Simon Fraser University

ENSC 452: Advanced Digital System Design
Spring 2018

Adaptive Noise Cancellation Headphones with FPGA

Gur Kohli
gkohli@sfu.ca
301215504

Svetlana Borkovkina
sborkovk@sfu.ca
301215298

January 23, 2018

Table of Contents

List of Figures

Introduction

We live in a world where noise has become a constant part of everyday life: cars passing by our windows in the middle of the night, people talking loudly next to us in a coffee shop, loud music playing on the street. Most of us are so used to the sounds surrounding us every second of every day, that we barely register them. Not all noise is bearable and is often undesired as in the case of noise inside an helicopter cabin. The headsets worn by the pilot and the passengers actively filter out the noise of the fast moving rotor blades, in absence of which, a conversation would be impossible. There are also times when noise level becomes excessive, when we are exposed to unpleasant sounds we cannot avoid, and that noise becomes a significant irritant. Our team has decided to implement adaptive noise control algorithm on Xilinx FPGA to eliminate ambient noise and deliver clean sound to the listener.

Background

To deal with unwanted noise, Passive Noise Control (PNC) is used, such as barriers and enclosures. A normal in-ear headphone uses an earbud that goes into a person's ear canal and provides a sufficient seal to block outside noise. Over-the-ear headphones have cushions around the headphones surrounding the actual speaker driver to minimize noise entering inside the headphones enclosure. Passive noise control methods are relatively simple and work on a wide variety of frequencies. However, they can be costly, large, or don't provide the best performance.
Adaptive Noise Control (ANC) is a more technologically advanced alternative to the passive noise control methods. Adaptive noise cancellation is a well-researched topic and nowadays this technology can be found even in moderately-priced headphones. ANC systems analyze the noise wave and create an anti-noise sound wave to cancel out the noise. Anti-noise waveform has the same amplitude, but inverted phase to the original noise. As a result of addition, those two waveforms cancel each other out and noise is eliminated [Fig 1]. ANC is often implemented using digital signal processing.

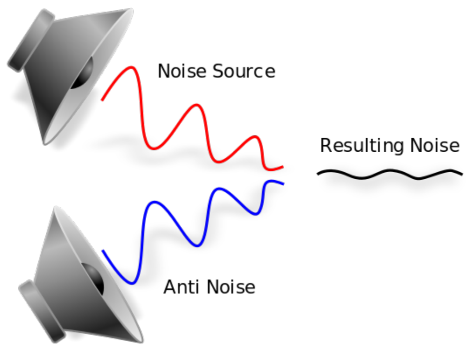

Figure 1: Cancellation of Noise [1]

Project Objective

In this project we are implementing ANC algorithm in VHDL on Xilinx FPGA to provide real-time noise cancellation. A user is able to connect headphones to the main FPGA board and hear a clear, noise-free sound, with ambient noise filtered out. The user is able to select the source of the input from either the Line IN input, carrying a music signal from a device such as iPhone, MP3 Player or a speech signal input, arriving from the microphone connected to Mic IN of Board B via Ethernet. The headphones are fitted with two set of microphones, inside and outside the headphone casing recording their respective sound. Using these two signals, the noise is calculated which is subsequently subtracted from the signal coming from the input source, the result of which is fed back to the headphones. This feed forward loop is keeps on going until the error with the least mean square is realized at which point the noise is most effectively eliminated.

System Overview

High-level Project Description

Adaptive Noise Cancellation removes unwanted ambient noise that leaks through the headphones and into the ears. There are two microphones, primary microphone and reference microphones that are the inputs to the ANC module. The primary microphone is placed inside the headphone casing, recording the signal sound coming out of the headphones mixed with the noise that leaks into the headphones. The reference microphone is attached to the outside of the headphones recording the ambient noise around the headphones but no signal. Because of the close proximity of both the headphones, the noise recorded by both the headphones are statistically similar.

The ANC module works on Least Mean Square (LMS) algorithm that employs an FIR filter with adaptive weights to find the difference between the primary signal and the reference signal, which is the anti-noise. This noise is fed back to the Adaptive Algorithm that calculates the new weights to be used by the FIR filter. This process keeps repeating itself until the LMS algorithm converges to the output with the least mean square error.

