Digital Polyphonic Synthesizer

ECE241
Final Project

Diego Barriga, Hakyum Kim

Detailed description

We built a digital audio synthesizer on the DE1-SoC FPGA that generates sound from PS/2 keyboard inputs and plays it through an audio codec. The system creates two waveforms, mixes them based on a user-selected percentage, and shapes each note using an Attack–Sustain–Release (ASR) envelope. All processing from keyboard decoding to waveform generation to final audio output—is implemented fully in hardware using Verilog. We got the inspired from Diego's synthesizer he had for his personal hobby.

Keyboard Input → Note Processing

- The PS/2 module reads scancodes, filters break codes, and outputs an 8-bit note value.
- This note value is converted into a frequency used by the waveform generators.
- Trigger signals also inform the ASR envelope when a note is pressed or released.

Waveform Generators

- Two separate waveform generators run in parallel
- Each generator produces a continuous digital sample stream at the target frequency.
- These raw waveforms are routed to the mixer regardless of which combination the user selects.

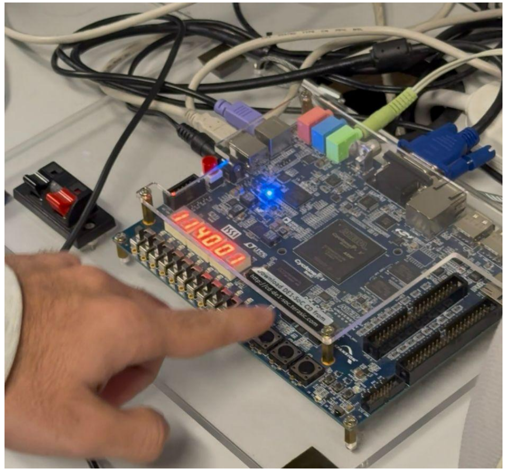
Waveform Mixing + Percentage Control

- The mixer takes the two waveforms and computes (Output = Wave1 x Audio% + Wave2 x (1- Audio%)
- Scaling and clipping logic prevent overflow and keep the signal within sae audio range.
- This produces a custom output based on the user's selected mix.

ASR Envelope → Final Audio Output

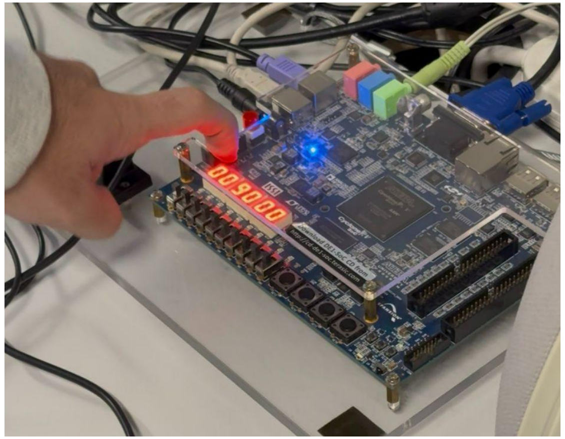
- The ASR module shapes the volume curve of each note (attack rise, sustain hold, release fade).
- The envelope multiplies with the mixed waveform to create dynamic and expressive audio
- The final signal is streamed to the audio codec via I^2C-controlled configuration and continuous sample output

