Department of
Computer Science and Engineering

Title: Implementing Encoding and Decoding Scheme Using NRZ-I

Data Communication Lab
CSE 308

Green University of Bangladesh

1 Objective(s)

- To be familiar with NRZ-I Line coding technique.

2 Problem analysis

Non Return To Zero (NRZ) line code is a binary code in which ones are represented by one significant condition, usually a positive voltage, while zeros are represented by some other significant condition, usually a negative voltage, with no other neutral or rest condition.

Non Return To Zero Inverted (NRZ-I) is a type of NRZ line coding. In NRZ-I usually binary 1 changes the state (toggle), and binary 0 keeps the state same to previous bit. Although binary 0 changing state and binary 1 keep the state unchanged is also NRZ-I. We will use the first way that is binary 1 changes the logic level and binary 0 keep unchanged the logic level.
Here is an example of NRZ-I line coding for bits 10111001 shown in Figure 1.

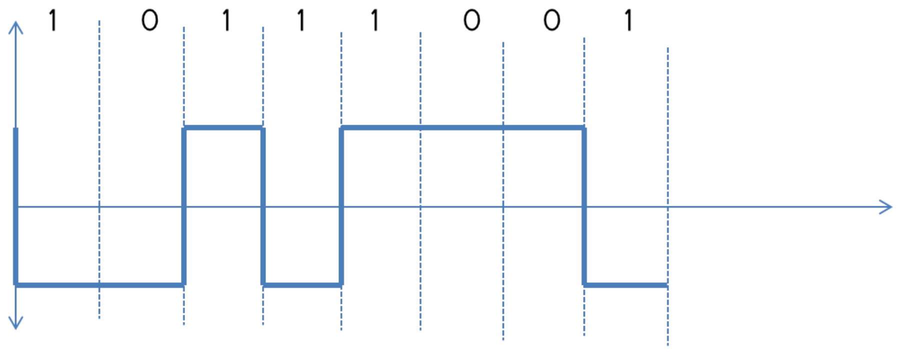

Figure 1: An example of NRZ-I line coding for bits 10111001

- **Conversion from bitstream to signal is called encoding.**
- **Conversion from signal to bitstream is called decoding.**

3 Coding With MatLab

For encoding we need the bitstream first. In MatLAB that can be defined as array or we can take input from user. Both ways are given below:

```
bits = [1 0 1 1 1 0 0 1];
```

or,

```
bits = input('prompt');
```

With bitstream we need bit rate that means how much bit(s) will appear in each second. This can be also defined as a variable or can be taken as input. Normally bit rate is considered as 1 bit per second. So we will define it to 1.

```
bitrate = 1;
```

We need a sampling for each bit. So that MatLAB needs to know how much sample point will be appeared for each bit. Low sampling can loss data or cannot define bits accurately. Very high sampling can cause slow your program. Here we will define it as 1000.

```
n = 1000;
```

MatLAB needs to know the total time needed for the bitstream. Sometimes total time T may be given or said to take as input. If not then we need to find it by dividing length of bitstream with bitrate.

```
T = length(bits)/bitrate;
```

We have a sampling for each bit we have declared previously. Now we need to find total sample point needed for whole bitstream. We can find it by multiplying sample point of each bit with the number of bits as below.

```
N = n*length(bits);
```

Now, we need to found time for each sample point. For this we divide the whole time (T) with number of total sample points (N).

```
dt = T/N;
```

We are near to the main process. Now we will define the time domain. That is the all values of X-axis. For this we need to declare an array from time 0 to time T by partitioning with dt. That is an array like this.

```
t = 0:dt:T;
```

For better performance we need to set all values of the result (amplitue domain) to 0. We can do this task with a build in function zeros().

```
x = zeros(1,length(t));
```

Here we need an additional variable to save the previous bit. The previous bit at the first point is given in the problem statement. If not by default we will declare it to 1. Here I am declaring it 1 that is the previous bit had positive voltage.

```
lastbit = 1;
```

