PDM-to-PCM Microphone Signal Conversion on FPGA Using CIC and FIR Filters

Yatian Liu
University of Michigan
Ann Arbor, US

Julia Lanier
University of Michigan
Ann Arbor, US

Kevin Fu
University of Michigan
Ann Arbor, US

Alanson Sample
University of Michigan
Ann Arbor, US

***Abstract*—Digital microphones are one of the most common sensors used in people's daily life. MEMS digital microphones usually output in pulse-density modulation (PDM) encoding and the output needs to be converted to pulse-code modulation (PCM) encoding for further analysis. This essay presents a PDM-to-PCM signal conversion system implemented on an Intel Arria V GX FPGA. The conversion is done using CIC and FIR filters, and the system can currently receive data from two microphones at the same time and output the converted PCM signals on GPIO pins. The code can be easily extended to support a larger microphone array in the future.**

I. Introduction

Digital microphones are one of the most common sensors used in people's daily life. Of all types of digital microphones, MEMS microphones are becoming more popular because of their smaller size compared to traditional condenser microphones. Pulse Density Modulation (PDM) is the most common used output interface of MEMS microphones since it requires minimal signal processing on the microphone and makes microphone synchronizing easier [1]. However, PDM signals are not as useful as the more commonly used Pulse Code Modulation (PCM) signals for analyzing and reproducing the input sound, so the output from PDM digital MEMS microphones needs to be converted to PCM signals before being further processed. Traditionally, PDM-to-PCM signal conversion is done by software codecs or Digital Signal Processors (DSPs). However, in recent years, FPGAs are preferred over DSPs in many research utilizing large arrays of digital MEMS microphones. FPGAs are advantageous over DSPs when processing data from a large microphone array, offering more IO ports and enabling a higher level of hardware parallelism [2]. This article demonstrates one such FPGA PDM-to-PCM conversion system using Cascaded Integrator-Comb (CIC) filters and Finite Impulse Response (FIR) filters, implemented on an Intel Arria V GX FPGA by our group. The current system can process data coming from two PDM digital microphones at the same time and output PCM data using GPIO pins. Data output to computers using Ethernet is currently being developed, and extending the current system to a larger array of microphones is another future goal.

II. Background

A. Pulse Code Modulation and Pulse Density Modulation

Pulse Code Modulation is the most common digital audio form. Most audio file formats and the Compact Disc (CD) all use PCM for encoding. In a PCM stream, the amplitude of the analog signal is sampled at uniform intervals, and each sample point is quantized to the nearest representable digital value [3]. Linear PCM is a specific type of PCM that quantize the analog signal using linear, uniform steps. It is also the most commonly used type of PCM. Some MEMS digital microphones output PCM signal using the Inter-IC Sound (I2S) interface, but the proportion of them is still small today due to the more complex hardware structure to produce a PCM signal on a serial bus.

Pulse Density Modulation represents analog signals using a radically different way. It only has a bit depth of 1, and does not quantize the analog signal directly to the nearest representable value (0 or 1 here). Instead, it uses delta-sigma modulation to create a signal whose density represents the strength of the analog signal [4]. By using a higher sampling rate and utilizing the error diffusion property of delta-sigma modulation, it can push the noise into higher, unused frequency ranges and preserve details in lower, desired frequency ranges. To sample audible sound with frequency ranging from 20 Hz to 20 000 Hz, PCM encoding will use a sampling rate of over 40 000 Hz from the Nyquist Sampling Theorem and a typical rate is 48 000 Hz. However, to achieve about the same audio quality, PDM encoding will use a much higher sampling rate, usually over 2 MHz, and one real-life example is the 2.8224 MHz used by Direct Stream Digital (DSD). A comparison between PCM and PDM encoding of the same sine wave is shown in Fig. 1. The noise distribution of a 2 MHz PDM encoding of a 1 kHz sine wave is shown in Fig. 2. We can see from the figure that in the 20 Hz to 20 000 Hz range the amplitude of noise is less than 10^{-2} of the amplitude of the 1 kHz input signal, which means more than −40 dB weaker volume.

Since PDM encoding represents an analog signal's amplitude in density and has significant noise in higher frequency, it cannot be used to directly analyze the input signal's amplitude properties or for playback. Superposition of signals is also hard. Therefore, PDM encoding is only used in MEMS microphones for simpler encoding hardware and smaller size, but the resulting signal needs to be converted to a PCM signal for further processing. The process usually includes decimation (i.e. down-sampling) and low-pass filtering, getting a signal with lower sampling rate and removed high frequency noise.

