VOXOS: Vocoder on FPGA

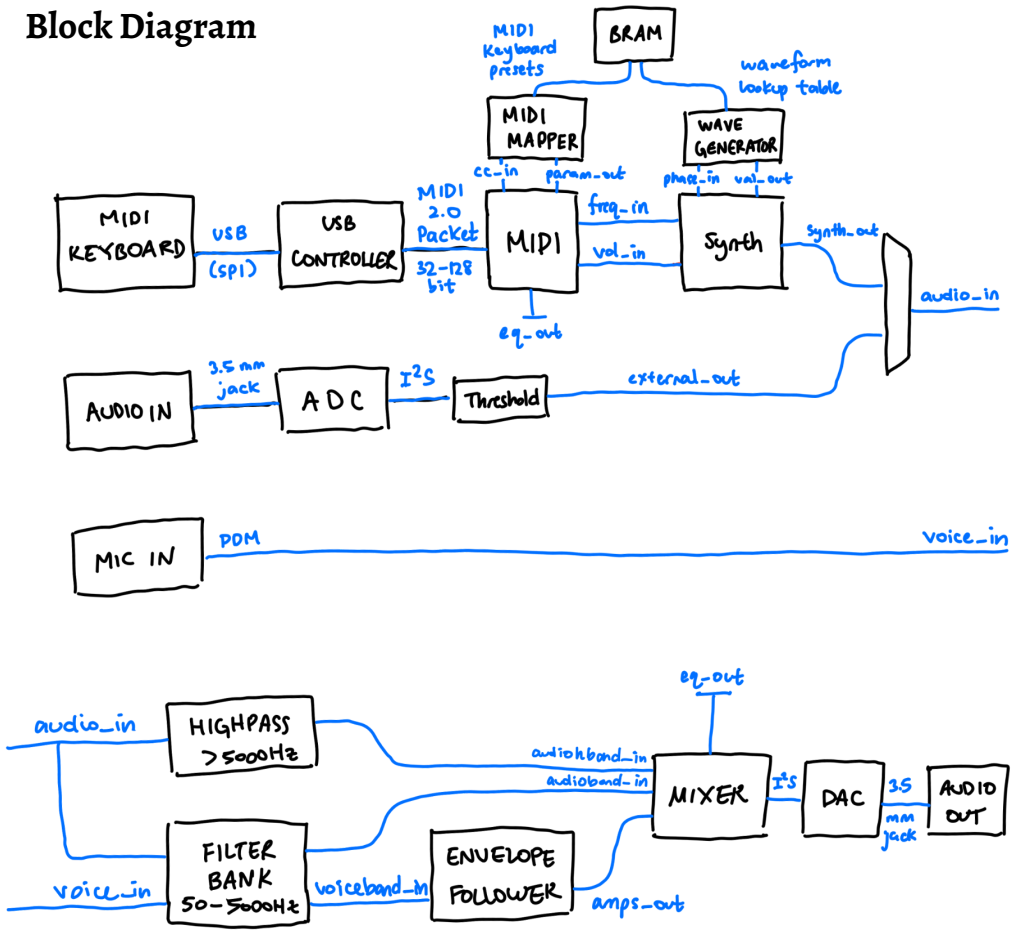
Block Diagram Report

Andrew Li (andyli@mit.edu)

1 Abstract

Originally meant as a compression device for analog voice data, vocoders have seen wide use in popular and electronic music as a creative tool to impart speech characteristics onto synthesized audio. I present VOXOS, a synthesizer and vocoder on an FPGA. VOXOS will be able to plug-and-play with any MIDI controller to control an onboard synthesizer or consume analog audio input. This input is then shaped according to the voice from the FPGA's built-in microphone. The clarity of speech reproduction depends on the number and quality of bandpass filters that partition the voice frequency band. Therefore, I aim to demonstrate the ability of the Spartan7-50's 120 DSP slices and to build something useful for music production (I can't sing).

