(i) == 0 dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse)) = max_voltage*ones(1,samples_per_pulse); else dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse)) = min_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([' NRZ-L for ',num2str(bit_stream),'']) ```	

2. For the bit_stream = [0 1 0 1 0 0 1 1], write a MATLAB code to generate Manchester line coded signal and plot it over a suitable time domain.

<table>
<tr><th>MATLAB Code</th><th>Output Figure</th></tr>
<tr><td><pre>
 clc;
clear all;
%RZ+I

bit_stream = [0 1 0 1 0 0 1 1];
no_bits = length(bit_stream);
%no_bits= 8
bit_rate = 1000; % 1 kbps
pulse_per_bit = 2; % for
differential manchester
pulse_duration =
1/((pulse_per_bit)*(bit_rate));
no_pulses =
no_bits*pulse_per_bit; %
no_pulse=6=8*2=16

samples_per_pulse = 500;
fs =
(samples_per_pulse)/(pulse_dura
tion); %sampling frequency
% including pulse duration in
sampling frequency
% ensures having enough samples
in each pulse
t =
0:1/fs:(no_pulses)*(pulse_durat
ion); % 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;
inv_bit = 0;
last_state = max_voltage; %+v
inv_last_state = min_voltage; %
inverse of last state, -v
</pre></td><td>
</td></tr>
</table>

<pre>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_pul se)+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_pul se)+1):(j+2)*(samples_per_pulse)) = min_voltage*ones(1,samples_per_ pulse); temp_cons = last_state; % temporary constant last_state = inv_last_state; inv_last_state = temp_cons; 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)])</pre>	

2. For the bit_stream = [0 1 0 1 0 0 1 1], write a MATLAB code to generate Differential Manchester line coded signal and plot it over a suitable time domain.

<table>
<tr><th>MATLAB Code</th><th>Output Figure</th></tr>
<tr><td>

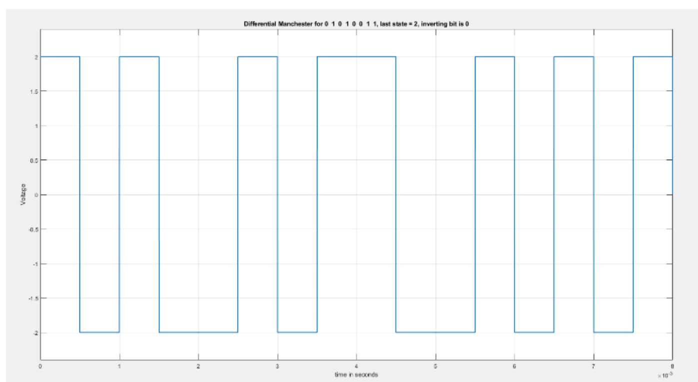
```
  clc;
clear all;
%RZ+I

bit_stream = [0 1 0 1 0 0 1 1];
no_bits = length(bit_stream);
%no_bits= 8
bit_rate = 1000; % 1 kbps
pulse_per_bit = 2; % for
differential manchester
pulse_duration =
1/((pulse_per_bit)*(bit_rate));
no_pulses =
no_bits*pulse_per_bit; %
no_pulse=6=8*2=16

samples_per_pulse = 500;
fs =
(samples_per_pulse)/(pulse_dura
tion); %sampling frequency
% including pulse duration in
sampling frequency
% ensures having enough samples
in each pulse
t =
0:1/fs:(no_pulses)*(pulse_durat
ion); % 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;
inv_bit = 0; % inverting bit
(in the lab sheet it is 1, but
it should be 0)
last_state = max_voltage; %+v
inv_last_state = min_voltage; %
inverse of last state, -v
```

</td><td>



</td></tr>
</table>

<pre>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)) = inv_last_state*ones(1,samples_p er_pulse); dig_sig(((j+1)*(samples_per_pul se)+1):(j+2)*(samples_per_pulse)) = last_state*ones(1,samples_per_p ulse); else dig_sig((j*(samples_per_pulse)+ 1):(j+1)*(samples_per_pulse)) = last_state*ones(1,samples_per_p ulse); dig_sig(((j+1)*(samples_per_pul se)+1):(j+2)*(samples_per_pulse)) = inv_last_state*ones(1,samples_p er_pulse); temp_cons = last_state; % temporary constant last_state = inv_last_state; inv_last_state = temp_cons; 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(['Differential Manchester for ',num2str(bit_stream),', last state = ',num2str(last_state),', inverting bit is ',num2str(inv_bit),''])</pre>	