Conductor Hero

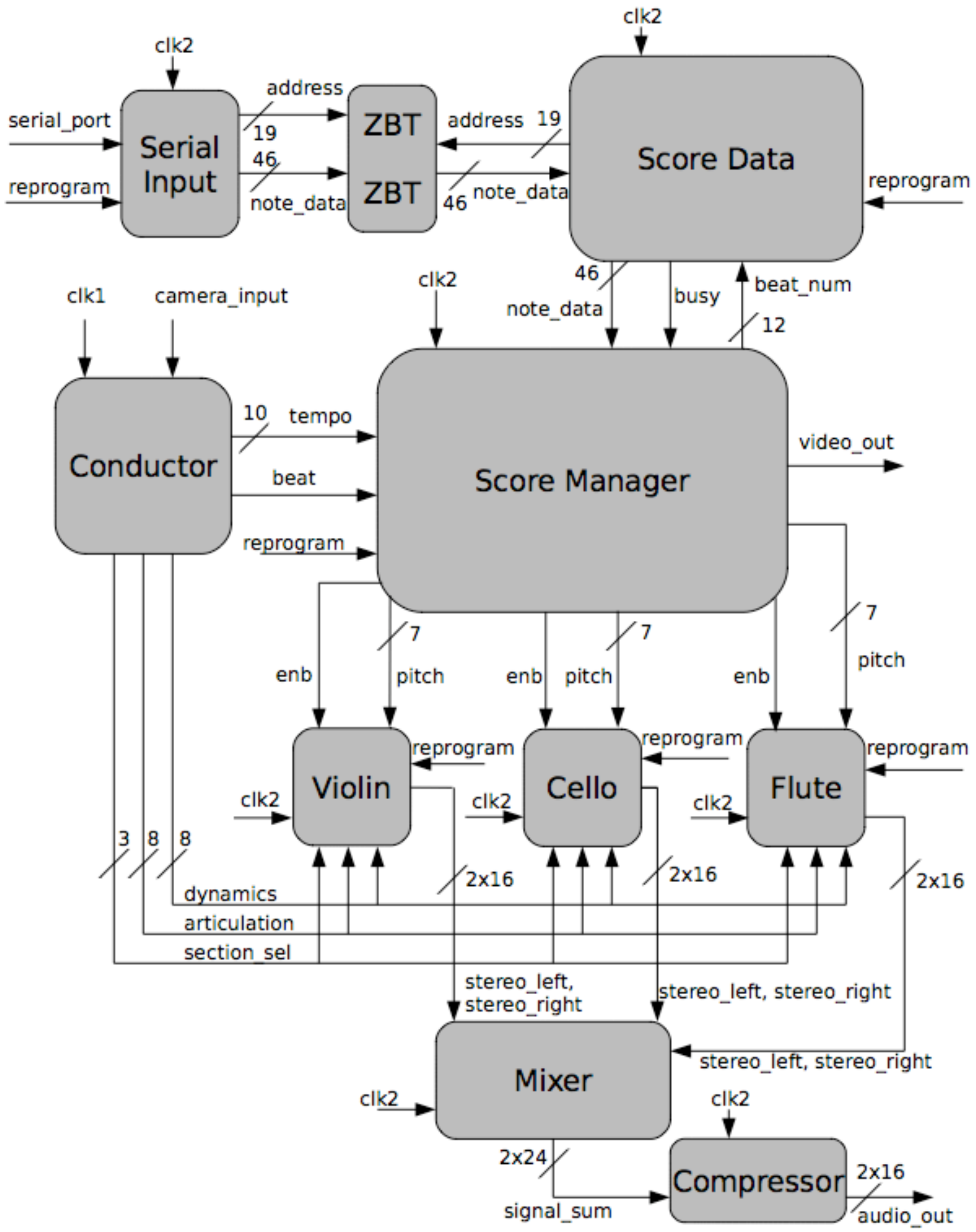
Natalie Cheung, Yuta Kuboyama, Edgar Twigg

Introduction

In our project we will simulate conducting an orchestra. The user will hold a baton with a colored ball at the far end. A camera will track the motion of the colored ball. As the user conducts, circuitry will extract dynamics, tempo, articulation, and section selection data from the ball's motion. The system will then synthesize the sound of an orchestra playing a pre-determined score at the user's commanded tempo. Whatever stylistic commands the user makes (dynamics and articulation) will only apply to the section he is currently commanding (as determined by the section selection data), and when he turns to face a different section, the previous section will continue performing with the dynamics and articulation it was last commanded with. Although our primary objective is to implement the audio functionality, if there is time then we will also implement some sort of graphical display for an even more immersive experience.

