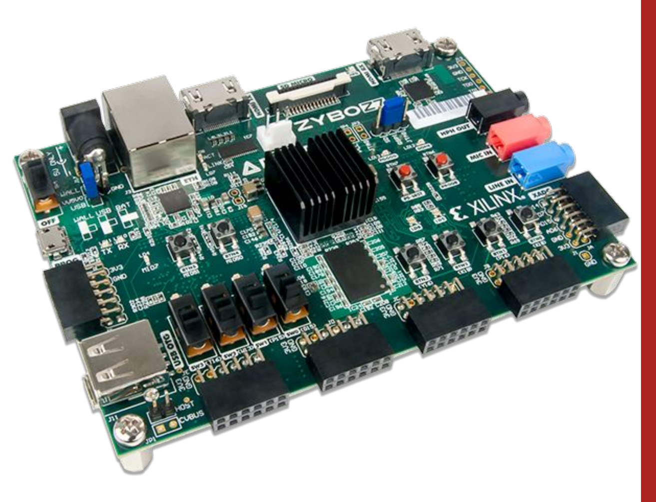


FPGA-based Robotic Effect Voice Changer

Author(s): Jiacheng Tu (jt842), Shuzhe Liu(sl2973), Xiangzhou Wei (xxw2)

Advisor: Hunter Adams

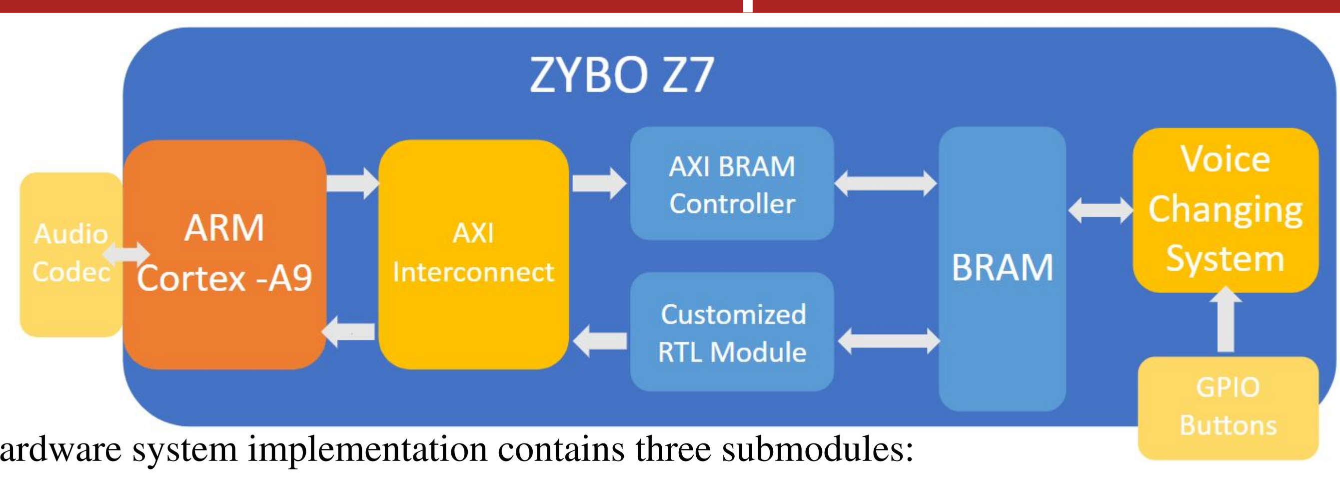
FPGA-Based Vocoder: Improving Privacy & Latency

Our project developed a real-time robotic voice changer with pitch-shifting capabilities, implemented on a Zynq-7010 ARM/Field Programmable Gate Array (FPGA) System on Chip Development Board.

This technology enhances privacy in the AI era by allowing users to mask their natural voices during voice chats or calls, thereby adding an extra layer of **security**.

