

Hybrid-Input Effects Pedalboard for Guitar

By
Connor Gooding
Justin Wilson

Final Report for ECE 445, Senior Design, Spring 2016

TA: Braedon Salz

4 May 2016

Project No. 63

Abstract

In the music technology industry, there are two types of stompbox guitar effect pedalboards: multi-effects pedalboards and custom single-effect pedal chains. The former provides a guitarist with many types of effects to try, with no one effect having a significantly high tone quality. The latter provides a guitarist with a more specialized effect per unit, and thus higher tone quality, and the cost of more external purchases to power, patch, and house all of the pedals. Integrating the two kinds of pedalboards is often awkward and presents few options for creative customization.

This Hybrid-Input Effects Pedalboard interrupts the digital signal chain is a multi-effects processor board that allows for interruption of the digital signal chain for the incorporation of up to three third-party single-effects pedals.

Contents

1	Introduction	1
2	Design.	2
2.1	Data Conversion - Hardware	2
2.1.1	Analog-to-Digital Converters	2
2.1.2	Digital-to-Analog Converters	2
2.2	Data Conversion - Software	3
2.2.1	Deserializer	3
2.2.2	Serializer	3
2.3	Parameter Control Hardware	4
2.4	Software Menu	4
2.5	Digital Effects.	5
2.5.1	Distortion	5
2.5.2	Tremolo.	5
2.5.3	Digital Delay	6
2.6	Digital Patch Bay	8
2.7	Power.	8
2.7.1	12 Volt Power Supply.	9
2.7.2	9 Volt Regulator	9
3	Design Verification.	10
3.1	Data Conversion - Hardware	10
3.1.1	Analog-to-Digital Converters	10
3.1.2	Digital-to-Analog Converters	10
3.2	Data Conversion - Software	10
3.3	Software Menu	10
3.4	Digital Effects.	10
3.4.1	Distortion	11
3.4.2	Tremolo.	12
3.4.3	Digital Delay	13
3.5	Digital Patch Bay	13
3.6	Power.	13
3.6.1	12 Volt Power Supply.	14

3.6.2	9 Volt Regulator	14
4	Cost	15
4.1	Parts	15
4.2	Labor.	15
5	Conclusion.	16
5.1	Accomplishments	16
5.2	Uncertainties	16
5.3	Ethical considerations	16
5.4	Future work	17
	Reference	18
	Appendix A Requirement and Verification Table	18
	Appendix B Full Circuit Schematic	25

1 Introduction

Making the transition from a multi-effects processor to a custom chain of single-effect pedals is a huge financial investment. A guitarist that currently owns a multi-effects processor and would like to experiment with single-effect pedals might desire a pedalboard that could integrate a few single-effect pedals with his or her existing multi-effects processor board. This pedalboard allows for such integration by isolating each of its three effects modules, and routing them through a software-controlled digital patch bay which can bypass them in favor of up to three analog single-effect pedals at the user's request. The board also provides three stock digital guitar effects: distortion, tremolo, and digital delay; up to three adjustable parameters per effect, and three built-in toggle switches for enabling or bypassing these effects while playing.

The pedalboard, as seen in Figure 1, accomplishes all of this by first using a 24-bit analog-to-digital converter (ADC), connected to the 1/4" input jack, to convert the analog guitar signal to 24-bit pulse code in Left Justified format and feeding it into the GPIO header of a DE2-115 FPGA Development Board. Inside the FPGA, a Deserializer converts that pulse code to a 24-bit parallel signal and passes it to the digital patch bay. The patch bay then sends the 24-bit sample through three effects units, which could either be digital algorithms inside the board or analog pedals connected via to the export channels. The router's patching and the parameters sent to any digital effects can be modified by the user using a software menu displayed on the LCD of the DE2-115 development board and the three Potentiometers that send DC control data to an 8-bit parallel out analog-to-digital converter. The Serializer then converts the 24-bit signal into Left Justified pulse code and sends it to a 24-bit digital-to-analog converter, which generates an analog signal from this data and sends it out to the 1/4" output jack for patching to an amplifier or other processing units.

The block diagram for the entire circuit may be seen in Figure 1.

