Design Project Report – Group C
A PYNQ Theremin

1. Overview

Our PYNQ design is a Theremin, which is a musical instrument that can produce a sound where its pitch, in our case, is based on the distance from the Grove Ultrasonic Ranger (USR) to an object. The gain or overall volume of the system can be adjusted from the Jupyter Notebook. The purpose of the project was to create said instrument on the provided PYNQ-Z2 board, with the features of a typical theremin.

2. Review Of System Architecture

In Vivado, the following block designs are used for the project.

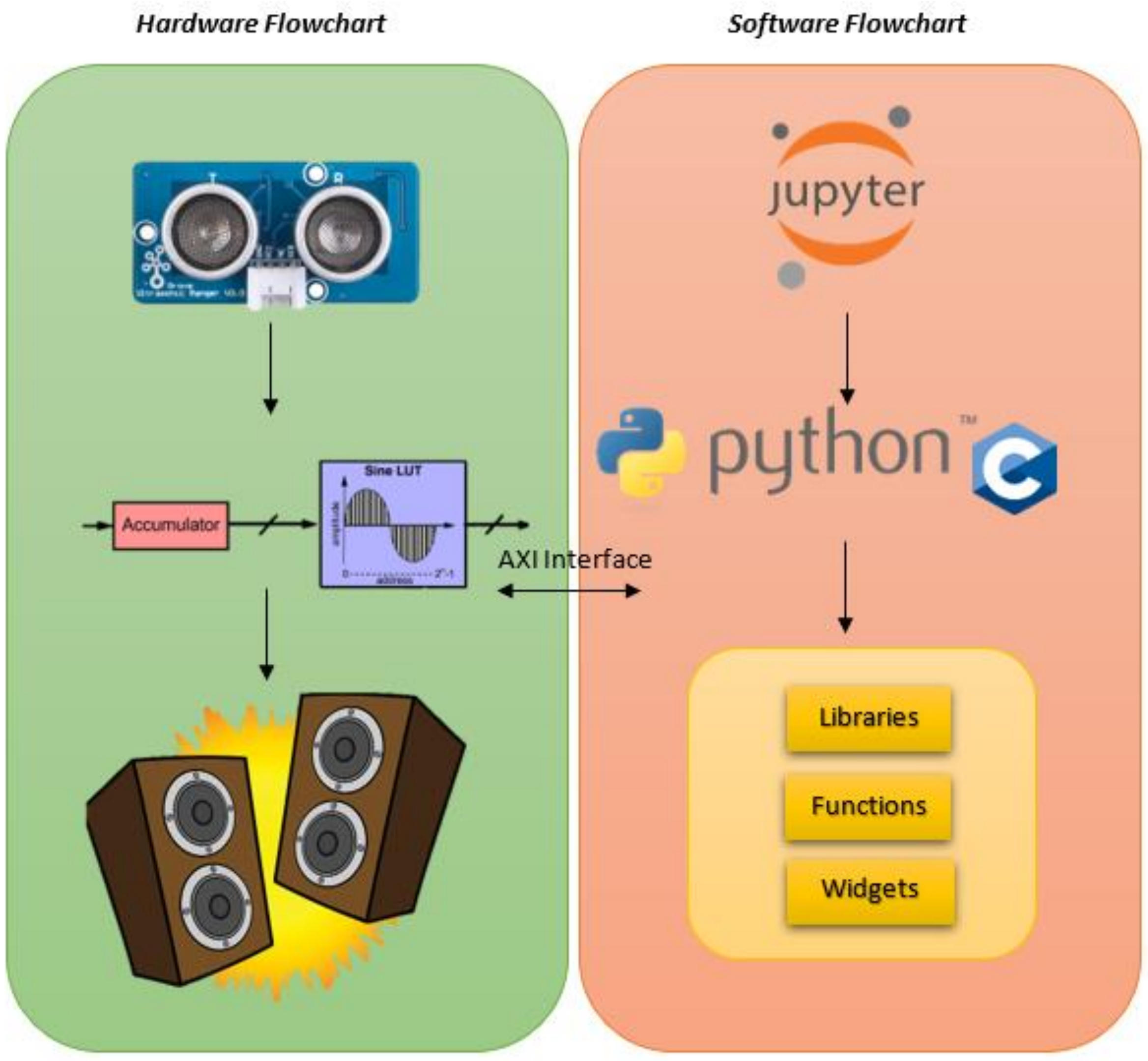


Figure 1 Hardware and Software Flowchart

2.1 *Numerically Controlled Oscillators (NCO) Design*

For designing the NCO, our design was a modified version of the given "AudioProcessing.slx" on the Further VHDL and FPGA Design MyPlace page (University of Strathclyde, 2022) and a PYNQ-Theremin online resource (Greer, 2020). Shown in in Figure 2 and Figure 3 respectively.

