Adaptive Noise Cancellation

Sharvani Vadlamani (sv2734), Huda Jafri (shj2127), Sayem Kamal (sk5336)
Spring 2026

Introduction

The goal of this project is to build a noise cancellation system. It will take in two audio signals through the onboard audio CODEC: a microphone signal containing speech mixed with background noise, and a reference signal containing just the noise. Then, a hardware adaptive filter on an FPGA estimates and subtracts the noise from the microphone signal, outputting clean audio through the speakers in real time.

A VGA display shows scrolling waveforms so the user can see the noise being removed. Also the user can control aspects of the system through a USB keyboard.
This project will use:

- DE1-SoC board (onboard audio CODEC, VGA output, USB port)
- VGA monitor
- USB keyboard
- Microphone (connected to mic in jack)
- Audio cable (to connect noise source to line in jack)
- Headphones or speaker (connected to line out jack)
- Phone or laptop to play the noise source

Overview

The hardware side handles all audio processing. An audio interface module configures and communicates with the onboard CODEC. Audio samples from two input channels feed into the adaptive filter module, which computes the cleaned output and sends it back to the CODEC for playback. A VGA module renders waveforms to the display.

The software side handles the user interface. It reads keyboard input and writes parameter changes to hardware registers over the Avalon bus. It also reads audio samples from hardware to update the VGA display. The time-critical audio loop runs entirely in hardware with no software involvement.

Algorithm

How Noise Cancellation Works
The microphone picks up speech plus background noise. A cable from the noise playing device into the line-in jack provides a clean copy of just the noise. The adaptive filter takes this reference noise and learns to transform it to match the noise picked up by the microphone. Subtracting the filter's output from the microphone signal removes the noise and leaves only the speech.

Speech passes through because the filter can only predict things correlated with the reference input. Since speech is uncorrelated with the noise source, the filter cannot remove it.

Least Mean Squares (LMS) Algorithm
We will use the LMS (Least Mean Squares) algorithm, a well known adaptive filtering method. On every new audio sample, it performs three steps:

- **Predict:** Run the reference noise through an FIR filter to estimate the noise in the microphone signal
- **Subtract:** Remove the estimated noise from the microphone signal. The result is the cleaned output
- **Adapt:** Adjust the filter coefficients to reduce the prediction error for the next sample

The algorithm has two main parameters: the filter length (how many taps in the FIR filter) and the step size (how aggressively the coefficients adapt). A longer filter can model more complex noise patterns. A larger step size converges faster but risks instability. The user can adjust the step size at runtime.

Resource Budget

The filter requires a small amount of on chip memory to store its coefficients and a buffer of recent reference samples. The VGA display requires three sample buffers for the three waveforms.

The main computational cost is the multiply accumulated operations in the filter. We plan to use a small number of parallel MAC units built from the FPGA's DSP multiplier blocks to ensure the filter computation completes within one audio sample period. This will use only a small fraction of the available resources.

User Interface

The VGA display shows three scrolling waveforms stacked vertically: 1) the noisy input, 2) the estimated noise, and 3) the cleaned output. A small text overlay shows the current step size and whether bypass mode is active.

The user has two keyboard controls:

- **Toggle bypass**: switch between noisy passthrough and filtered output
- **Adjust:** The adaptation speed

For the demo, a phone plays a steady noise through its speaker near the microphone while a cable from the phone's headphone jack runs into the board's line-in. The operator speaks into the microphone, and the audience hears cleaned audio through headphones connected to line-out. The user can toggle between noisy and clean output, making the effect immediately obvious.

Major Tasks

- **Audio CODEC interface:** configure the CODEC and get basic audio passthrough working
- **LMS adaptive filter module:** implement the filter in hardware and verify in simulation
- **System integration:** Connect the filter to live audio and verify noise cancellation on real signals
- **VGA waveform display:** Render three scrolling waveforms with parameter overlay
- **User interface:** keyboard controls for bypass toggle and step size
- **Final improvements:** tune filter parameters

Further Improvements/Limitations

- Normalized LMS (NLMS) automatically scales step size based on how loud the input signal is, so it works well on both quiet and loud noise without manual adjustment. May require additional resources.
- If time permits, we can find a way to only update coefficients when the user is not speaking. This is because when you speak, the error signal contains your voice, and the filter may try subtracting your voice