Now for each bit if that bit is 1 than a range of the amplitude domain is redefined with value 1 or -1 depending on the previous bit. If If the bit is 0 we will redefine a range of values of amplitude domain with same values on previous bit. Here we will continue a loop from 1 to length of bitstream. Inside the loop we will check whether the bit is 1 or 0. If the bit is 0 then we will fill a range from x((i-1)*n+1) to x(i*n) with the value saved in the variable lastbit. Here lastbit will remain unchanged. But if the bit is 1 the we will fill range from x((i-1)*n+1) to x(i*n) with opposite value of the lastbit and negate lastbit. That is if i=1 then we will fill from 1 to n'th (sample points per bit) position of the array with 1 or -1 (depending on last bit) or with lastbit value if bit is 0. If i=2 then we will fill from (n+1) to 2*n'th position. Here is the loop required.

```
for i=1:length(bits)
  if bits(i)==1
    x((i-1)*n+1:i*n) = -lastbit;
    lastbit = -lastbit;
  else x((i-1)*n+1:i*n) = lastbit;
  end
end
```

Everything required for encoding is done. We will now plot the result (amplitude domain) on respect to time domain. Here is the plot. Here 'Linewidth', 3 means the signal line will be bold three times than original signal.

```
plot(t, x, 'Linewidth', 3);
```

4 Implementation in MatLAB

4.1 Code For Encoding

```
bits = [1 0 1 1 1 0 0 1];
bitrate = 1;
n = 1000;
T = length(bits)/bitrate;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t));
lastbit = 1;
```

```
for i=1:length(bits)
  if bits(i)==1
    x((i-1)*n+1:i*n) = -lastbit;
    lastbit = -lastbit;
  else x((i-1)*n+1:i*n) = lastbit;
  end
end
plot(t, x, 'Linewidth', 3);
```

4.2 Decoding

For decoding we need a counter that will count how much bit has been decoded. We will define the initial value of the counter to 0.

```
counter = 0;
```

Here we also need a variable to store lastbit. We will use the same variable we used in encoding. But as the value of that has been changed so we will define it's value again.

```
lastbit = 1;
```

Now we need a loop again to decode the signal. The loop will run from 1 to length of t. That is same times of total sample point. In the loop we will test whether t(i) is greater than the counter. If it is true then we will increase the value of counter. And will assign the value 1 to result(counter) array and negate lastbit (previous bit) if x(i) is not same to the lastbit (previous bit) or we will assign 0 to result(counter array. In this way we will assign or test the second sample value of each bit to the result bitstream. And for other samples the if condition (external if) will be false. For example, At first the value of t(1)=0 and that is not greater than counter so nothing occured. Then t(2)=0.001 and that is greater than 0. So counter increased to 1 and the bit is set to the value of 1 or 0 (depending on voltage level and lastbit). Next values of t(i) will be skipped (if condition will be false) as long as it is not 1.001. For each bit same things will be held. The loop looks like below.

```
for i = 1:length(t)
  if t(i)>counter
    counter = counter + 1;
    if x(i)!=lastbit
      result(counter) = 1;
      lastbit = -lastbit;
    else result(counter) = 0;
    end
  end
end
```

4.3 Display

After this decoding is also done. We need to display it. We can display that by just writing the variable without semi-colon. Or this can shown with disp() function.

```
disp('NRZ-I Decoding:');
disp(result);
```

4.4 Full Code (Encoding and decoding)

```
bits = [1 0 1 1 1 0 0 1];
bitrate = 1;
n = 1000;
T = length(bits)/bitrate;
N = n*length(bits);
dt = T/N;
```

```
t = 0:dt:T;
x = zeros(1,length(t));
lastbit = 1;
for i=1:length(bits)
  if bits(i)==1
    x((i-1)*n+1:i*n) = -lastbit;
    lastbit = -lastbit;
  else x((i-1)*n+1:i*n) = lastbit;
  end
end
plot(t, x, 'Linewidth', 3);
counter = 0;
lastbit = 1;
for i = 1:length(t)
  if t(i)>counter
    counter = counter + 1;
    if x(i)!=lastbit
      result(counter) = 1;
      lastbit = -lastbit;
    else result(counter) = 0;
    end
  end
end
disp('NRZ-I Decoding:');
disp(result);
```

