CMPE 125L Lab 5 - Spring 2019
The 1980's All Over Again

Prof. Heiner Litz

Check-Off: May 4th and 5th, 2019 (in lab)
Canvas: May 3rd (Monday), 2019 11:59pm
125 Points (25 Report, 100 Functionality)
Extra Credit (25 pts)

1 Objective

In this lab, you will design and build a simple audio synthesizer. You will generate sounds that will be played through the AC97. The synthesizer will have 4 "voices" that can produce different sounds. The switches on the board select which voices are simultaneously active. You should be able to adjust the frequency of the voices using the left/right buttons and the volume using the up/down buttons. The voices are mixed together to produce a combined sound that is played on left and right channels of the headphone jack. If multiple voices are active the frequency modulation should only apply to the last enabled voice. The volume adjustment should apply to all active voices.

2 Prerequisites

- Read through this **entire** lab assignment.
- Follow the in-class notes to use the DCM_SP, Block RAM, and IOBs.
- Read the AC97 manual and find out how to initialize and interface with it.
- Post questions in the forum if you are uncertain about the specification!

3 TA Review

Your lab TA will cover the following items in the first portion of the first lab:

- What's required
- Some high-level discussion of the AC97 interface (pointers to documentation too!)
- Encouragement :)

4 Requirements

You may submit (via Canvas) multiple Verilog files, one or more Verilog test bench files, your UCF file, and the associated lab write-up. Your design should be synchronous and well-organized. Your Verilog should include appropriate comments and meaningful names of modules and variables.

4.1 Overview

Your design should play synthesized voices through the headphone jack on the AC97. This assignment involves three main tasks:

- interfacing with the AC97
- implement the different voices
- developing a simulation testing strategy

Work in phases! Get the headphone output to work first with one voice. Then add frequeny modulation and volume control, then frequency superposition and finally work on the 4th voice stored in a block RAM.

Some more specific details:

- Make sure your design works in Verilog both BEFORE and AFTER implementation to debug. This means that you need to perform post-implementation simulation and finally test it in hardware.
- You must output 18-bit audio in both left and right channels from the AC97. Check the volume before putting any headphones in your ear!
- Your design must not have any generated clocks that do not come from a DCM_SP and a BUFG. However, you may generate any clock of your desired frequency if you use a DCM_SP and a BUFG in the correct way.
- If you do use multiple clocks from different BUFGs, you must appropriately address synchronization between the two clocks. These must be synchronized appropriately to properly interface with the AC97.
- You should use a fixed sampling rate of 48kHz and 18-bit samples for left and right channels.
- The global reset should reset all of your state synchronously.

4.2 Inputs/Outputs

As in previous labs, you must correctly assign the inputs and outputs to the appropriate switches, buttons, and LEDs. This should be done using the UCF file.

- The reset button should reset your design.
- The up/down buttons should adjust the volume of the generated sound.

- The four switches (SW3-SW0) enable/disable four voices.
- The left/right buttons should adjust the frequency of the last enabled voice (using SW3-SW0).

The four voices that you must implement are:

- square wave (SW0)
- saw tooth wave (SW1)
- sine wave (SW2)
- customizable wave of your choice (SW3)

The first three waveforms are typical in "old school" 80's synthesizers. You can find more on Wikipedia at this URL (for example): `http://en.wikipedia.org/wiki/Square_wave`.

The final waveform is often used in "wavetable" synthesizers. You should include a single period of a sound and scale to the correct frequency. The customized wave should be stored in block block memory. You need to consider how many points should be stored and how you interpolate between the points in order to approximate a real wave at the different note frequencies. It could include (intentional) distortion, additional harmonics, chords, etc. You can implement this any way you want. This is your chance to be creative. Static is not a valid customizable wave! :)

The voices are activated using the switches. **If more than one switch is active, you must combine the audio from the multiple voices!** If none are active, it will be silent. Combining audio can be done with a simple weighted average of the audio signals. Choose a reasonable set of frequencies for your waveforms. For instance, with 20 left/right button presses, cover the entire frequency spectrum of the human ear.

