ECE532: Digital Systems Design

Final Group Project Report

April 7, 2025

Audio Vocoder

Group 32:

Sherry Zhang

Table of Contents

1. Overview

This project aims to create a voice encoder, or vocoder, using a Nexys Video Board. A voice input audio is collected using a microphone then imposed onto a carrier signal, and the resulting synthesized audio is output through earbuds. The audio input and output devices are connected to the board's 3.5mm audio jacks. The carrier signal is selected by a 16-button keypad, which chooses between a set of preset waveforms generated by a digital synthesizer.

1.1. Project Background and Motivation

A vocoder is a device that analyzes and synthesizes a voice input for various purposes including voice transformation, data compression, voice encryption. It splits the input into various frequency bands, then applies characteristics of each input band onto a carrier signal which may be produced by a synthesizer or other musical instruments.

Vocoder technology differs from conventional autotune processing, which simply corrects the pitch of the input; instead of outputting a modified version of the input signal, vocoders output synthesized audio signals that carry characteristics collected from the input signal.

There are many applications for vocoders, ranging from voice encoding for information masking to modern music production. The team took a particular interest in the use of vocoders in the music industry and sound design. This project also applies many concepts introduced in digital signal processing courses, allowing the team to apply our understanding of DSP concepts to an FPGA project.

1.2. Goals

The main project goal is to record a segment of audio, then play back a synthesized version of the signal with the chosen carrier. This is accomplished by designing a custom audio processing IP block, which separates the input signal into frequency bands, uses an envelop detector to extract significant characteristics, applies those traits to the carrier signal, then combines the frequency bands into one output signal. The carrier signals are generated using a Direct Digital Synthesizer (DDS) module, and are selected using a 16-button keypad PMOD programmed to select between the preset pool of generated signals.

1.3. Block Diagram

Figure 1 illustrates the block diagram of the implemented vocoder system. See Appendix A for an additional block diagram figures that illustrates the components within the custom audio processing IP block.

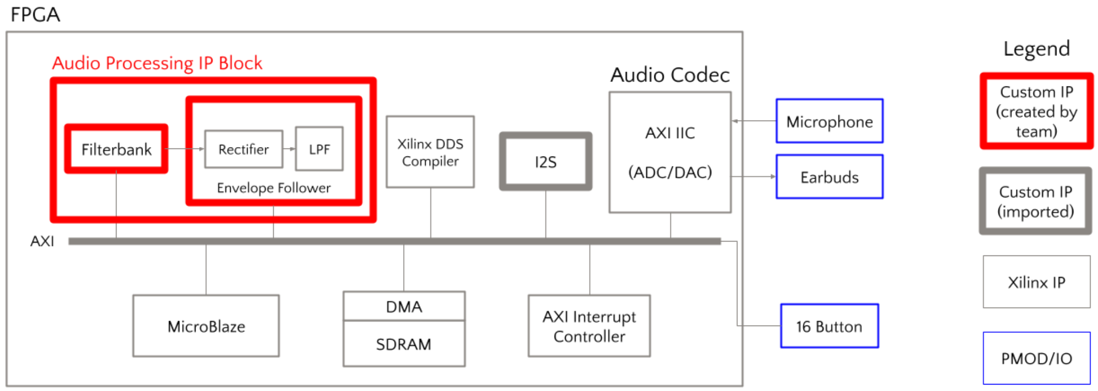

Figure 1: Block Diagram of the Project System

1.4. IP Summary

The audio data path is primarily composed of three main blocks: audio codec, I2S, and our custom audio processing IP block. The raw analog audio enters the system through the ports attached to the audio codec included in the Nexys Video board. The codec is controlled by the AXI IIC block, which is available in the Vivado Library. This audio codec processes the data and convert it to a digital signal, which would then be received by the I2S. The I2S module is a custom block provided in the 2023 Nexys Video DMA Audio Demo [1], which is able to convert audio between parallel and serial data formats. This allows the data to fit the width formats required for other blocks within the design. The last main component is the custom audio processing block, which is composed of two filter banks, full wave rectifiers, low pass filters, and a vector multiplier. This block was designed by the team to impose the voice input onto the synthesized signal input. The audio data path is split into two directions depending on if it would bypass the filter bank and audio processing module.