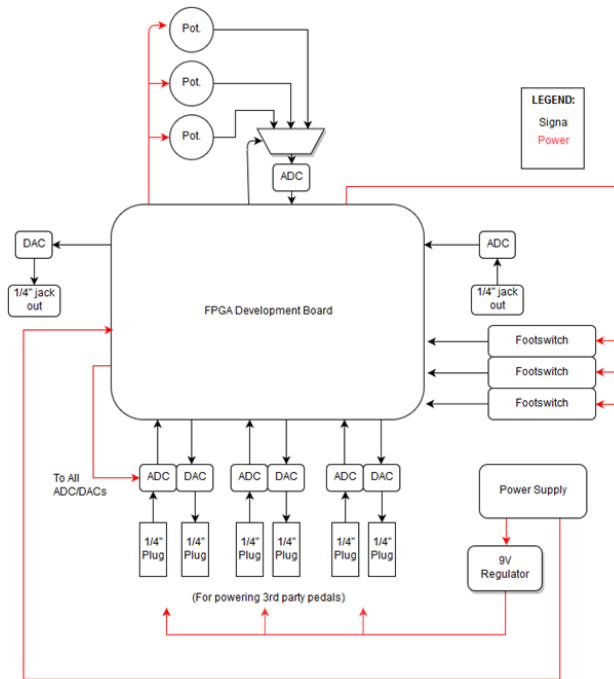


Figure 1: Block diagram of the full system

2 Design

2.1 Data Conversion - Hardware

2.1.1 Analog-to-Digital Converters

Conversion of analog audio data to the digital realm is usually done in a resolution of at least 16 bits to preserve a strong dynamic range and sensitivity. The Nyquist frequency for audio signals must be at least 20 kHz to prevent any audible aliasing, as that is the upper threshold of the audible frequency range for humans. The PCM1808 by Texas Instruments is a 24-bit stereo input analog-to-digital converter that can sample at up to 96 kHz. For the purposes of this project, it is programmed to sample at 52.083 kHz, which correlates to a satisfactory Nyquist frequency of 26.042 kHz.

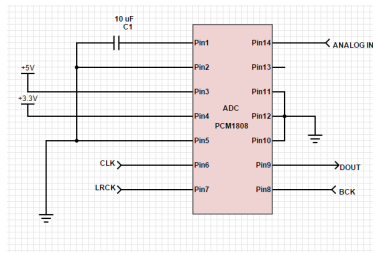


Figure 2: Schematic for the PCM1808 ADC

The PCM1808 takes two channels of analog data in and converts them both to one serial data line of 24-bit pulse code in the PCM Left Justified format by means of Delta-Sigma modulation, oversampling, and decimation. The chip receives a 20 MHz system clock signal (SCKI, pin 6, Figure 2), a 2.5 MHz bit clock signal (BCK, pin 8, Figure 2), and a 52.083 kHz left-right clock signal (LRCK, pin 7, Figure 2). Each bit of the pulse code is output on the positive edge of the bit clock, and the left-right clock has a period 48 times larger than the bit clock to encompass 48 bits per cycle, or one 24-bit signal per channel per cycle. Pins 10-12 (Figure 2) are grounded to set the chip to slave mode and Left Justified format. In slave mode, the chip detects automatically that the system clock is 384 times the frequency of the left-right clock. The chip also requires two power sources: a 5V supply for the analog side and a 3.3V supply for the digital side.

2.1.2 Digital-to-Analog Converters

Pulling the digital audio data back in the analog realm accurately requires a digital-to-analog converter that also expects a sampling rate of 52.083 kHz and a 24-bit pulse code in PCM Left Justified Format.

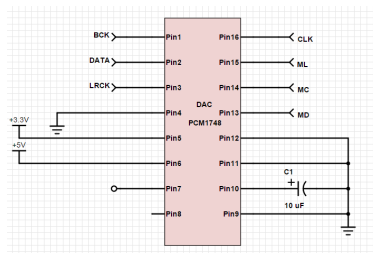


Figure 3: Schematic for the PCM1748 DAC

The PCM1748 is the partner chip to the PCM1808 and satisfies both of those requirements. It uses the

same three clocks as the PCM1808 (SCK, BCK, LRCK) to convert one line of 24-bit pulse code in PCM Left Justified format to two stereo analog audio signals. This chip is slightly more complicated; it has a 3-pin mode control (Figure 3), which for the purpose of staying in slave mode and expecting 24-bit Left Justified format data, is set to latch the default setting of the mode register upon start-up. Like the PCM1808, it also requires a 5V analog-side supply voltage and a 3.3V digital-side supply voltage.

