Experiment-4

Aim: To study Lines Codes & Implementing them in Matlab/Octave.

Software Used: Octave.

Theory: Line coding is the process of converting digital data to digital signals. Line coding converts a sequence of bits to a digital signal. At the sender, digital data are encoded into a digital signal; at the reciever, the digital data are recreated by decoding the digital signals.

Characteristics of Line Coding:

1. Signal element vs Data Element: A data element is the smallest entity that can be represent a piece of info. In digital data comm; a signal element is shortest unit of a digital signal.

2. Data Rate versus Signal Rate: The data rate defines the number of data elements sent in 1s. The signal rate is the number of signal elements sent is 1s. The unit is Baud. The data rate is called bit rate.

3. Bandwidth: Digital signal that carries information is non-periodic. The bandwidth of a non-periodic signal is continous with an infinite range. However, most digital signals we encounter in real life have a bandwidth with finite values. The effective bandwidth is finite.

4. Baseline Wandering: In decoding the reciever calculates a running average of recieved signal power. This average is called baseline. A long string of 0s & 1s can cause drift in the baseline.

5. DC Component: When the voltage level in a digital signal is constat for a while, the spectrum continues very low frequencies

6. Self Sychronization: To correctly interpret the signals recieved from the sender, the reciever's bit interval must correspond exactly to the sender's bit intervals. If the reciever clock is faster or slower, the bit intervals are not matched & the reciever might misinterpret the signal

7. Built-in Error Detection :- It is desirable to have a built in error detecting capability in the generated code to detect some of or all the errors that occured during transmission. Some encoding schemes that we will

8. Immunity to Noise & Interference :-

9. Complexity :- A complex scheme is more costly to implement than a simple one.

Each line code has advantages & disadvantages :-

The unipolar NRZ line code has the advantage of using circuits that require only one power supply, but it has the disadvantage of requiring channels that are DC coupled, because the waveform has a non-zero DC value.

The polar NRZ line code does not require a DC coupled channel, provided that the data toggles between binary 1's & 0's often and that equal of 1's & 0's are sent. However, the circuitry that produces the polar NRZ signal requires a negative voltage power supply as well as the positive voltage power supply.

The manchester NR line code has the advantage of always having 0 DC value, regardless of the data sequence, but it has twice the bandwidth of the unipolar NRZ or polar NRZ code because the pulses are half the width.

Experiment - 4

Aim: To Study Line Codes and Implementing them in Matlab/Octave

NRZ FAMILY:

1. NRZ-L unipolar
2. NRZ-L Polar
3. NRZ-M Unipolar
4. NRZ-M Polar
5. NRZ-S Unipolar
6. NRZ-S Polar
7. NRZ-AMI
8. RZ-AMI

RZ FAMILY:

1. RZ unipolar
2. RZ bipolar

MANCHESTER:

1. Bi -ϕ -L

1. NRZ-L unipolar

$$s(t) = \begin{cases} 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1 \\ 0, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

2. NRZ-L Polar

$$s(t) = \begin{cases} 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1 \\ -1, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

3. NRZ-M Unipolar

1. Mark based differential encoding (XOR) and create $mark[nT_b]$
2. $$s(t) = \begin{cases} 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1 \\ 0, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

4. NRZ-M Polar

1. Mark based differential encoding (XOR) and create $mark[nT_b]$
2. $$s(t) = \begin{cases} 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1 \\ -1, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

5. NRZ-S Unipolar

1. Space Based differential encoding (XNOR) and create $space[nT_b]$
2. $$s(t) = \begin{cases} 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1 \\ 0, & for\ 0 \to T_b\ \ if\ b[nT_b] = 0 \end{cases}$$

6. NRZ-S Polar

1. Space Based differential encoding (XNOR) and create $space[nT_b]$
2. $$s(t) = \begin{cases} 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1 \\ -1, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

7. NRZ-AMI

$$s(t) = \begin{cases} \pm 1, & for\ 0 \to T_b\ if\ b[nT_b] = 1, \text{where sign toggles for every occurance of 1} \\ 0, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

8. RZ-Unipolar

$$s(t) = \begin{cases} 1, & for\ 0 \to \frac{t_b}{2}\ if\ b[nt_b] = 1 \\ 0, & for \frac{t_b}{2} \to t_b \\ 0, & for\ 0 \to t_b\ if\ b[nt_b] = 0 \end{cases}$$

