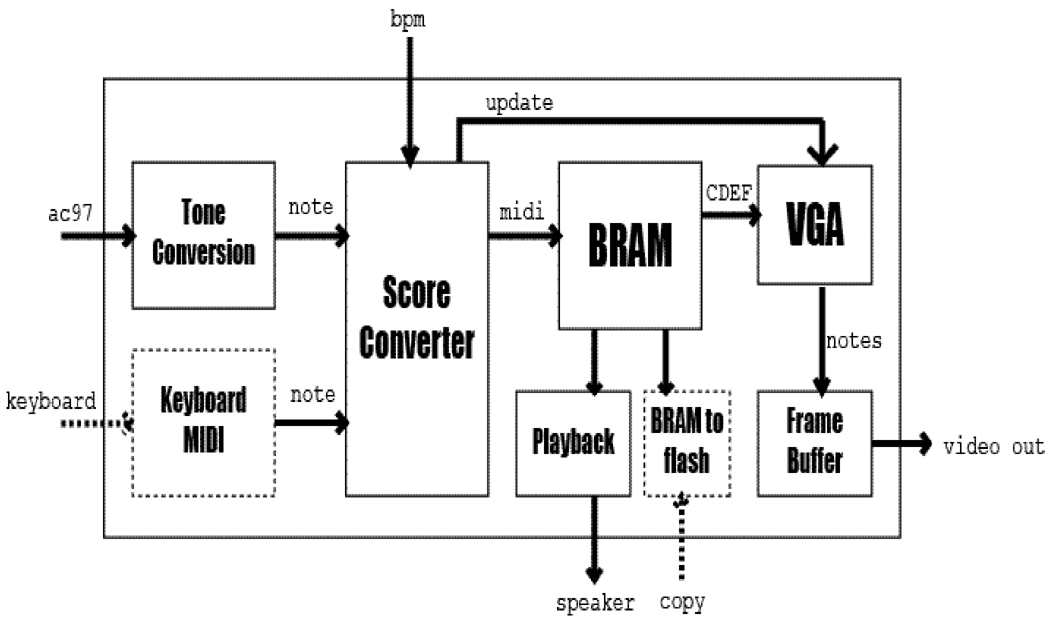
Perfect Pitch Sheet Music Maker

6.111 Final Project Proposal

Adam McCaughan (amcc@mit.edu) and John O'Brien (jweob@mit.edu)



A description of the project in words, stating what your system is
going to do and how you plan to implement it.

* A block diagram.
* A set of specifications that define in detail what your system does (in terms of inputs and outputs) and what tests will be used to prove that it functions properly.
* A statement of how the project work is to be divided among the partners. The block diagram should be referenced.

The project should be partitioned into separately testable subsystems, each subsystem is to be the responsibility of a single partner.

The proposal should be typewritten. Typically it should be two to five pages in length, single-spaced, plus the block diagram and any figures you may need.

The Proposal Conference:

Each project proposal must also be discussed with your TA so that everyone understands what it is you are attempting and whether your basic design approach is sound. Each project group should sign up for a 30 minute meeting -- your TA will be sending you some email to set up a time. Be sure to bring extra copies of your Proposal with you to the presentation so that the mentor can follow your talk without your having to draw your block diagram on the board.