The MicroBlaze processor was included to control I/O interactions. It is connected to the AXI Interconnect, UART, DMA, and Interrupt Controller modules to handle interactions with the on board buttons and the 16-button keypad. All MicroBlaze and AXI related IP can be found in the Vivado library.

2. Outcome

2.1. Initial Proposal

The original proposal specified similar inputs and outputs to that of the final project, but also included real time processing of the audio signal, allowing the user to manipulate their vocal input signal almost instantaneously. The system would have to impose the pitch characteristics of the analog voice input onto the carrier signal, carrying a minimum of one frequency component at a time.

2.2. Design Changes

The project was scaled back from its initial proposal due to various challenges that arose from the design process. The final project closely resembles the Nexys Video DMA Audio Demo available on the Diligent Reference Manual [1]. The team attempted to implement some additional modules, including an audio processing module and additional PMOD interfaces for more audio playback options.

The team was unable to implement real time audio processing. Instead, the audio input would be saved to memory and later recovered when audio playback was requested.

2.3. Project Functionality

The latest version of the project results in an output similar to that of the basic audio demo. Due to last minute complications, no functional filter bank was implemented, and selecting between different audio data paths (filtered/unfiltered) was also not fully implemented.

2.4. Future Works and Improvements

Since many of the desired functions and components were not implemented successfully, future projects could focus on completing the partial work done for these elements.

In previous versions of the project, the 16-button keypad was functional, and was able to assign a select signal to choose between output signals, however this version was not recovered for the final project. This functionality can be reimplemented relatively easily, as the decoder and hardware connections were preserved, thus primarily requiring debugging in the software.

Additional debugging or rebuilding of the audio processing unit is required in order to use the latest custom IP block. As of present, the custom audio processing block is not able to be successfully integrated into the system. Although the individual biquad filters have been tested through simulation, additional simulations and testing in hardware of the remaining elements would allow for a better understanding of the issues with the block. Additionally, the IP was built in an older version of Vivado (2018.3) compared to the remainder of the project (2023.2), so it may be more beneficial to follow the structure of the block and simply rebuild it in the correct version. Another other potential area to begin debugging include the changing the operating frequency (48kHz) to match the rest of the project (100 MHz), or finding a way to handle the different clock drivers.

If given the opportunity to restart the project, the team would know how to handle audio input and output using the 2023 Nexys Video Audio Demo [1], and would be able to dedicate more time to carefully implementing the audio processing module. This would most likely include adding the modules individually into the system, rather than integrating everything simultaneously as a large IP block. Once functionality is verified, the various components could then be wrapped up to make the block diagram cleaner and easier to follow.

3. Project Schedule

3.1. Milestone Comparison

The table below compares the initial milestones set for the project proposal and the updated milestones, which were reviewed week to week in order to reflect the progress of specific areas within the project.

Table 1: Description of the milestone goals and achievements for this project

Milestone	Original Milestone	Updated Milestone
1	• Test input data collection; testing input devices (ie microphone) and checking that we can collect the data as desired	• Refined the project proposal • Researched existing IPs for audio processing modules • Researched I/O connections and how to collect the desired data

2	• Get a basic audio processing block to work:
	  o Research existing IPs
	  o Try to implement them in hardware
	  o Attempt to build and test some custom blocks to implement these DSP blocks
	• If not achieved, at least some exploration in possible options for audio processing should be explored and ready to present.
	• Begin implementing PMODs (microphone and audio ports)
		• Researched additional information on IIR filters to implement envelop detection and bandpass filtering
		• Attempted to build and test some custom blocks to implement DSP blocks
		  o A simple low pass filter was implemented with the filter and DDS IPs from the Vivado library
		• Began implementing Nexys Video DMA Audio demos
		  o An earlier version was used, but it was not functional

3	• Show an initial project build with roughly connected components
	  o At least functional input data collection
	  o Some beginning tests in hardware to verify the processing that is done on the audio signal
		• Determined logarithmic spacing of filter bank
		• Designed filters using Matlab FilterDesigner
		• Continued Nexys Video DMA Audio demo implementation
		  o No significant progress compared to the previous week