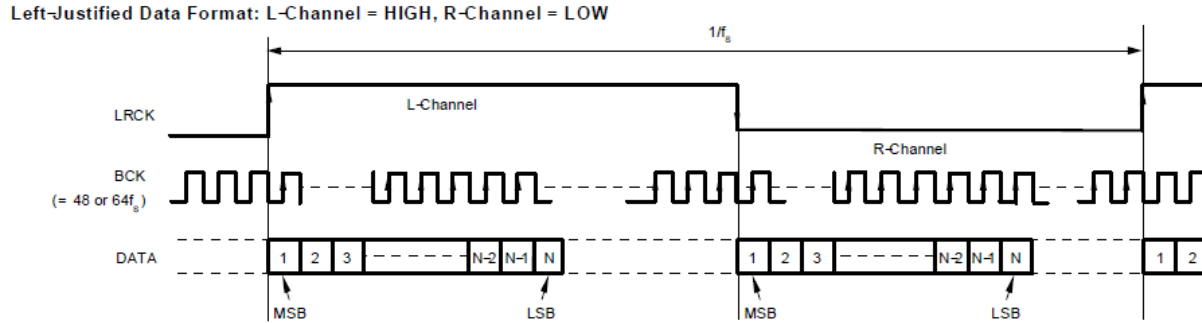


Figure 4: Left-Justified 3 wire serial timing diagram

2.2 Data Conversion - Software

As mentioned before data conversion that was used is 24-bit PCM Left-Justified, which is sent and received using a 3 wire serial interface with a bit clock (BCK), left-right channel select clock (LRCK) and data which is sent on BCK. The timing diagram can be seen in Figure 4.

2.2.1 Deserializer

The Deserializer is a module instantiated on the FPGA that serves to convert the 24-bit PCM Left-Justified data coming from the PCM1808 into 24-bit parallel data on the positive edge of the LRCK. The SystemVerilog module symbol may be seen in Figure 5.

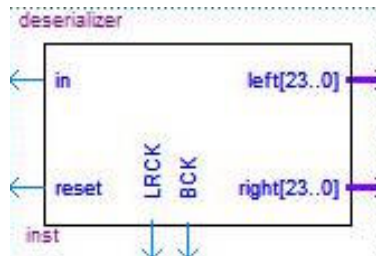


Figure 5: Deserializer module in SystemVerilog

2.2.2 Serializer

The Serializer is a module instantiated on the FPGA that converts 24-bit parallel data into 24-bit PCM Left-Justified data going to the PCM1748 on the positive edge of LRCK. The SystemVerilog module symbol may be seen in Figure 6.

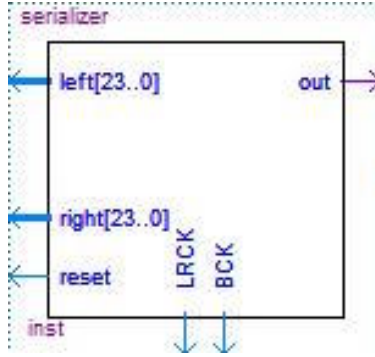


Figure 6: Serializer module in SystemVerilog

2.3 Parameter Control Hardware

To accomplish parameter control $1k\Omega$ potentiometer is connected to 5V, and the sweeper is sent to a mux, which selects between three potentiometers and sends the selected value to an ADC which goes into the FPGA. Figure 7 shows the schematic for each individual potentiometer, and how the potentiometers are sent to the multiplexer may be seen in the main block diagram in Figure 1.

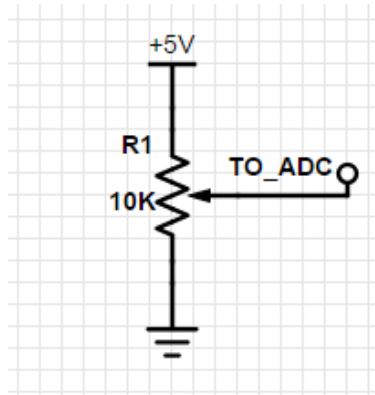


Figure 7: Schematic used to wire up the Potentiometers

2.4 Software Menu

The software menu is the main system to change parameters and digital patching for the whole system. A sample diagram showing possible paths through the software menu can be seen in Figure 8. The LCD that displays the contents of the menu at any given time is a 16 character two line display, and as such each block in Figure 8 represents a frame that would be written to the LCD.