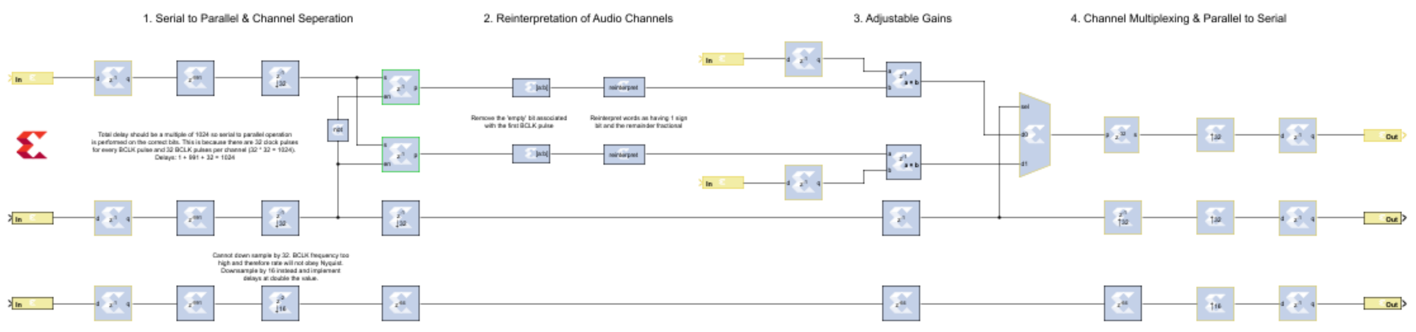

1. Serial to Parallel & Channel Seperation
2. Reinterpretation of Audio Channels
3. Adjustable Gains
4. Channel Multiplexing & Parallel to Serial
Total delay should be a multiple of 1024 so serial to parallel operation is performed on the correct bits. This is because there are 32 clock pulses for every BCLK pulse and 32 BCLK pulses per channel (32 * 32 = 1024). Delays: 1 + 991 + 32 = 1024
Remove the 'empty' bit associated with the first BCLK pulse
Reinterpret words as having 1 sign bit and the remainder fractional
Cannot down sample by 32. BCLK frequency too high and therefore rate will not obey Nyquist. Downsample by 16 instead and implement delays at double the value.

