Zixi Liu, CK Ong & Jacqui De Sa
October 30th, 2014

6.111 Project Proposal: Chordination

INTRODUCTION

Chordination is an FPGA-based hardware device that calculates and plays a chord progression out of a melody sung by a user in real time. It can serve as a tool for composition & arrangement, a way to improve improv performance skills, or just as a fun way of exploring musical possibilities. Chordination implements digital logic within record/playback functions, processing of pitches, and within a state machine that calculates possible new chords based on previously-played chords and the current pitch the user is singing.

MOTIVATION

All of the members in our project team are currently part of a cappella groups here at MIT and have taken music theory in the past. The style of composition we learned in music theory -- to generate a melody, decide possible chords to harmonize to that melody, then identify the best voicing to proceed with for the harmony -- is user-driven and often difficult and time consuming to accomplish. This style of composition is ill-suited to those who do not have the time or music theory background to try and generate harmonies for their arrangements or new musical ideas. Chordination provides an accessible way for users to play around with musical ideas and hear harmonizations for their tunes without undergoing the painstaking process of analysis. It allows users the freedom to focus on melody and flow and provide inspiration for them to take a Chordination-generated harmony and tweak it to their needs and interests.

BACKGROUND

While there are several existing programs that provide support for music composition, including Sibelius, Finale, and the website Noteflight, none of these major systems support the composer in providing harmonization suggestions, instead leaving those decisions to be entirely driven by the user. Although there are existing programs such as Ludwig (http://www.write-music.com/), none of these programs provide real-time chord generation that harmonize with a user's voice -- instead, they require the user to type up and input a melody which the program can then calculate on. As such, Chorination is the first real-time machine-driven harmonization project of its kind.

PROPOSED APPROACH

We divided the implementation into four major modules: the Filtered FFT Module, Chord Calculation Module, Voicing Module and Synthesis Module. The Filtered FFT Module takes in the microphone input of the user singing and outputs a filtered pitch to the Chord Calculation Module. The Chord Calculation module takes 4 inputs: the filtered user input (from the Filtered FFT Module), the key signature (through switches), the tempo (through switches) and the history of chord progression, from which it will produce the potentially possible chords. The voicing module

takes in the possible chords calculated by the Chord Calculation module and constrains this input into a selection of three pitches to harmonize with the sung user note based on voicing rules from classical music theory. Finally, the Synthesis Module takes in the specific list of pitches provided by the Voicing Module and plays a chord constructed by the user's voice modified in frequency to match this list of pitches. This playback can be recorded and looped for further user analysis.