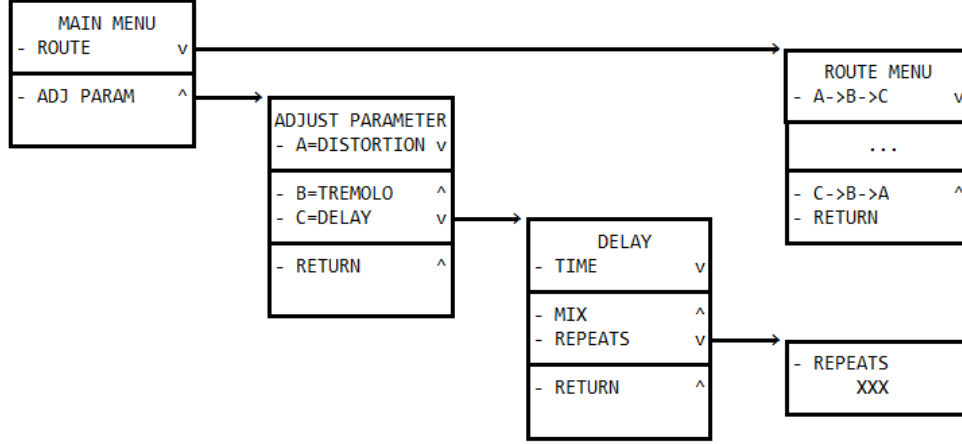


Figure 8: Sample of the Menu System

2.5 Digital Effects

2.5.1 Distortion

The Distortion Unit consists of two sub-modules: the Clipper and the Driver. It is also dependent on two user-controlled parameters: Drive and Volume.

The Clipper takes the input signal and clips the maximum possible value of the output to a threshold value that is dependent on the Drive parameter. This relationship can be described in a linear piece-wise equation:

$$y[n] = \begin{cases} x[n], & x[n] \leq (2^{23} - 1) - 16448 \bullet D \\ \leq (2^{23} - 1) - 16448 \bullet D, & x[n] > (2^{23} - 1) - 16448 \bullet D \end{cases}$$

where D is the Drive parameter, an integer ranging from 0 to 255.

When a digital audio signal is clipped like this, it gains harmonic content but loses volume. To compensate for the volume loss, the Driver equalizes the amplitude of the signal. This equalization ranges from $-\infty$ to +10 dB SPL. This takes the form of a piece-wise logarithmic equation:

$$y[n] = \begin{cases} x[n] \times 10^{(V-127) \frac{2}{127}}, & 0 \leq V \leq 127 \\ x[n] \times 10^{0.64V}, & 127 < V \leq 255 \end{cases}$$

where V is the Volume parameter, an integer ranging from 0 to 255.

2.5.2 Tremolo

The tremolo unit submits the digital audio data to elementary periodic modulation using a 1-10 Hz square wave as the control signal. It is dependent on three user-controlled parameters: Rate, Depth, and Duty Cycle. The sequential logic is dependent on the positive edge of the left-right clock (LRCK).

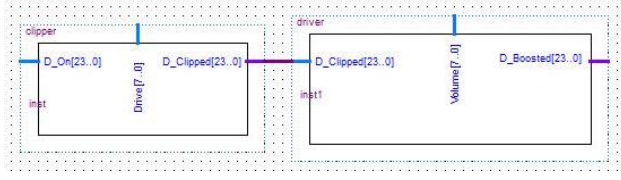


Figure 9: Block diagram for the distortion module in SystemVerilog

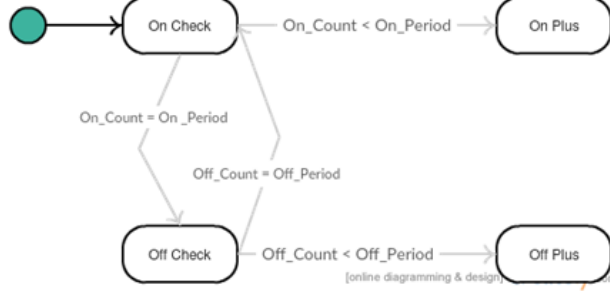


Figure 10: State Machine for Tremolo

The finite state machine of the Tremolo Unit enables and bypasses the Attenuator module when the square wave is in its On or Off cycle, respectively. It sends incrementation and clearing commands to an On Counter and an Off Counter, which count to a number of samples specified by the Duty Cycle and Rate of the control waveform.

The Rate parameter is converted to the Total Period using the following equation:

$$T = \frac{52803 \times 255}{18 \times R + 510}$$

where T is the Total Period and R is the Rate parameter, an integer ranging from 0 to 255.

The Duty Cycle parameter is converted to a ratio for determining the Off Counter threshold using the following equation:

$$N = \frac{64 \times (255 - D)}{255}$$

where N is the Duty Numerator and D is the Duty Cycle parameter, an integer ranging from 0 to 255.

The Depth parameter is used in the same way to determine the ratio by which the signal is multiplied during the Off cycle for attenuation.