Our project is composed of three parts: the conductor interface, score management, and audio synthesis. The conductor interface takes movements from the conductor's hands and sends the score management system the tempo, dynamics, and articulation of the piece. Furthermore, the conductor interface will be designed to cue which section of the orchestra needs to play louder or softer. The score management system takes in MIDI data and instructs each section of the orchestra to play a note when the conductor motions to play. Additionally, the score management system produces an orchestra visual on the computer screen. The audio synthesis unit takes in the dynamics, articulations, and pitch from the other modules and outputs the appropriate audio signals.

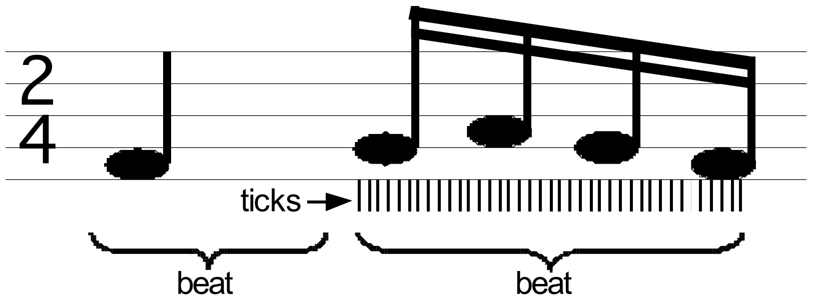
System Block Diagram



Time Representation

In order to understand how the score is represented and played back, one must first understand how we model time. The song's time is measured in beats. Each beat represents one beat as conducted by the conductor. So for music in 4/4 time, one beat is represented by a quarter note. In 6/8 time, one beat is represented by a dotted quarter note.

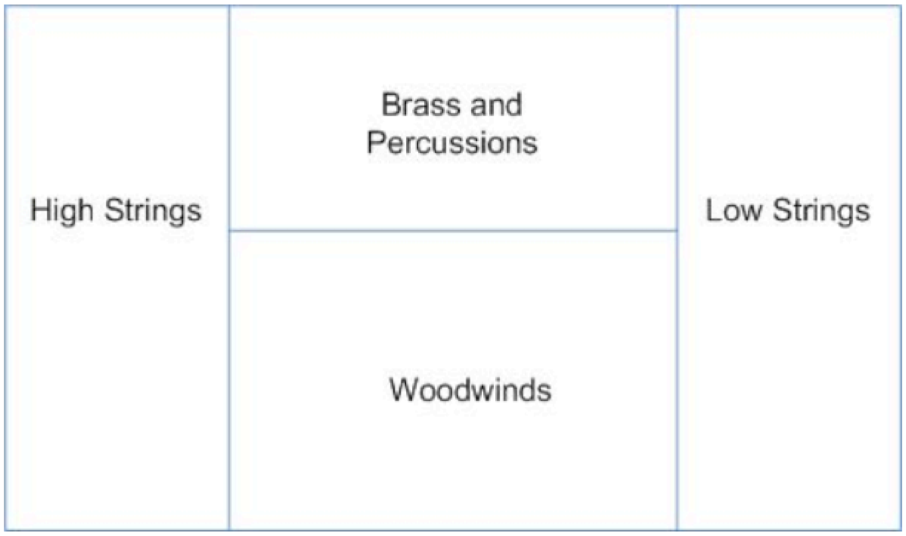
Each beat is divided into 128 ticks. This high resolution allows triplets to be represented fairly accurately, even though they can not be perfectly represented by any base two number.