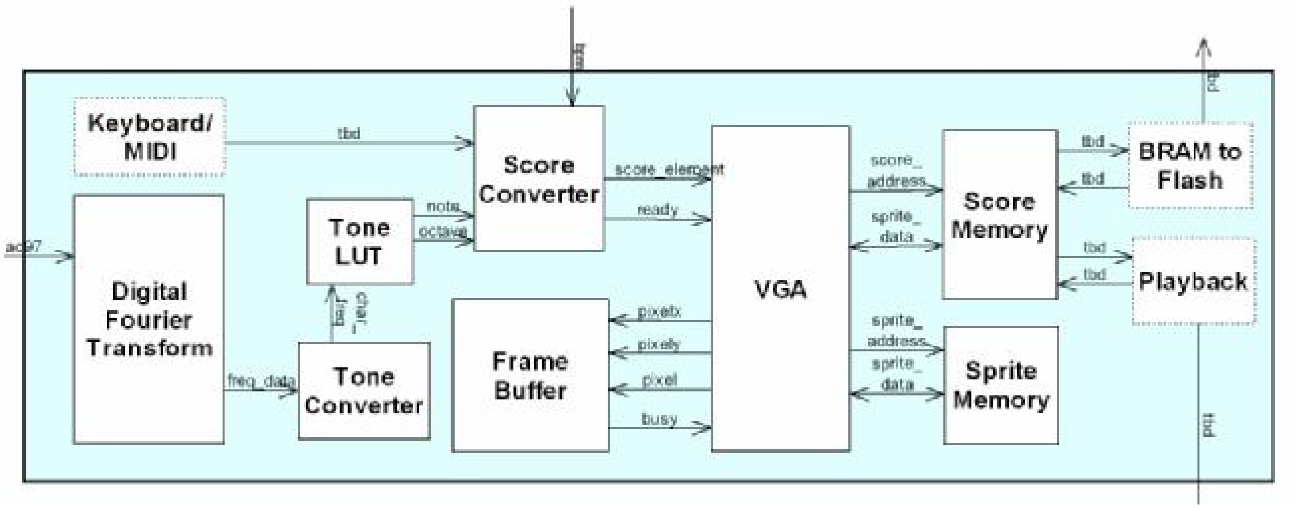
Abstract

This project will allow music to be transcribed automatically. It will consist of a microphone, input switches and a video display. The device will analyze music played into the microphone and update the notes that have been played in real time on the display. The display will show the notes in standard sheet music format. The switches will be used to input settings such as the key or timing of the music.
Initially the device will be made to deal with a sequence of individual notes from an instrument that will give a pure tone (such as a synthesizer).

Other optional features that could then be added include:

- Ability to recognize richer notes – for example wind instruments.
- Ability to recognize chords.
- Synthesized playback of recorded scores
- The option to input notes directly from a keyboard or midi cable
- A graphical mouse driven interface to replace the input switches
- The option to write a midi file containing the score to a flash memory card.

Block Diagram



Specification

System Specification and Evaluation

The system will analyze music played into the microphone and display the equivalent musical score on a monitor. The music must consist of single notes only and the tones must be pure and free from noise. The score for the music must be known beforehand so that it can be compared with the device's output. The beats per minute value must be known. To fulfill these criteria a recording of music synthesized from a software program such as Sibellius will be used to demonstrate the system's functionality. The system will be judged to function correctly it can accurately represent the score of the recording on the monitor.

Digital Fourier Transform

The digital fourier transform receives PCM frames from the ac97 and fourier transforms them in order to output the frame in the frequency domain.

Tone Converter

The tone converter module's primary function is to take a frame of Fourier-transformed data, analyze it, and determine what (if any) tone it corresponds to. In the initial stages of testing, this input data coming from the digital fourier transform will be a pure, single-frequency tone. In this case, the tone converter will simply find the frequency of the maximum amplitude and then proceed to determine whether or not the intensity of the sound is great enough to justify labeling it as a note. If it is in fact loud enough, char_freq will change from its value of zero to the characteristic frequency. Once able to identify the characteristic frequency of a pure tone reliably, the tone converter module will be outfitted with a host of logic to be able to determine the characteristic frequencies of complex instruments (which have many harmonics), such as the piano or saxophone.

Tone LUT

The tone lookup table has a very simple purpose. It receives the characteristic frequency from the tone converter and outputs the corresponding note, A-G#, and the corresponding octave the note is being played in. It does this by rounding imperfect frequencies to the nearest relevant frequency, then inputting that matched frequency into a ROM which matches the frequencies for each note in each octave.

Score Converter

Score converter's function is to denote the time at which a note and octave is first received from the tone LUT, and then keep track of how long the note lasts for. Then, based on the user-tempo input bpm, it determines precisely what length note (in terms of 1/32nds of a beat) occurred. It then outputs the start frame, note, octave, and length to VGA alongside a ready signal.

