Asher Voris
ECEN 3002
Prof. Robinson
9 December 2020

Audio Codec Project Writeup

This project aimed to drive the audio codec chip on the DE10 standard and play audio to it in the form of a mif file. The codec is configured using the i2c communication bus, for which a master was written. Success in playing music with the codec was not achieved. After many hours of debugging (which were performed because the project is extremely close to functioning), the problem has been fully realized; however, a solution was not found.

The i2c master functions properly in the following regards: waiting in the idle state, creating the start condition, sending the device address, sending the read write bit, sending the register address, sending data, and creating the stop condition. The master fails in the following regards: reading acknowledgement signals. The exact point of failure has been found; however, a solution has not. It is suspected that state machine logic is to blame and that the error is something obvious.

Debugging was performed in two ways, although three were attempted. Firstly, a test bench was created to allow prototyping of the master and the codec driving signals. The test bench output is shown in the next section. Next, signal tap was inserted into the design; however, a “signaltap found conflicting placement requirements for partitions preserving placement” error occurred which could not be resolved. Instead, output for the i2c bus was outputted on the DE10’s expansion header and external 4.7kOhm pullup resistors and a 3.3V source were provided to the bus. (It would have been convenient for the DE10 board designers to include the i2c bus on this head so a logic analyzer could be used, but they did not.) Although the codec was not on the bus in this configuration, this allowed the timing of the master logic to be analyzed. It was found that the acknowledgement state takes two clock cycles when it should take one. It is assumed that the cause of this issue is due to the state machine logic being used. An attempt to remedy this was to allow the master to stretch the clock signal high during the acknowledgement state, something which is not defined in the i2c protocol. This of course did not remedy this issue. Scope output is shown below, note for the purposes of driving an open bus, the acknowledgement is seen as a logical high versus a logical zero as if the codec were connected to the bus.

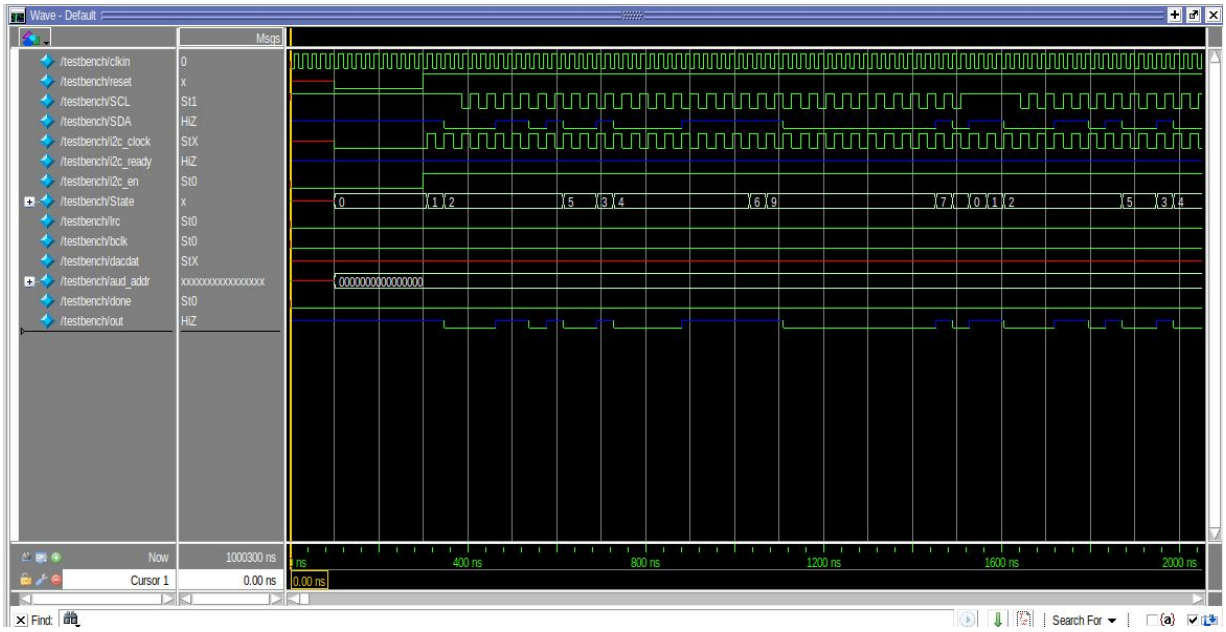
Figure 1: Scope Output of the Open I2C Bus

In this image, the start condition can be clearly seen as well as proper data(yellow) setting on the negative clock edge of SCL(green). This is data showing an attempted reset register write. First the start condition is seen. The first 7 bits are the device address followed by the R/W bit. The next “2 bits” are the acknowledge signal (logical high for reasons discussed above). This obviously did not function with the codec on the bus, but it can be seen that the SDA line is properly tristated during acknowledgement. Next the register address is sent followed by another “2 bit” acknowledgement. Finally 9 bits of data are sent to the register followed by the final acknowledgment. Because of the strange acknowledgement behavior (and the fact that a logical one is seen), the stop condition cannot be properly seen. However, when the acknowledgement is seen as a logical zero and the clock is not stretched, a proper stop condition occurs.