BLOCK DIAGRAM

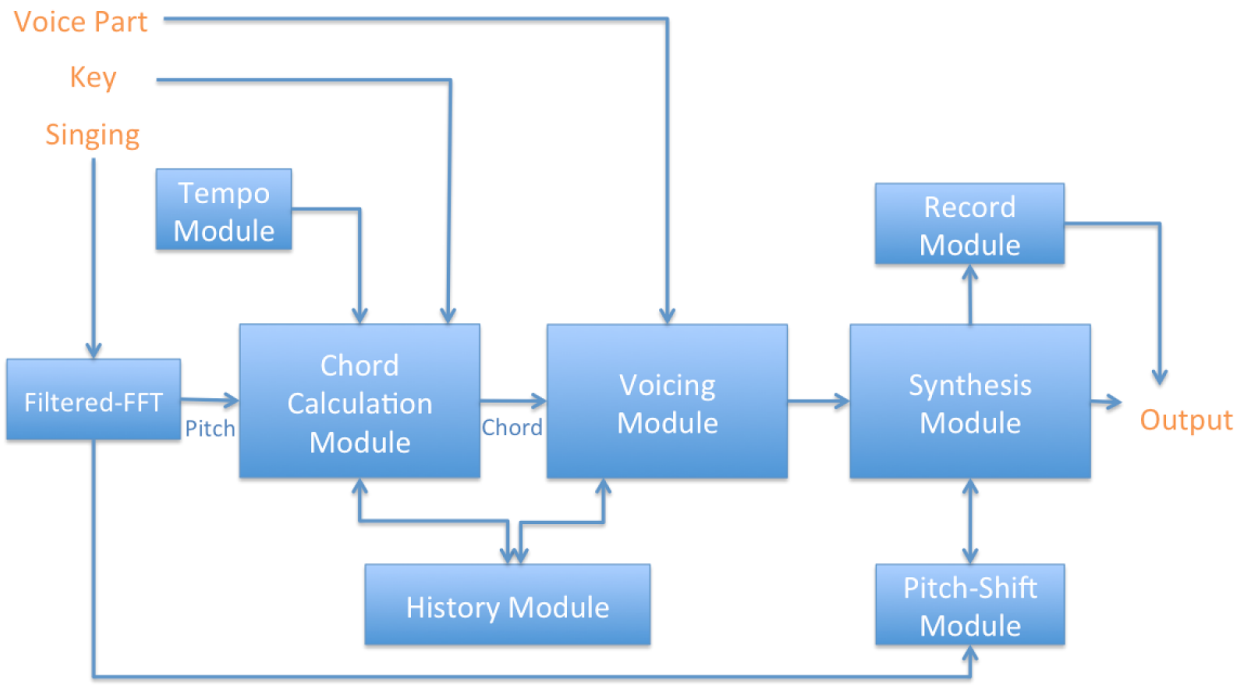

Figure 1: Chordination Module Block Diagram

Filtered-FFT Module

The Filtered-FFT module takes the user's singing through a microphone as an input. The Filtered-FFT module first takes the Fourier Transform of the input signal to get the pitches of the melody. Considering the quality of the microphone and the background noise, the signal is put through a Low-Pass filter to remove noise and overtones and obtain a single signal, which is the pitch the user is singing. The output of the Filtered-FFT Module (which will be referred to as the filtered user input) will be an input to both the Chord Calculation Module and the Synthesis Module.

Chord Calculator Module

The chord calculator module is responsible for supplying the voicing module a list of possible chords that harmonize within a given note -- that is, its role is to determine potential combinations of pitches to play with a sung note.

Generating possible chords requires several inputs: a tempo, which is used to determine how often chords should be recalculated, the filtered user input, which is a pitch (note) to harmonize with, a key a harmonization should be calculated in, and a history of previous chords in order to calculate the current one. The pitch, which is given to the module in Hz, is converted from an inputted frequency to one of seven Western music note names by calculating which note name is closest to the given frequency (there exist several charts such as http://www.phy.mtu.edu/~suits/notefreqs.html which can provide the basis this conversion). Additionally, users are able to input one of twelve major keys -- keys being notes which anchor songs by providing a pitch to center a song around -- which provide a basis for the chord calculator module to calculate and constrain possible chords using knowledge of music theory rules. Each pitch given to the module has, according to Western music theory, four possible chords it could be associated with. Chords are structured as a series of three or four notes: a root note (which forms the basis of a chord), a note which is a musical interval of a third above the root (which corresponds to a frequency ratio of 5:4 for a major third compared to its root), a perfect fifth (which corresponds to a frequency ratio of 3:2), and possibly a seventh of some variety. As such, there are four possible chords for any given note the user can sing.

Furthermore, information on previously-played chords has to be kept as classical music theory dictates that only certain chords can progress to other chords. This history of past chords can be represented as a state machine, as shown in Figure 2, where the Roman Numeral states indicate how far the base of a chord (its “root”) is from the note of the key signature; lines leading from one state to another represent possible transitions that can be made.