2 Block Diagram



Top: Input stage from analog in or the onboard synthesizer controlled via a partial MIDI implementation

Bottom: Output stage where audio and mic input pass through a customizable filterbank and are mixed to produce the vocoded result

3 Design Goals

For VOXOS, I have a few goals that guided the design. In particular, the synthesizer should:

- Comfortably produce all notes on the standard piano (27.5 Hz to 4.186 kHz)
- Support ASDR (attack-sustain-delay-release) envelope shaping
- Support vibrato and pitch bend of ±2 semitones
- Support generating sine, triangular, and square waves

I chose to implement subtractive synthesis due to familiarity and ubiquity. As a result of these requirements, I'll implement a portion of the MIDI spec:

- Note numbers 21 - 108 corresponding to the standard 88 piano keys
- Channel numbers will be ignored
- Well-known Continuous Control (CC) messages for modulation, pitch-bend, and ASDR envelope control

Finally, to push the FPGA, I'll start with the following guidelines for the voice processing stage:

- Should capture the wideband voice frequency range (50-5000 Hz)
- Should support at least 10 bandpass filters, which is comparable to the Roland VP330, a commercial musical vocoders from 1980

3.1 Stretch Goals

- **Synth:** Support generating customizable waveforms, such as sawtooth and ramp
- **MIDI:** Support generic CC messages for filterbank EQ that can be mapped per controller and stored
- **Voice:** Support up to 20 bandpass filters, with ability to switch between the number for lo-fi or hi-fi output

4 Overview

4.1 Clocking

Like in Lab 7, the entire system will be run on one clock domain, 98.3 MHz. This allows VOXOS to generate 11-bit samples at 48 kHz for higher-quality audio. I might experiment with higher bit depths for better audio. This limits the signal-to-noise (SNR) ratio to ~68 dB, which is right around the SNR of the onboard PDM microphone and still allows us to reproduce the entire audio spectrum. As a result, I have a 2048 clock cycles between audio samples to process the current MIDI input, apply the desired envelope, filter, and mix the signals.

4.2 Input Stage

I will implement a USB controller that communicates via SPI with the onboard USB controller to provide 5V to the MIDI controller, and to form valid Universal MIDI 2.0 packets for the downstream module. These range from 32 to 128 bits depending on type. The MIDI module decodes these packets and decides the frequency `freq_out` to generate based on note and CC pitch-bend and modulation messages. It also recognizes CC ASDR envelope messages, which allows the MIDI module to decide on the volume `vol_out`. As a stretch goal, the module would call the MIDI Mapper to lookup the meaning of MIDI controller-specific sliders that output generic CC messages in BRAM. This would return a one-hot encoding of parameter type, such as EQ for each voice bandpass filter or the number of bandpass filters used. In this way, arbitrary keyboard sliders can hook into the vocoder ecosystem.

MIDI CC #	MIDI CC PURPOSE	VALUE	MIDI CC DESCRIPTION
MIDI CC 0	**Bank Select** (MSB)	0-127	Allows user to switch bank for patch selection. Program change used with Bank Select. MIDI can access 16,384 patches per MIDI channel.
MIDI CC 1	**Modulation Wheel** (MSB)	0-127	Generally this CC controls a vibrato effect (pitch, loudness, brighness). What is modulated is based on the patch.
MIDI CC 2	Breath Controller (MSB)	0-127	Oftentimes associated with aftertouch messages. It was originally intended for use with a breath MIDI controller in which blowing harder produced higher MIDI control values. It can be used for modulation as well.
MIDI CC 3	Undefined (MSB)	0-127	
MIDI CC 4	**Foot Pedal** (MSB)	0-127	Often used with aftertouch messages. It can send a continuous stream of values based on how the pedal is used.
MIDI CC 5	Portamento Time (MSB)	0-127	Controls portamento rate to slide between 2 notes played subsequently.
MIDI CC 6	**Data Entry** (MSB)	0-127	Controls Value for NRPN or RPN parameters.
MIDI CC 7	**Volume** (MSB)	0-127	Controls the volume of the channel.

Figure: a selection of MIDI CC messages. Some have well-known meaning, others are available for mapping. Source

The Synth module then calls the Wave Generator, which is essentially a 11-bit lookup table to produce the wave value given a phase. As a stretch goal, the Generator would call BRAM to lookup these values in

memory, allowing configurable waveforms. This value is scaled by the `vol_out` to produce the synthesized sound.

Besides the onboard synthesizer, VOXOS also support analog audio input for a wider sound palette. After going through the ADC, the input is thresholded so that the vocoder does not produce sound from background noise.