2.5.3 Digital Delay

The Digital Delay unit sends the current digital audio sample to an SRAM chip that serves as a circular buffer, pulls five repeat values from memory, and sums the repeats together with every sample to add echo to the signal. It is dependent on three user-controlled parameters: Time, Repeats, and Mix. The sequential logic is dependent on the positive edge of the bit clock (BCK).

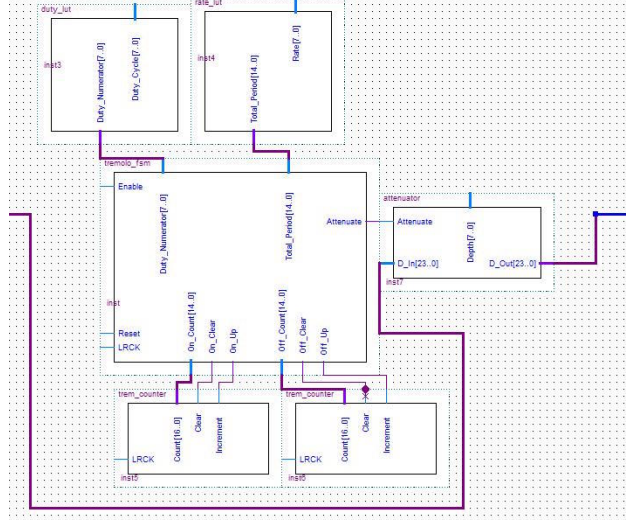


Figure 11: Block diagram for the tremolo module in SystemVerilog

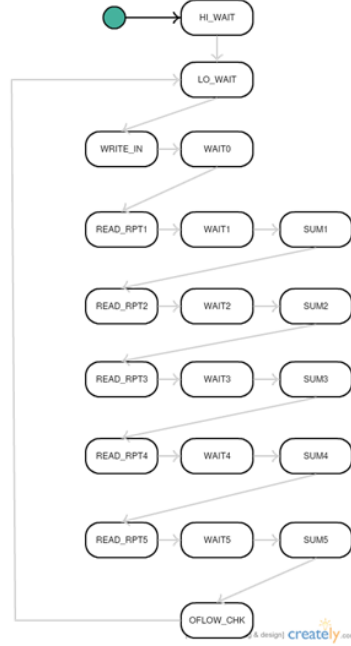


Figure 12: Block diagram for the tremolo module in SystemVerilog

The Delay algorithm first takes the input sample and writes it to SRAM at a base address, which is incremented for every cycle of the left-right clock. It then decrements the SRAM address by a step value derived from the Time parameter and reads the data at that address to a repeat register. This derivation can be described by the exponential relationship:

$$S = 2605 \times 1.0161845455^T$$

where S is the step size and T is the Time parameter, an integer ranging from 0 to 255.

After each read operation and a WAIT state, the read value is multiplied by a coefficient determined by the Mix knob and what number repeat it is. Each repeat, when enabled, is half of the amplitude of the previous repeat, except for the first repeat. The first repeat is attenuated by the coefficient determined by Mix. This derivation can be described by the linear relationship:

$$A = \frac{64}{255} M$$

where A is the attenuation coefficient and M is the Mix parameter, an integer ranging from 0 to 255.

Lastly, the Repeats parameter unmutes each repeat as it is increased, leaving only one repeat unmuted when it is set to 0 and all of the repeats unmuted when it is set to 255.

2.6 Digital Patch Bay

The Digital Patch Bay handles both the routing and the third party exports for other pedals. The basic design is quite simple, as there are three DSP blocks, as seen in Figure 13 as A, B, and C, which in the final design are modules containing Distortion, Tremolo and Delay, respectively, and a multiplexer/demultiplexer combo on the inputs and outputs of the DSP module. Getting more specific, Figure 14 shows in a general sense what the internals of A, B, and C are. As seen, the DSP has parameter controls coming in as selected by the aforementioned potentiometers, a stomp switch input to turn the effect on or off, and audio in. The output of DSP goes into a multiplexer, and depending on whether you have the external pedal plugged in or not, a multiplexer selects the appropriate audio output.

