Loading Dock Orchestra
Final Report

1st Tanner Packham
Department of EECS
Massachusetts Institute of Technology
tpackham@mit.edu

2nd Tzu-An Sheng
Department of Physics
Massachusetts Institute of Technology
tasheng@mit.edu

***Abstract*—We utilize the capabilities of an FPGA to drive multiple stepper-motor containing electronic devices (printers, floppy drives, etc) to play the different frequency components of an incoming audio signal. This design is implemented using the Nexys 4 FPGA board together with a PMOD IS2S analog-to-digital converter module. We evaluate its performance by measuring its overall latency and its ability to differentiate and play different frequencies of incoming audio.**

***Index Terms*—Field programmable gate arrays, Fast Fourier Transform, digital signal processing, audio, stepper motors**

I. Introduction

The need to harmonize a melody arises constantly whenever a songwriter tries to come up with inspiration for the latest hit, a singer wants to practice without a band, or simply a group of partying friends are looking for some novelty in their choice of music. We propose a project that involves building a portable device that listens to an incoming audio, pick up a melody and play it in real time.

II. Physical construction

The physical construction of our device addresses two important needs:

- Firstly, it needs to digitize an incoming stream of analog audio.
- Secondly, it is able to drive a stepper motor in a given direction and speed. The speed of the stepper (in steps per second) can be heard as an audible frequency.

Our device is fully contained within a peripheral board that can be connected to two of the PMOD ports on Nexys 4. These ports contain 8 I/O pins each, which serve to send and receive the data needed to control the analog to digital converter (ADC) and the stepper motors. The PMOD ports also provide a 3.3 V supply voltage that is used to drive the logic on both the stepper motor and the ADC board. The peripheral board has a female XLR input jack for a microphone connection and a female 1/8 inch jack used for a line-in audio connection. It also has a 4-pin JST connector through which a stepper motor can be plugged into the board, and a 2-pin JST connector through which the board is supplied with 24 V.

The ADC we choose to digitize the audio is the PCM1808, made by Texas Instruments [1]. It is a separate circuit mounted to a perfboard using header pins. The perfboard, shown in Fig. 1, is connected to the Nexys 4 via 90 degree male header pins into the PMOD port. The ADC needs 3 clock signals to function, as well as 3 separate pins that can be pulled high or low to specify the functional mode. Data is transmitted through another pin. All 7 of these pins are directly wired to the PMOD I/O pins. The ADC has a left and right audio input, where the left input is directly connected to the line-in jack and the right input is connected to the XLR input with a small 23 dB preamp in between. This preamp uses the LM358 op-amp and serves to bring the microphone output up to approximately the same level as line-in. The 5 V supply needed to drive this section of our circuit is generated using an LM7805 voltage regulator connected to the 24 V power supply for the stepper motor.

The stepper motor circuit uses an A4988 stepper motor driver, which is once again a separate circuit connected to the peripheral perfboard using header pin connectors. It has inputs for stepper motor step and direction, as well as inputs to enable the chip and select microstepping modes. All input pins of the A4988 are connected directly to the PMOD pins of the FPGA.

To reduce noise in our peripheral board, many filtering capacitors (both ceramic and electrolytic) were used. These were placed near the power input pins of every subsection of our circuit that has power supply and ground connections (i.e. ADC, A4988, op-amp, voltage regulator). The largest capacitor(1000 μF) was placed across the 24 V input for the stepper motor.

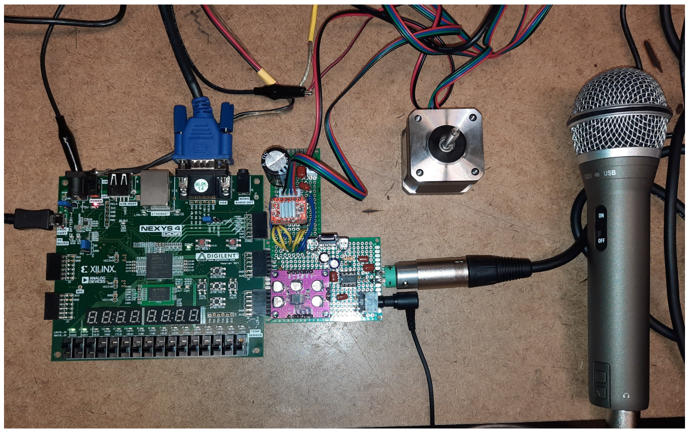

Fig. 1. (Top left to bottom right) The FPGA and Nexys 4, peripheral board, stepper motor, microphone and audio line-in

III. Audio signal processing

The source code of this project can be accessed on https://github.mit.edu/tasheng/loading-dock-orchestra.