Figure 2 AudioProcessing.slx

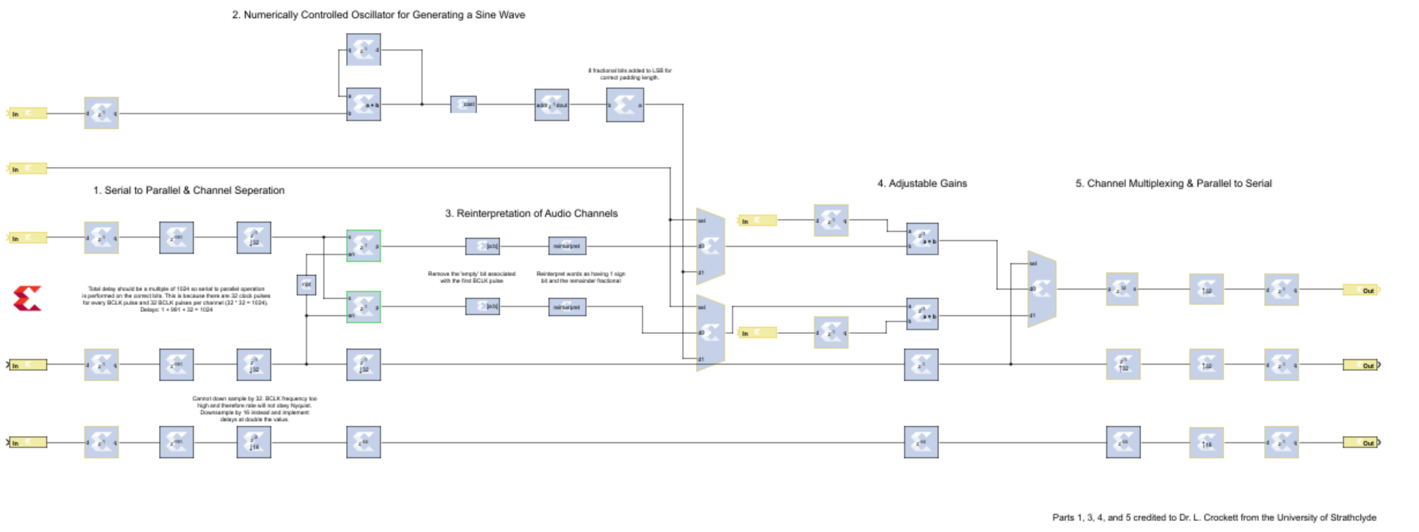

2. Numerically Controlled Oscillator for Generating a Sine Wave
8 fractional bits added to LSB for correct padding length.
1. Serial to Parallel & Channel Seperation
3. Reinterpretation of Audio Channels
4. Adjustable Gains
5. Channel Multiplexing & Parallel to Serial
Parts 1, 3, 4, and 5 credited to Dr. L. Crockett from the University of Strathclyde

Figure 3 Online design by Ryan Greer

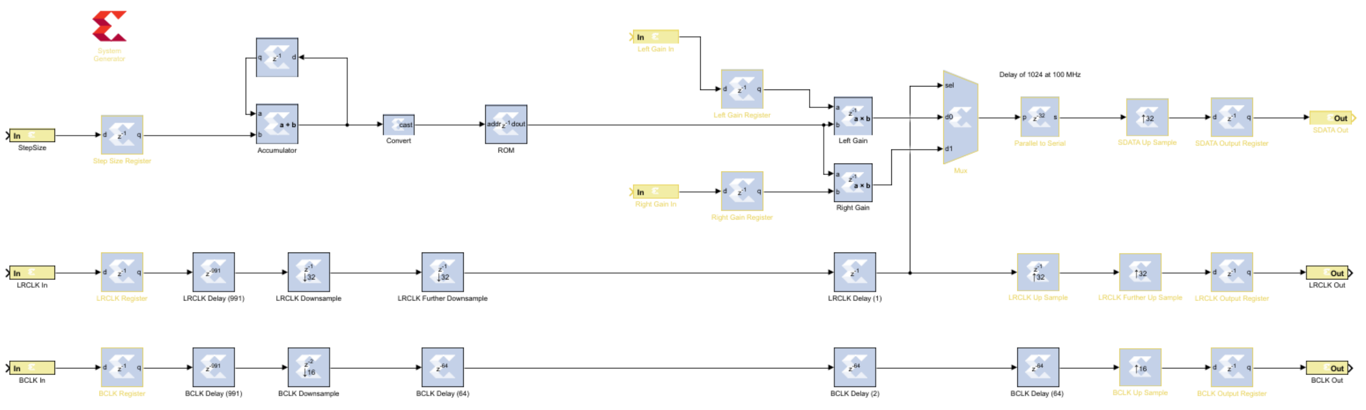


Figure 4 Project NCO design

The removed parts that can be seen were parts that added unneeded complexity to the project. For example, no separate audio channels are needed so they were removed and the NCO is made to go directly to the two Xilinx gain multipliers. Also, there is no "NCO Enable" due to us wanting it to be on continuously.

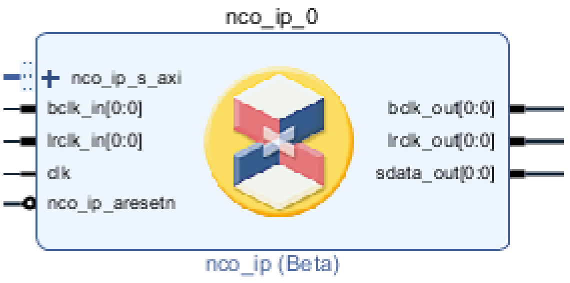


Figure 5 NCO in Vivado Block Design

In Figure 5, the finished IP for the NCO in Vivado can be seen.