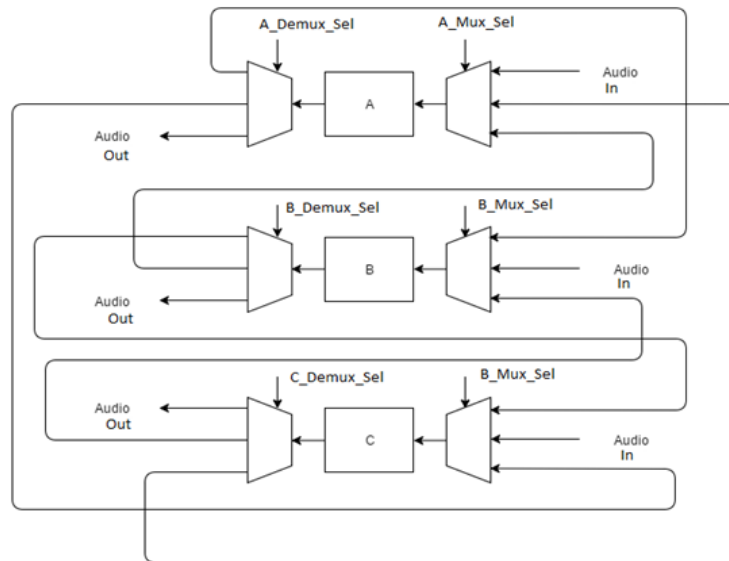


Figure 13: Basic design of the Router

2.7 Power

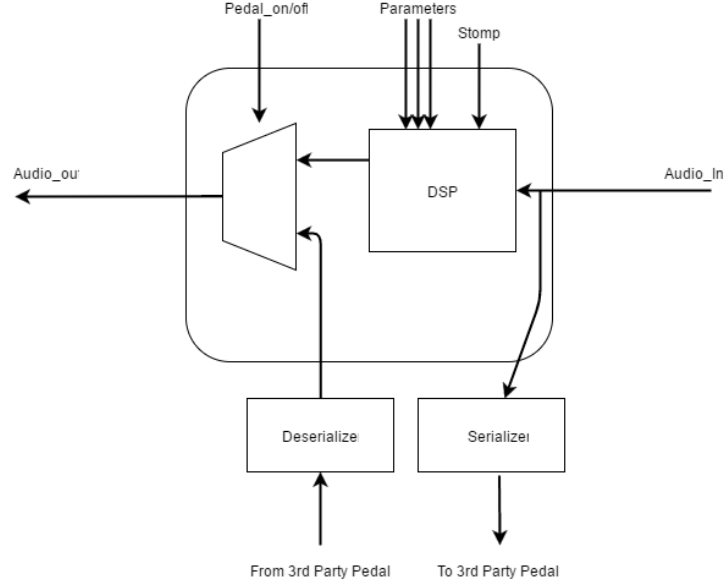


Figure 14: Internals of A, B, C

2.7.1 12 Volt Power Supply

For the 12V power supply the system is rather simple. The device used is simply an off-the-shelf unit with a variable voltage adjust level. The power supply we used was the Meanwell RS-35-12.

2.7.2 9 Volt Regulator

There also needs to be 9V power exports for the third party pedals, as per the design. As such, a 9V regulator was used, the PTN78060W, to be precise. The schematic for how the PTN78060W was used can be seen in Figure 15.

$$R_s = 54.9k\Omega * (1.25V)/(V_o - V_m) - R_p$$

$$V_o = 9.0V; R_p = 6.49k\Omega; V_m = 2.5V$$

Using the equations above with the constants as per the datasheet, the value 4.876k Ω was calculated.

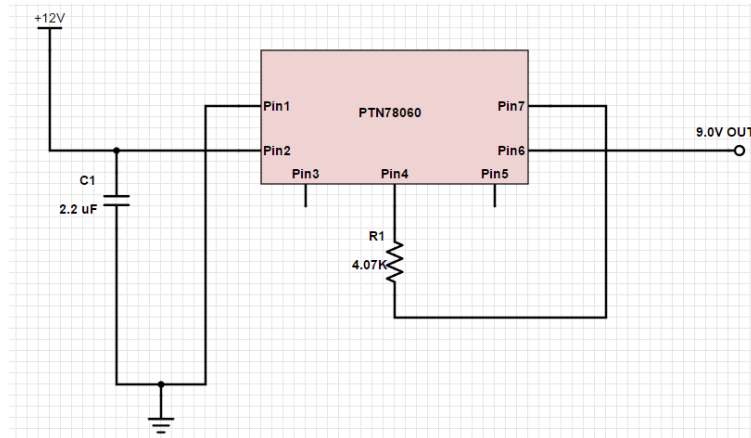


Figure 15: 9 Volt Regulator

3 Design Verification

3.1 Data Conversion - Hardware

3.1.1 Analog-to-Digital Converters

The PCM 1808, once properly set up, powered, and fed clock signals, successfully fulfilled its functions. The figure below shows the output of the PCM1808, which is generating 24-bit PCM Left Justified data from the noise on an unplugged input as it crosses the zero threshold.