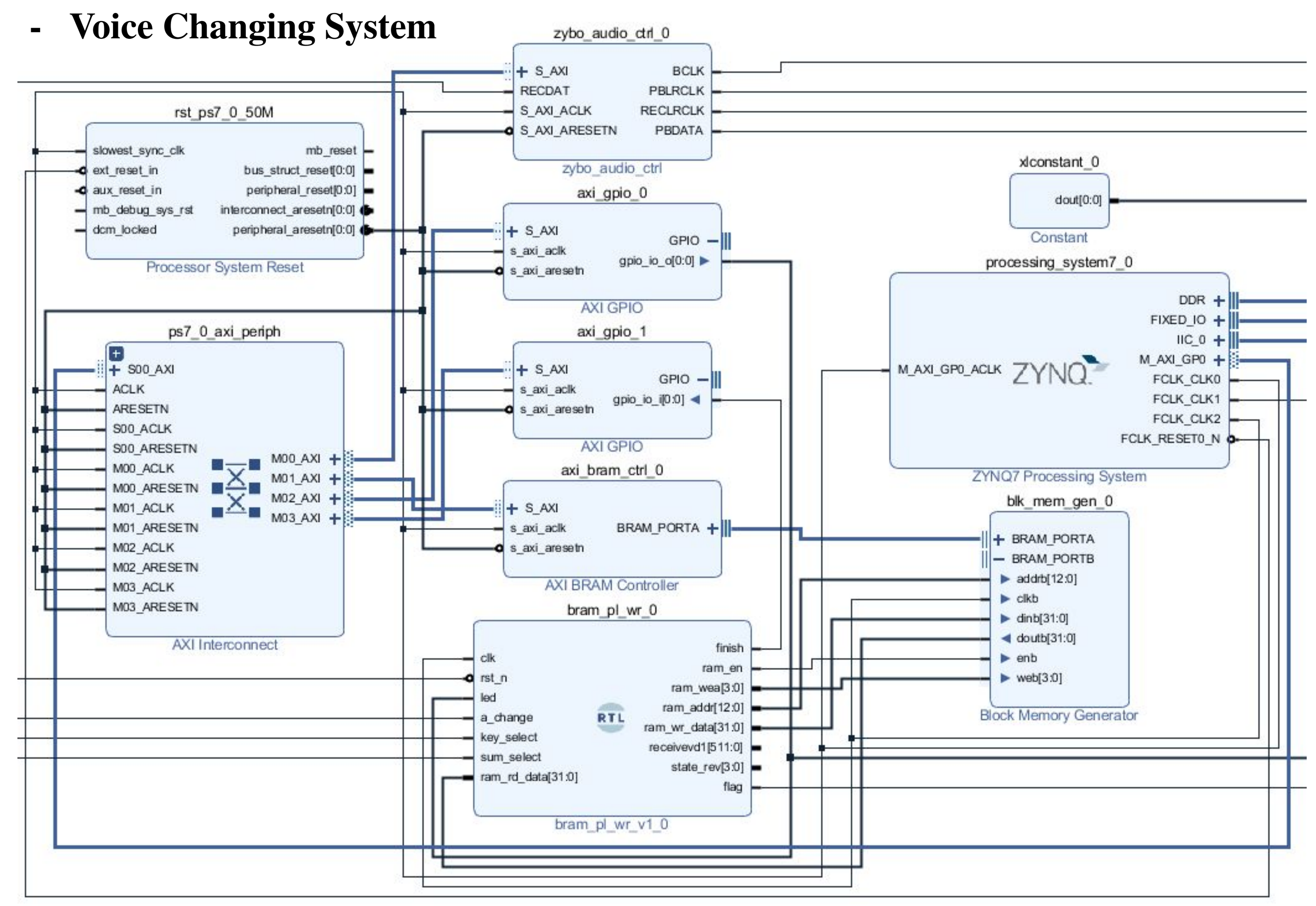
Additionally, it **compresses** sound and can generate robotic and futuristic effects, useful in the **music** industry and for **entertainment**. Leveraging FPGA's **parallel processing capabilities**, our system offers improved speech processing performance and **reduced latency**.

Hardware Implementation



Hardware system implementation contains three submodules:

- **Audio Codec Driving Controller**
- **Data Loopback Communication Module**
- **Voice Changing System**



A high-quality stereo audio codec, managed by a High Performance Computing (HPC) processor, uses Inter-Integrated Circuit Sound protocols with 24 kHz sampling rate.