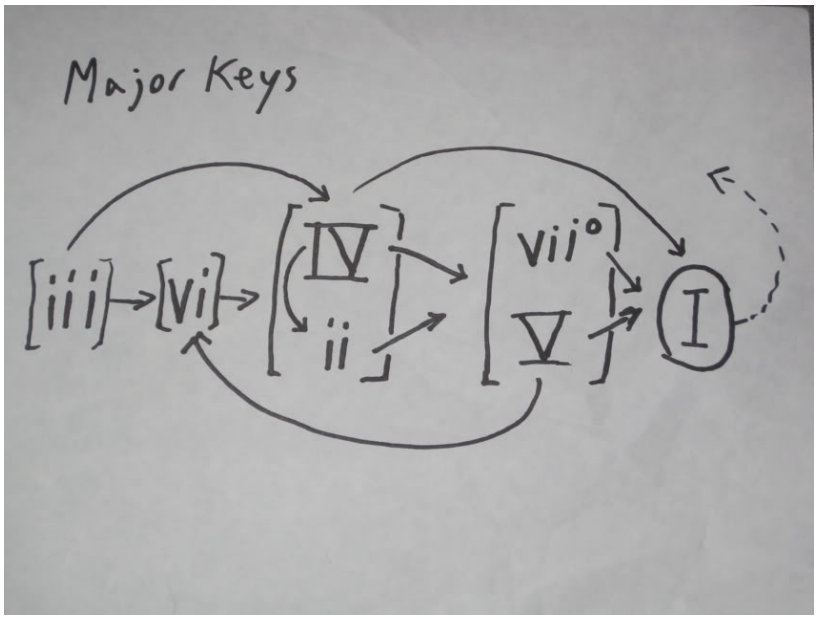

Figure 2: State Transition Diagram for Chords in a Major Key

It is the responsibility of the chord calculator module to take in a current pitch, calculate the possible chords that can be made using this pitch, then see which chords are possible to transition to next using the state transition diagram above. After finding potential next states (potential chords), these chords are outputted to the voicing module to assign each note of the chord to each of the four voice parts based on rules of classical composition.

Voicing Module

The voicing module is responsible for finding out exactly which notes to play (as opposed to the chord calculator, which finds out which collections of notes are possible to play). It takes in chords from the chord calculator in the form of groups of notes, as well as the chord calculator's ranking for which chords would be best to play based on the music theory described in the Chord Calculator Module above. The Voicing Module uses established classical choral music theory in order to apply different rules which constrain the possible notes that could be played with the current note being sung. These rules include rules on jumps (how close or far away are two consecutive notes for the same voice part), motion (whether notes move in the same direction or different directions and to what degree) as well as voice crossing and overlapping (whether a higher voice part sings lower than a lower voice part and vice versa), and the distance in pitch between parts.

Another important constraint to take into account is the user's voice part, which determines how many synthesized voices will sing higher and lower than the main voice. The user enters their voice part via switches on the labkit. For a four-part harmony, the voice parts are usually soprano, alto, tenor and bass, in decreasing order of how high each part sings. This constraint

is necessary since if that the voicing module always assumes that a soprano is singing, if a bass sings into the devices all the harmonized notes will be low and unrecognizable.

The voicing module uses these rules to calculate the most ideal combination of notes to play, which is any combination that does not break any of the voicing constraints provided by classical Music Theory as well as most satisfies the rankings provided by the chord calculator.

Synthesis Module

The synthesis module takes the output from the voicing module and the filtered user input, integrates them to generate the final output to a speaker and records the output if chosen to. The synthesis module contains three sub-modules: the main synthesis module, the pitch-shift sub-module and the record sub-module.

First, the synthesis module will take the filtered user input, and the chord/pitches from the voicing module into the pitch-shift sub-module. The pitch-shifting sub-module will make three copies of the filtered user input and shift the pitch up or down according to the chord generated from the voicing module. The result is a complete 4-voice part chord generated from the original user input. The result is feed to the main synthesis module.

Then, the main synthesis module will modify the volume of the 4-voice part chord so that the user's singing part is louder than the harmonizing parts. The modified chord will then be output to the speaker and the recording sub-module.

The recording sub-module will record the output from the main synthesis module. The user can choose to record by turning on a switch on the FPGA. The memory of the recording sub-module is large enough to record a typical song of duration no more than 5 minutes. If the user tries to record for more than 5 minutes, the memory will only hold the first 5 minutes of the recording. After recording a song, the user can playback the recording by turning on another switch. The playback is played in a loop.

EVALUATION MATRICES

Our three evaluation metrics are error rate, lag time, and correctness of the outputted chords. Error rate represents the number of times that Chordination will be unable to find a chord and appropriate voicing which works for an input pitch. Lag time represents the amounts of time it takes for Chordination to calculate a new chord and transfer from an old chord to a new one. Correctness quantifies whether the final outputted chord is in fact correctly voiced and is a valid chord in and of itself.

TIMELINE

Week 1: Finish Verilog Coding, start to implement bench tests
Week 2: Finish bench tests, start to debugging
Week 3: Finish debugging, fully test the modules and connect the modules
Week 4: Implement additional functions