9. RZ-Bipolar

$$s(t) = \begin{cases} 1, & for\ 0 \to \frac{t_b}{2}\ if\ b[nt_b] = 1 \\ 0, & for \frac{t_b}{2} \to t_b \\ -1, & for\ 0 \to \frac{t_b}{2}\ if\ b[nt_b] = 0 \\ 0, & for\ 0 \to \frac{t_b}{2}\ if\ b[nt_b] = 0 \end{cases}$$

10. RZ-AMI

$$s(t) = \begin{cases} \pm 1, & for\ 0 \to \frac{T_b}{2}\ if\ b[nT_b] = 1, \text{where sign toggles for every occurance of 1} \\ 0, & for\ \frac{T_b}{2} \to T_b\ if\ b[nT_b] = 0 \\ 0, & for\ 0 \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

11. Manchester

$$s(t) = \begin{cases} 1, & for\ 0 \to \frac{T_b}{2}\ if\ b[nT_b] = 1 \\ -1, & for\ \frac{T_b}{2} \to T_b\ if\ b[nT_b] = 1 \\ -1, & for\ 0 \to \frac{T_b}{2}\ if\ b[nT_b] = 0 \\ 1, & for\ \frac{T_b}{2} \to T_b\ if\ b[nT_b] = 0 \end{cases}$$

Code

```
% octave pkg to load signal based utils
pkg load signal
clc;
clear alll;
close all;
%Inputs
b = round(rand(1, 10))
t = 0 : 1/100 : 0.99;
inc = t(2) - t(1);
% Line Codes
NRZ_L_U = [];
NRZ_L_P = [];
NRZ_M_U = [];
NRZ_M_P = [];
NRZ_S_U = [];
NRZ_S_P = [];
RZ_U    = [];
RZ_B    = [];
MAN = [];
sign = 1;
NRZ_AMI = [];
RZ_AMI  = [];
for i=1:length(b)
    if b(i) == 1
        NRZ_L_U = [NRZ_L_U ones(size(t))];
        NRZ_L_P = [NRZ_L_P ones(size(t))];
        RZ_U    = [RZ_U ones(1, length(t)/2) zeros(1, length(t)/2)];
        RZ_B    = [RZ_B ones(1, length(t)/2) zeros(1, length(t)/2)];
        MAN = [MAN square(2*pi*t, 50)];
        NRZ_AMI = [NRZ_AMI sign*ones(size(t))];
        RZ_AMI  = [RZ_AMI sign*ones(1, length(t)/2) zeros(1, length(t)/2)];
        sign = sign*(-1);
    elseif b(i) == 0
        NRZ_L_U = [NRZ_L_U zeros(size(t))];
        NRZ_L_P = [NRZ_L_P -ones(size(t))];
        RZ_U    = [RZ_U zeros(size(t))];
        RZ_B    = [RZ_B -ones(1, length(t)/2) zeros(1, length(t)/2)];
        MAN = [MAN -square(2*pi*t, 50)];
        NRZ_AMI = [NRZ_AMI zeros(size(t))];
        RZ_AMI  = [RZ_AMI zeros(size(t))];
    end
end
```

```
% mark encoding
mark = 0;
for i=1:length(b)
    mark = [mark xor(b(i), mark(i))];
end
mark = mark(2:end);
for j=1:length(mark)
    if mark(j) == 1
        NRZ_M_U = [NRZ_M_U ones(size(t))];
        NRZ_M_P = [NRZ_M_P ones(size(t))];
    elseif mark(j) == 0
        NRZ_M_U = [NRZ_M_U zeros(size(t))];
        NRZ_M_P = [NRZ_M_P -ones(size(t))];
    end
end
```

```
%space encoding
space = 0;
for i=1:length(b)
    space = [space not(xor(b(i), space(i)))];
end
space = space(2:end);
for k=1:length(space)
    if space(k) == 1
        NRZ_S_U = [NRZ_S_U ones(size(t))];
        NRZ_S_P = [NRZ_S_P ones(size(t))];
    elseif space(k) == 0
        NRZ_S_U = [NRZ_S_U zeros(size(t))];
        NRZ_S_P = [NRZ_S_P -ones(size(t))];
    end
end
```

```
%Plotting
t1 = 0 : inc : length(b) - inc;

subplot(6, 2, 1);
stairs(t1, NRZ_L_U);
ylim([-1.2, 1.2])
ylabel('NRZ L U')

subplot(6, 2, 2);
stairs(t1, NRZ_L_P);
ylim([-1.2, 1.2])
ylabel('NRZ L P')


subplot(6, 2, 3);
stairs(t1, NRZ_M_U);
ylim([-1.2, 1.2])
ylabel('NRZ M U')


subplot(6, 2, 4);
stairs(t1, NRZ_M_P);
ylim([-1.2, 1.2])
ylabel('NRZ L U')


subplot(6, 2, 5);
stairs(t1, NRZ_S_U);
ylim([-1.2, 1.2])
ylabel('NRZ L U')


subplot(6, 2, 6);
stairs(t1, NRZ_S_U);
ylim([-1.2, 1.2])
ylabel('NRZ L U')


subplot(6, 2, 7);
stairs(t1, RZ_U);
ylim([-1.2, 1.2])
ylabel('RZ U')


subplot(6, 2, 8);
stairs(t1, RZ_B);
ylim([-1.2, 1.2])
ylabel('RZ B')


subplot(6, 2, 9);
stairs(t1, NRZ_AMI);
ylim([-1.2, 1.2])
ylabel('NRZ AMI')


subplot(6, 2, 10);
stairs(t1, RZ_AMI);
ylim([-1.2, 1.2])
ylabel('RZ AMI')


subplot(6, 2, 11);
stairs(t1, MAN);
ylim([-1.2, 1.2])
ylabel('Manchester')


%pause in octave
pause
```

