



DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

**Title: Implementing encoding and decoding
scheme using AMI and Pseudo ternary**

DATA COMMUNICATION LAB
CSE 308



GREEN UNIVERSITY OF BANGLADESH

1 Objective(s)

- To be familiar with AMI and Pseudo ternary Line coding technique.

2 Problem analysis

In telecommunication and data storage, Bipolar is a line coding scheme in which there are three voltage levels positive, negative, and zero. The voltage level for one data element is at zero, while the voltage level for the other element alternates between positive and negative. Bipolar line coding is two types AMI and Pseudoternary.

- **Alternate Mark Inversion (AMI)** A neutral zero voltage represents binary 0. Binary 1's are represented by alternating positive and negative voltages.
- **Pseudoternary** Bit 1 is encoded as a zero voltage and the bit 0 is encoded as alternating positive and negative voltages i.e., opposite of AMI scheme.

Here is an example of AMI and Pseudoternary line coding for bits 010010. shown in Figure 1.

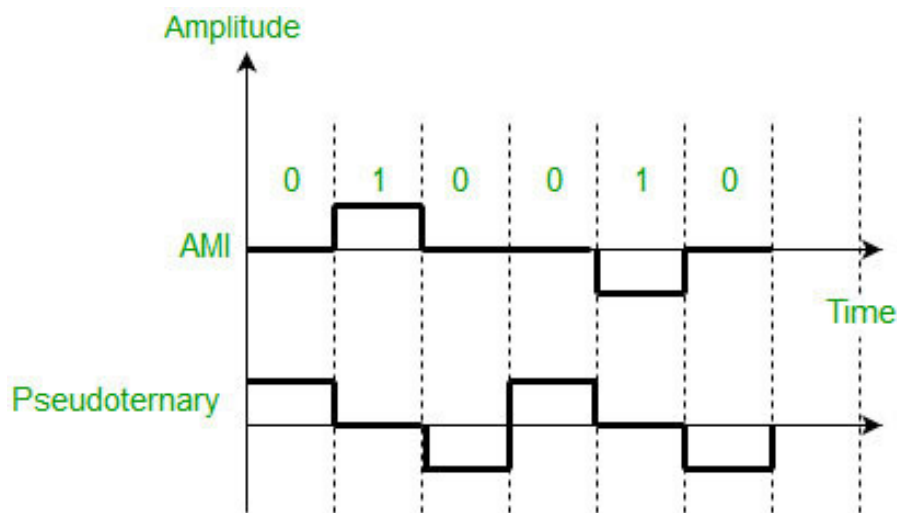


Figure 1: An example of AMI and Pseudoternary line coding for bits 010010

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

In this lab we will only implement Alternate Mark Inversion (AMI) because concept of Pseudoternary is reverse of AMI line coding. Where, 1 is positive voltage level and 0 means opposite voltage level.

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 bit-stream 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 (amplitude 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;
```

It is quite similar to NRZ-I. Here 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 the bit is 0 then the amplitude domain value will remain 0 (no change needed as they are previously defined as 0).

Here we will continue a loop from 1 to length of bit-stream. Inside the loop we will check whether the bit is 1 or 0. If the bit is 1 then we will fill a range from $x((i-1)*n+1)$ to $x(i*n)$ with opposite value saved of lastbit. And lastbit will be changed (negated). If the bit is 0 don't do anything, it is previously defined as 0. Here 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 0 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;
    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 = input('prompt');
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;
    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 opposite of the lastbit ($x(i) == -lastbit$) 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 occurred. 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 occurred. 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('AMI Decoding:');  
disp(result);
```

4.4 Full Code (Encoding and decoding)

```
bits = input('prompt');  
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;  
    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('AMI Decoding:');  
disp(result);
```

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

input: 1 0 1 1 1 0 0 1

Output:

6 Discussion & Conclusion

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

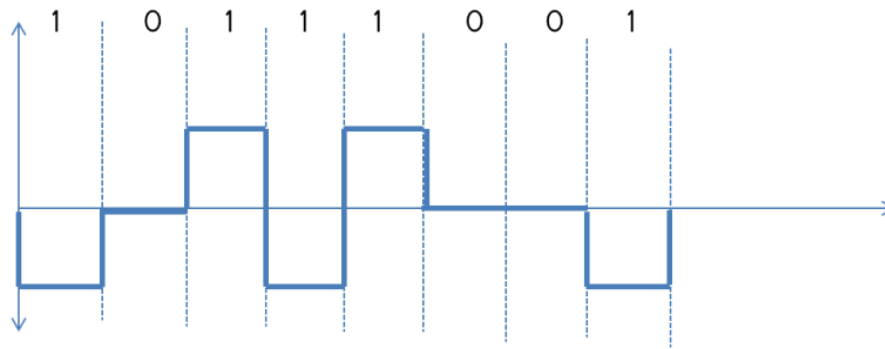


Figure 2: Output

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

1. Implementing Encoding and Decoding Scheme Using AMI for bit-stream 11001010

8 Lab Exercise (Submit as a report)

- Implementing Encoding and Decoding Scheme Using Pseudo ternary.

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.

10 Resources

- <https://www.youtube.com/watch?v=myI7t0tt9ak>