ECSE 436 **McGill University**

Prof. Bajcsy **Fall 2016**

Laboratory Assignment 2 (part 1)

Due: with Parts 2 and 3

Reading: Lecture and lab/tutorial notes, handouts on Altera FPGA signal processing boards, digital signal processing and coding. Related material in the reference textbook by Lin and Costello (available online) and Proakis, Manolakis Digital Signal Processing text.

Problems:

1. FPGA Signal Processing Boards

Go over the demo/example described in the FPGA handout and start implementing this demo on your signal processing board.

2. Viterbi Decoding Algorithm for the Hamming (8,4,4) Code

A Hamming codeword **c =** (c1, c2, c3, c4, c5, c6, c7, c8) has been transmitted over an AWGN channel with noise mean being 0 and variance equal to □2 □0.5 . The received vector **r = c + noise** is observed to be **r =** (0.54, - 0.12, 1.32, 0.41, 0.63, 1.25, 0.37, -0.02).

(a1) Using Matlab, the code book of the (8,4,4) code and material discussed in the lecture/lab, determine the closest of all 16 codewords of this code to the **r** that was received. Submit your script to obtain the answer.

(a2) How many floating point operations (additions, multiplications and comparisons) does the approach from part (a) use to decode a codeword?

(b1) Design an optimized trellis diagram for this code that has at most 4 states per stage.

(b2) Redo part (a1) in Matlab by using a Viterbi algorithm decoder for the Hamming code using its 4-state optimized trellis. Determine and implement the appropriate edge weights for individual trellis edges, then use the survivor path approach to determine your answer. Submit your stage-by-stage distance matrices and results of each stage survivor search.

(b3) Redo part (a2) for the improved decoder from part (b2)

(c1) Use Matlab to implement and test the Viterbi decoder from part (b2).

(c2) Use your Viterbi decoder from part (c1) to simulate bit error rate performance of the (8,4,4) Hamming code on the additive Gaussian noise channel. Use 'randn' function to generate Gaussian noise samples and set the noise variance so that the SNR ranges between 0 dB and 10 dB, with a step of 1 dB. Submit your BER vs. SNR plot as well as the script/s used to obtain it. (HINTS: Calculate the errors only for the message bits and simulate each SNR value until you obtain 100 decoding errors or until you run 10,000 packets, whichever is less. Use the 'semilogy' command to plot your results in log scale.)

3. Speech Processing and Acoustics: Convolution and the Impulse Response

An overall effect of a room on any given sound signal *x[n]* can be modeled as a linear time-invariant system and can be completely described by the room's impulse response *h[n].* As mentioned in class, you can measure impulse response of any specific room by producing a very short sound-burst (an impulse such as a gun-shot or a clap) at the source and then measuring the resulting signal at the

microphone which picks up not only the direct line-of-sight contribution from the sound source, but also echoes produced by reflections off walls and room objects. File "h.wav" contains the impulse responses h[n] for a specific room and source/microphone setup.

(a) Using Matlab, plot the impulse response for the channel and explain how the delay of the channel (i.e., the time taken for the sound to travel from source to microphone) can be computed by looking at the impulse response plot. What is the distance of the microphone from the source of impulse ?

(b) The file "speech.wav" is a voice recording from an *anechoic* chamber (a studio with little or no echo), with the microphone very close to the voice. Convolve this recording with the impulse response h[n] and listen to the result before and after convolution. Explain what you hear.

(c) Repeat part (b) for an impulse response h'[n] = h[n] + h[n-3000] where the delay is in number samples. Again, explain what you hear.

HINTS: In Matlab, x=wavread('speech.wav') will read audio file named "speech.wav" into the vector "x". Because a computer can only store a discrete sequence of numbers, each file is a vector $v[n]$ sampling the original response $h(t)$ so that $v[n] = h(nT)$ for indices $n = 0,1,2$ etc. In this experiment the sampling rate for all recordings is $T = 1/16000$ seconds. Convolution can be implemented with the "conv" command and you can listen to any vector of data by using the command sound(v,16000)which plays back the vector $v[n]$ at a rate of 16000 samples per second.

(d) Describe how you can play the signal backwards in time and discuss what you hear.

(e) Play the signal at different playback speeds (13000, 14500, 17000, 18500, 20000) and describe how the signal quality/intelligibility changes.

(f) To observe the effect of aliasing, sub-sample the signal (2:1, 3:1, 4:1, 5:1 and 10:1), then play it back appropriately. Describe what you hear and sketch an appropriate signal spectrum plot, illustrating what is happening.

(g) Quantization is another signal processing operation that is performed (e.g., to save on data storage needs) but it can affect significantly quality of the play-back signal.

Quantize your signal using appropriate uniform quantizer that has 16, 8, 4, 2 and 1 bits. Play the quantized signals back and discuss the impact of quantization on the quality of the sound.