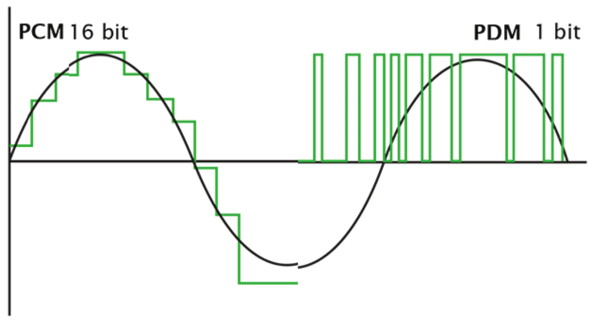


Fig. 1: Comparison of PCM and PDM encoding.

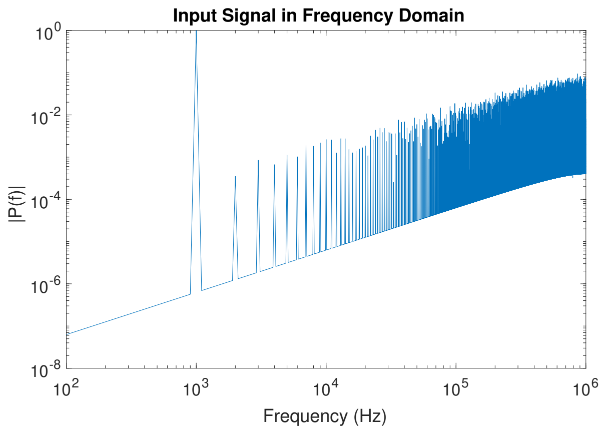


Fig. 2: FFT of a 2 MHz PDM sample of a 1 kHz sine wave.

B. FPGA in Microphone Array Signal Processing

The PDM to PCM conversion is usually done by specific Digital Signal Processors (DSPs) or software codecs. These methods are in particular good for a small number of microphones since the amount of data needed to be processed is limited and do not require very high efficiency. However, in researches utilizing a large PDM microphone array, Field-Programmable Gate Array (FPGA) is considered as an alternative way to implement PDM to PCM conversion in recent years[2].

To implement a real-time signal processing system for a large microphone array, there is a greater computation demand. FPGAs are suitable for this purpose since they can be used to built highly customized circuits and it is easy to instantiate multiple instances of the same hardware on them using Hardware Description Languages. If we want to implement PDM to PCM conversion for a 50-mic microphone array on an FPGA, we can just instantiate one filter for each microphone and save the aggregated output to internal or external storage devices. On the other hand, if the same system is implemented using specific DSP chips, we may need a separate chip for each channel and a dedicated PCB is needed for the interconnections of all the chips, making the design process longer and the final product less flexible. Software codecs running on general purpose processors will also have a inferior performance than FPGA-based solutions since it is very hard to achieve a high level of hardware parallelism. In addition, due to advancements of FPGA fabrication technology, more hardware resources are available on a single FPGA chip, which enables the possibility to implement the whole signal processing system on one chip. All these advantages of FPGAs in large microphone array's signal processing make them the preferred choice in recent researches.

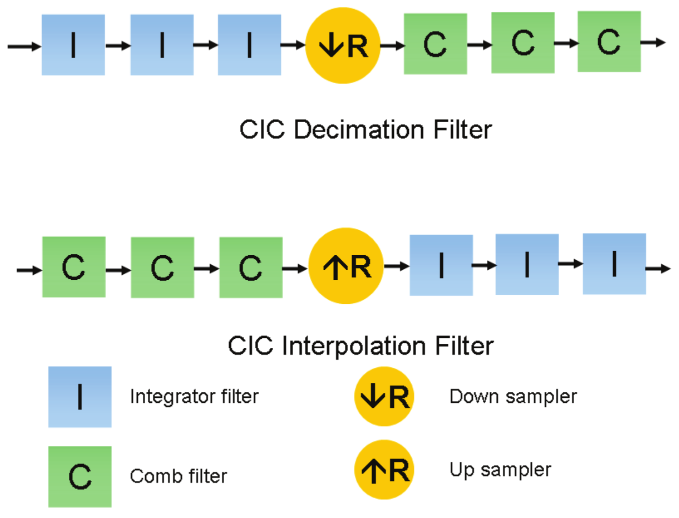


Fig. 3: Structure of CIC decimation and interpolation filters.