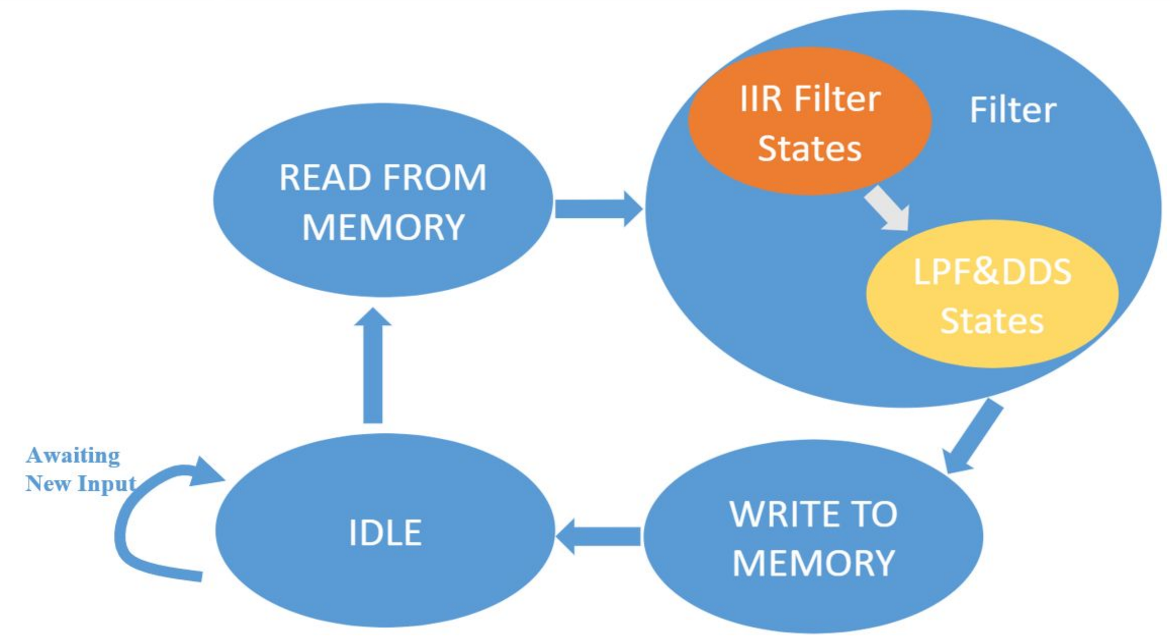
The Block RAM (BRAM) connects to the Advanced eXtensible Interface (AXI) Master Bus to facilitate audio data transfer between the HPC processor and FPGA, utilizing an AXI BRAM Controller and a customized Register Transfer Level (RTL) design module as a bridge.

The Voice Changing System comprises three stages: Infinite Impulse Response (IIR) bandpass filters, Low Pass Filters (LPF), and Direct Digital Synthesis (DDS) to add a robotic effect to audio. General Purpose Input/Output (GPIO) buttons are used to adjust pitch shifts for various sound effects.

