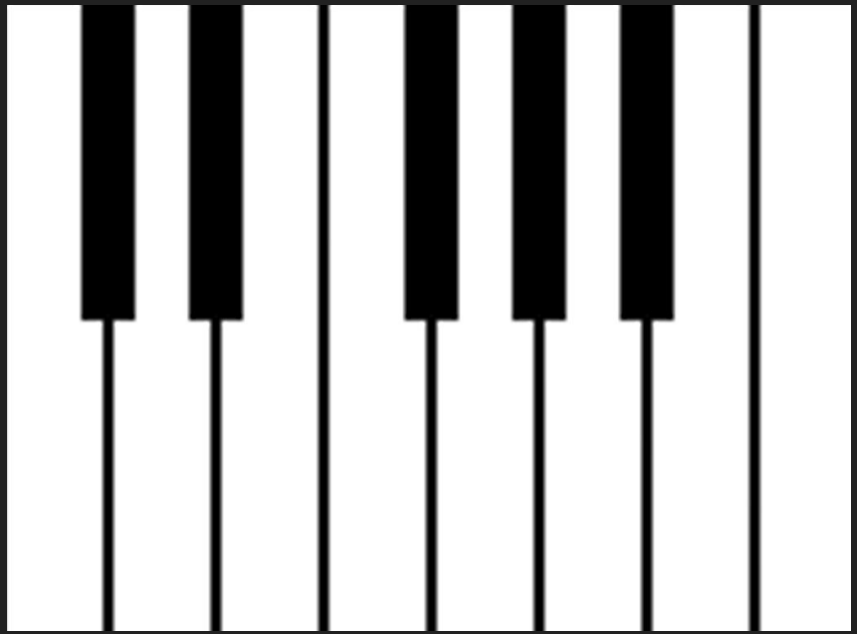
Piano Synthesizer Utilizing DE1-SoC + PS/2 Keyboard

ECE241 Project

Harish Babu
Dylan Ho

Project Description



Piano Musical Instrument Synthesizer

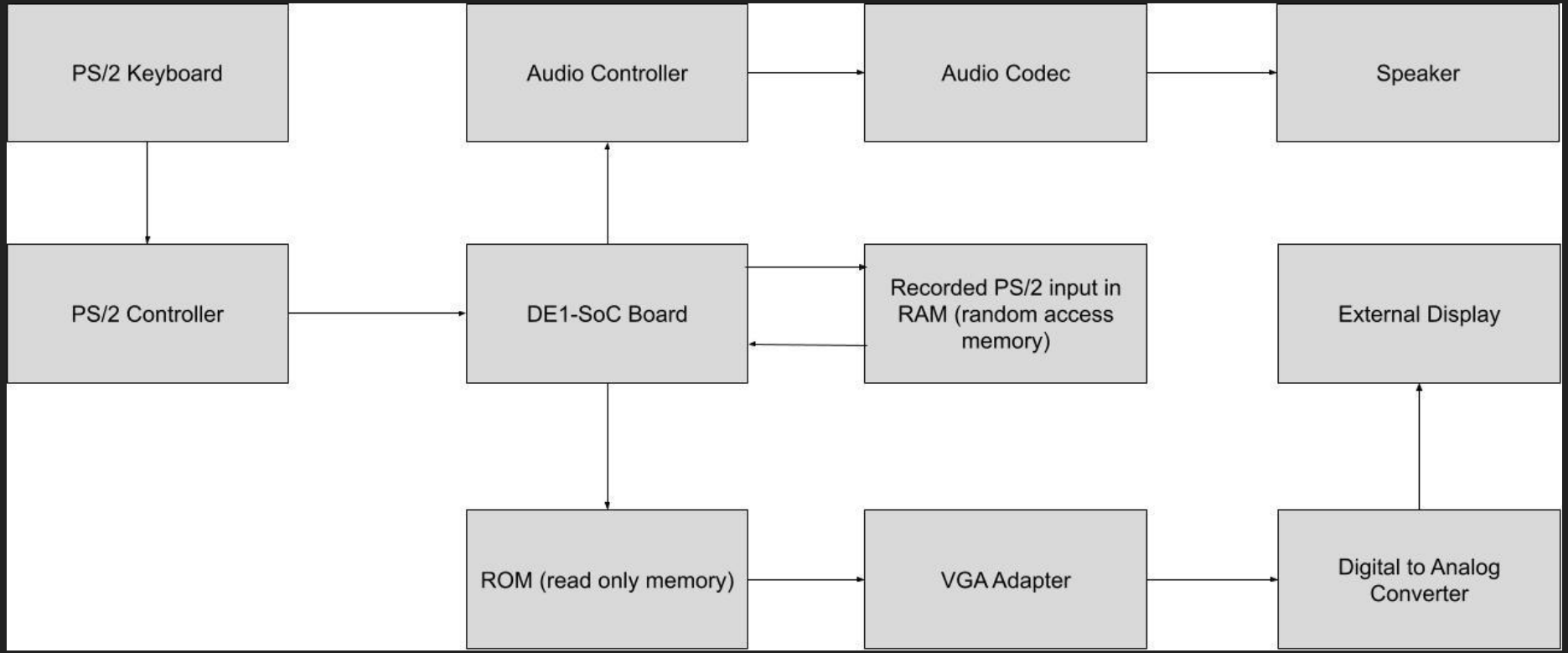
The objective of the project is to develop a musical instrument that converts a user input into an audio tone. Our instrument is based off of a traditional piano. It features 13 unique tones and 13 unique inputs.