A. Audio input

The audio is fed into a PMOD IS2S adapter that converts the analog audio to digital at a sampling rate of 44.1kHz. A dedicated module takes care of this step. The signal then goes through a low pass filter module in which the frequencies above 4kHz are filtered out. To do this, we calculated the appropriate coefficients for a finite impulse response filter with our desired specifications, then generated a Vivado FIR compiler IP core. The output audio is then down-sampled to 8kHz and fed into the Fast Fourier Transform (FFT) module.

B. Fast Fourier transform

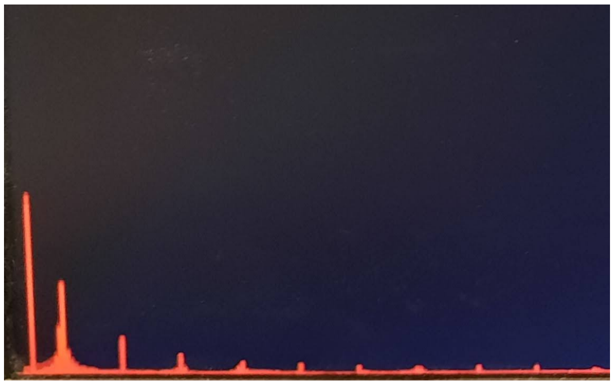

Fig. 2. FFT Output with the audio pipeline

We then use a pipeline similar to the one provided in lecture. Here, 1024 samples are collected before passing into an FFT IP block. The time series is padded with 1024 zeros to assure that we can get a 2–3 Hz frequency resolution at a throughput of 8 FFT frames per second. The data are decomposed into its various frequency components. The real and imaginary parts of each frequency bin are squared, summed, and then taken square root of to provide the amplitude of the FFT outout. This happens in 2 separate modules. The frequencies and corresponding amplitudes are stored in a FIFO for use by the instrument selection module.

C. Peak finding and output

The instrument selection module searches for the local maxima in the spectrum by looking at every five bins with strictly decreasing 2nd-order derivatives in the discrete sense. It then picks N maxima with highest amplitudes, where N is the number of external instruments. If there were 3 instrument peripherals connected to the FPGA and 4 frequency peaks, the 3 highest peaks would be sent along to the hardware audio peripherals to generate sounds at the given frequencies. The stepper motor control module will step any stepper motor (in floppy drives, printers, etc.) at the frequency that the instrument selection module tells it too. It will reverse stepper direction occasionally in the case that it is driving a device that is linearly actuated, e.g. a scanner. In this report, we present a design with 2 hardware peripherals obtained from the loading dock. Due to the modular nature of the design, the number of instruments can easily be expanded in the future.

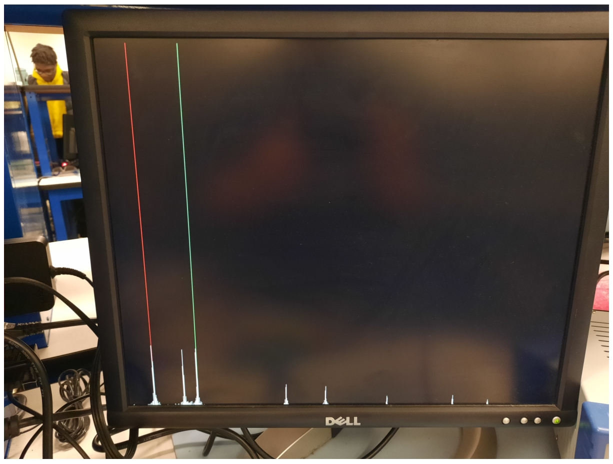

Fig. 3. Finding the largest (red) and the 2nd largest (green) peaks

The combination of all these modules will allow us to accomplish the goal of reproducing and harmonizing to live audio on a variety of scrap electronic devices. A system that plays music on scrap electronics has been built before, but these systems need someone to prepare a custom MIDI file to make them play music. The novelty of using an FPGA in our system is that we can instantly make it play any song because the FPGA will very quickly decide how to play the instruments depending on the input given to it.

