Rave-in-a-Box

6.111 Fall 2018 Final Project Proposal

Team Members: Samuel Cherna, Matthew Reeve, Joshua Gruenstein

October 30, 2018

We intend to create a musically-controlled laser light show, capable of processing incoming audio and using extracted features to laser project engaging visuals on a nearby surface. This project has numerous technically interesting components: audio processing using Fourier-based methods, the storage and interpolation of vector graphics, and real-time control of galvanometers. At a very high level, we present the following block diagram:

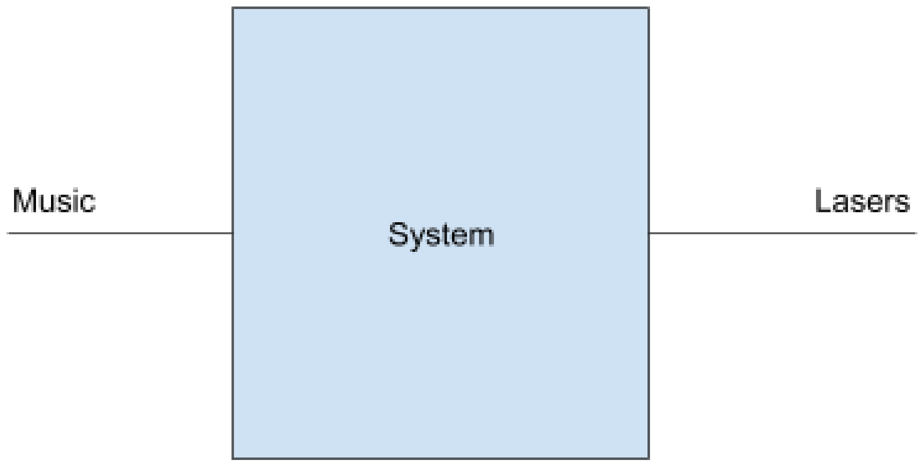

Figure 1: Extra-Macro Block Diagram

Our end goal is to project sequences of vector graphics, chosen by scene feature similarity to audio features extracted from music. These sequences of images will contain transitions also parametrized by audio features. Below is a block diagram representing our intended design for an FPGA implementation of this system, upon which we will elaborate throughout the remainder of this proposal. Note that this diagram is rendered at the macro level, and does not show the submodules involved in each module displayed.

Macro System View

Our system overall interacts with the following IO. The macro input to the system is the signal from the ADC on the Nexys4, which connects to the audio-in port on the board. Our output is two signals that go to a 12 bit DAC (MAX525), which are then fed into the

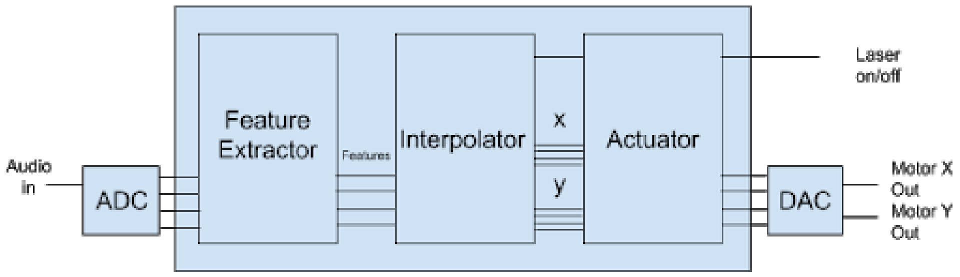

Figure 2: Macro Block Diagram

galvanometer control boards to modify the angle of our X and Y mirrors. We have three main modules that encompass all other modules in our system: FeatureExtractor, Interpolator, and Actuator. These three modules may contain submodules, but do not contain each other, and will be developed primarily by Sammy, Josh, and Matt respectively.

