# Sound Source Localizer

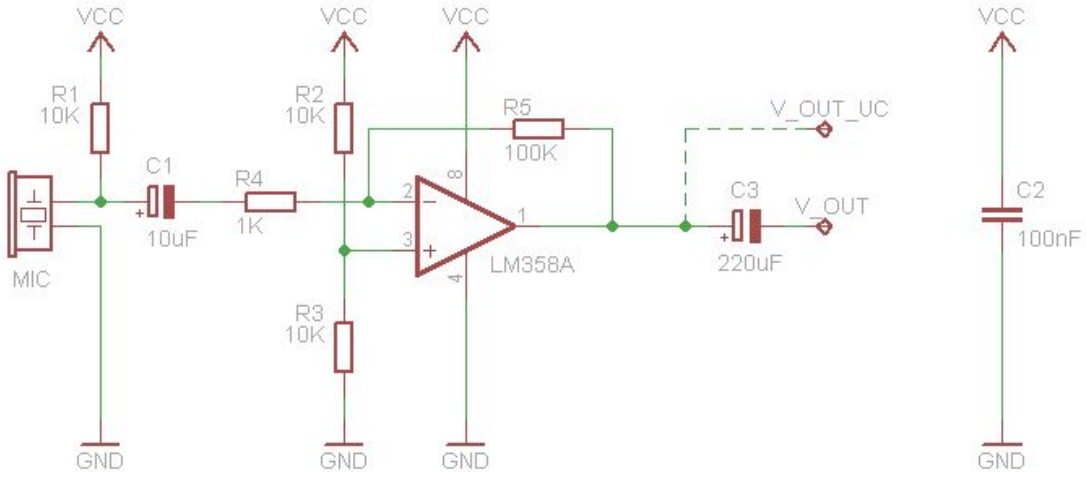
Joren Lauwers, Changping Chen

## Abstract

In our final project, we will build a sound source localizer - a system that positions the source of human speech on a 2d map. We will sample audio data at a high sample rate on the Nexys 4 FPGA board from three carefully positioned, cheap condenser microphones, and triangulate the source of the signal in real-time based on the difference in time of arrival of the same sound in each microphone. Five main components of the project are (a) setting up a pre-amplification stage for each of the microphones, (b) sampling signal using onboard ADC and filtering noise, (c) computing signal's delay between microphones by performing computation-intensive real-time cross-correlation in parallel, (d) triangulating sound source through trigonometry, and (e) visualizing the result via a VGA-rendered spotlight.

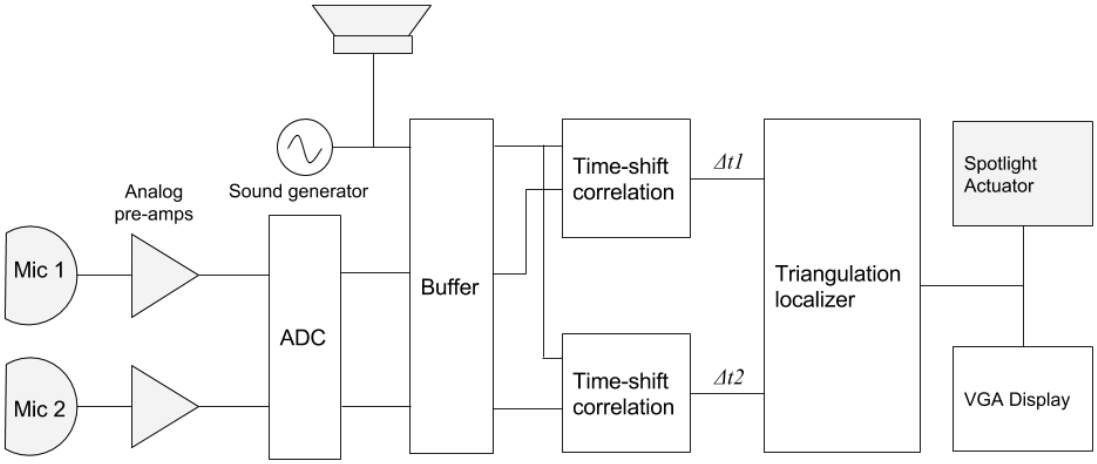
## Circuit: Audio Input Preamplifier

To sample the small audio input signals from microphones, we need a preamplification stage for each of the microphone signals. Here we will use a LM358 op-amp circuit from an online guide[1]. The exact circuit is reproduced below.




[1] https://lowvoltage.wordpress.com/2011/05/21/lm358-mic-amp/

# Modules




## Input Sampling with ADC

The Artix-7 FPGA on the Nexys 4 board features two 12-bit analog-to-digital converters (ADCs)[2] . The two ADCs can be configured to simultaneously sample two external-input analog channels, at a sample rate of 1MSPS. The sampling module will use this feature and sample the audio input from preamplifiers simultaneously on the same sampling clock $CLK_s = \frac{1}{t_s}$, and write the digital values to the sample buffers described below.

To test this module, we will drive the ADC input with two synchronized sine waves, where one has double the frequency of the other, examine the sampled digital values on logic analyzer. They should line up perfectly at a few infection points.