Input is captured using a PS/2 Keyboard. Our piano logic is processed on the DE1-SoC board and outputted to a VGA display and a set of stereo speakers for visual and audio output.

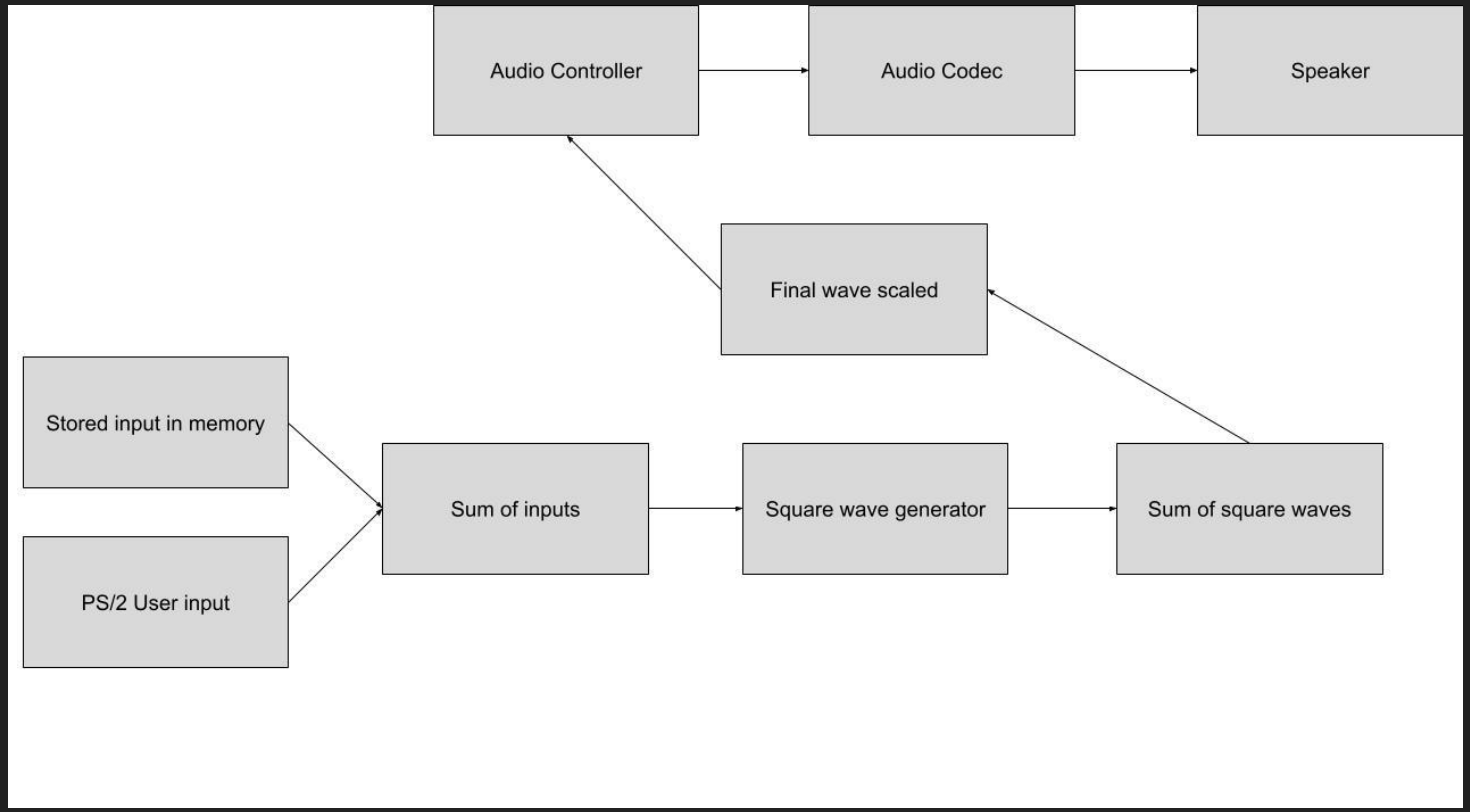
In addition to operating as a live input musical instrument, the synthesizer is capable of capturing 10 seconds of user input, and playing it back at will.