4	• Show functional input and processing • All hardware components should be working individually • Software driver should be in the debugging stage, but should at least not be causing critical errors	• Successfully replicated Nexys Video DMA Audio demo (integrated I2S and IIC modules to handle audio input/output) • Added 16-button PMOD and the required GPIO hardware blocks

		o Software integration and interrupt handling in progress • Debugging of filter bank functionality
5	• Have a primitive vocoder working • Software can continue being debugged to refine certain features but base structure and function should be working	• 16-button successfully implemented; the correct buttons were being detected when pressed, and they could select between different audio data paths in the hardware • Attempted to output filtered output using custom filter IP • Connected all filter bank channels • Continued filter bank debugging
6	• Buffer week for additional features to be implemented, or for further debug in case milestones 4/5 are delayed o The team struggles most with software; we are preparing this buffer week to give us more time to either debug our main vocoder function or add additional features to refine the design	• Continued filter bank and audio processing debug work o Integration of audio processing IP into the main block diagram proved to be very challenging and complex, so the entire milestone focused on this task

3.2. Discussion

Throughout the project, our milestone expectations changed significantly due to slower-than-anticipated progress. We refined our weekly milestone goals by explicitly stating subtasks to better understand what needed to be done. Due to very ambitious initial milestones, most weeks involved milestone reframing.

For milestone 1, our goals changed as we realized that implementing basic input output systems required more research. For the following weeks, the team continued work on researching and implementing filters, but many fundamental tasks like handling audio IO were left incomplete. Milestone 3 was a low-progress week that served as a reality check, particularly around I/O difficulties, leading us to narrow our focus onto successfully handling our system’s input collection. We made notable progress on implementing PMODs and connecting input/output and filter components for the remaining milestones, but continued struggling with implementing a functional audio processing module. The team did not anticipate the complexity of the filter bank, rectifier, and audio mixer integration, leading to insufficient time for debugging of that component of the design despite allocating a buffer week in the initial plan.

4. Detailed IP Block Description

4.1. IIC: Audio Codec Control

The Inter-Integrated Circuit (IIC) IP core in Vivado can be found in the Vivado library. It serves as a master device that sends configuration commands to the audio codec. An example includes, ADAU1761, an audio codec chip that converts signals between analog and digital representations, is configured by the IIC IP core.

The audio codec is a device that converts analog audio signals from a microphone or line input into digital data and vice versa for playback through speakers or headphones. By properly configuring the codec through the IIC interface during system initialization, this device assists in handling input and output audio signals, reformatting them into compatible data format such as I2S.

4.2. I2S: Audio Reformatting

The Inter-IC Sound (I2S) IP is a custom block provided in the DMA Audio Demo [1]. It is responsible for transmitting and receiving digital audio data between the audio codec and the rest of the FPGA. After receiving a signal from the audio codec, the I2S interface handles outputting audio samples in sync with the codec's clock. For incoming audio recording, it will take the serial data stream from the codec and output parallel data samples. The reverse is done when audio playback is requested; the parallel data stream is converted to a serial data stream that can be sent to the codec.

In this project, the I2S is connected to the AXI bus/DMA to read/write the audio files to memory. This allows the input audio to be recorded and stored, then later read when the carrier synthesized signal is chosen.

4.3. Audio Processing Custom IP

4.3.1. Biquad filter and Filter Bank

The biquad filter IP core is a custom IP block that implements 4th order IIR Butterworth filters as cascaded biquads. The filter coefficients were designed using Matlab FilterDesigner, allowing them to cover different ranges of frequencies. These filters are used in the filter bank IP core, which connects pairs of filters to form a 24-channel filter bank that splits the input signal into segments based on specified frequency ranges. A table with the coefficients and frequency ranges of each filter pair can be found in Appendix B.

4.3.2. Audio Processing IP

The overall audio processing IP block uses the custom filter bank IP, as well as a rectifier and mixer module. The filter behaviour is described above. The rectifier module takes the absolute value of the signal. The mixer module acts as a vector multiplier, which combines the values of the voice input and synthesized carrier input signals.

4.4. MicroBlaze

The MicroBlaze processor block can be found in the Vivado library. It controls the system logic and runs C code to record or playback audio depending on the interrupt signals triggered by PMOD interactions.

