# Sound Localization Project

Matheu Campbell (mgc2171), Dawn Yoo (dy2486), Peiran Wang (pw2593), Elvis Wang (yw4082)

February 27, 2024

## 1 Overview

In this project, the FPGA, interfaced with a microphone array, will map the movement of a sound source over time as it moves through a room, then visually represent the movement in an animation. The microphone array will lie stationary in the center of the room, and the audio source will be a wireless speaker that we manually move around.

Computation will take place in three phases. In the first phase, the FPGA will record data from the microphones and store it in memory. To simplify calculations, the source will be a single narrow-band high-frequency audio signal in an otherwise quiet room with minimal reverb. In the second phase, the FPGA will calculate the angular offset of the audio source with respect to the location of the microphone array. Finally, the angle vs. time data will be displayed on a screen as an animation.

### 1.1 Audio Source

To maximize spatial resolution, we will select a high frequency (low wavelength) audio signal. To keep the audio source within normal human hearing, we will limit the frequency to 20 kHz. The microphone arrangement will assume a frequency of interest of between 10 and 20 kHz.

### 1.2 Microphone Arrangement

The microphone array will contain 8 microphones. A sparse array like this will hopefully be just enough to broadly resolve the location of the sound.

To maximize spatial resolution, the minimum distance between any two microphones will be at least the wavelength that corresponds to our frequency of interest. Assuming a lowest possible target frequency of 10 kHz, the microphones should be at least 3.43 cm apart, but likely will be farther.

The microphones may take a linear or planar arrangement. We will start with a linear arrangement, fine-tuning as necessary.

### 1.3 Animation

After recording audio data, the FPGA will compute the angle over time data and display an animation. The animation will use standard VGA resolution with a 60 Hz frame rate and one bit per pixel. Playback would ideally show audio source movement at the speed it actually happened, but memory constraints will determine how long a recording sessions can be as well as how long the animation can be. The animation will show a top-down view of the room with microphones represented by dots. A cone centered at the array will show the calculated location of the audio source over the course of the recorded session.

## 2 Algorithms and Interfacing

### 2.1 Data Acquisition

Audio data will be recorded at the highest sampling rate that will permit at least a 20 second recording, keeping memory constraints in mind. At each sample, we aim to record data from all of

the microphones as close to simultaneously as possible, as the relative arrival times of the signals is crucial to being able to resolve the audio source.

Because we'll need many microphones, we'll connect them to the board via its GPIO pins. Both digital and analog MEMS microphones are available. In the design document, we'll consider which to choose in the context of the larger design (would we rather write communication protocol or work with the FPGAs ADCs and which fits more naturally into the larger system)?

Once we decide on a permanent microphone arrangement, we can build a housing to fix them in place and manage their cables. Before then, a breadboard will probably be sufficient.

### 2.2 Beamforming for Sound Localization

We will implement the delay-and-sum beamforming technique to evaluate the audio coming from a given direction. With this algorithm, we delay the signal in time received by each microphone according to its position in the array and the desired direction of sensitivity. Then, we sum the delayed signals to find the strength of the signal coming from the target direction. By repeating for all directions, we can determine where the source of an audio signal is. We will implement FFT and basic filtering on the FPGA, but we will assume a narrowband signal with negligible interference, allowing us to delay and sum the signals with minimal interference from other frequency components adding noise to the system.

The delay-and-sum algorithm will be computed as a multiplication of frequency content by complex coefficients in the frequency domain, and we will do it in C (not on the FPGA).

## 3 Major Tasks

The subsystems can be tested and combined incrementally.

1. **Microphone Array**
   Record and save data over a given time period, then playback to verify accuracy.
2. **Beamforming Algorithm**
   Implement and verify the beamforming algorithm in C with simulated data.
3. **Drawing and Animating**
   Animate an audio source path given simulated angle vs. time data.
4. **FFT and Filtering**
   Implement FFT and basic FIR filtering on the FPGA.

Figure 1: Progress Chart