4.3 Output Stage

Vocoders extract the strength of the voice in each frequency band and amplify the source audio in those bands by those extracted strengths. The result is voice-sounding audio, whose clarity is improved with more frequency bands.

In the output stage, VOXOS uses a filterbank to isolate these frequency bands. I'll start with 10 biquad bandpass filters that split the 50 Hz - 5 kHz band exponentially. In particular, biquad filters provide the steep rolloff and high-Q that is needed for clarity in each voice band, and are easy to implement. The system will store the previous two samples and two outputs in BRAM, and use its Direct Form I topology with fixed-point arithmetic by scaling the samples.

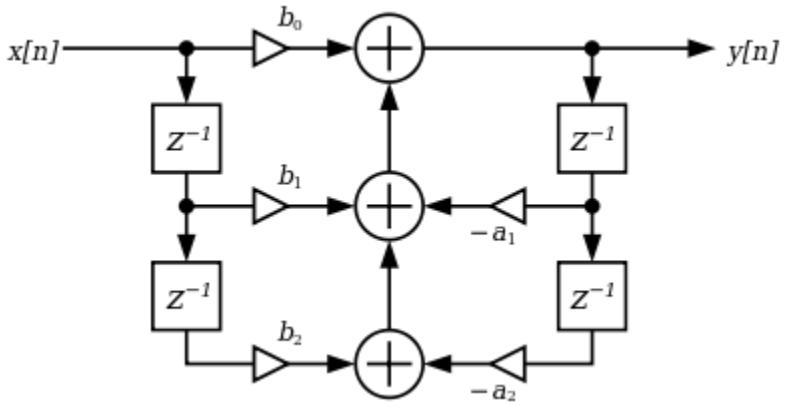


Figure: Biquad filter represented graphically, where the previous two inputs are scaled and added, then the last two outputs are subtracted. Source

Each audio and microphone sample, `audio_in` and `voice_in`, will run through these biquad filters. `audio_in` will additionally run through a separate highpass to capture audio characteristics above the voice frequency band. Depending on the limited number of DSP slices, filtering can also be performed on each sample for each band serially by loading each filters' coefficients from BRAM.

Next, each voice sample from each band `voiceband_in` will run through an Envelope Follower module to produce the strength of each band. This is essentially a lowpass filter on the magnitude of the input. Finally, these samples are all normalized and mixed, potentially as mentioned before, according to some EQ parameters and sent to a DAC.

4.4 Memory Requirements

There are lots of components to VOXOS that require memory:

- Waveform lookup tables
- **Stretch:** Additional waveforms
- Note frequency lookup table
- **Stretch:** MIDI CC message type mapping
- Previous inputs and outputs for biquad filter
- Biquad coefficients

This is a rough estimation of the memory I'll need:

Purpose	How much?	Assumptions
Waveform LUT	164 Kb	16-bit wave values, 2048 11-bit phase values, at most 5 waveforms
Note Freq. LUT	3 Kb	128 possible MIDI notes and 20-bit frequency values: scale everything up 100 and operate only on integers for cent-precision, the maximum MIDI note corresponds to ~8 kHz, so we need to fit up to ~800 kHz. 20 bits gives us just over 1 MHz.
Biquad Prev. I/O	>1 Kb	11-bit audio samples, 4 audio samples per filter, at most 20 filters
Biquad Coeffs.	3 Kb	32-bit coefficients (scale to eliminate decimal), 5 coefficients per filter, at most 20 filters
MIDI CC Mapping	3 Kb	128 possible MIDI CC values, at most 5 bits to encode 32 parameters to map to (at most 20 filters to change EQ of), at most 5 MIDI controllers mapped at a time

This gives a rough upper bound of ~256 Kb which comfortably fits in the provided 2 Mb BRAM onboard. Size requirements might change depending on audio bit-depth, but due to clocking constraints that may not be viable.

5 External Hardware

For this project, I'll be testing on a ubiquitous Akai MPK Mini controller:

I'll need an audio ADC for analog input, preferably a breakout board with a 3.5mm jack and I2S support.

6 Summary

VOXOS aims to be a mostly-featured and configurable synthesizer and vocoder. My minimum viable design accomplishes most of these goals and is extensible to accommodate stretch goals. Further, it will highlight the DSP capabilities of the FPGA by performing high-quality audio synthesis, filtering, and mixing while also being a fun instrument to play and integrate.