Expanded FPGA Synthesizer: Progress Report

Mel Murphy

David Hanson

Jonathan Law

Portland State University
ECE 544 Final Project
Spring 2021

Project Description

Our goal for this project is to build a synthesizer on the Artix-7 FPGA. The synthesizer will accept MIDI signals on an interrupt basis and write them to a sequence. It will then play notes out of the PMOD I2S jack corresponding to the MIDI signals it stored. It will also have MIDI output functionality such that it can send MIDI signals to external instruments. The buttons, switches, and rotary encoder will be used to set parameters of the sequence, including tempo, and these parameters will be displayed on the seven-segment display and the OLED RGB display. The software will be coded in C, and it will run on a Nexys A7-100T FPGA board. We'll use Vivado to build the hardware, and Vitis to build the software.

Progress to-date

So far, we've accomplished the following:

Audio output

We've completed a build of an I2S module that accepts PCM samples over AXI and writes them to the PMOD I2S2 audio output device so that they can be heard using a device connected to its ⅛" barrel jack. The I2S module sends an interrupt when it is ready to receive a 24-bit sample, and the system sends the sample through a synchronizer so that it can enter the audio clock domain from the system clock domain unaltered. The samples are calculated and stored to a buffer of 8,192 samples, then sent on an interrupt basis at a rate of 48 KHz.

Signal Generator

The synthesizer is able to generate pulse/square waves, sawtooth waves, and triangle/ramp waves. It can generate multiple signals simultaneously and mix them to create a combined signal.

OLED Display Menu

The OLEDrgb display shows a menu to the user that can be used to select values for a series of parameters including system-level parameters (sequence, mode, load/store), sequence-level parameters (tempo, subdivision, swing, volume, pattern), and note-level parameters (velocity, frequency, duty cycle/waveshape, on/off). The display is navigated using the PMOD rotary encoder and the pushbuttons, which are both read by the system on an interrupt basis.

S**equencer**

We ported over the code for the sequencer from Mel's ECE 540 project. The sequencer is able to store a series of note values and read them back in sequence. It can be set to play notes forwards, backwards, forth-and-back, or in a random order. The notes are sequenced using an interrupt-enabled AXI timer to set the note intervals and maintain a consistent tempo. The sequencer also lights the LED corresponding to the active step in the 16-note sequence. 16 sequences can be stored in the bank.

MIDI Input

We packaged a pre-existing open source MIDI receiver and added an interrupt controller so that it can be accessed by AXI. With this functionality, the synthesizer is able to read MIDI input on an interrupt basis. The incoming note values are translated into a corresponding frequency and sent to the signal generator, which generates a tone. It's able to accept NOTE ON and NOTE OFF messages, as well as modulation messages. The modulation messages can be used to alter the signal generator's waveshape parameter.

MIDI Output

The synthesizer is able to generate MIDI messages for NOTE ON and NOTE OFF. These can be sent to external instruments. We tested the functionality and were able to play a sequence of notes on a Moog Sub Phatty.

FreeRTOS Implementation

To maximize performance, we ported the system over to FreeRTOS so that we can schedule multiple tasks to handle different areas of functionality. For example, we can process audio samples, update the display, and update the sequencer in different tasks while responding to MIDI signals, button or rotary encoder changes, sequence timer countdowns, or I2S sample send messages on an interrupt basis.

Progress to-be-done

We still are working on the following and intend to get them done in time for project delivery:

VGA Display - David

To further demonstrate what the system is doing, help with changing variables, and because visuals always add to a project demo, development is in progress to display some data to an external monitor through VGA. This was done in ECE 540 and especially in David's 540 final project. Much of the implementation will be the same as in that previous class but some work still needs to be done to properly hook it up without adversely affecting the rest of the system.

Memory - David

One thing currently limiting the system is the lack of space to store larger amounts of data. The solution currently being pursued is through the existing Vivado IP catalog. Two additional blocks are getting added to the block diagram. These are the AXI BRAM Controller and Block Memory Generator. The hope is that going about the memory in this way will keep things simple and interfacing with the memory will be similar to the other AXI peripherals.

Menu - Mel and Jonathan

We still need to connect the menu to the parameters that it changes. To do so, we're creating an API so that the sequence parameters can be accessed through functions.

Seven-Segment Display - Mel

The seven-segment display is currently blank. We're going to write out the system tempo to the right-hand side, and add another useful parameter to the left-hand side--possibly the currently playing note value. The decimal points are also currently unused, and we may use these to show note subdivisions or tempo swing.

Stretch Goals

Hardware floating point calculations - Jonathan

Port over the current software sample calculations over into the hardware to reduce the load on the processor. Software will need to send user menu parameters to hardware to change the sample settings.

Audio Input - Mel

We're looking at expanding the I2S module to also allow for audio input. This should be fairly straightforward, as the clocks and other functionality already exists for output. The DRAM memory would be especially useful here for storing audio samples, and we may be able to write input samples to the VGA display as well as output samples.

VGA expansions - David

The initial goal of including the VGA display is to be able to show the current waveform being outputted by the board. Time permitting, we may be able to add more data to the display. One current idea is to list all the menu items on one side of the screen. Another is to be able to choose which waveform is being displayed (not just what is playing).

Tempo and Frequency-based RGB Lighting - ?

The onboard RGB lights that we used in project 1 could add an interesting visual aspect to the instrument. Using the PWM outputs of the Nexys4IO module, we could change the colors and intensity of the lighting based on the notes played by the sequencer.

Polyphony - Mel

We've made a couple attempts to play multiple tones simultaneously but have been fairly unsuccessful so far--multiple oscillators aren't difficult, but managing the MIDI inputs the allow for chords has proven to be. We'll need to alter the hardware to add granularity to the interrupts, and likely expand the active note register such that the hardware stores which notes are active and which are currently inactive. This could be done using a 128-bit series of 4 registers which stores a 1 for an active note and a 0 for an inactive note. We'd then do 4 successive register reads to see which notes across the MIDI spectrum are currently active.

Originally Proposed Milestones

<u>Week 8 (5/16 - 5/22):</u>

- Submit project proposal ✅
- Package the MIDI input hardware as an AXI module for use in Vivado ✅
- Design basic processing and timing functionality so that the sequencer codebase can be adapted for use with a Microblaze ✅

<u>Week 9 (5/23 - 5/29):</u>

- Design sound generator ✅
- Add I2S output functionality ✅
- Add MIDI output functionality ✅
- Add button ✅, switch ⏩, LED ✅, OLED screen ✅, and 7SegD functionality ⏩

<u>Week 10 (5/30 - 6/5):</u>

- Pursue stretch goals ⏩
- Fine-tune sequencer and sound generator ⏩

<u>Finals Week (6/6 - 6/9):</u>

- Prepare demo ⏩
- Prepare project report ⏩