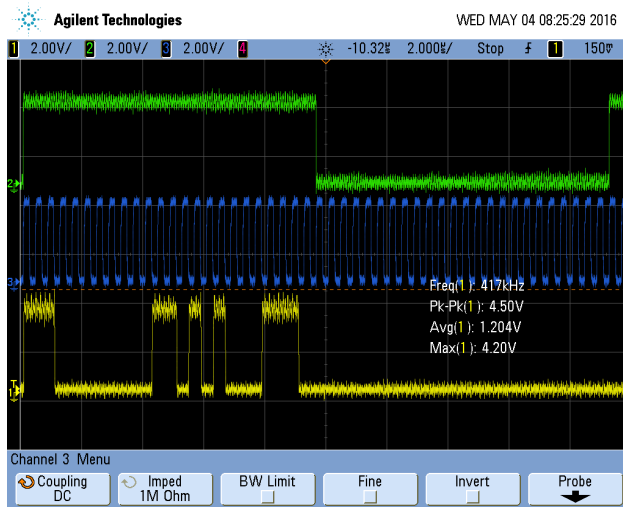


Figure 16: 24-bit PCM Left Justified data displayed on the oscilloscope, PCM1808 verification

3.1.2 Digital-to-Analog Converters

The PCM1748's operation was verified by connecting it directly to the output of the PCM1808, sending a 2.5V peak-to-peak voltage waveform into the PCM1808, and recording the output. The output is noisy, requiring some high-pass filtering, and its amplitude is 2V, but the sinusoidal shape can be seen in the Figure 17.

3.2 Data Conversion - Software

The Serializer and Deserializer were verified by direct connection in a top-level file in SystemVerilog and an RTL simulation. A test signal 0xAFFFFFFF was sent in pulse code to the Deserializer while LRCK was high, and the same signal was pulsed out of the Serializer on the next positive edge of LRCK. This is displayed in Figure 18.

3.3 Software Menu

Unfortunately the full software menu was not realized in the end design. While one could ascertain that they were navigating through the menu by setting certain flags to trigger LEDs on the board, the LCD was not displaying any information, and as such the software menu failed verification.

3.4 Digital Effects

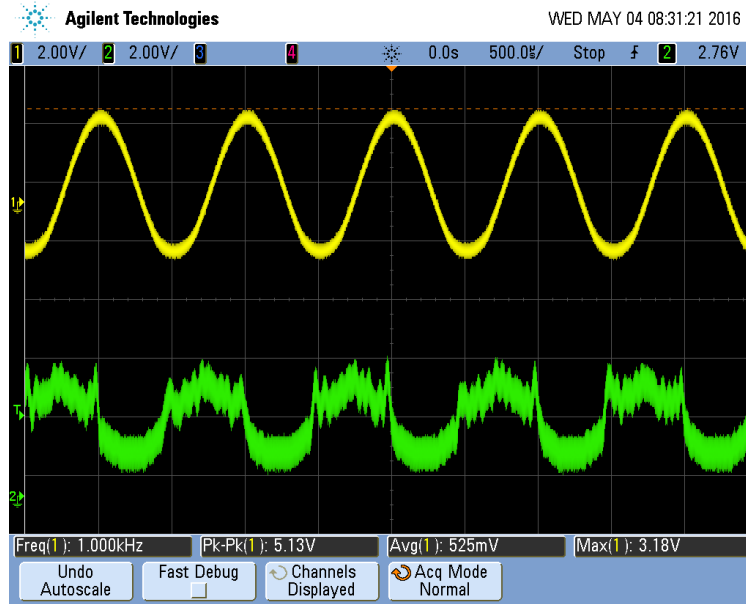


Figure 17: Sinusoidal signal transmitted through ADC/DAC pair, PCM1748 verification

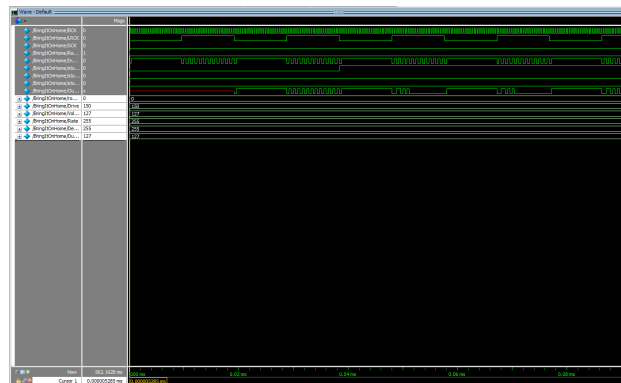


Figure 18: Successful Deserialization and Serialization in ModelSim Altera

3.4.1 Distortion

Figure 19 shows the input-output relationship of the Distortion effect when the Drive parameter is increased. The threshold is lowered as the parameter increases, and both positive and negative values of the input are clipped correctly without overflow issues.