Pictures of your project working & detailed description

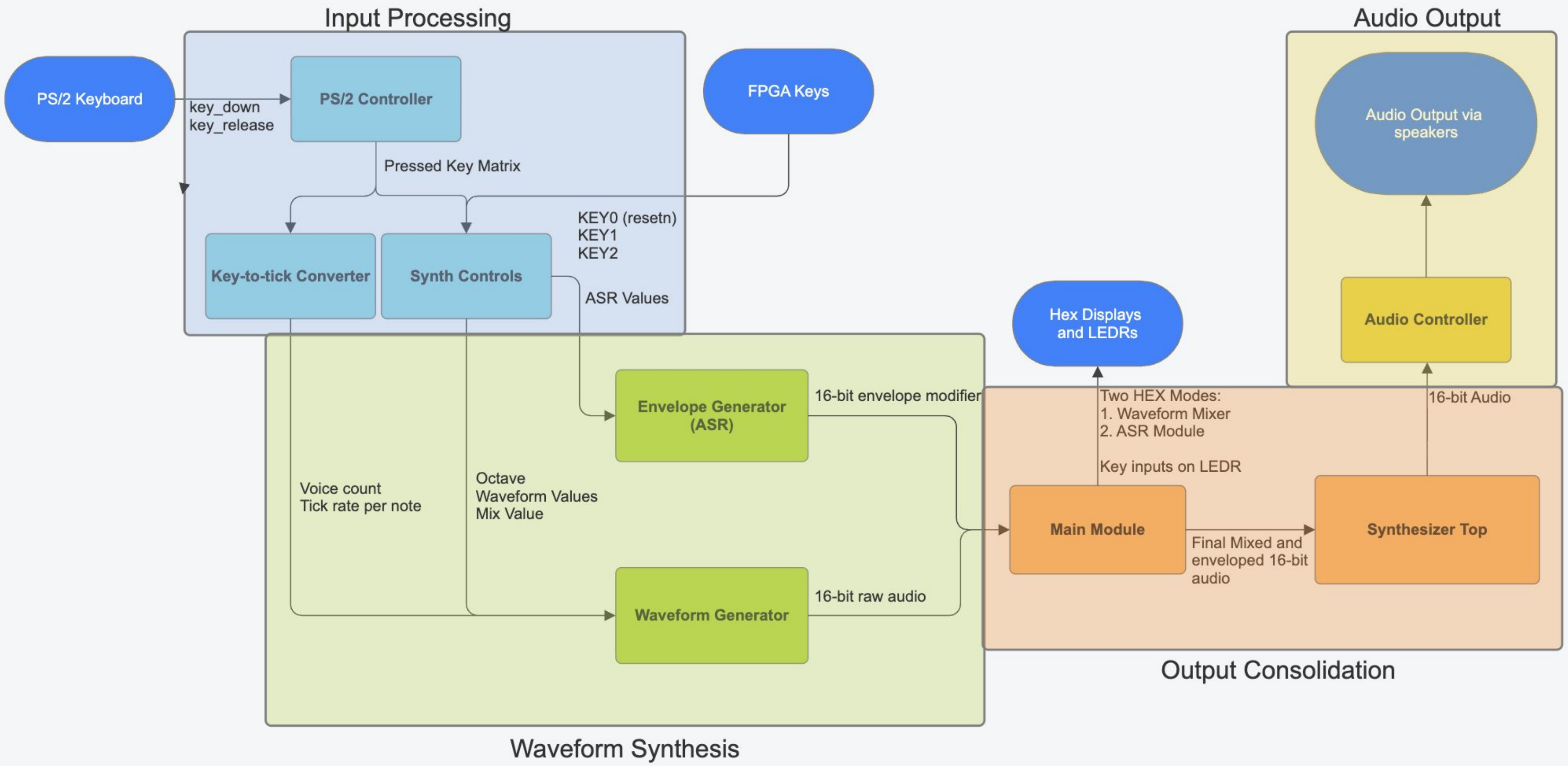


- **Waveform Generator and Mixer:** Users select two waveforms (saw, pulse, triangle, square) using the arrow keys on the PS2 Keyboard, which the FPGA converts into waveform selection values and displays on the HEX display in real-time.
- **Hardware Generation:** The FPGA generates the selected waveforms internally using hardware-based counters and lookup tables. These waveforms are continuously updated and output as digital signals, ensuring real-time waveform generation without relying on software.
- **Waveform Combination:** The FPGA combines the waveforms in real-time by performing a weighted addition. The mixer adjusts the amplitude of each waveform according to the mix ratio, ensuring that the two signals blend smoothly. The resulting output waveform is a dynamic mix of the two inputs, ready to be sent to the next stage of the audio pipeline.