This anti-noise is further an input to Signal Processor and Multiplexer (SPM) module. The SPM module accepts this anti-noise and two input signals, the Line IN input that carries a music signal from a device such as iPhone, MP3 Player and a speech signal from DDR3 memory via AXI bus. User can choose between the latter two inputs, implemented via a multiplexer. The chosen input is added to the anti-noise waveform and sent as output to the audio codec which further sends it via Line OUT to the headphones.

The speech signal input, arriving via AXI bus to the SPM module starts from the microphone connected to Mic IN of Board B. The signal is recorded by a sound recorder algorithm using the ARM processor running on top of Petalinux. The sound is recorded in real time and streamed via Ethernet to Board A where another instance of Petalinux running on the ARM processor receives it. It’s then stored in the DDR3 memory.

System Block Diagram

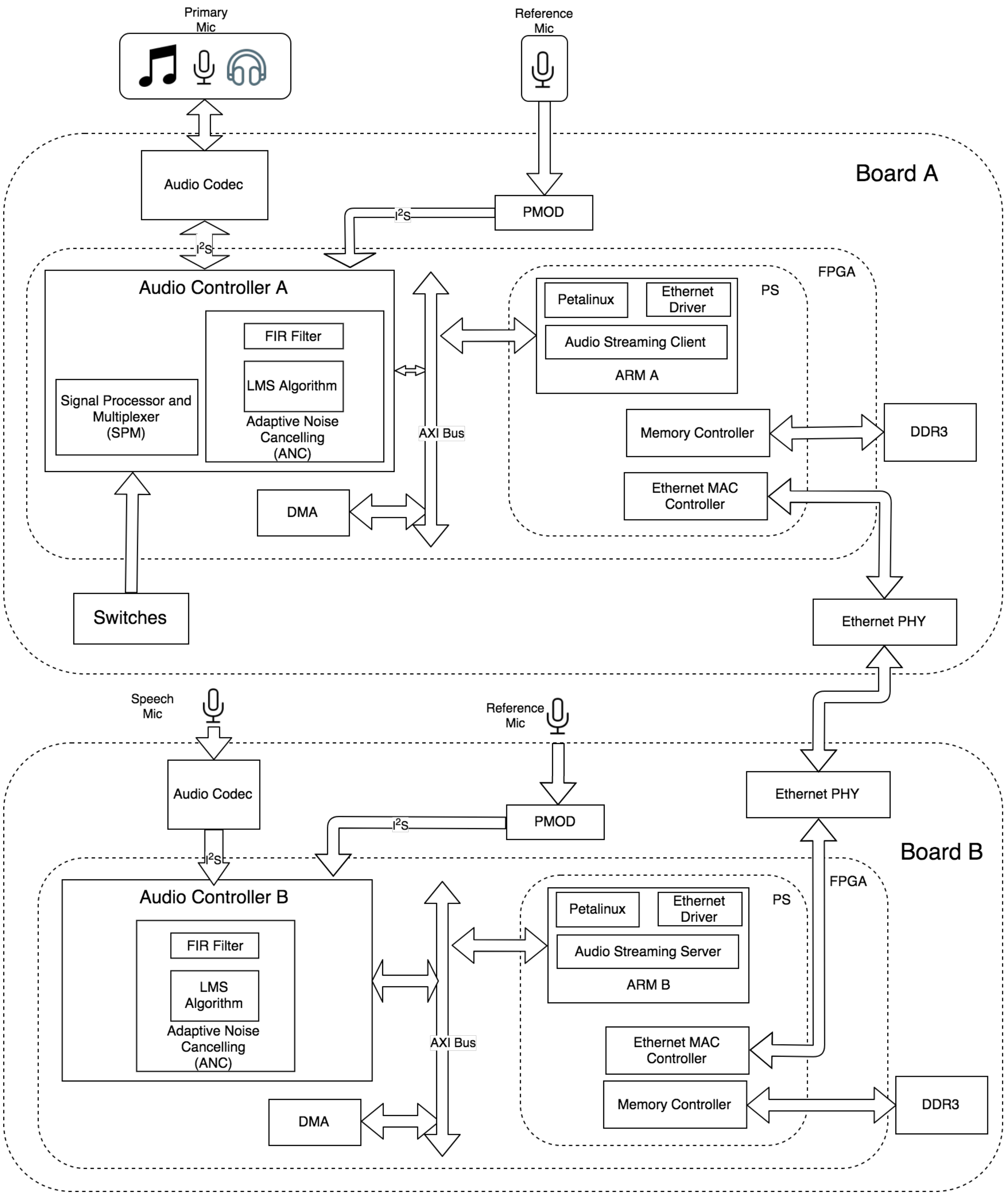

Figure 2: System Model

Use Case Diagram

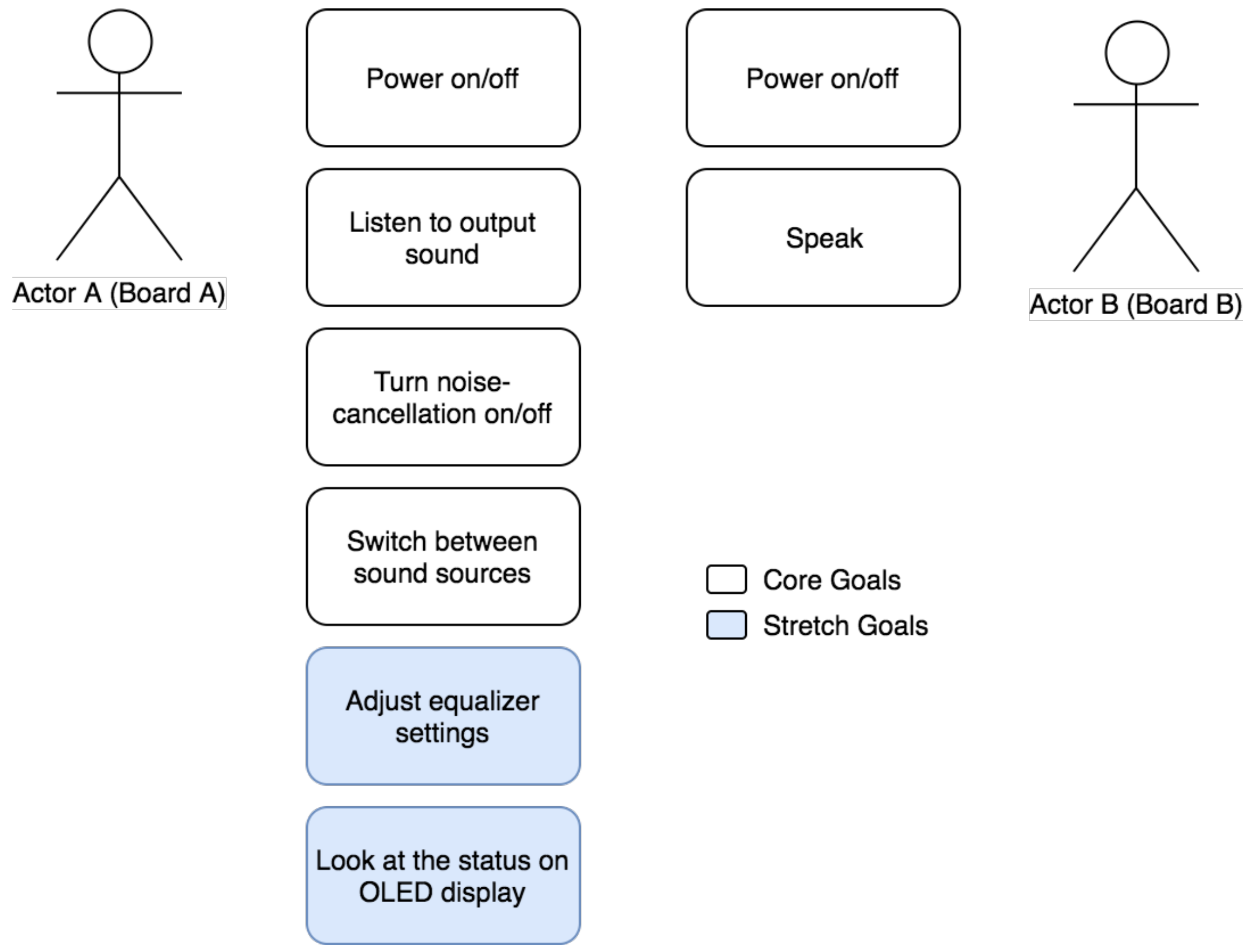
Power on/off
Listen to output sound
Turn noise-cancellation on/off
Switch between sound sources
Adjust equalizer settings
Look at the status on OLED display
Power on/off
Speak
Actor A (Board A)
Actor B (Board B)
Core Goals
Stretch Goals