CornellEngineering

Electrical and Computer Engineering

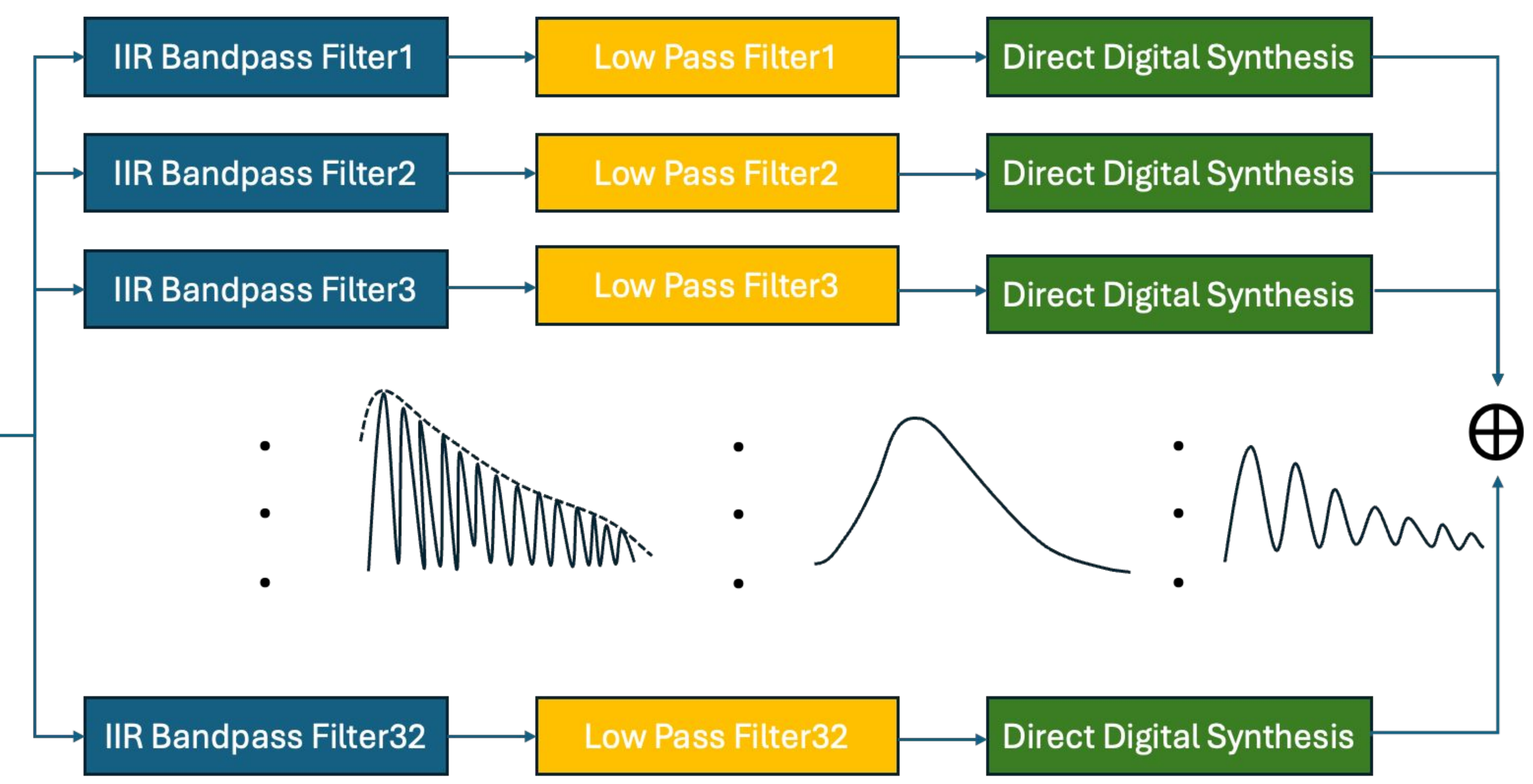
System Control Flow



The Finite State Machine (FSM) on the FPGA starts in the IDLE state, awaiting new audio data input storage in BRAM. It then reads the data and processes it through internal state machines for IIR Filters, Low Pass Filters, and Direct Digital Synthesis within the Top-Level Filter state.

After processing, the audio data is written back to BRAM, and the FSM returns to the IDLE state, sends a ready signal to the HPC processor, and waits for the next input.

Voice Changing System

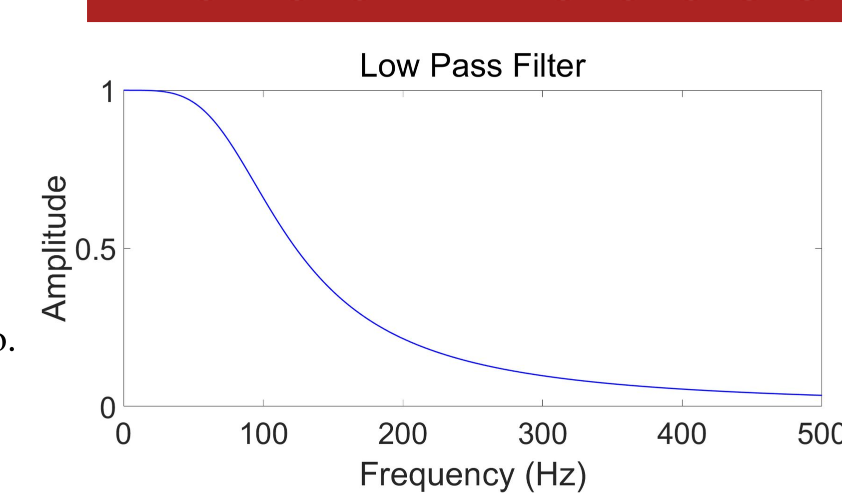


To modulate voice, an audio jack connects an audio source to the development board, where the sound is stored in BRAM. The input is processed through **IIR band-pass filters** to create the **Mel Frequency spectrum**, separating the sound into various frequency bands to extract voice features.

The **Low Pass Filter** restricts the frequency of changes in sound, resulting in a smoother sound signal. Then, **Direct Digital Synthesis** recreates a sound recognizable to the human ear by carrying the sound across frequency bands. This allows pitch shifting by adjusting the carrier frequency.

Once encoded, the sound is stored in BRAM for output through speakers.