4.5. AXI GPIO

AXI GPIO blocks are found in the Vivado library. These blocks are used to connect PMODs or on board buttons to the system, connecting their behaviour to the rest of the design.

5. Design Tree Description

All relevant code for the project has been uploaded to GitHub at the following address:

https://github.com/sherry30397/ece532-vocoder-project/tree/main

The bulk of the project closely resembles the 2023 Nexys Video DMA Audio demo, available on the Diligent Reference Manual [1]. The only notable hardware differences include the custom IP block, the decoder for handling the 16-button PMOD, and the DDS and GPIO blocks for synthesizing alternate audio sources. Most of the base software files are also similar to those provided by the demo. There are additional modules and changes to demo.c and userio.c to accompany the addition of the 16-button keypad into the system.

```
(The project tree below only includes the important files or the files that have changed from the DMA Audio demo)
ece-vocoder-project/Nexys-Video-DMA-hw.xpr/Nexys-Video-HW/
├── Nexys-Video-DMA/
│   ├── src/
│   │   ├── userio/
│   │   │   ├── userio.c  # added functions to handle 16 button interactions
│   │   │   └── userio.h  # header file for userio.c
│   │   ├── demo.c        # added 16 button interrupt and behaviour cases for some button presses
│   │   ├── demo.h        # header file for demo.co
│   │   └── ...
│   └── ...
├── Nexys-Video-HW.ipdefs/repo/local/ip
│   ├── biquad filter          # custom ip for biquad filter
│   ├── d_axi_i2s_audio_v2_0   # i2s IP; from demo
│   ├── filterbank             # custom ip for filter bank with 24 biquad filter channels
│   └── ip_repo                # custom ip for audio processing custom IP block; uses filterbank and biquad sub-ips
├── Nexys-Video-HW.srcs
│   ├── constrs_1/imports/constraints   # constraints file for Nexys board ports
│   ├── sources_1/
│   │   ├── bd/
│   │   │   ├── design_1/              # block diagram of whole project system
│   │   │   └── design_2/              # block diagram for audio processing custom IP block
│   │   ├── imports/Downloads/
│   │   │   ├── decoder.v              # 16-btn decoder
│   │   │   └── sign_extend.v          # extend input width to fit filter/i2s modules
│   │   ├── new/
│   │   │   ├── top.v                  # top wrapper for block design and other modules
│   │   │   └── ...
│   └── ...
├── Nexys-Video-HW.xpr      # project file
├── ...
└── README.md
```

Figure 2: Design Tree

6. Advice for Future Students

Since no notable technical work was completed for this project, the following section focuses more on project management and planning.

Future students who intend on working on this project should ensure that a proper debugging plan is in place for their components, especially for custom IP blocks. Rather than implementing the IP block as a whole, it may be beneficial to test and debug each small IP block individually. It's much simpler to isolate problems with the overall IP block integration when each building block's functionality is already verified. Additionally, it is important to scale back project goals to allow for reasonable progress when faced with a challenge. Due to complications with the project, the team ended up with a non-functional demonstration, which could not showcase the partially integrated components. A backup plan for a project that is still functional and demonstrable still showcases partial integration of various, even if it's not the full version that is initially proposed.

7. Video Demonstration

A short video demonstration explaining the project can be found at the project location mentioned in Section 5.