C. Cascaded Integrator-Comb Filter

Cascaded Integrator-Comb (CIC) filters are a special class of FIR filters used in multi-rate digital signal processing, in particular interpolation and decimation. Compared to standard FIR filters, they are more economical since they do not include multipliers and use limited storage components [5]. As its name suggests, a CIC filter is a cascade of integrator filters and comb filters. The structure of CIC decimation and interpolation filters are shown in Figure 3. The difference between the two configurations are the order of the two types of filters and down/up sampler selection. The number of integrator filters and comb filters are always equal.

CIC filters have three parameters: number of stages (N), diffrential delay (M), and rate change factor (R). N refers to the number of integrator filters and comb filters, M refers to the delay of the comb filters, and R refers to the up/down sampling rate. Frequency response (magnitude) of a CIC filter can be represented using these three parameters:

$$|H(e^{j(2\pi f)})| = \left|\frac{\sin(RM\pi f)}{\sin(\pi f)}\right|^N, \qquad f \in [0,1). \tag{1}$$

(Here f is normalized w.r.t. the input signal's sampling frequency.) Using this equation, we can plot the magnitude of the frequency response of a CIC filter. One example is shown in Fig. 4. As we can see from the example, a CIC filter has low-pass qualities, filtering out most components with frequency over f_0/R (f_0 is the input signal's sampling rate).

In PDM-PCM conversion, CIC filter is used to decimate the PDM signal with higher sampling rate and get a rough

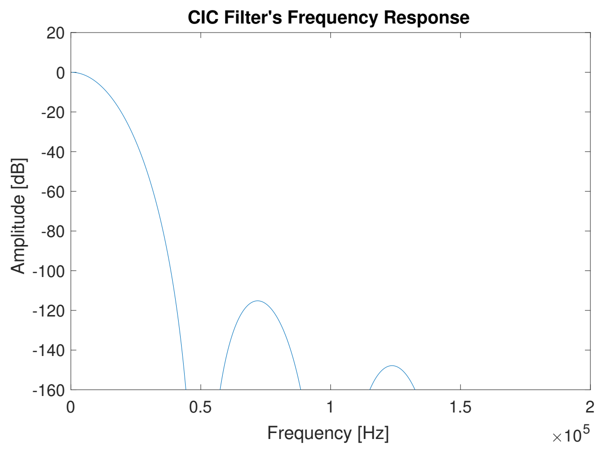


Fig. 4: Frequency response (magnitude) of a CIC filter with $N = 9$, $M = 1$, and $R = 8$. The input signal's sampling rate is 400 kHz.

PCM signal with fewer hardware resources than standard FIR filters. However, since the CIC filter's frequency response is not ideal, an additional FIR filter working under the output sampling rate is needed to rectify it. This will be discussed in the following subsection.

D. *Finite Impulse Response Filter*

Finite Impulse Response (FIR) filters are one general class of digital filters. The other general class of digital filter has an opposite name, Infinite Impulse Response (IIR) filter. They are classified by the form of their impulse response, and as its name suggests, the impulse response of an FIR filter has a finite duration and satisfies $\exists N \in \mathbb{N}\ \forall n > N\ h[n] = 0$. Given this property, the difference equation of an FIR filter always has the form [6]

$$y[n] = \sum_{i=0}^{N} (b_i \cdot x[n-i]), \tag{2}$$

since this implies the impulse response of the system is

$$h[n] = \sum_{i=0}^{N} (b_i \cdot \delta[n-i]) = \begin{cases} b_n & \text{if } 0 \le n \le N \\ 0 & \text{if } n > N \end{cases}. \tag{3}$$

From this formula, we know that an FIR filter is uniquely determined by its sequence of parameters (b_i) and we can characterize an FIR filter solely by a finite sequence of parameters.

Although CIC filter is more economical than FIR filters for decimation, as we can see from Fig. 4 that the amplitude attenuation from 0 to f_0/R is rather smooth, which is not ideal for a low-pass filter. In addition, Since the output signal's sampling rate frequency is f_0/R, from the Sampling Theorem we would like to cut off at at most $f_0/2R$ instead of f_0/R in order to reduce aliasing. Therefore, an FIR filter is needed to compensate for a CIC filter's frequency response. The Intel CIC filter IP core provides a MATLAB script for generating the parameters for an FIR filter that can compensate a CIC