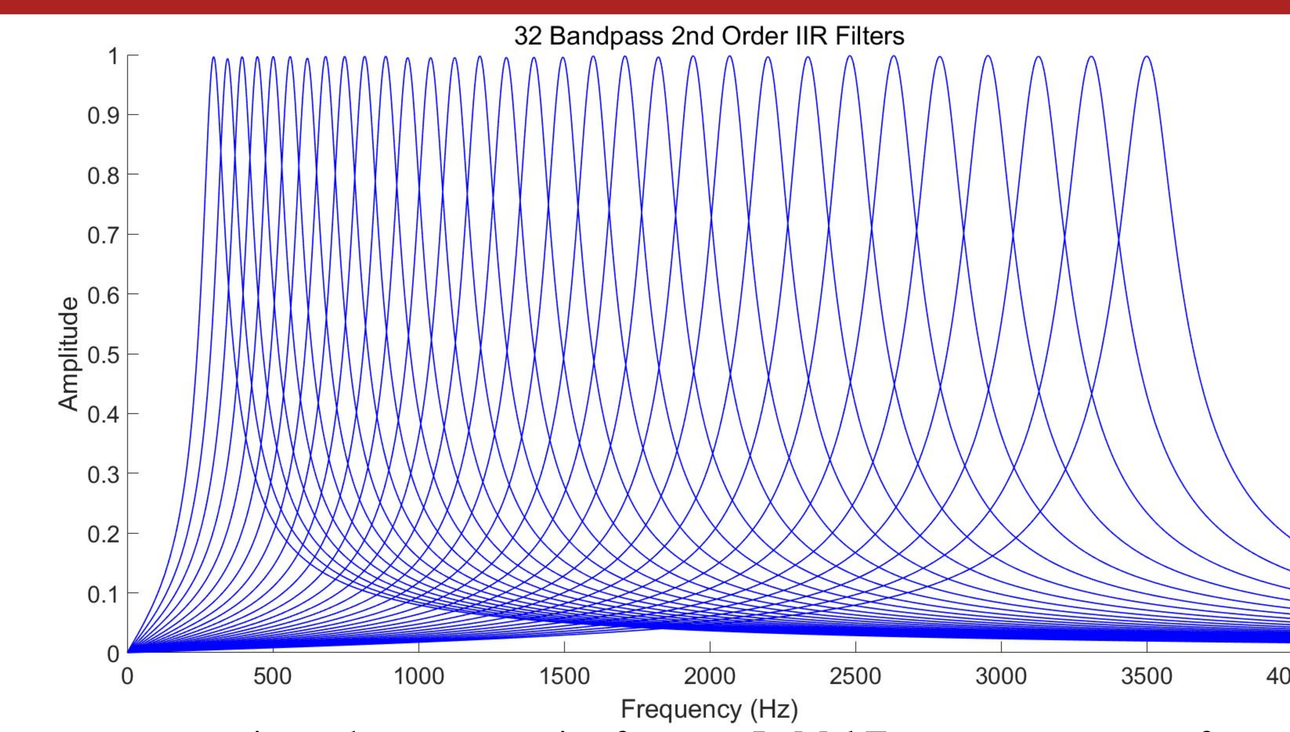
Voice Processing & Synthesis



A low-pass filter used after an IIR band-pass filter controls the rate of frequency change in the audio output, matching the natural limits of human vocal changes. This ensures that the synthesized signal's variation does not exceed the human voice's maximum rate of change. Testing shows that a cutoff frequency around 100 Hz effectively aligns with these typical rates.

To synthesize sound, we use Direct Digital Synthesis with a sine wave lookup table. This allows us to generate carrier waves at different frequencies. The frequencies are set by incrementing through the lookup table, with each increment amounting corresponding to an IIR filter's center frequency. Changing the increment amount lets us adjust the pitch.