Figure 3: Use Case Diagram

Requirements and Testing

Module	Requirement	Validation / Test
ANC	The LMS algorithm should have an average convergence time of less than 2 seconds	Observe convergence times of varied noisy environment and verify average convergence time
Audio Codec	The Line OUT should have consistent audio quality	There should not be any dropout in music/ speech signal with/ without noise cancellation

Audio Codec	The primary and reference mic should not have more than 1 ms delay between their recorded signals	Play a tone close to both the mics and calculate the delay between their recorded signals.
ANC	The LMS algorithm should not have a computation time of greater than 5 ms	Connect speaker to Line OUT, enable noise cancellation and play a single tone close to microphone and observe the delay in hearing the tone back
ANC	The FIR filter should be reliable	Develop a test bench for the filter and compare with expected output
SPM	The SPM should not get input from DDR3 when the operating system is writing to it.	The mutexes responsible for read/write to memory should have correct value
Linux	The audio stream should have a reliable connection of less than 1 drop/minute.	The stream will be run for a long duration and the dropouts measured
Linux	The audio stream should have a persistent connection.	The stream should reconnect when Ethernet cable is removed and reattached
Audio Codec	The speech mic on Board B should record audio correctly	Connect the input signal from mic to Line IN temporarily and observe the output signal
Linux	Have reliable Ethernet connection	Test Ethernet connectivity and observe the number of dropouts

Extended Requirements

1. Implement an Equalizer controlled by switches that would filter on/off frequency range associated with that switch. This equalizer filter would reside in the SPM module and process the input signal before it's added to the anti-noise wave.
2. Use the directional up/down push buttons to respectively increase or decrease the bounds of the Equalizer, from 20 Hz to 20 kHz. The directional left/right button would specify whether the upper or lower bound is to be changed.
3. Display current state of the system on OLED screen: whether the filtering is on or off, which input is selected, and equalizer settings.

Algorithm Flowchart

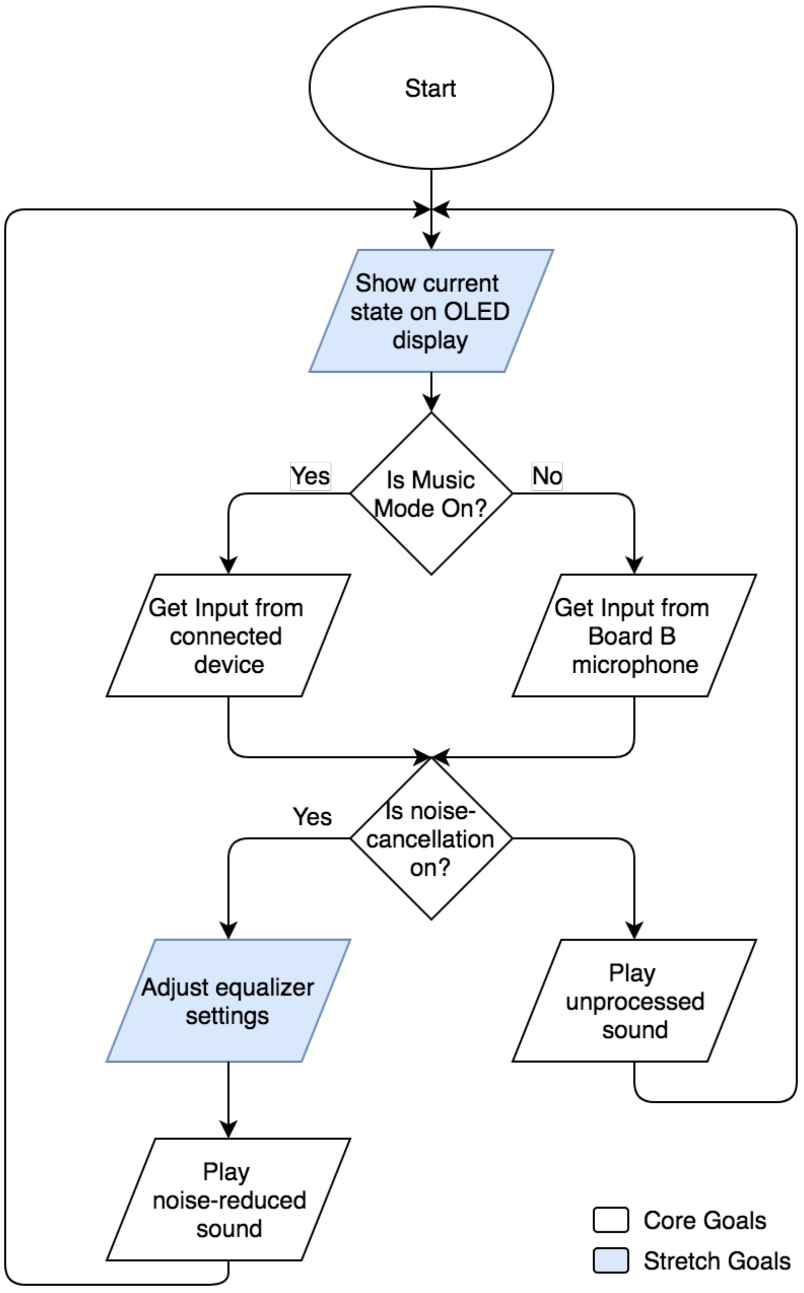
Start
Show current state on OLED display
Is Music Mode On?
Yes
No
Get Input from connected device
Get Input from Board B microphone
Is noise-cancellation on?
Yes
Adjust equalizer settings
Play unprocessed sound
Play noise-reduced sound
Core Goals
Stretch Goals