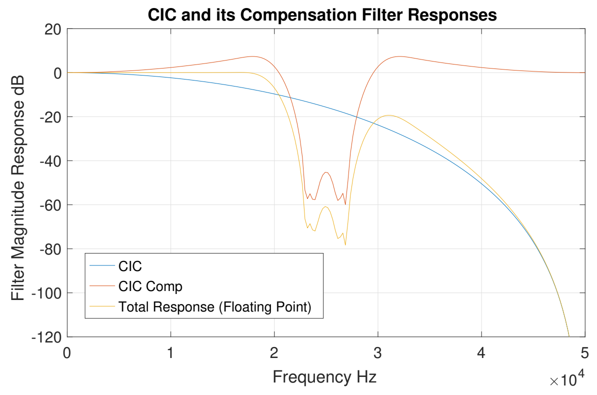


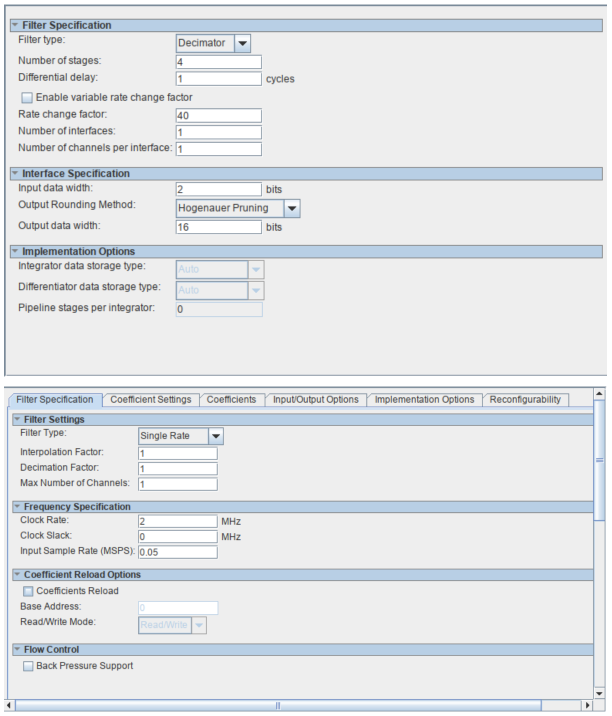
Fig. 5: A CIC filter's frequency response before and after an FIR filter's compensation. The parameters of the test is given in the text body.

filter of certain parameter, and our group used that script to design the FIR compensation filter. The frequency response of the CIC filter used in our system before and after FIR filter's compensation is shown in Fig. 5. The parameters for the CIC filter is $N = 4$, $M = 1$, and $R = 40$. The input PDM signal's sampling rate is $f_0 = 2$ MHz, which gives an output PCM sampling rate of $f_0/R = 50$ kHz. The FIR compensation filter is intended to make the total response cut off at $f_c = 20$ kHz. Since the FIR filter works at f_0/R, its frequency response is symmetric w.r.t. $f_0/2R = 25$ kHz, and this intrinsic property limits its ability to rectify the frequency response. The total response for f from 0 to 25 kHz is considerably better as a low-pass filter, but the response for f from 25 kHz to 50 kHz is almost the same before and after compensation. That said, adding the FIR compensation filter still improves the system's overall cut-off performance, so it is used in our final system.

III. Methods

We built the PDM-to-PCM signal conversion system on an Intel Arria V GX FPGA. To build the system, we utilized the CIC filter and FIR filter IP cores provided in Intel's IP catalog. In additional to the basic parameters of the two types of filters, the IP cores also have additional useful parameters such as output rounding settings and flow control support. The configuration interfaces for the CIC and FIR filter are shown in Fig. 6. As mentioned in the previous section, a MATLAB script provided by the CIC filter IP core is used to generate the parameters of the FIR filter.

In addition to setting the parameters of the two filters, a PLL is used to generate synchronized clocks at different frequencies, and a data multiplexer is written to output the processed PCM signals from multiple microphones on the same GPIO pins. The current system is able to process data from two PDM microphones at the same time and output the multiplexed results on GPIO pins, and transferring the results to a PC using Ethernet is under development.



