Warren Cheng
Rob Hendry
Ashwin Ramachandran

Embedded Systems Spring 2012 Proposal
2/21/12

Midi Synthesizer

I. Introduction

We propose to build a MIDI Synthesizer, which will take input stimuli from a MIDI controller (such as a keyboard) and produce synthesized tones mimicking various instruments. Our system will also have the capability to further manipulate the synthesized notes by sending the sounds through a digital signal processor (DSP). This project essentially requires the completion of three key objectives: understanding and decoding of the MIDI protocol, designing hardware to synthesize the notes, and creating a sufficient DSP to manipulate the sound signals. We are planning to implement a synthesizer using the frequency modulation, and additive synthesis methods, however, we might just hone in on one particular method as our project continues.

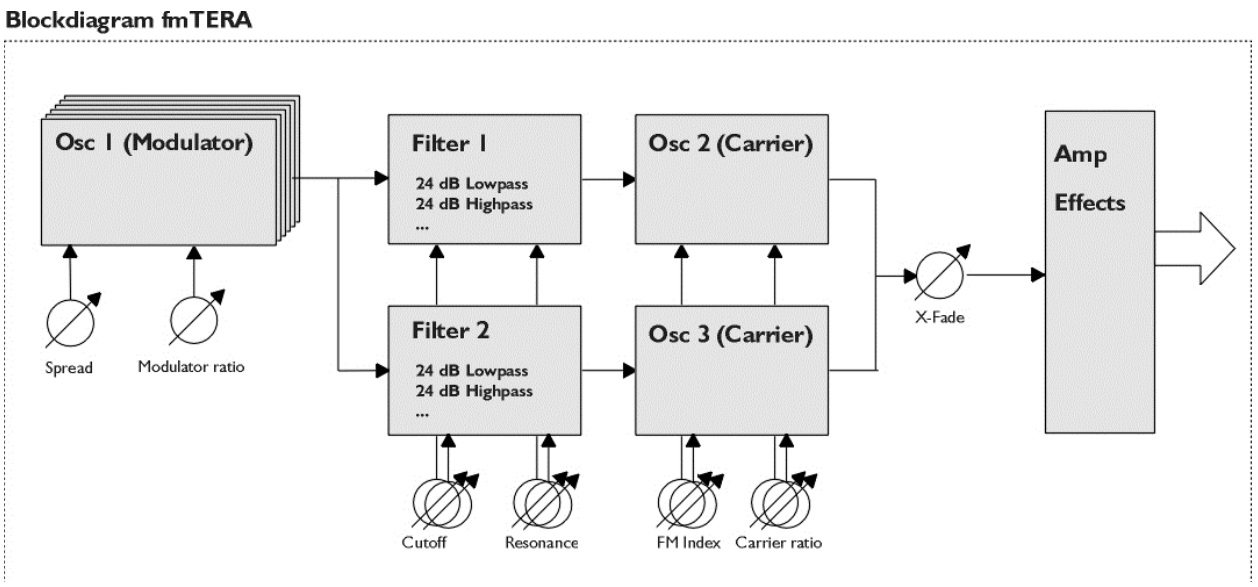
II. MIDI Basics

In order to implement the design, we first need to conduct research on the MIDI protocol according to the MIDI 1.0 specification. In general, MIDI signals can be transmitted via a MIDI port as well as USB and Ethernet connection standards. Due to the physical layout of the DE2 board, implementing a MIDI to RS-232 or USB connection would be the most feasible.

The basic MIDI protocol says that a MIDI message begins with a status signal. This 8-bit signal indicates whether the controller wants to turn a note on, off, change control, change the program (change the instrumental sound), and so on. The upper nibble of a status signal indicates the operation, and the lower nibble indicates the channel to perform the operation on. MIDI supports up to 16 channels. In order to play a note, the note-on status signal (with its corresponding channel) is transmitted, followed by two more bytes. The first byte is the note itself (where the upper nibble is a number from 0-127, and the lower nibble is the channel) and the second is the velocity (or how long the note is held out). Each number between 0 and 127 is mapped to one note according to the MIDI protocol. The rest of the different statuses operate along a similar principle. One thing to note is that MIDI feature signals not implemented must be ignored by the MIDI synthesizer. We will continue to further study this protocol for implementation on our synthesizer.

III. Synthesis by Frequency Modulation

An FM synthesizer consists of two oscillators which generate signals (in our case, digital representations of signals) called the *carrier frequency* and *modulation frequency*. Basically, the carrier frequency is a simple sinusoid that produces a pure tone at the desired pitch, while the modulation frequency adds variation to the sinusoid to produce different timbres. The amount of modulation frequency that is applied to the carrier frequency can be varied; the ratio of this relationship is called the *index of modulation*. Building an FM synthesizer will involve designing a system with one or more modulation oscillators, one or more carrier oscillators, and logic to combine the digital signals into a single output waveform. The image below shows an example FM synth schematic with programmable modulator filters to increase the variety of timbres of which the synthesizer is capable.



http://www.virsyn.de/en/E_Products/E_TERA/e_tera.html

IV. Digital Signal Processing

While DSP in itself is a very broad field, it pertains to any processing or manipulation of discrete time signals. In our case, synthesized signals must be sampled using an ADC converter and stored into memory for further processing by the Altera DE2 board. Through the use of digital filters, we can add audio effects to the original sound such as flanging, chorusing, reverberation, and multitap delays to just name a few. These filters directly impact the harmonics, amplitude, frequency, or phase of the sampled signal in order to create distinct new ones. Of course, these effects can be replicated in the analog realm with real filter components, but that isn't the purpose of this project. Given that all filters have a certain transfer function associated with them, extensive simulations of these digital filters can be produced using Matlab.

Rough Project Timeline

By Milestone 1

- To gain a thorough understanding of the MIDI protocol
- Begin learning about the implementation of synthesizers

- Learn more about DSP and DSP filters and their effects on sounds
- Begin simulation of DSP filters in Matlab

By Milestone 2

- Backend implementation of MIDI
- Begin synthesizing simple instruments/sounds
- Implementation of DSP filters

By Milestone 3

- Develop specified instrument implementations
- Finish interface between keyboard input and synthesizer output
- Get the DAC audio converter working to allow us to play the output on speakers