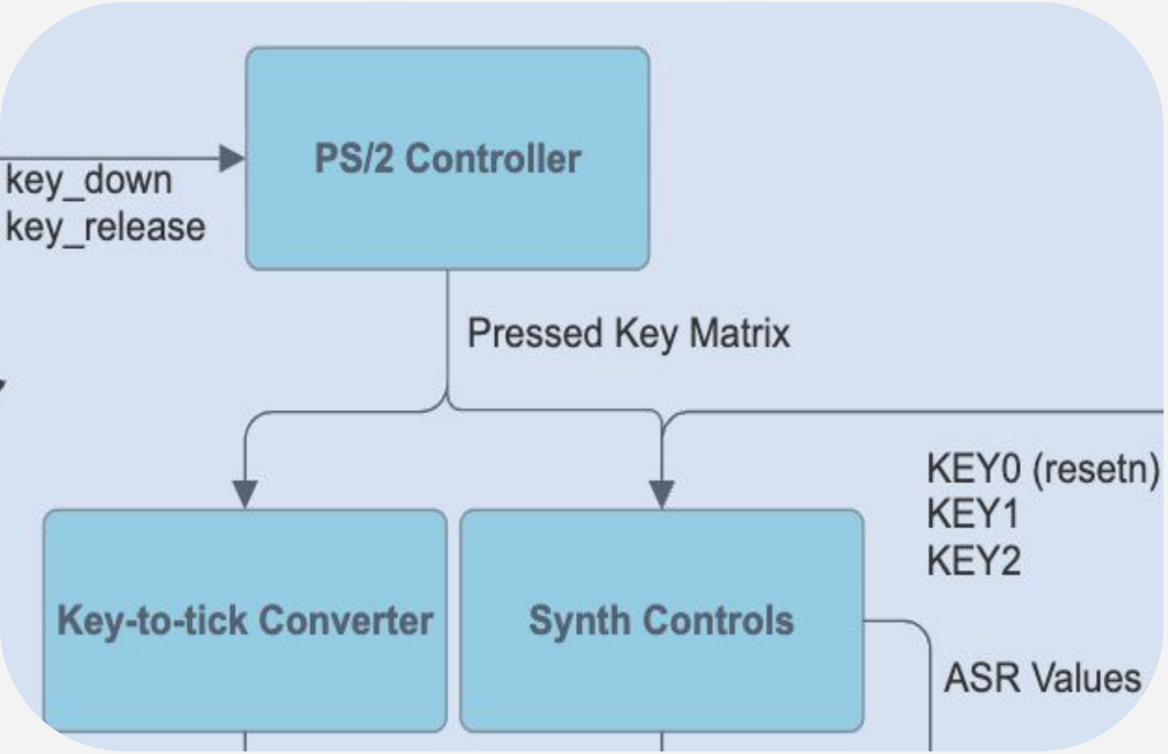
- **Attack:** When a key is pressed, the Attack phase starts, and the FPGA controls how the sound's volume ramps up from zero to its maximum level. This is achieved by gradually increasing the amplitude of the waveform over a user defined period. The Attack timing is controlled by internal counters, and the amplitude is adjusted in real-time to create smooth or sharp volume transitions.
- **Sustain**: After the Attack phase, the Sustain phase holds the volume steady as long as the note is held. The FPGA continuously checks the key's state (pressed or released) and keeps the output waveform at a constant amplitude during this phase. The sustain time is also controlled by a real-time counter, ensuring the volume remains consistent until the key is released.
- **Release:** Once the key is released, the Release phase begins, and the FPGA smoothly reduces the amplitude of the waveform to zero over a user-defined period. The FPGA uses another counter to gradually fade the volume, adjusting the signal's amplitude to simulate a natural decay.



Top-Level Project Block Diagram



Input Processing

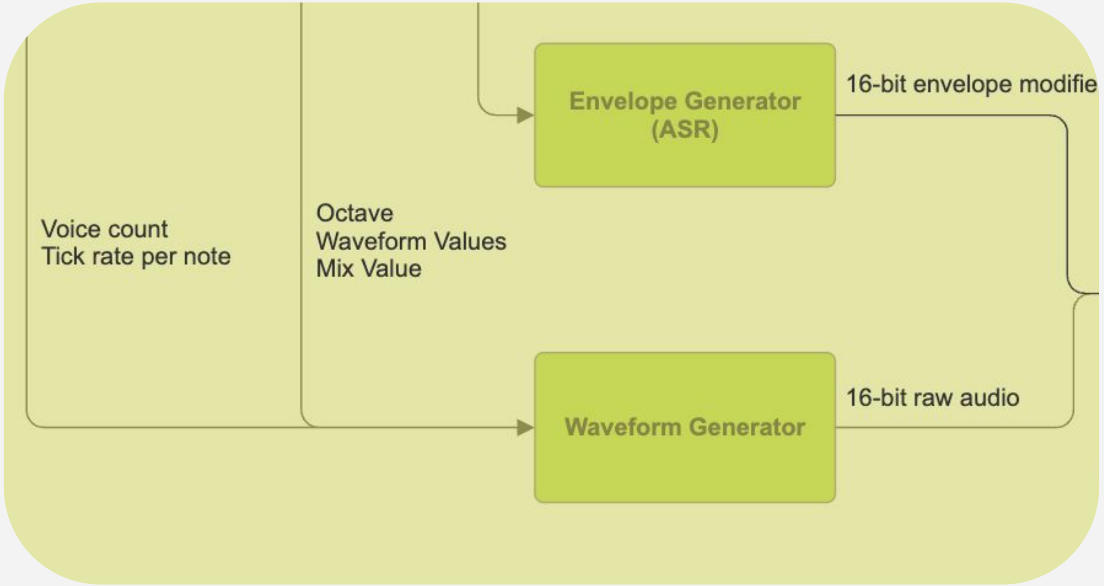


Transforms user input from keyboard and FPGA keys into usable information in the program.