4.3 AC'97

Read the AC'97 specification `http://www-inst.eecs.berkeley.edu/~cs150/Documents/ac97_r23.pdf`. In particular, Section 4 of the AC-Link interface that is used to communicate between the FPGA and the Codec. You should have an understanding of PCM, the concept of sampling rate and amplitude and how to represent sound via PCM. Understand the concept of time multiplexing and slots. Determine the slots that you need to transfer stereo sound as well as how to drive slot 0,1,2. Checkout the timing diagram (Figure 11. AC-Link Output Frame) to determine how to transfer data to the codec and develop an FSM that generates the sequence of Figure 11. Understand clocking covered in Section 4.2 of the specification. Note that the BIT_CLK of 12.88 MHz is used to send data over the external interface to the AC'97 codec, while your internal design should run at SYSCLK. You should phase match the AC'97 clock with the SYSCLOCK and correctly synchronize data between your clock domains. Write a ac97_controller Verilog module with the interface shown below which takes in slots as parallel data and serializes them on the time-multiplexed AC'97 interface.

```
 module ac97_controller (
   //Internal Interface
   input SYSCLK, //up to 125MHz
   input SYSTEM_RESET, //active on 1
   //Note, some slots utilize less than 20 bits
   input [19:0] slots [12:0];

   //External Interface connected to AC'97
   input BIT_CLK, //12.288 MHz
   input SDATA_IN,
   output SYNC,
   output SDATA_OUT,
   output RESET,
);
```

4.4 Clock Generation

As in the last lab, you must generate clocks using a DCM_SP module. You also need to worry about synchronizing with the AC97 chip, so be careful to phase match the clock appropriately!

5 Synthesis / Implementation

In order to properly synthesize your design at the correct frequency, you need to specify the clock signal PERIOD in the UCF file. As an example,

```
NET sys_clk_pin PERIOD = 20000 ps;
```

Once this is done, Xilinx can make sure all critical paths between flip-flops satisfy this constraint. If you use an additional clock to interface with the AC97, you must also specify this and worry about synchronizing data between the two clocks.

6 Test Bench and Simulation

Before you demonstrate your design on the FPGA, you should simulate it with your test bench. Your simulation should demonstrate the operation sufficiently to give confidence that the design will work in hardware. Include the results of your simulation in your final report. Your test bench can be implemented in non-synthesizable Verilog.

7 Lab Submission

Your lab will be submitted via your Canvas account. Please log in to Canvas using your UCSC account and attach the following files to your "Lab 5" assignment submission:

- lab5.v (with a top level module lab5)

- lab5_test.v (a test for the lab5 module)
- lab5.ucf
- lab5.bit
- lab5_report.pdf

Include any other Verilog files as needed. **Note that the final report must be submitted in PDF format.** If your bit file is too large, please zip it but not the other files.

If you have a partner, only one person should submit these files. Specify the names of partners in the report.

Make sure to confirm that your assignment is SAVED and SUBMITTED before the deadline. You may resubmit your assignment an unlimited number of times up until the due date.

7.1 Check-off

For this lab, as with most labs, you will need to demonstrate your lab when it is finished to the TA and get it signed off. This must be done by the end of lab section, so plan to be done early as there may be a queue of students! You will also need to submit your lab files using Canvas (including the final report) the evening that the lab is due.

7.2 Lab write-up

In the lab write-up, we will be looking for the following things. **Writing a succinct yet complete report is best.** We do not break down the point values; instead, we will assess the lab report as a whole while looking for the following content in the report.

- Include an example of the results of your simulation.
- Include a high level diagram of your design.
- Describe the high level diagram and state machines in your design.
- Discuss your test strategy.
- How many on-chip resources did you use (LUTs, block RAMs, etc.)?
- How did you configure your clock to interface with the AC97? Were there synchronization issues?
- How many clock domains did you have?
- What did each partner contribute to the lab assignment?
- Anything else interesting?

7.3 Extra Credit

There are two possible pieces of extra credit:

- Be creative in your song to receive up to 15 points extra credit. Make it fun. This is subjective.
- Implement a "volume meter" out of the LEDs that shows the total volume of all frequencies averaged over the last 100ms. 10 extra points

Extra credit must be demonstrated during check-off and must be included in the written report.