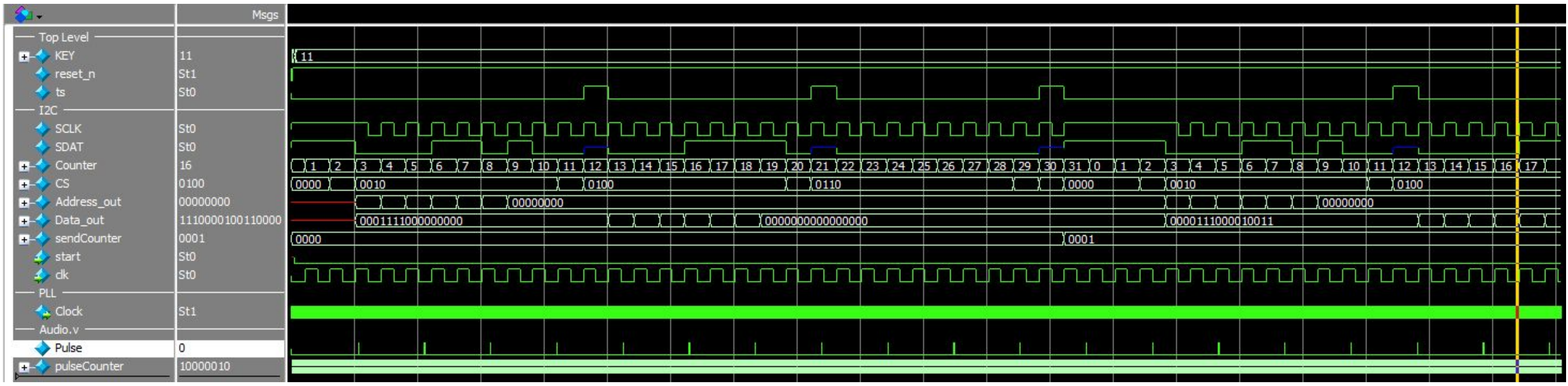
Matt Hartnett
ECEN 3002
12/09/2020

Project 2 Writeup

While I was unable to get music to play through the audio codec, there were still many things that I accomplished with this project. The main design goal was to program the audio codec using I2C and then play music out of a ROM. In order to program the audio codec chip, you need to use an I2C state machine, which I got operating in testbench, but unfortunately I was unable to get conclusive results if my I2C state machine operated on the board. In order to test it, I tried several different ISSP configurations as well as SignalTap, but neither resulted in evidence that I was receiving an ACK after sending my address/data. I also managed to program the I2C state machine in such a way that it would send multiple commands to audio codec to configure it before starting sending the data out of memory.

One of the largest stumbling blocks that I ran into was getting the timing right for my I2C state machine so that the chip would (theoretically) read the same data that I was sending. I spent a lot of time where the I2C would send the data one clock cycle off of the intended signal. I also had a few issues getting the correct frequency for the I2C clock, which has to be specifically set up with a clock divider. While this did take a while to resolve, I do believe that the I2C is successfully sending the data that I want it to. In order to start the I2C state machine, I used KEY[1] to start the programming and KEY[0] as a reset. You can see this reflected in the screenshot below by the slight activity on the KEYs at the very beginning.



I apologize for the lack of clarity in the screenshot, but as you can see, the SDAT line changes so that the address and data is programmed when captured on the falling edge of SCLK.

Towards the bottom of the screenshot, you can see the pulse signal that I am using to time the output of the memory file and send to the AUD_DACLRCK. It uses the 12MHz clock from the PLL with a maximum count of 250 to create the proper ratio for the sampling rate of the data provided. While I was not able to test bench the memory block, it should be sending data off of the 12MHz clock to the AUD_DACDAT line. With the memory file being read out and the I2C programming of the chip, the audio should work, but unfortunately it does not. A few possible causes would be my I2C not sending the addresses correctly (since I was unable to test the ACK), the memory file not sending the correct data out (since I couldn't simulate) or another issue that is due the specifics of the chip. Overall, I think that I completed most of the requirements of the project, but I ran out of time to troubleshoot all of the issues, but I am confident that with more time, I could have gotten it to work and play audio out.