Block Diagrams

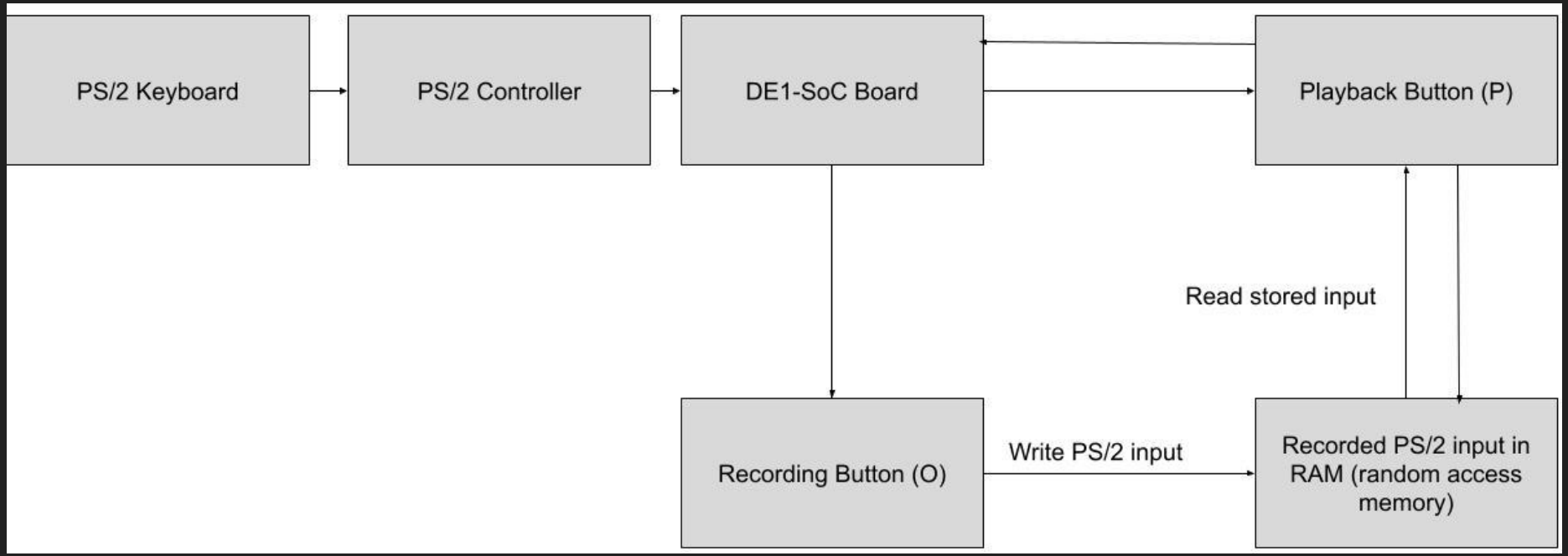
High Level Block Diagram



Audio Block Diagram

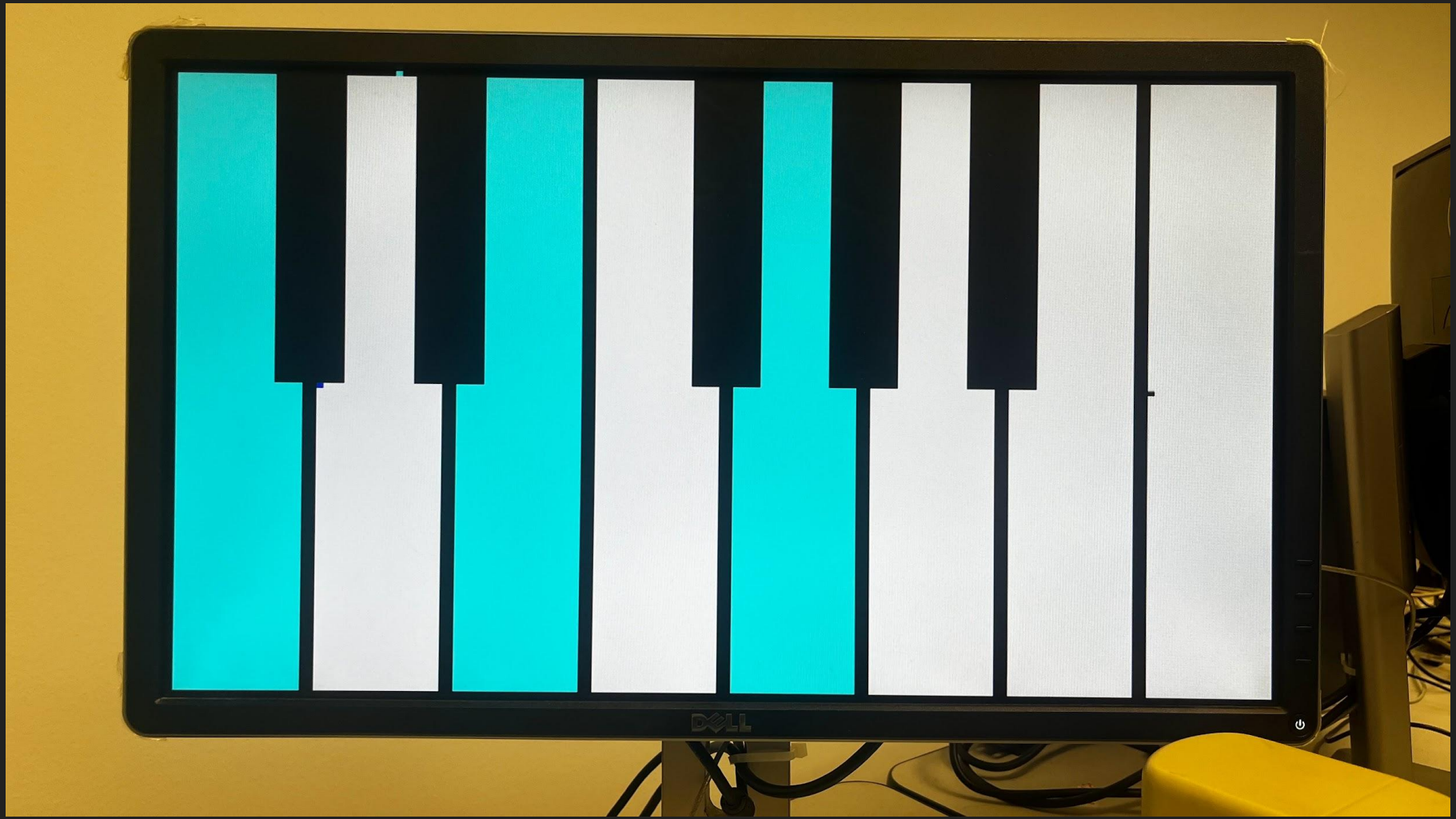


Recording Block Diagram



Videos + Demos


DELL
ABSOLUTELY
FOOD OR DRINK
ON THIS DESK
ENJOY
MULTI-MEDIA
SPEAKER SYSTEM
EP-691A



Bugs/Issues

Audio Quality with Multiple Notes

The logic that manages the output wave to the audio controller creates a sine wave for each individual note, and sums the notes together. The final amplitude of the note is then scaled by the total amount of notes being played.

Due to the way the final wave is summed and scaled, there is a noticeable loss in audio quality. When more notes are being played, the volume also is reduced.

Multiple Key Press Bug

During the transition between single key output and multiple key output, an issue with multiple key presses not registering past F4 was encountered repeatedly. This issue would occur seemingly randomly, and no common factors were identified when the issue arose

Possible causes:

- PS/2 Keyboard hardware issues (shared traces for multiple keys, N-key rollover limits)
- Logic issues in code

VGA Artifacting

A visual bug occurs when a key is highlighted. Pixels appear in spots all over the screen, and in the incorrect color. Potential causes:

- X/Y value being passed to VGA controller is driven by multiple counters, resulting in potential overlap where the X and Y coordinate are not being reset to zero before the next counter begins.
- On board clock speed potentially not fast enough to avoid transitions between VGA states being seen.
- Colour being sent to the VGA is cycling through colours from many MIF files, potentially causing overlap colours at certain XY locations on screen.