Conductor Interface

Conductor Module

We plan to borrow a video camera from the equipment room for the conductor module. This module detects the motion of the baton and sends the tempo and beat to the score manager. It also outputs the dynamics and articulation for a section of the orchestra and sends it to the audio synthesizer. Within the module, we will use a video decoder to retrieve the data, convert it from analog to digital, and save the data on the ZBT. Furthermore, we will analyze the data by finding the baton velocity and marking each change in velocity as a beat in the music. The module will output a square wave where the switch from high to low or low to high indicates another beat. We will output the tempo as a ten bit number – the greater the number, the faster the piece. Additionally, the dynamics and articulation outputs are each eight bit binary numbers – the greater the output, the louder and the more legato the piece becomes, respectively. Lastly, we have added a three bit output that determines which section the conductor is directing his motions to. In order to direct different sections, we decided that the user will give specific hand movements to conduct the orchestra. To point to a specific section of the orchestra, the user must conduct in a certain location. We decided to break down the video detector area to different sections, as shown in the figure below. The module only registers the user's moves, when the user conducts in the particular section for at least three seconds. If the user moves to different locations within three seconds, the module will revert to the previous section that the user conducted in. Furthermore, if the conductor wishes to only conduct a section of the orchestra, say only the woodwinds, the conductor will use his right hand to instruct the sections. In order to increase the volume of the whole orchestra, the conductor will use both hands, meaning both sensors, to conduct the dynamics and tempo of the orchestra.



Testing Procedures and Milestones

To test the module, we decided to first test if the hand motions were tracked by the video camera correctly. After establishing that it works, we want to test that the module is counting the beats. To test this, we will use a led light – it will flash on if the output is high, off if the output is low. In order to test the tempo, dynamics, articulation and section selection outputs, we will be using the hex display on the labkit to indicate each output when we perform different sets of hand motions.

Score Manager

Note Data Format

Score data is stored in ZBT memory in the Note Data format. The first 12 bits specify what beat the note is in. The second 7 bits specify what tick the note starts on. The third 13 bits specify how many ticks the note lasts for. The fourth 7 bits specify which instrument should play the note. And the fifth 7 bits specify the pitch the note should play. These bit assignments were carefully tuned. At a very slow 50 bpm, 128 ticks per beat allows for a time resolution of 9 ms, which is safely less a person's ability to distinguish time interval differences (approximately 30 ms). We can address 128 instruments and command them at 128 different pitches (significantly more than a piano's 88 keys).

MIDI to Orchestra Hero Score Converter

A computer program will be written that converts MIDI data into a bit stream in the Note Data format, described above. This program will then download the bit stream to the labkit's ZBT memory via an RS-232 port. Each Note Data takes up 46 addresses in memory, but since the memory is addressed in 72 bit chunks (if you connect two 36 bit ZBTs in parallel) the last 26 bits are wasted. A song 700 measures long with 4 beats per measure and an average of 20 notes per beat (a fairly complex and long symphony) would take up 3.85 Mb of RAM, which will just fit on the labkit's 4.5Mb of ZBT SRAM.

Serial Input

This module is responsible for communicating with the RS-232 port and loading the ZBT memories with the supplied bit stream. It is controlled by the program input. When program goes high and then back down again (on the falling edge of program), the module raises its busy output to high and starts downloading a bit stream from the RS-232 jack to the ZBT ram using the labkit's MAX3222 chip. Two ZBT's are addressed in parallel, so that two 512kx36 SRAMs become one 512x72 SRAM. When the bit stream has been completely loaded, busy returns to zero.

Score Data

This module abstracts away the complexities of dealing with the ZBT ram for the Score Manager. The Score Data module takes a 12 bit number representing a beat number as an input from the Score Manager. If this beat number is higher than the beat number of the current memory address, then its busy signal goes high. Each clock cycle, it will put the current Note Data on its pins and increment the memory address. It does this until it has read up to the beat the Score Manager is asking for.

When it's reset goes high, it clears its memory address counter to zero.

Score Manager

The Score Manager is responsible for taking beat and time data from the conductor, as well as score data from the memory, and turning it into pitch and enable signals for each of the instruments. It does this in the following way:

The Score Manager has an internal BRAM 46 bits wide by 256 addresses. Each address contains one full Note Data (thus limiting the maximum number of notes per beat to 256).

Each time the beat input changes, the Score Manager rewrites the BRAM from the top with the next beat's data.

The Score Manager has another internal BRAM that is 13 bits wide (the note duration bit width) by 128 (the number of instruments). This BRAM keeps track of how long every note an instrument is currently playing has left to play.

There is also a set of output registers for every instrument: an enable, and an X-bit wide pitch.

The beat input from the conductor module switches its state every time the beat changes. The score module uses a level to pulse converter to detect both of these edges (rising and falling). When the beat changes, the Score Manager increments its beat output and resets its tick counter. As long as Score Data's busy signal is high, the Score Manager takes one Note Data element from the Score Data module per cycle.