Appendix A: Audio Processing Custom IP Block Figure

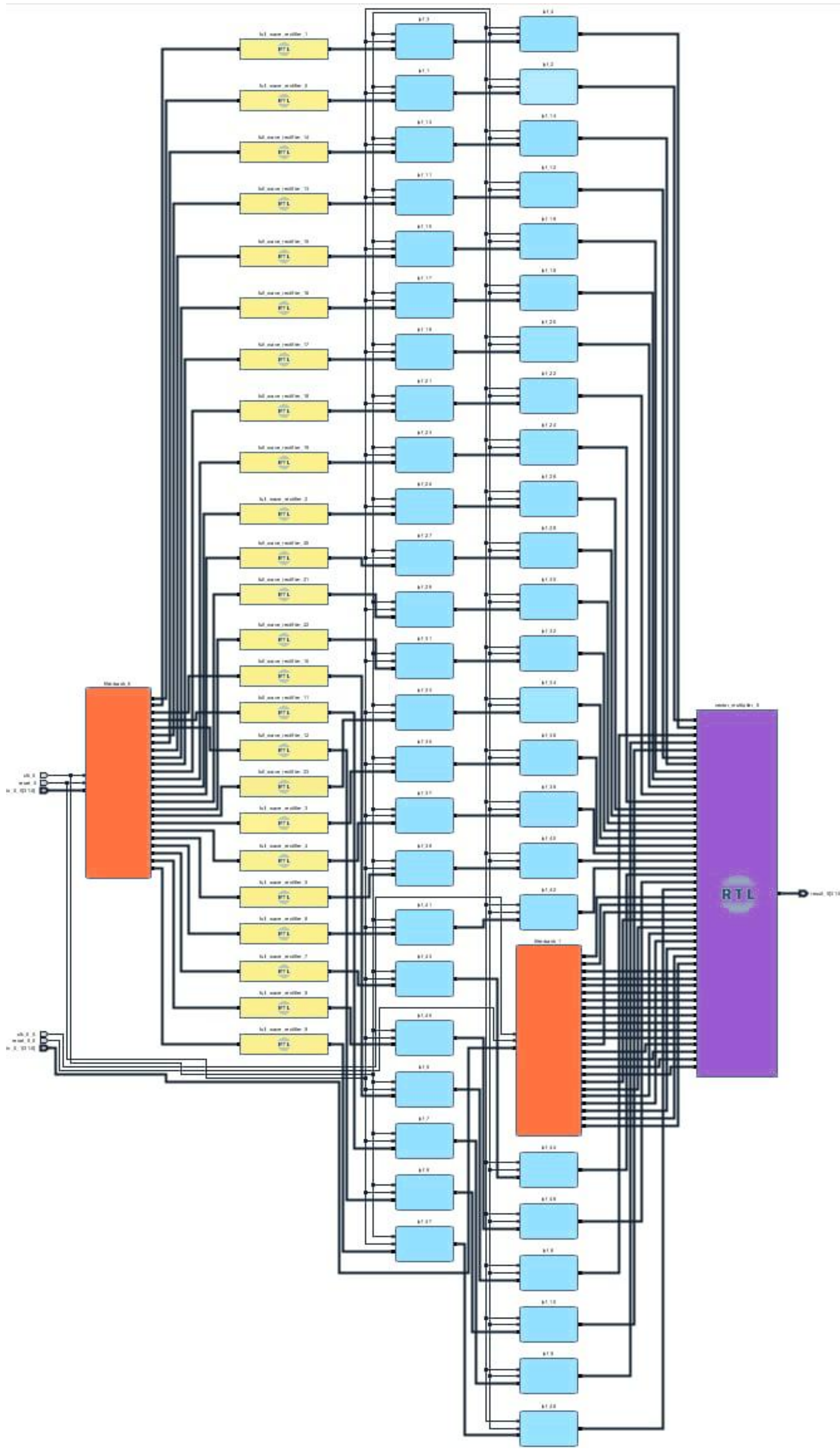

Figure 3: Block Diagram of audio processing custom IP block. It is made up of filter banks (red), full wave rectifiers(yellow) and low pass filters (blue), and an audio mixer (purple)