Modules are:

- PS/2 Controller: Takes PS/2 input and returns scan codes, we then store pressed key(s)in a matrix.
- Key-to-tick converter: Takes the key scan codes and returns musical note and its corresponding tick rate
- Synth Controls: Takes the key scan codes and uses them to control synthesizer parameters

Waveform Synthesis

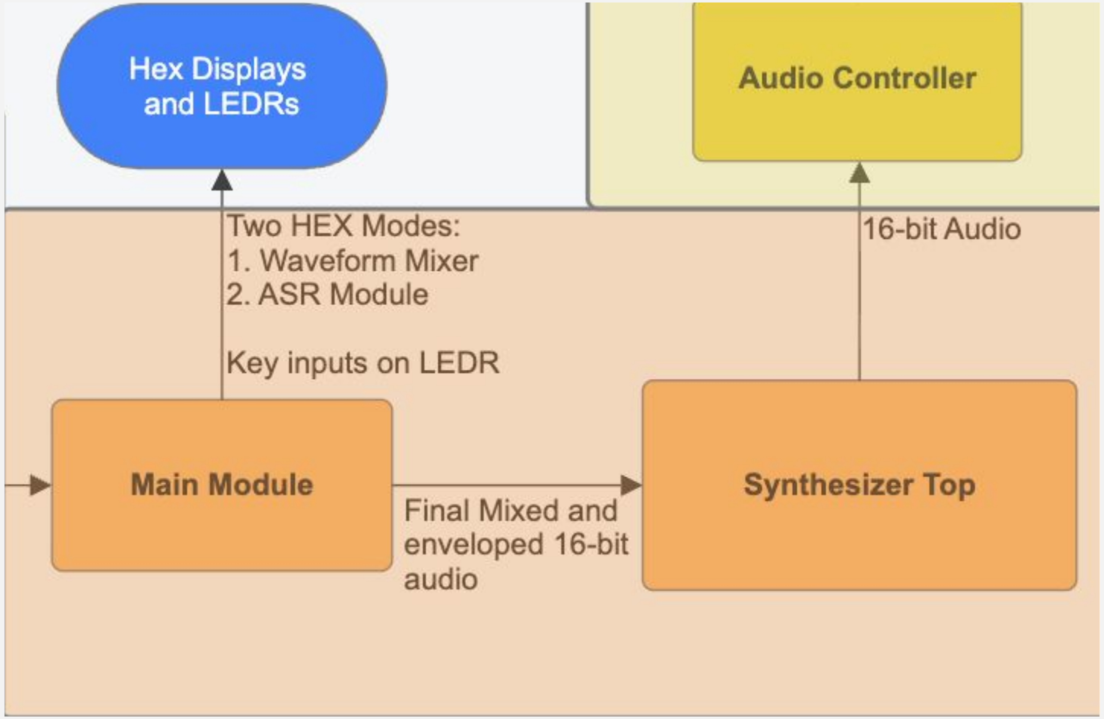


Synthesis process module. Waveforms we will use for the output as well as the envelope modifiers are done separately.

We have two modules:

- Envelope Generator: Takes attack, sustain and release values from synth_controls which are used in parameters in the envelope FSM.
    - Waveform Generator: Generates waveform1 & waveform2 for each voice according to correct tick rate, mixes between waveforms and creates final voice mix. Applies octave modifiers before sending a 16-bit audio out