Output For B = 1 0 0 0 1 1 0 0 0 0

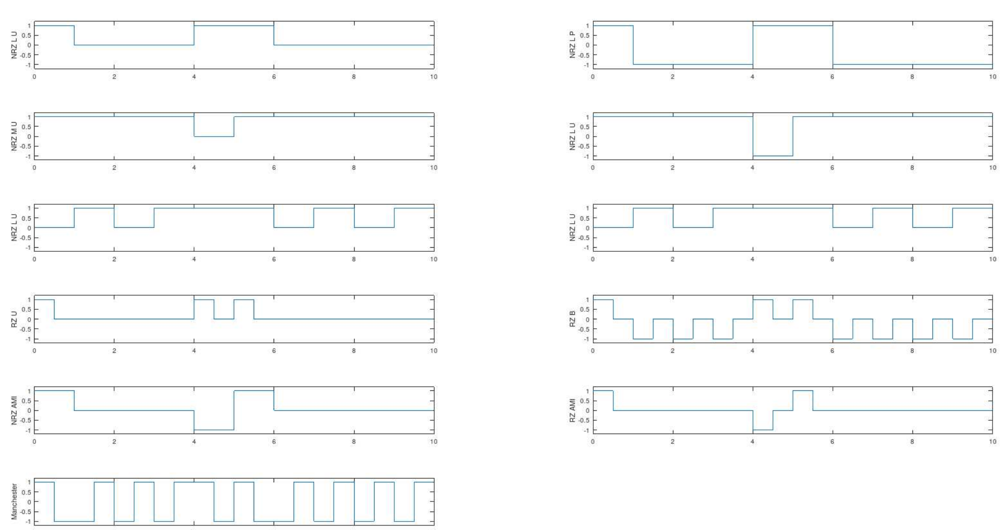

Figure 1: Line Codes

Q1. Compare the different line codes with respect to the bandwidth they use.

→ Manchester Code are considered worst line code as far as bandwidth is considered since they have the highest bandwidth. RZ bipolar also has very large bandwidth. NRZ polar is the considered best line code considering bandwidth since they have half the bandwidth of Manchester. Delay line code have the lowest bandwidth

Q2. Compare the different line codes with respect to the ~~bandwidth they use~~ probability of error they may have?

→ AMI is best line code in case of error in them as they have the ability to detect error (violation of rules). Polar & Bipolar signals have a limited error detecting ability. Unipolar & Manchester have zero error dection capability.

Q3. Compare the different line codes with respect to the DC power they require for transmission.

→ NRZ polar has the highest DC component among line codes. NRZ unipolar also has high DC component. RZ line codes have significantly low DC power. Manchester codes have '0' DC power component.

Q4. Compare the different line codes with respect to the self-clocking ability.

-> Manchester line code has the highest selfclocking ability among line code. RZ bipolar also has selfclocking ability. NRZ have no selfclocking ability.

Q5. Site a specific application of line codes.

-> NRZ encoding :- RS232 based protocols.
NRZ coding is most commonly used coding scheme & the reference for all coding schemes

-> RZ coding is used primarily in optical transmission system because it minimizes power consumption & the effects of system dispersion on optical signal distortion

-> Manchester encoding :- Ethernet networks, embedded clock application because it forces at least once transition per bit

-> Differential manchester encoding : token ring networks

-> Modified AMI :- WAN.