Voice Feature Extraction

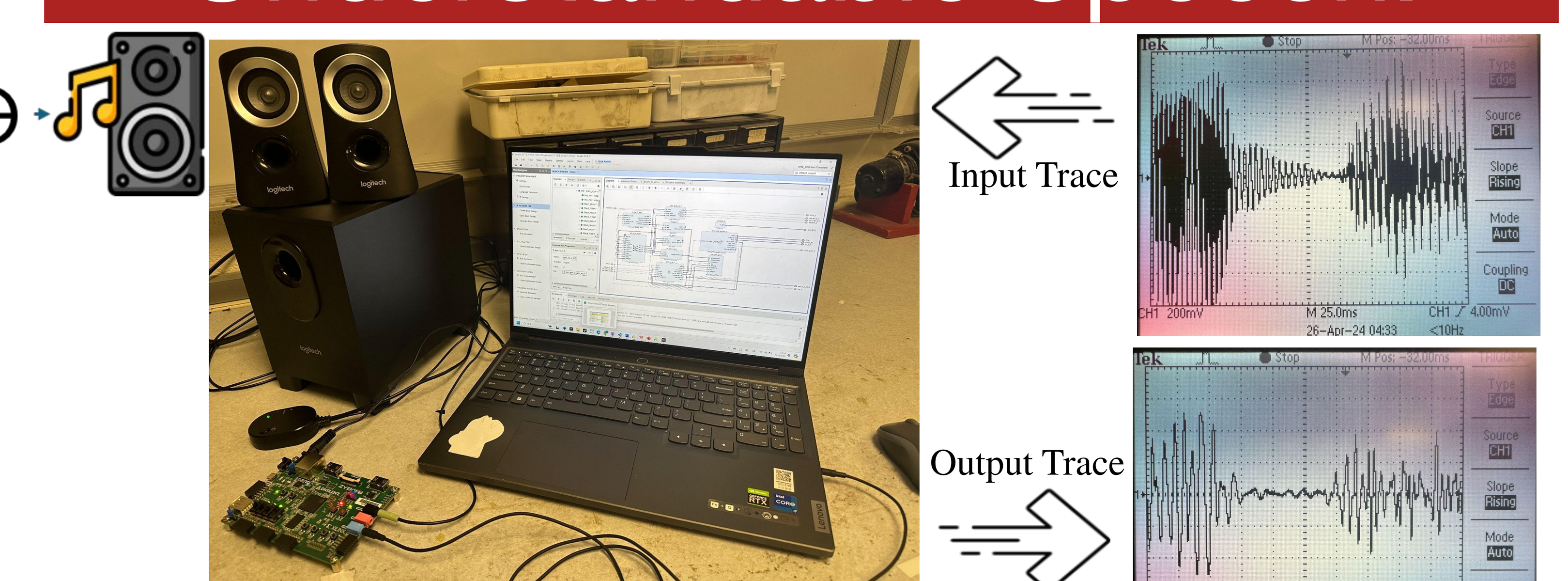


Mel Frequency spectrum is used to extract voice features. In Mel Frequency spectrum, frequency bands are evenly spaced on the Mel scale, which aligns more closely with the human auditory system than linear frequency spectra. Since the human ear is more sensitive to lower frequencies, this method provides a more accurate representation of sound. The formula provided converts between the Mel scale and normal frequency. In our setup, the Mel scale (m) features uniformly spaced values. The diagram above shows the spectra for the 32 band-pass filters we employed.

$$f = 700 * (e^{\frac{m}{1125}} - 1)$$

We use second-order IIR band-pass filters to isolate frequencies at their calculated Mel scale centers. Each filter's edges are defined where the amplitude drops to about 50% of the peak at the center frequency. MATLAB is used to compute coefficients for all 32 filters and to automatically generate Verilog code for them.

Understandable Speech!



The figure above shows our final system, consisting of our Zybo Z7, a speaker and a personal computer, with output port of the codec connected to the speaker and the line-in connected to the computer. As human voice is input into our board, the speaker plays the robotic effect version of the voice.

The figure on the right side shows the input and output trace of the sound wave speaking "I am a robot". It can be observed that the robotic effect sound output matches our expectation shown in the central figure of this poster: the output trace is more compressed but still carries the same message: "I am a robot".

Acknowledgements

Special thanks to V. Hunter Adams for his consistent mentorship and guidance throughout this project, consistently helping us blueprint the project and giving us valuable technical advice.

Thanks to Bruce Land for his suggestions on how to improve the project and advice on our voice changing system.

[1] "DSP," *people.ece.cornell.edu*. https://people.ece.cornell.edu/land/courses/ece5760/DE2/fpgaDSP.html

[2]"Zybo Z7 DMA Audio Demo - Digilent Reference," *digilent.com*. https://digilent.com/reference/programmable-logic/zybo-z7/demos/dma-audio (accessed Apr. 22, 2024).

[3] "DDS", vanhunteradams.com, https://vanhunteradams.com/DDS/DDS.html