VGA Graphics

A bug occurs where the blue highlight of a key does not disappear until all keys being pressed are released. This issue is caused by how the VGA FSM handles resetting the image. Potential fixes:

- Implementing new states and drawing over the key instead of resetting to background image
- Creating a new wire that acts as a flag when a key is released (similar to keyPressed), this can be done with an additional module
- Create equivalently dimensioned mif files that will colour over highlighted keys (to erase them) and create additional states in the VGA FSM for them

Future Work

Improving Audio Quality

Changing the algorithm used to create the final square wave will improve the audio quality of individual notes and chords. There are various ways to do this:

- Generating sine waves instead of square waves
- Changing how notes are scaled, and develop scaling factors based on the total frequency and amplitude, not number of notes
- Adjust frequencies of notes so they work together as a square wave

Implementing MIDI Controller Support

In the future, it may be possible to utilize an external MIDI keyboard controller instead of a PS/2 keyboard as an input device. This could be done by using a microcontroller (arduino) to receive inputs from the MIDI controller and convert them into a bitwise code or PS/2 signal that would be inputted into the FPGA.

Work Distribution

Harish:

- PS/2 Controller research and implementation
- VGA Controller research and implementation
- On-board memory research and implementation for VGA elements and recording feature

Dylan:

- Audio controller research and implementation
- Multi note implementation for both live and recorded playback
- Keyboard logic