Filter Specification

Filter type: Decimator

Number of stages: 4

Differential delay: 1 cycles

☐ Enable variable rate change factor

Rate change factor: 40

Number of interfaces: 1

Number of channels per interface: 1

Interface Specification

Input data width: 2 bits

Output Rounding Method: Hogenauer Pruning

Output data width: 16 bits

Implementation Options

Integrator data storage type: Auto

Differentiator data storage type: Auto

Pipeline stages per integrator: 0

Filter Settings

Filter Type: Single Rate

Interpolation Factor: 1

Decimation Factor: 1

Max Number of Channels: 1

Frequency Specification

Clock Rate: 2 MHz

Clock Slack: 0 MHz

Input Sample Rate (MSPS): 0.05

Coefficient Reload Options

☐ Coefficients Reload

Base Address: 0

Read/Write Mode: Read/Write

Flow Control

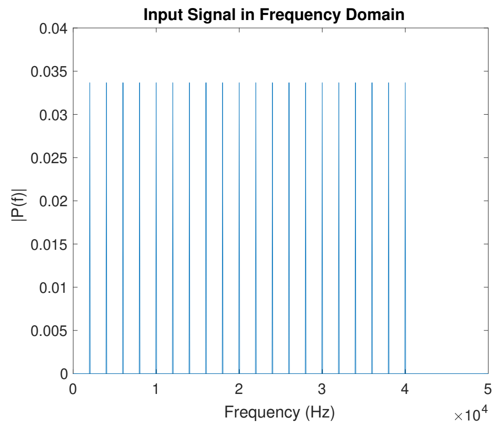
☐ Back Pressure Support

Fig. 6: Configuration interfaces for the CIC (above) and FIR (below) filter IP cores.

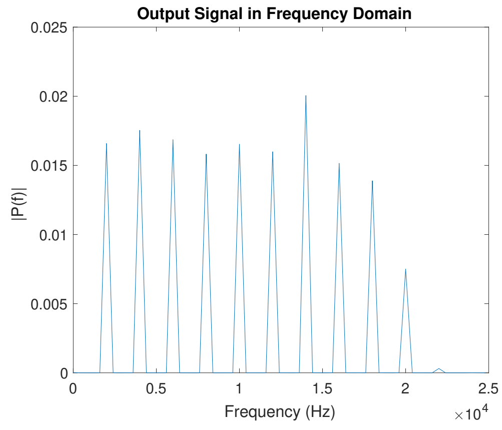
IV. Results

We test the completed PDM-to-PCM conversion system based on CIC and FIR filters using both simulation and real hardware test. For simulation, we conducted the test for one-mic setup. The filter's parameters are the same as the settings at the end of Section II-D. The input analog signal is a series of sine waves at different frequencies overlaid together. The sine waves' frequencies is an arithmetic sequence from 2 kHz to 40 kHz and they all have the same amplitude. It is sampled using software delta-sigma modulation to generate a 2 MHz PDM signal, and the signal is fed to the conversion system using on-board RAM. The output PCM signal is extracted from the simulator and then imported to MATLAB for analysis. The FFT of the input analog signal and the output PCM signal is shown in Fig. 7. As we can see from the result, the sine waves are indeed low-pass filtered with a cut-off frequency of 20 kHz. There are slight distortions in the 0 to 18 kHz frequency range and the 20 kHz spike is considerably attenuated, but overall the system convert the input PDM signal to an PCM signal with reasonably good fidelity.

For the real hardware test, we compiled the Verilog HDL code for the system and uploaded the design onto the board, and then connected two PDM microphones we have and used digital oscilloscope to capture the outputs on the GPIO pins. The output data are then converted to a PCM audio file and we evaluated the playback quality qualitatively. Currently, the playback quality is good for one-mic setup without multiplexing but is a little noisy for two-mic setup, but we have good reason to believe the noise in the two-mic setup are caused by the oscilloscope's error when capturing high frequency data and this will be resolved after Ethernet is used to transfer the output data.



(a) FFT of the input analog signal.



(b) FFT of the output PCM signal.

Fig. 7: Analysis of the simulation results.

V. Conclusion and Future Work

In conclusion, this essay presents a PDM-to-PCM microphone signal conversion system implemented on an FPGA. The system mainly consists of CIC and FIR filters, and can currently receive data from two microphones at the same time and output the converted PCM signal on GPIO pins. The modular characteristic of Verilog HDL code makes the code for this system easily extensible for a larger microphone array setup, and the current code serves as a basis for the future system with a larger microphone array. Future work for this project includes completing the Ethernet data transfer subsystem, extending the current code base for a larger microphone array, implementing beamforming algorithms for object detection, and exploring applications of object detection and other possible functions in the field of embedded security and human-computer interaction.

References

[1] J. Lewis, “Analog and digital mems microphone design considerations,” *Technical Article MS-2472. Analog Devices*, 2013.

[2] B. Da Silva, A. Braeken, and A. Touhafi, “Fpga-based architectures for acoustic beamforming with microphone arrays: trends, challenges and research opportunities,” *Computers*, vol. 7, no. 3, p. 41, 2018.

[3] Wikipedia. (2020) Pulse-code modulation — wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Pulse-code%20modulation&oldid=980465874

[4] ——. (2020) Pulse-density modulation — wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Pulse-density%20modulation&oldid=954286453

[5] E. Hogenauer, “An economical class of digital filters for decimation and interpolation,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 2, pp. 155–162, 1981.

[6] Wikipedia. (2020) Finite impulse response — wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Finite%20impulse%20response&oldid=964563582