In terms of logic development, the i2c has a master with states for idle, start, stop, sending the device address, read write bit, device address acknowledge, register address, register address acknowledge, data to slave, stop, and a state for trapping and illuminating led0 if a nack occurs. The clock source originally used was a 51.2MHz pll divided down to the codec maximum of 512KHz; however, after debugging this clock rate was reduced to around 350KHz in an attempt to eliminate any clock based errata. The i2c master itself has a phase delay generation for the SCL line and the SDA line. The SCL is only enabled on the negative edge of the input clock and the SDA is only enabled during the positive edge. This ensures that data is set during the negative edge of SCL and read during the positive edge. Using the state status of the state machine, the SCL line can be enabled or disabled for states that require the clock and those that don't. . An altera tristate buffer was also instantiated to allow bidirectional communication on the SDA line of the i2c bus. A hard coded tristate buffer was attempted; however, it did not function. It is hypothesized that perhaps it is being optimized out during synthesis. The master module is called by the write registers module, which is a degenerate solution to push data onto the bus. Although not ideal for a generic master core, it should suffice for driving a single device like the codec.

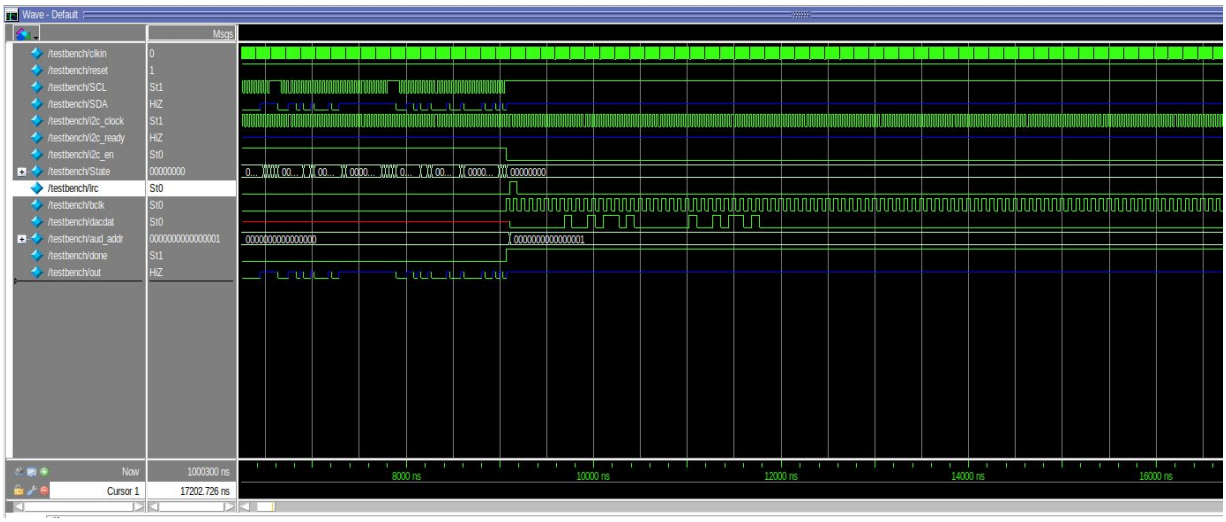
The top module functions as follows: A 12MHz pll is used for the codec master clock and bclk, a 51.2MHz clock is used for the previously mentioned i2c clock. The write registers module also outputs a "done" signal which is used to enable the master clock and bclk to the codec after the registers have been written. The DACLRCK clock is generated using the 12MHz pll and a counter set to a value to correspond to the 8KHz frame signal. This frame signal is then passed not only to the codec itself but also to a ROM which reads the mif file. The rom outputs a 16 bit word every DACLRCK positive edge as per specification. This word is passed to the Audio Data module which drives the left channel for the first 16 periods of the master clock and then the right channel for the next 16 periods of the master clock for every DACLRCK pulse. After the 32 pulses the line is driven low as per the codec specification. All codec based modules are also enabled using the write register done signal to ensure no data is pushed to the codec before it is finished configuring. Modelsim output is shown below demonstrating the complete transaction.

Figure 2: I2C Modelsim Output



A reset register write with all attempts at a solution removed. The start and stop conditions can be seen as well as proper tristating of the output line. The bottom waveform, out, demonstrates what the master would see on the SDA line during operation. Note the strange pattern of bits after the ACK cycles that insert an additional clock edge adding a zero to the address and registers data.

Figure 3: Codec Data Input



Output demonstrating data going to the codec. Clock signals can clearly be seen enabling after the I2C transactions are complete. Symmetric data can then be seen on the left and right channels of the data bus to the codec. If this waveform was more zoomed out, A repeating frame signal could be seen occurring every 125uS to correspond with an 8KHz period.

It is firmly believed that if the acknowledgement error could be resolved this design would be functional. In terms of output nothing substantial was achieved. Noise can be heard from the codec, including some music, if the acknowledgements are ignored; however, it is noisy as is expected since the codec is not properly configured.