IV. Evaluation

The design uses 92KBit of RAM. 2048×32 bits are for the FIFO that hold the output of the square-and-sum module, and 1024×28 bits function as a ROM look-up table that maps the FFT output frequency bin numbers to real frequency numbers. 240 DSP are used for the square-and-sum and the CORDIC square root modules.

There are three clocks in this design. A 100 MHz system clock that drives most of the modules, a 65 MHz clock that interface with the VGA. and a 22.57921 MHz clock for the PMOD audio input. According to the post router timing report, the worst negative slack is 1.952 ns with 1-motor output; we meet all the timing requirements for this case. With 2-motor output, the worst negative slack becomes -1.648 ns. Nevertheless, there is no significant impact on the performance. We measure the overall latency by probing the audio input and the stepper output with an oscilloscope. The latency is measured to be 22–49 ms. The throughput is limited by the theoretical limit of the FFT sampling window at 8 Hz.

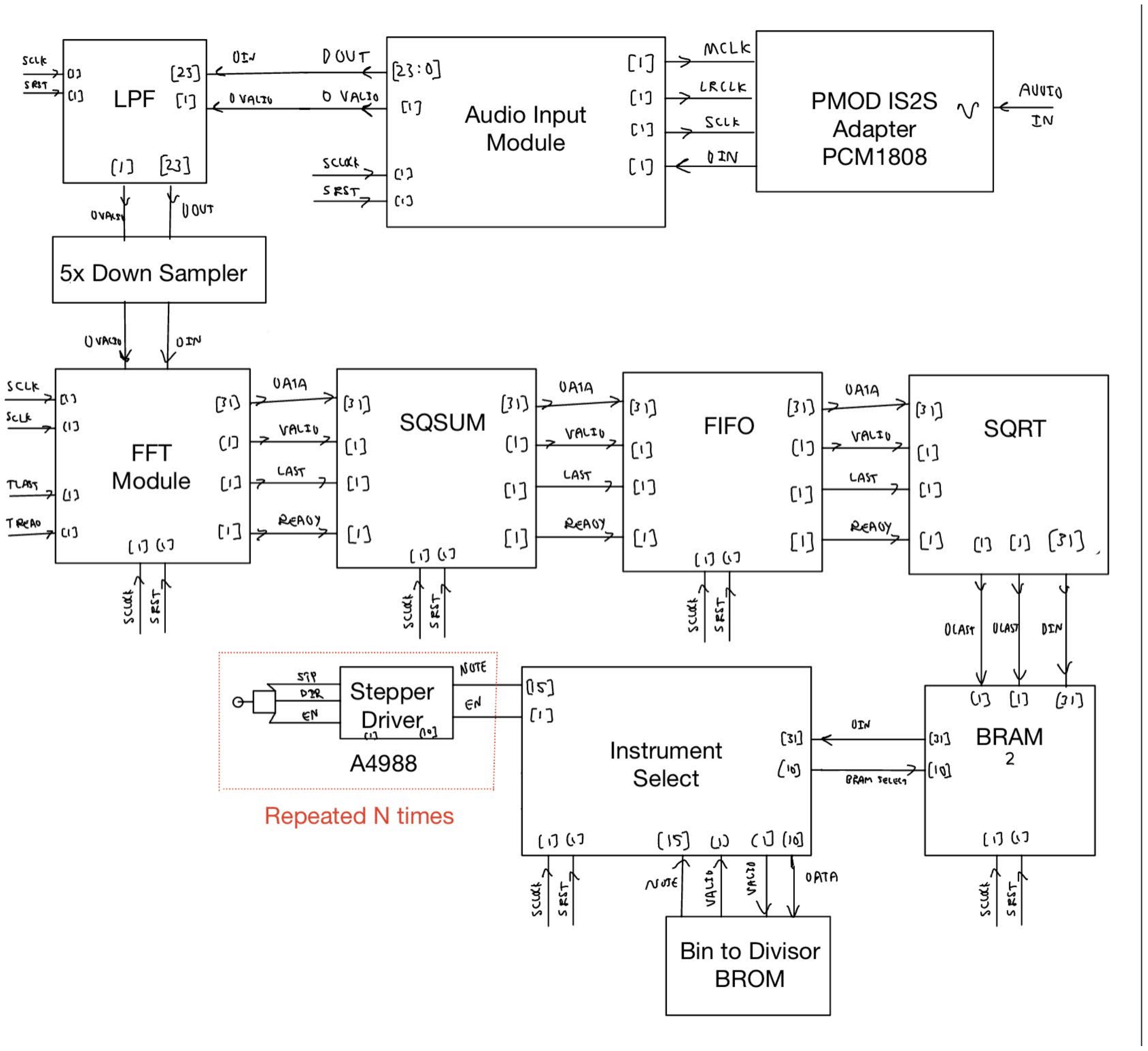

Fig. 4. Block diagram of the signal processing pipeline

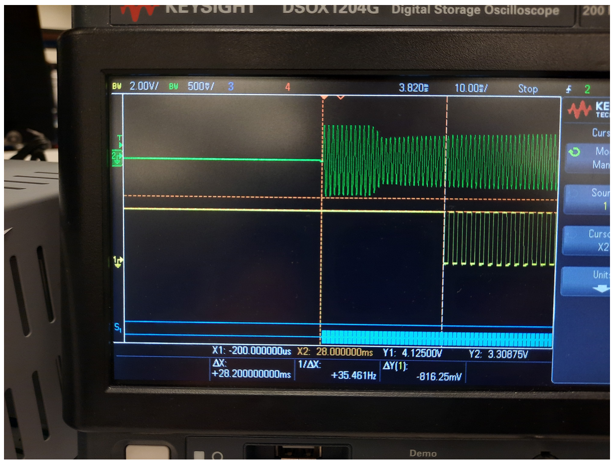
KEYSIGHT DSOX1204G Digital Storage Oscilloscope
X1: -200.000000us X2: 28.000000ms Y1: 4.12500V Y2: 3.30875V
ΔX: +28.200000000ms 1/ΔX: +35.461Hz ΔY(1): -816.25mV

Fig. 5. Latency measurement

The physical setup was first evaluated by creating a new top level module to serve as a temporary test-bench. This skeleton module communicate with both the ADC and the stepper driver. The ADC was tested by routing the most significant bit of the ADC data output to one of the unused PMOD pins. When a sine wave was fed into the audio input, a square wave of the same frequency was produced on said pin. This confirmed that the ADC