2.2 Ultrasonic Ranger Circuit (iop_pmoda)

At the early stages of the project, the main idea was to implement as much of the stuff was "ready" as possible. The use of the ultrasonic sensor with the *Grove Adapter* was done in one of the in-campus labs – therefore it was approximately known what to expect in terms of functionality and how to evolve the designs around that idea. The code found in the notebook inside */base/pmod/pmod_grove_usranger.ipynb* uses the iop_pmoda hierarchy (pmoda) of the base overlay, highlighted in Figure 6.**Error! Reference source not found.** Amongst others, the pmoda consists of a microcontroller (microblaze) which compiles and runs c code (part of the code from the notebook) to control the ultrasonic ranger sensor. The microblaze microcontroller sends a signal to the ultrasonic ranger and then expects the echo of it through the same pin. Then, through a timer interrupt, it checks when the signal is detected and "raises" a flag for the change from low to high. This interrupt flag is then used inside the code to calculate the time taken for the ultrasound wave to travel – and with the calculation shown in Equation 1**Error! Reference source not found.** - a distance is given (in meters) or multiplied by 100 given in centimetres (cm). (ReadTheDocs, 2021)

$$\text{distance} = \frac{1}{2} * 340\frac{\text{m}}{\text{s}} * \ 10^{-6} * \text{time}$$

Equation 1. Ultrasonic wave distance calculation

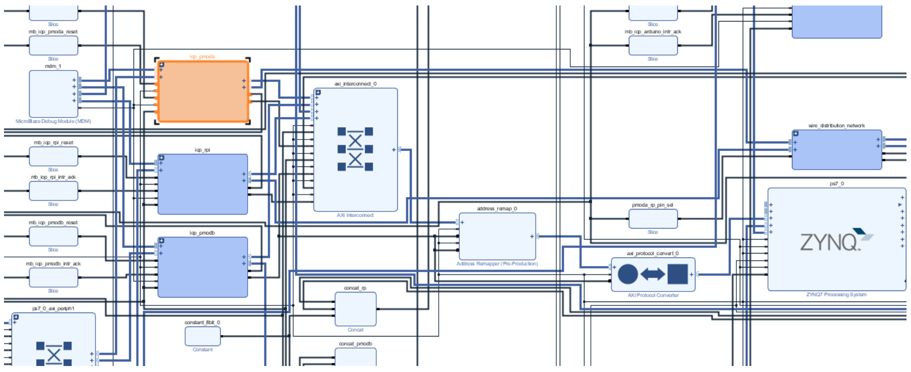


Figure 6. PMODA hierarchy

2.3 Jupyter Notebook

Using the Jupyter Notebook the user can interact with the Theremin. Values like the volume/gain can be changed and other adjustments, like the duration of the sound, and the amount of readings taken. It was split into multiple sections for ease of comprehension, for example Libraries, Functions, Widgets etc. More detailed and user-friendly documentation exists at the start of the project notebook. A single value for the Left and Right channel gain was decided on, as that would make the Theremin much easier to use. A duration slider was also added to easily display to the user how long the theremin would sound an output. Both values had to be entered before the final cell.

3. Testing

Testing was made with the individual parts tested individually to ensure correct operation. Help from lecturers and TAs was given in various stages of the project which will be noted.

3.1 NCO

As aforementioned, the NCO was modified from previously made NCOs (Numerically Controlled Oscillators) in Matlab Simulink. For testing purposes a constant was added in the 'Step Size' input so correct NCO function is ensured.

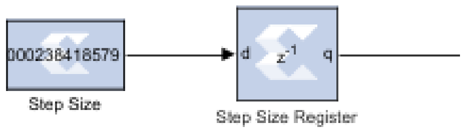
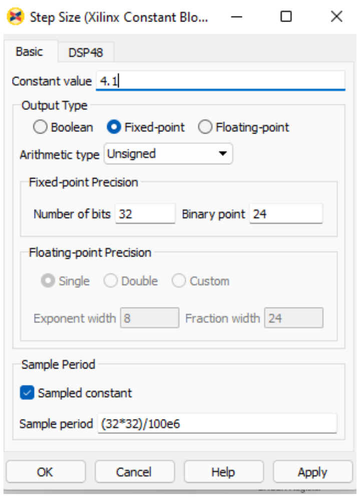


Figure 7 Constant Step Size for testing



Step Size (Xilinx Constant Blo...