Output Consolidation

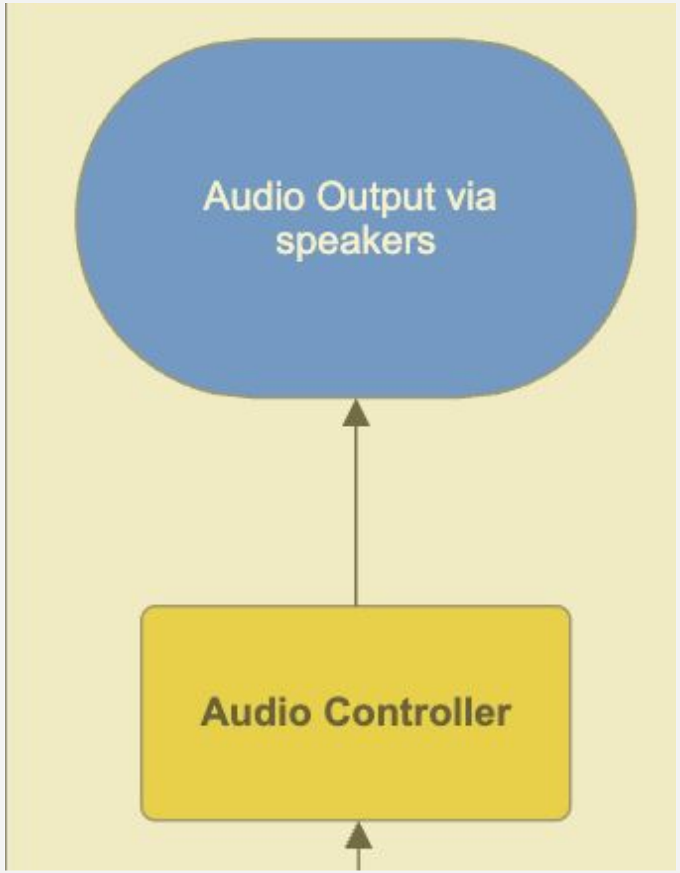


In these modules, we consolidate the output mix, combine the envelope and waveform outputs and pack final mix into 16-bit audio buses to send to audio controller.

Modules are:

- PS2 Main(Main Module): Combines waveform generator output and envelope modifier for final audio output, displays relevant info on HEX displays
- Synth Top: Takes final polished audio and sends to audio controller

Audio Output



Audio Output Process

Once the processed 16-bit audio is then transferred to the Audio Controller, which handles the necessary timing and control to ensure smooth output. The Audio Controller converts the signal into a form that can be played through the speakers, delivering the final sound to the user.

Bugs and issues encountered

During the development of the synthesizer system, several bugs and issues arose, ranging from timing and synchronization problems to challenges with data transmission. Below is a summary of these issues and the steps we took to resolve them.

ASR Release

We had achieved successful audio output by detecting pressed musical notes and storing them as voices . However, once you released them, the waveform generator stopped producing. In consequence, release did not output anything regardless of value. We fixed this by creating a temporary register to store the last notes played, which then were played for the release cycle of the envelope.

ASR Attack function error

In the envelope FSM, the program would step through the phases (Attack, sustain, release) for different conditions. However, attack was looping over and over like an oscillator due to not enough bit space for the phase to complete. In consequence, it was never able to finish the attack cycle and step into sustain.

Transmission issue

The system repeatedly transmitted the same data, causing audio distortion. This was due to the lack of logic to detect changes in keypresses. We added change detection in the synth_controls module using a keydown register to track key states and detect rising edges. Additionally, edge detection logic was implemented in PS2_Main to compare the current and previous key states, ensuring data was only transmitted when a new key was pressed. This solved the issue and improved audio performance.

MIXER PROBLEM

To avoid distortion from mixing closely-tuned notes, we implemented voice separation and individual mixing. Each note is processed separately, with a mix percentage balancing waveform 1 and waveform 2. The pitch is adjusted by octave, and the final output combines the voices: one for voice1, two averaged, or three mixed together. This ensures cleaner, more musical output.

Future Work

Building upon the establishment we've made, the synthesizer could still be improved with additional features.

- Features such as filter, pitch modulation, chorus and distortion
- Expand the sound bank of the synthesizer (additional waveforms)
- Increase the number of waveforms being mixed in real-time.
- Connect input to midi controller keyboard
- Implement user-defined effects (reverb, delay, and modulation.
- Add a virtual user interface
- Recording feature

Cutoff: 0.728 Res: 0.52

Rate: 5.70 Hz

A: 3.40 ms D: 400 ms S: 0.00 R: 170 ms

A: 830 ms D: 7700 ms S: 0.29 R: 480 ms

Final Work Distribution

Diego Barriga:
Developed the waveform generator module and connected input processing modules into synthesis modules. Developed the envelope generator module and configured the consolidation of output to send to audio buses.

Hakyum Kim:
Optimized the synthesizer control module to manage user inputs for octave, waveform, mix, and ASR settings efficiently. Also optimized the main module by refining key detection and reducing latency for improved performance.

We developed modules to translate PS/2 scan codes into usable and context-focused data to use in our modules. Wired and debugged all modules to ensure proper flow between user input, synthesis and visual and audio outputs.

THANK YOU