was working. The stepper was tested by giving it a fixed frequency to step at (controlled by the switches on the Nexys 4) as well as routing the MSB of the ADC to the stepper driver. This had the effect of stepping the motor at the same frequency as the incoming audio (if the given audio is a sine wave). These tests proved to be very successful.

In our first preliminary test of the whole system, we fed the incoming audio signal (sampled at 44.1 kHz) directly into the FFT module via an finite impulse response low-pass filter set to 4 kHz. Fig. 2 shows the FFT output of an incoming sine wave through the 3.5mm headphone jack. We test several notes around C3 (130 Hz) and observe that the frequency separation at the lower end of the spectrum performs poorly. This has improved after we down-sample the audio. It is also clear from Fig. 2 that a 60 Hz interference peak is always visible. As a result, we constructed our peak finding module to ignore frequencies below 60 Hz.

In our final test, the signal went through the entire described in Fig. 4. The two motors successfully played at different frequencies from the input. As expected, the reproduced pitches are accurate at the higher range and mediocre at the lower end, where the notes are closer to each other. We have achieved our ideal goal.

Tanner Packham contributed to the physical construction of the driver boards, the interface between Nexy4 and the peripherals, the PMOD adapter, the audio input module, the down sampler and the stepper driver modules. Tzu-An Sheng contributed to the low-pass filter, the FFT module and its output, and the peak finder module.

References

[1] Texas Instruments. *PCM1808 Single-Ended, Analog-Input 24-Bit, 96-kHz Stereo ADC*. Revised Aug 2015. https://www.ti.com/lit/ds/symlink/pcm1808.pdf