By looking at the tempo input (which arrives as bpm x 4), the Score Manager determines when enough clock cycles have passed that one tick should have passed. When this happens, the tick counter is incremented.

Each time the tick counter increments, the Score Manager checks to see if the current tick equals the start tick of the current tick. If it is equal, then the Score Manager sets the enable and pitch for the appropriate output register to the appropriate value. The BRAM that keeps track of playing note durations is also set to the appropriate value.

Once all notes that need to start on that tick have been initialized, each counter for the play duration BRAM is decremented. If any of these counters decrement to zero, the enable for the corresponding instrument is turned off.

Testing Procedures and Milestones

First, we will test the RS-232 interface. The goal will be to take a hand generated Note Data binary file on a computer and read it on the labkit. We will debug the system by making sure that the appropriate numbers for beat number, start tick, duration, etc. show up on the alphanumeric display.

Second, we will test the ZBT SRAMs and Score Data module. We will debug the system by clocking the Score Data module and ZBT rams on a manual switch, so that we can see the Note Data items go by on the alphanumeric display each clock cycle.

Next, we will debug the Score Manager module. We will do this by setting a fixed tempo input and flipping a manual switch to simulate the beat input. The enables will be connected to LEDs and the pitches will be connected to the alphanumeric display.

Finally, we will write and debug software to automate the creation of Note Data bit streams from MIDI data.

Audio Synthesis

Instrument Modules

These modules synthesize audio signals by combining various types of signals and simulating the waveform produced by the actual instrument it represents. Each instrument module can produce only one note, and each output signal is characterized by its pitch, dynamics, and articulation. Higher pitch value will correspond to a higher pitch, and higher dynamics value will correspond to a louder output. A higher articulation value will correspond to a more legato audio output signal, and a lower the articulation value will correspond to a more staccato audio output signal.

Each instrument module has its own section selector value, which determines which section the instrument is in (strings, brass, bass, etc). The module takes in a dynamics value (dynamics), an articulation value (articulation), and a section selector value (section_sel) from the conductor module, and every time the value of section_sel matches an instrument module's section selector value, the dynamics and articulation values of the instrument module updates to the values given by the conductor module. The dynamics and articulation value will reset to their default values when the reprogram signal (reprogram) is asserted.

The instrument module will also take in an enable signal (enb) and a pitch value (pitch) from the score manager module, and will output the audio signal at the given pitch for the duration of the enable signal.

The output audio signal consists of two weighted 16-bit signals to create a stereo signal (stereo_left, stereo_right). The weighting of the left and right channels of this stereo signal will be determined by the location of the instrument in the orchestra.

Mixer Module

This module takes in the left and right audio signals (stereo_left, stereo_right) produced by the instrument modules, and mix the left and right channel signals separately by summing them together. Each of the inputs is a 16-bit signal, and in order to prevent clipping in the mixing process, the mixed output will be stored as a 24-bit signal. The mixer module outputs the two 24-bit signals for left and right channels of the stereo signal.

Compressor Module

This module will take in the two 24-bit left and right channel signals from the mixer module, and applies a gain on those signals. The value of the gain is set to maintain a reasonable volume level and changes slowly in time. In this way dynamic changes can be heard while keeping sustained dynamics easily audible. The outputs of the compressor module are two 16-bit left and right channel signals, and they will be sent to the speakers via an AC97 DAC.

Testing Procedures and Milestones

The first step in developing the audio synthesis block would be developing the instrument modules. In order to test the instrument modules, the pitch, dynamics, articulation, section selector, and enable bits will be set to various values, and the output stereo signal will be directly connected to a speaker. The input signals will be altered to test if the speaker output has the desired behavior. The quality of the speaker output (how similar it is to the real instrument) would be evaluated as well.

Once the instrument modules are complete, the next step is to be able to mix the instrument module outputs. To test the mixer module, the output audio signals from the instrument modules will be sent to the mixer module. The output of the mixer module will be connected directly to a speaker. The input instrument audio signals will be turned on and off to see if the mixer module outputs the summed signals correctly.

Finally, the compressor module will be tested. The signals from the mixer module will be set as the inputs to the compressor module, and the output of the compressor module will be connected to a speaker. The dynamics value on the instruments will be raised to check that there is no clipping in the speakers. Then, the dynamics value will be dropped to see if the speaker output does not become too quiet.