Figure 4: Algorithm Flowchart

Design Methodology

We will loosely follow Agile methodology while developing the project. Weekly milestones will serve as course-set sprints. We will establish internal deadlines two days before the course milestone reviews, which will allow us to test our work and ensure compatibility, while also giving us extra time to fix any unexpected problems. We will have regular face-to-face meetings at least once a week, not including the scheduled lab time, to discuss progress, issues, and adjust the project scope accordingly.
We will start work on the project with peripherals essential to our core requirements (such as microphones and headphones), as well as establishing Ethernet connection between the boards. After setting up peripherals and getting audio streaming from one board to another through Ethernet, we will proceed to implement the noise control algorithm.

Gur: I'm following an Agile development methodology for this project with a top down approach. I will develop the higher level components first as separate tasks with a soft deadline. Once developed, I will create the module connections to the bottom layer and such.

Lana: I will follow Agile methodology as much as it is applicable in a hardware project. I will start with building small parts of the project and connecting them to the parts developed previously as early as possible to ensure compatibility. I will thoroughly test every part of the project to catch bugs on early stage of the development. I will also be checking that the whole system is still performing as expected after every new change.

Division of Labor

In order to maximize learning for both team members, we decided to switch tasks every week. We divided the milestones in such a way that we both could have a complete understanding of the whole project and could potentially implement it independently after course completion. Every week after completing assigned milestones, we plan on thoroughly explaining our work to each other, including any problems we encountered and suggestions for making progress. By the end of the course both of us will have learned how to use various peripherals, setup Linux on FPGAs, connect FPGAs together through Ethernet, perform digital signal processing, and implement adaptive noise control.

Gur: ANC algorithm to find the optimal anti-noise waveform. Having always been interested in active noise cancellation in headphones, the process of it is a topic i'm interested to learn and implement myself.

Lana:
I will start on getting the output sound to work with VHDL first, and then try to do the same thing in Petalinux. I will work on adding PMOD mic to the FPGA and streaming audio signal from one

board to another through Ethernet. After that, I will implement Basic Filter and Equalizer. If time permits, I will also get OLED display to work so user can see the status of the system.
Learning needs:

1. Performing signal processing in VHDL. I have only tried implementing FFT in VHDL before, but I have never done anything else related to digital signal processing in hardware. I have taken multiple DSP-related classes, so I am familiar with some concepts, but trying to transfer the knowledge into VHDL would take some learning.
2. Streaming audio from one board to another through Ethernet. I don’t have any experience with setting up Ethernet, and at this point I am not sure what process we should follow to get audio stream from one board to another via Ethernet. I will definitely need to do a lot of research on the topic.