The FeatureExtractor is responsible for reading audio input, executing Fast Fourier Transforms on incoming data for analysis, and exporting features about the current music. These feature bits are then read by the Interpolator, which is responsible for selecting a scene based on features from the FeatureExtractor, reading scenes from a stored representation in ROM, and interpolating between coordinates stored in the scene ROM. This outputs the values X, Y, LaserOn which represent where the projector should point the laser, and whether to turn the laser on and off. Finally, the Actuator receives X,Y,ON/OFF from the Interpolator, and actually issues commands to the galvanometer by interfacing with the MAX525 DAC over SPI to ensure the coordinates and commands suggested by the Interpolator are always enforced. These modules only need to read the value at any frame from the module before it, so they should be easy to compose into one system (no additional interfacing logic is necessary).

Feature Extractor

We will extract two main features from our incoming music, both based on a running spectrogram of the music computed from the FFT. The first feature is the chromagram, which tells us about the most prominent pitch classes being played, and the second is the tempogram, which tells us about the tempo / beat locations in the music. The spectrogram will be computed as in the Vivado sample above, by doing the following things: oversampling the ADC 16x to get 14-bit precision, storing the samples in BRAM (4096 samples), passing the samples from the BRAM to the 4096-point XFFT IP Core and getting the magnitude by squaring the real and imaginary parts and then taking the square root, using IP Cores as well.

From the spectrogram, the chromagram can be computed by summing the spectrogram values in the bins that correspond to each pitch class (the corresponding bins for each pitch class can be stored in a ROM). This can be easily pipelined to compute efficiently at sample rate. We can then extract further by outputting the indices of the most prominent few pitch classes.

The tempogram can be computed in parallel, by first computing the energy novelty

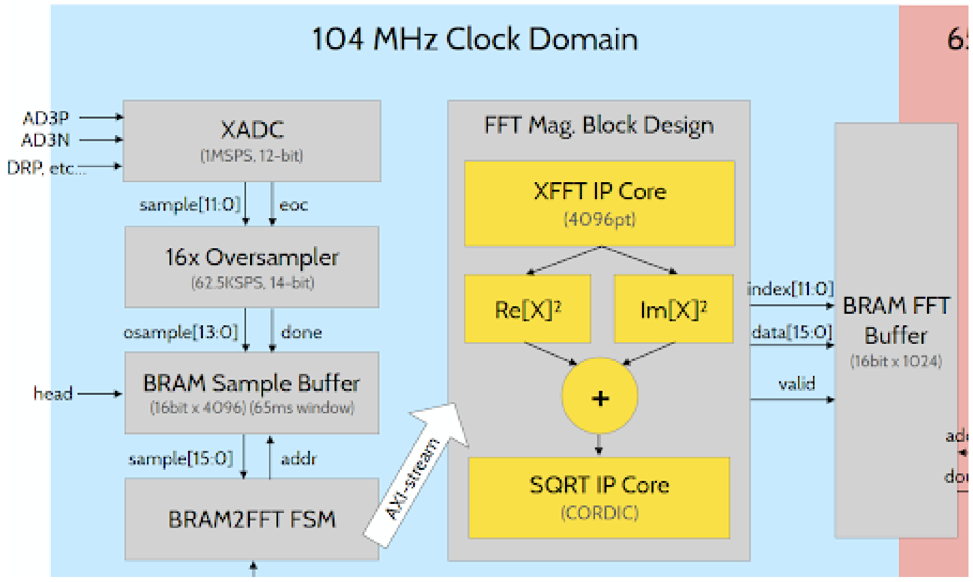

Figure 3: Spectrogram FFT Sample Diagram

from the spectrogram, and then computing another FFT on the energy novelty curve. This second FFT does not need nearly as big a window size, so it is less resource-intensive. The energy novelty is computed by subtracting the current spectrogram window from the previous spectrogram window (keeping only positive values), which gets the change in energy for each frequency band, and then summing the changes across all frequency bands. Once we have the tempogram, we can send out beat pulses at regular intervals using the clock and a counter.

These two features, the most prominent pitch classes (output as three 4-bit wires, each indicating the index of the prominent pitch class), and the beat (output as a single wire, which is 1 during a beat and 0 otherwise), are sent to the interpolator module in order to fetch and display the correct graphic.