5 Sample Input/Output (Compilation, Debugging & Testing)

input: 1 0 1 1 1 0 0 1
Output:

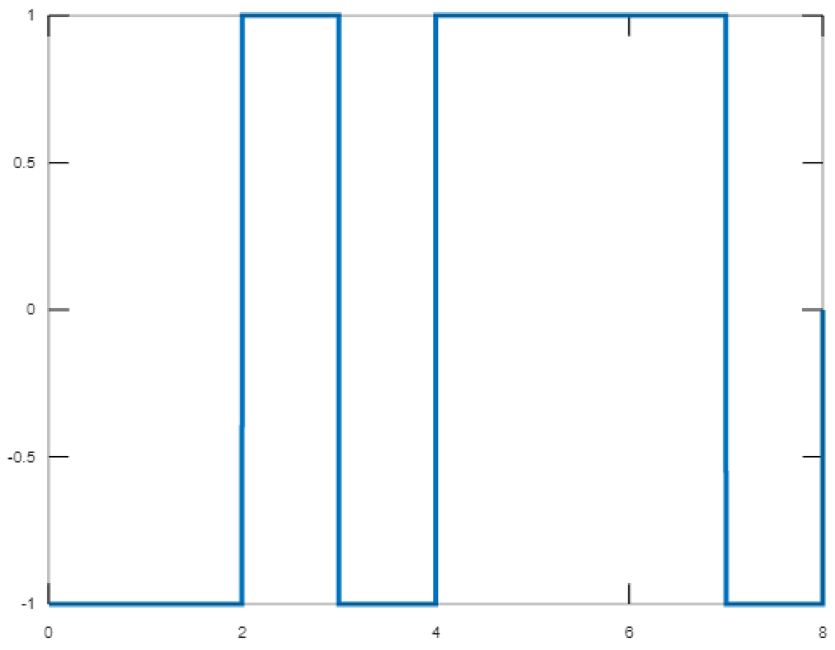

Figure 2: Output

6 Discussion & Conclusion

Based on the focused objective(s) to understand about the algorithms, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

7 Lab Task (Please implement yourself and show the output to the instructor)

1. Implementing Encoding and Decoding Scheme Using NRZ-L

7.1 Problem analysis

Non Return To Zero Level (NRZ-L) is a type of NRZ line coding. In NRZ-L usually binary 1 maps to logic level high, and binary 0 maps to logic level low. Although binary 0 maps to logic level high, and binary 1 maps to logic level low is also NRZ-L. We will use the first way that is binary 1 maps to logic level high and binary 0 maps to logic level low.

Here is an example of NRZ-L line coding for bits 10111001 shown in Figure 3.

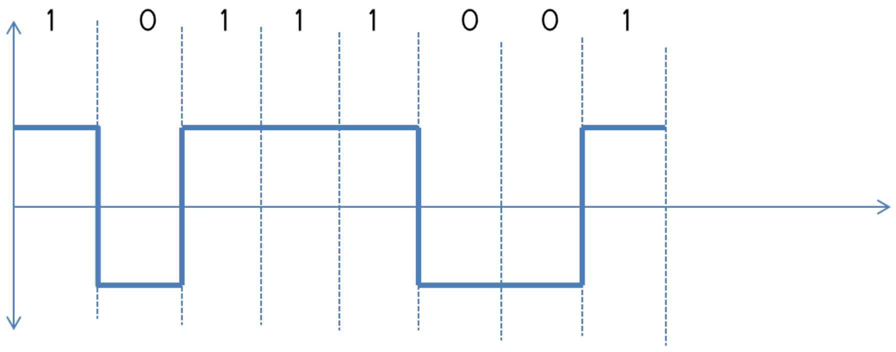

Figure 3: An example of NRZ-L line coding for bits 10111001

8 Lab Exercise (Submit as a report)

- Implementing Encoding and Decoding Scheme Using NRZ-L.

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.