VGA

The VGA module converts information about new notes into a graphical representation in the style of conventional sheet music.

The module receives note data from score converter via the score_element wire. Score_element tells VGA the start time, tone, octave and duration of a new note. Score converter informs VGA of a new signal on score_element using the ready signal.

When a new note is inputted VGA looks up a note of the required duration and type in the sprite memory using the sprite_data and sprite_address signals. It then arranges the sprite on a "slice" of musical staff 1/32nd of a beat wide. Where on the staff the note is placed depends on its tone.

To decide which sprite to select for a particular note VGA must look at its tone (high notes have tails pointing down, low notes have tails pointing up). Since collections of short notes of the same duration are connected with bars (to make reading them easier) VGA must also look at the values of the notes directly before and after the current one. Since the VGA unit cannot guess the value of the next note to be played it may have to update the last note displayed as well as the current one.

The pixel coordinates for the slice are arranged to represent the start time of the note. Once the pixels and their coordinates have been computed frame buffer is updated to represent the new note. The output is buffered and is only asserted when the busy signal is low. This is to prevent VGA from contending for the bus while frame buffer is reading from its ZBT SRAM. Once a note has been displayed its information is saved in the score memory with the score_address and score_data signals.

Sprite Memory

The sprite memory is a ROM containing pixel values for all the types of note required by VGA to display sheet music. Each sprite consists of a 2D array of pixel values. The pixels can take one of three values – white, black or transparent. For the sprites to be correctly arranged on the musical staff by VGA they must have a common reference point (i.e. each type of note stored in the sprite memory must have their centre at the same pixel coordinate).

Score Memory

Score memory keeps track of all the notes that have been played so far. Although it is not envisioned that this information will need to be accessed in normal operation (VGA should only need to know about the last two notes) it may be needed for debugging the display module and in special cases where the whole screen needs to be refreshed (for example to take in to account a change in time signature). It will also greatly simplify the addition of optional modules such as playback and write to flash.

Frame Buffer

The frame buffer stores the current display image in a ZBT SRAM. This avoids the need to recalculate all the pixel values every new frame – only pixels whose values change need to be changed. The frame buffer must read information from the ZBT at a fast enough rate to supply the 65Mhz pixel signal to the display. Between reads it must accept pixel update information from VGA on pixel, pixelx and pixely.

Resources

Audio

The ac97 chip included on the labkit has a sampling frequency of 48kHz with an 18bit resolution. The highest note that can be played on a piano keyboard (the range of which we expect to be able to represent) is 4.4kHz, giving a Nyquist frequency of 8.8kHz. Therefore we should not experience aliasing problems. The sampling frequency and resolution of the ac97 actually exceeds that encoded in CD audio (44kHz with 16bit resolution).
A high quality microphone will be required to ensure that minimal distortion and noise enters the signal.

Memory

The labkit provides 4MB of ZBT SRAM and 2.6Mbit of BRAM (144x18kbit). The main memory requirement of the project will be the frame buffer. To provide a resolution of 1024x768 pixels with 3 bits per pixel it will need a capacity of 2.4Mbit. This can easily be accommodated in a ZBT SRAM. If the refresh rate is 60Hz then the ZBT will need to provide a new 3 bit pixel value at the rising edge of the 65 Mhz

pixel clock. The ZBT can output 36 bits every clock cycle, and can be clocked at up to 167 Mhz so it should be able to meet these demands.
The other memory demands of the project have been calculated to not exceed 200kbit (estimate 50 kbit for sprite memory, 64kbit for note memory and less than 5kbit for tone lookup table). This will not place a strain on the BRAM available.

Organization

Adam McCaughan will be responsible for the digital fourier transform, tone converter, score converter and tone lookup modules. John O'Brien will be responsible for the VGA, score memory, sprite memory and frame buffer modules. This splits the project into an audio analysis subsystem and a video display subsystem, both of which can be tested independently. It is important that the timing specifications and format of the data passed to the video system on score_element and ready are carefully defined to ensure smooth integration of the two sub_systems.