Interpolator

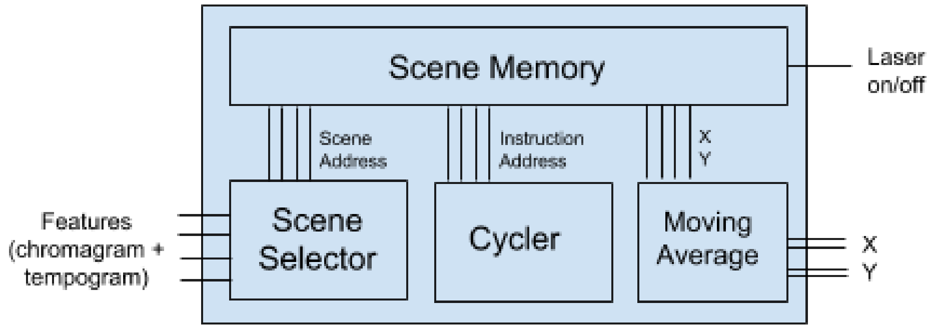

Figure 4: Interpolator Module Block Diagram

The interpolator is responsible for reading chromagram and tempogram features from the feature extractor, selecting appropriate scenes, and interpolating between points in scenes to create a continuous curve over time of coordinates the laser should project at. This functionality is accomplished by four submodules: the scene memory, the scene selector, a cycler, and a moving average module.

Scenes are stored in block ROM as a series of instructions, where each instruction contains a 12 bit X coordinate, a 12 bit Y coordinate, and a single bit representing whether the laser should be on or off at this point. Each scene contains the same number of instructions for simplicity (if there are less vertices in the vector graphics than instructions, we can pad by either repeating the sequence, or repeating the last instruction). Each scene has a unique address, and each instruction in a scene has an address one greater than the previous instruction. Thus the address format for our memory is Scene Address, Instruction Address, to which the memory outputs X, Y, On/Off.

The scene selector is responsible for hashing chromagram and tempogram features into a scene address. We can draw out a feature space represented by the chromagram and tempogram features, and will choose scene addresses for scenes such that the hash function we create will display appropriate scenes for the given features. The universal hashing theorem guarantees that such a hash function exists, and as such we plan to use some experimentation in Python to solve for the parameters our hash function will need, then use those parameters in some simple combinational Verilog that will make up the majority of the submodule.

The cycler is responsible for simply counting up through the instruction address space and overflowing to zero, at a rate that the laser can keep up with. Thus the scene memory will cycle through points in any given scene, which will then be read by the moving average module and incorporated over time using a rolling average to linearly interpolate between instruction points, generating a smooth path for the laser to follow. This module will output 12 bit X and Y coordinates suitable for encoding into SPI commands to the DAC, as is done by the actuator module.

If we achieve the above functionality in the interpolator module, we may reach for additional functionality such as creating scenes parameterized by chromagram/tempogram features (by increasing the complexity of our hash function and the number of scenes in memory), or creating smooth transitions between scenes (by creating an additional module in between the scene memory and moving average that applies some changing transition function over time to incoming points).

Actuator

In order to project vector graphics we must output analog signals to both our X and Y mirror control galvanometers. This is a nontrivial task, as SPI communication requires complex state machines capable not only of receiving incoming data, but also sending commands. We have selected the MAX525 as our DAC, which has 5 channels (of which we will use 2) each with 12 bits of precision. We have deemed this to be sufficient, as this provides 4096x4096 points upon which we can project, which we expect to be far greater precision than our motors and mechanics can actually achieve.

Hardware Notes/Limitations

The two main hardware limitations that we anticipate facing are the speed / resources required for the FFT / signal processing, and the storage required for all of the different scenes in memory. However, for both of these, we can scale down our performance in order to match the resources available. For example, if our signal processing takes too long or requires too much hardware, then we can use a smaller window size and tradeoff frequency precision. If we run out of memory for the scenes, then we can either use fewer scenes or use an SD card.

We will require the following external hardware for our project: a set of galvanometers (already lent to us by Gim), an MAX525 12 bit 5 channel DAC, a laser pointer, and a MOSFET to turn the laser on or off. Matt will build an aesthetically appealing wooden enclosure and wire up the hardware to the FPGA. We intend to purchase all materials for our project (including a Nexys4 board) so we can have fun nerdy raves at our fraternity.