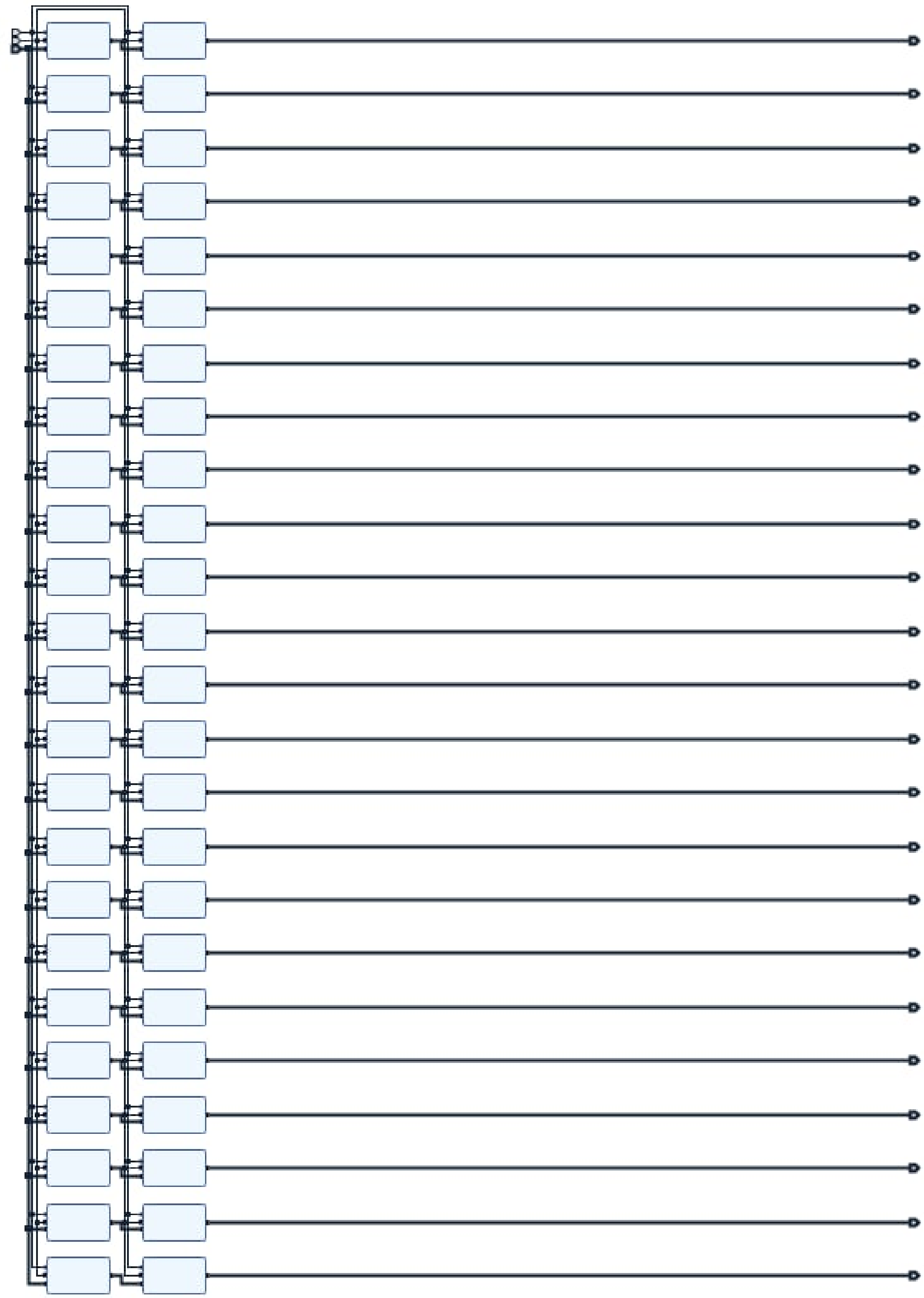

Figure 4: Filterbank components - each filterbank channel is comprised of a pair of cascaded biquad filters. There are 24 total frequency channels

