A blue circle with text and symbols

Description automatically generated

**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.

|  |  |
| --- | --- |
| MATLAB Code | Output Figure |
| 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  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.

|  |  |
| --- | --- |
| MATLAB Code | Output Figure |
| 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\_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;  inv\_bit = 0;  last\_state = max\_voltage; %+v  inv\_last\_state = min\_voltage; % inverse of last state, -v    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  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)]) | A blue lines on a white background  Description automatically generated |

1. 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.

|  |  |
| --- | --- |
| MATLAB Code | Output Figure |
| 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\_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;  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    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\_per\_pulse);  dig\_sig(((j+1)\*(samples\_per\_pulse)+1):(j+2)\*(samples\_per\_pulse)) = last\_state\*ones(1,samples\_per\_pulse);  else  dig\_sig((j\*(samples\_per\_pulse)+1):(j+1)\*(samples\_per\_pulse)) = last\_state\*ones(1,samples\_per\_pulse);  dig\_sig(((j+1)\*(samples\_per\_pulse)+1):(j+2)\*(samples\_per\_pulse)) = inv\_last\_state\*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(['Differential Manchester for ',num2str(bit\_stream),', last state = ',num2str(last\_state),', inverting bit is ',num2str(inv\_bit),'']) | A graph with blue lines  Description automatically generated |