## Input Sample Buffer

An input sample buffer is composed of S sample registers of 12-bit audio input stored in FIFO order. Internally it should use a circular buffer. It should provide a write interface for accepting new sample appended to the tail, and a read interface for accessing the $n$'th sample stored in the buffer. Note that this buffer must be implemented using registers, since parallel reads are required for cross-correlation module below.

[2] http://www.xilinx.com/support/documentation/data_sheets/ds197-xa-artix7-overview.pdf, http://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

## Cross-correlation

This module calculates the cross correlation score (dot product) of two input signals when one is time shifted by a lag $m$ over a window size of W. It's a common method of comparing the similarity of two time series when one lags behind another. This module is used later to calculate the time delay between when a signal is sent and the same signal is received.

The cross correlation of two inputs A and B over a finite sample window of W where B lags behind A by m samples, is given by

$$C(n,\ m) = \sum_{k=0}^{W-1} A[n+k]B[n+k+m]$$

We want to calculate the cross-correlation score for each possible lag $m$, use the lag associated with the highest score to calculate the difference in time of arrival.

Notice that $C(n,m)$ depends on the future value $B[n+W-1+m]$ and the current value $A[n]$, so we need to store at least $m+W$ samples, or $S=m+W$. The requirement on $m$ is discussed in the next section.

Another implication from the equation above is that our score will be delayed by $S$ sampling cycles, but this delay is negligible in human perception.

**Example**

In the example below, we choose W=5 and S=10. Suppose there exists a signal

```
0  0  0  0  0  1  2  3  4  5  4  3  2  1  0  0  0  0
```

This signal arrives at mic 1 after a delay of 2 sampling cycles, and at mic 2 after 5 sampling cycles. At one point, the sample buffer for the two mics will contain:

```
Mic 1:  2  3  4  5  4  3  2  1  0  0
Mic 2:  0  0  1  2  3  4  5  4  3  2
```

Based on this dataset, the relative delay of signal received at mic 2 compared to mic 1 is 3 cycles. If we calculate the cross correlation score for $m \in [0,\ 5]$, we expect the score for $m=3$ to have be the highest. Here $C(n,\ 3) = (2)(2)+(3)(3)+(4)(4)+(5)(5)+(4)(4)$.

Observe that with each streaming input sample, the difference in the new and old correlation score for the same lag m is

$$\begin{aligned} C(n,\ m) - C(n-1,m) &= \sum_{k=0}^{W-1} A[n+k]B[n+k+m] - \sum_{k=0}^{W-1} A[n-1+k]B[n-1+k+m] \\ &= A[n+W-1]B[n+W-1+m] - A[n-1]B[n-1+m] \end{aligned}$$

Therefore, in each cycle, we only need two multiplications, one subtraction and one addition to compute the next value.

## Music Player

The music player module plays a pre-recorded snippet of songs through the onboard audio output, which is connected to a speaker at a certain distance away from the two microphones. It's important to have audio output synchronized to our sampling clock, to reduce randomized error in time delay calculation below. Our whole system's goal is to locate this music player.

## Delay Calculator

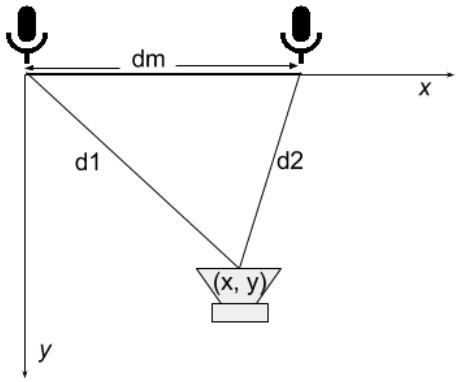
This module calculates the time delay of the same signal when sent or received at two different locations, which linearly relates to the distance between the locations through the speed of sound $c = 343.2\ m/s$. This delay is critical for sound source triangulation module to be discussed later.

In the cross-correlation module, we set $S = W + m$. In this particular application of delay calculation, we want $W$ sufficiently large to capture enough samples so that the same signals can be identified as such. Each increment of $m$ corresponds to one sampling period $t_s$. With $d$ being the maximum distance between the sound source and a microphone, we know the maximum time delay of a signal $max(t_d) = \frac{d}{c}$. Therefore, the same maximum time delay in terms of number of cycles $max(m) = \frac{max(t_d)}{t_s}$. Here, assuming $d = 1m$ and $t_s = 10\mu s$, we get $max(m) = 292$.

Based on the estimated distance above, this module instantiates 293 cross-correlation modules for the pair of $(speaker, mic1)$ and another 293 for the pair of $(speaker, mic2)$. The Artix-7 manual suggests there are at least 45 multiplier units on board, but pipelining is necessary to implement $293 * 2$ multiplications required by the cross-correlation modules per sample period. Lucky, the system clock rate is much higher than the sampling rate.

To calculate the distance given the sample delay $m$, we use $d = mt_sc$.

### Source Triangulation




With two distances calculated from the previous module, this module produces the $x$ and $y$ coordinate of the speaker in real-time. This triangulation is requires solving two trigonometric equations which are difficult to perform on FPGA, so we will make a lookup table to map pairs of distances to their respective coordinates.

## External Components

- 2x 3.5mm stereo jack female connectors
- 1x speaker