Basic | DSP48

Constant value 4.1

Output Type

Boolean [✓ Fixed-point] Floating-point

Arithmetic type Unsigned

Fixed-point Precision

Number of bits 32 Binary point 24

Floating-point Precision

Single Double Custom

Exponent width 8 Fraction width 24

Sample Period

[✓ Sampled constant]

Sample period (32*32)/100e6

Figure 8 Step size configuration

Decisions for the final NCO design were made from testing and independent research. In the beginning, it was assumed that a single clock would be required for the audio data, but eventually it was discovered that a left and right channel clock was required even though the data would be single channel. Issues with Vivado version were also discovered which were resolved by upgrading to the 2020.2 version and by downloading an earlier PYNQ image. The final version was decided that it was functional once it was implemented on the PYNQ board and it outputted a constant sound. For this test, the given code on the PYNQ SD card for the Grove Ultrasonic Ranger was used.

The NCO was tested with some small differences in the final code since the USR was not used.

```
nco.write(0x8, step_size_calculation(read_distance_cm()*25))
```

The code snippet above is the one in the final version whereas the one below was used for testing.

```
nco.write(0x8, step_size_calculation(800))
```

If the testing snippet is written, the USR will not be used for reading the distance and instead a value of 800 will be inputted. This was done to purely test the NCO and its output on its own so that correct functionality is confirmed.

3.2 USR Testing

Testing for the USR had to be done in stages and it had a variety of different approaches until one that worked was found. Initially, the idea was to implement a custom IP using Simulink in MATLAB, but was proven to be rather difficult because of the bidirectional pin on the USR (SIG).
For the second approach, it was thought to use the custom IP generator tool by Vivado and configure the corresponding input and outputs in the circuit to the ones on the *Grove Adapter*. This would enable the whole circuit to be controllable through code. This concept turned out to be rather difficult because of the complexity involved in both the constraints and IP files (to make the custom IP) as well as the testbenches and the required code (for testing), since there wasn't much experience in working with Vivado.
Therefore, following the structure of the iop_pmoda hierarchy not only solved the aforementioned problems, but also using something pre-existent meant the crossed-results could be checked and functionality of the two designs could be compared to find faults – a much better solution. The use of the c code meant it was easier to understand and troubleshoot if needed, since knowledge of c language exceeded that of the python corresponding.
During the testing phases, it was decided to try and make the whole overlay smaller in the sense of deleting the IP blocks and hierarchies that the project didn't need. This was done following the concept of "remove-till-failure" but failed, as the moment a single block was removed, the program would stop functioning as expected.

3.3 Full Project

To test the entire project, the cells in the notebook were ran and changes were made accordingly.

4. Approach To Project Management

The project was split into two parts, NCO development and PMOD with USR development. Multiple Zoom meetings took place and various face to face meetings as well. Forum posts were made for the more difficult aspects of the project to try and clarify the direction that should have been taken and how to resolve certain errors. If any challenges arose they were discussed within the group for what the next steps should be. In some cases two group members did the same task as to discard or find different solutions for the same problem.

5. Reflections

Reflecting from this project a variety of new material was learned that was not necessarily taught in lectures and some was further expanded upon. Further experience creating custom IPs and implementing them into Vivado was gained. Usage of the PYNQ-Z2 board also provided further exposure to this kind of software and hardware development which could be useful for future career endeavours since this type of technology is slowly being more accepted and more widely used in the industry. Independent research was required for a variety of parts for the project. Posts in the forum and online search helped with aspects such as how to implement an NCO with the audio codec and to create an IP for the Grove Ultrasonic Ranger. Due to the amount of testing required to create a working project, skills were also enhanced in using Simulink, Vivado and especially the Jupyter Notebooks since a variety of code and commands were used to decide how the project functioned.

6. References

Greer, Ryan. 2020. GitHub. [Online] 14 May 2020. [Cited: 13 May 2022.] https://github.com/RyanMan1/PYNQ-Theremin/tree/master/Source%20Files.

ReadTheDocs, Pynq. 2021. *Pynq ReadTheDocs.* [Online] 2021. [Cited: 28 04 2022.] https://pynq.readthedocs.io/en/latest/pynq_libraries/pynq_microblaze_subsystem.html.

Taylor, Adam. 2020. *PYNQ and IOP on Ultra96 Breakout.* [Online] 21 Dec 2020. [Cited: 14 May 2022.] https://www.hackster.io/adam-taylor/pynq-and-iop-on-ultra96-breakout-84564c.

University of Strathclyde. 2022. MyPlace. [Online] 2022. [Cited: 13 05 2022.] https://classes.myplace.strath.ac.uk/mod/resource/view.php?id=1696060.