Appendix B: Filter Bank Coefficients

		Filter1					Filter2				
Channel #	**Ending frequency**	**a1_1**	**a2_1**	**b0_1**	**b1_1**	**b2_1**	**a1**	**a2**	**b0**	**b1**	**b2**
1	61.43162512	0x8013	0x3FEE	0x000C	0x0000	0xFFF4	0x8010	0x3FF1	0x000C	0x0000	0xFFF4
2	75.47689129	0x8018	0x3FE9	0x000F	0x0000	0xFFF1	0x8015	0x3FEC	0x000F	0x0000	0xFFF1
3	92.73336182	0x8020	0x3FE3	0x0013	0x0000	0xFFED	0x801B	0x3FE7	0x0013	0x0000	0xFFED
4	113.9352224	0x8026	0x3FDE	0x0016	0x0000	0xFFEA	0x8020	0x3FE2	0x0016	0x0000	0xFFEA
5	139.9845174	0x802F	0x3FD6	0x001C	0x0000	0xFFE4	0x8028	0x3FDB	0x001C	0x0000	0xFFE4
6	171.9895279	0x803C	0x3FCC	0x0022	0x0000	0xFFDE	0x8033	0x3FD3	0x0022	0x0000	0xFFDE
7	211.311924	0x804B	0x3FC1	0x002A	0x0000	0xFFD6	0x8040	0x3FC9	0x002A	0x0000	0xFFD6
8	259.624698	0x8061	0x3FB0	0x0034	0x0000	0xFFCC	0x8052	0x3FBB	0x0034	0x0000	0xFFCC
9	318.9833424	0x807B	0x3FA0	0x003F	0x0000	0xFFC1	0x8067	0x3FAD	0x003F	0x0000	0xFFC1
10	391.9133021	0x809F	0x3F8A	0x004E	0x0000	0xFFB2	0x8085	0x3F9A	0x004E	0x0000	0xFFB2
11	481.5174211	0x80CF	0x3F6E	0x0060	0x0000	0xFFA0	0x80AC	0x3F82	0x0060	0x0000	0xFFA0
12	591.607954	0x810E	0x3F4E	0x0075	0x0000	0xFF8B	0x80DF	0x3F66	0x0075	0x0000	0xFF8B
13	726.8687609	0x8165	0x3F26	0x0090	0x0000	0xFF70	0x8125	0x3F43	0x0090	0x0000	0xFF70
14	893.0545846	0x81DD	0x3EF4	0x00B1	0x0000	0xFF4F	0x8184	0x3F18	0x00B1	0x0000	0xFF4F
15	1097.235889	0x8283	0x3EB8	0x00D9	0x0000	0xFF27	0x8208	0x3EE4	0x00D9	0x0000	0xFF27
16	1348.099676	0x836D	0x3E6D	0x010A	0x0000	0xFEF6	0x82C0	0x3EA3	0x010A	0x0000	0xFEF6
17	1656.319078	0x84B5	0x3E13	0x0146	0x0000	0xFEBA	0x83C1	0x3E54	0x0146	0x0000	0xFEBA
18	2035.007454	0x8689	0x3DA4	0x018F	0x0000	0xFE71	0x852E	0x3DF3	0x018F	0x0000	0xFE71
19	2500.276301	0x8924	0x3D1E	0x01E8	0x0000	0xFE18	0x8735	0x3D7E	0x01E8	0x0000	0xFE18
20	3071.920728	0x8CE3	0x3C7A	0x0256	0x0000	0xFDAA	0x8A1D	0x3CED	0x0256	0x0000	0xFDAA
21	3774.261651	0x923F	0x3BB6	0x02D9	0x0000	0xFD27	0x8E47	0x3C3D	0x02D9	0x0000	0xFD27
22	4637.180536	0x99EE	0x3AC7	0x037A	0x0000	0xFC86	0x9441	0x3B66	0x037A	0x0000	0xFC86
23	5697.390726	0xA4DC	0x39AB	0x043C	0x0000	0xFBC4	0x9CCE	0x3A5F	0x043C	0x0000	0xFBC4
24	6999.999425	0xB43F	0x3859	0x0527	0x0000	0xFAD9	0xA8F3	0x391C	0x0527	0x0000	0xFAD9
lpf	100Hz cutoff	0x80A6	0x3F5D	0x0001	0x0001	0x0001	0x818A	0x3E78	0x0001	0x0001	0x0001

References

[1] Digilent Inc., "*Nexys Video DMA Audio Demo*," [Online]. Available: https://digilent.com/reference/programmable-logic/nexys-video/demos/dma-audio?s[]=nexys&s[]=video&s[]=audio&s[]=demo

[2] AMD, "*DDS Compiler IP Core*," [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/dds_compiler.html

[3] MIT FPGA Team 35, "*Real-Time Vocoder Final Report (2023)*," [Online]. Available: https://fpga.mit.edu/videos/2023/team35/report.pdf

[4] AMD, "*Efficient Filter Structures for IIR Filters (WP330)*," [Online]. Available: https://docs.amd.com/v/u/en-US/wp330

[5] AMD, "*FIR Compiler IP Core*," [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/fir_compiler.html

[6] AMD, "*Audio Formatter IP Core*," [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/audio-formatter.html

[7] S. Sethares, "*Channel Vocoder in MATLAB*," University of Wisconsin-Madison, [Online]. Available: https://sethares.engr.wisc.edu/vocoders/channelvocoder.html