Project Milestones

Week 1

	Gur	Lana
Goal	Install Petalinux and set up Ethernet communication	Play sound over codec on Line OUT
Demo	We would create a file on one board and send it through Ethernet to another board	We would hear some pre-recorded sound coming out of the headphones connected to Line OUT
Purpose	Board-to-board communication for streaming audio	Preparation for playing any audio

Week 2

	Gur	Lana
Goal	Connect Line IN with Line OUT	Play a saved audio file from Petalinux on Line Out
Demo	We would hear a song played from a phone connected to Line IN from headphones connected to Line OUT on the same board	We would hear a saved sound after connecting headphones to Line OUT at either board
Purpose	Setting up receiving sound from a mobile device	Preparation for playing audio coming from another board through Ethernet

Week 3

	Gur	Lana
Goal	Connect Mic port to Petalinux on Board B	Connect PMOD MIC and Mic port to LINE OUT on Board A
Demo	We would be able to record an audio file using Board B Mic	We would hear a voice coming from PMOD Mic or Mic input on Board A
Purpose	Receiving audio from microphones for future ANC processing	Receiving audio from microphones for future ANC processing

Week 4

	Gur	Lana
Goal	Implement simple signal processor without filtering	Stream sound coming from Board B mic to Board A Line Out
Demo	Sound output will be a distorted version of signal input	On Board A we would hear sound coming in to Board B Mic port in real-time
Purpose	Learning the basics of signal processing implementation in VHDL	Preparation for performing ANC on speech coming from a microphone

Week 5

	Gur	Lana
Goal	Implement signal multiplexer (input toggle button)	Implement Basic Filter and add filter on/off button The Basic Filter would simply subtract current noise waveform
Demo	We would be able to switch between Board A headphones sound source using a toggle button: either sound from a Board A Line IN, or from Board B microphone	We would hear either a non-processed sound or a sound after noise cancellation, depending on the button press.
Purpose	Enable user selection of signal source	Provide audio feedback for the user

Week 6

	Gur	Lana
Goal	Implement Adaptive Filter	Implement Equalizer for a particular frequency
Demo	Results of noise filtering would be significantly improved and noticeable to a human ear.	We would input a sound file with known frequencies and play the output to demonstrate that certain frequency range gets modified (for example, filter out low frequencies).
Purpose	Core algorithm of the project	Improving output sound quality

Week 7

	Gur	Lana
Goal	Add switch control to the Equalizer	Connect OLED display
Demo	We would be able to adjust equalizer setting through interaction with switches and buttons and hear the difference.	OLED display would display current status: whether noise is currently being cancelled or not, which input source is selected, and what settings equalizer is currently at.
Purpose	Adding user interaction	Adding visual feedback for the user

Design Concerns

Gur: A potential pressure point for this project would be transferring speech from Board B to Board A and subsequently using it in the Audio Controller. Having never worked on any project involving hardware and software integration, I do not know what sort of problems to expect. To mitigate this, we will both actively work on this portion, realizing any issues faced by the other member.

Lana: My main concern is whether we would be able to implement the adaptive filter in a way that would provide a significantly noticeable noise reduction. Even though we found research on adaptive filters that were successfully implemented, I expect debugging to be quite a challenge. We expect it to take more than one week to complete, so even though we put Adaptive Filter Implementation close to the very end of our project, we plan on starting to work on the implementation well in advance.

References

[1] "Active noise control - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Active_noise_control. [Accessed: 23-Jan-2018].

[2] A. Di Stefano, A. Scaglione, and C. Giaconia, "Efficient FPGA Implementation of an Adaptive Noise Canceller," in *Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05)*, Palermo, Italy, pp. 87–89.

[3] S. Thilagam and P. Karthigaikumar, "Implementation of adaptive noise canceller using FPGA for real-time applications," in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India, 2015, pp. 1711–1714.

[4] Sinjari, A., Amory, R. and Alsaigh, R. (2015), Proposed Active Noise control System by using FPGA. *IJCSI International Journal of Computer Science Issues*, 12(3), pp.219-224.