Figure 20 shows the input-output relationship of the Distortion effect when the Volume parameter is increased. The signal is muted when the parameter is set to 0, and it increases in amplitude as the parameter is increased. Once the signal has been boosted past the maximum possible magnitude, clipping occurs as intended until the parameter approaches its maximum. This final region displays undefined behavior, where the clipping gouges the signal twice per peak.

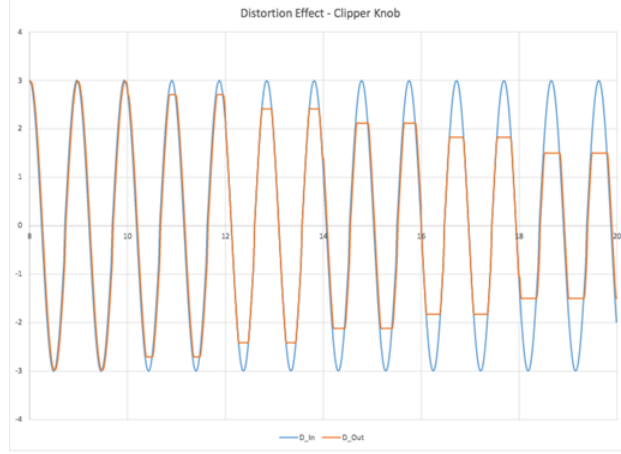


Figure 19: Effect of Drive Parameter on Signal

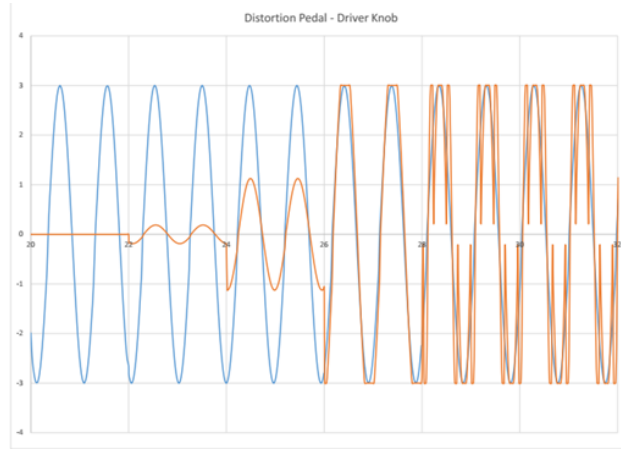


Figure 20: Effect of Volume Parameter on Signal

3.4.2 Tremolo

Figure 21 displays the effect of all three parameters on the modulation envelope cast onto the input signal. The simulation window 0.5s divisions. At zero Rate, maximum Depth, and half Duty Cycle, the envelope is a 1V peak-to-peak, 1 Hz square wave with a DC offset of 0.5 V. When Rate is maximized while holding the other two parameters constant from the previous example, the square wave's frequency increases to 10 Hz. When the Depth is set to 127, or halfway, the amplitude of the OFF cycle is 0.5 V. When the Duty Cycle is set to 200, it is evident in Figure 21 that the ON cycle lasts longer than the OFF cycle. Lastly when the Duty Cycle is set to 0, the signal is completely muted, as the OFF cycle is now operating at 100%.

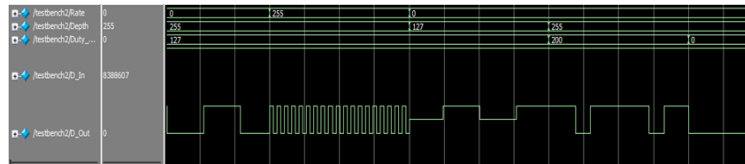


Figure 21: Tremolo Envelope, RTL Simulation in ModelSim

3.4.3 Digital Delay

The digital delay did not operate as intended, as an SRAM of the required size (2 MB) could not be instantiated effectively in the Quartus testbench and ModelSim simulation. However, Figure 22 demonstrates that the relationship between the Time parameter and the SRAM address spacing per repeat functions as intended, as well as the relationship between the Mix parameter and the amplitude of the repeats and the relationship between the Repeats parameter and the muting of later repeats.

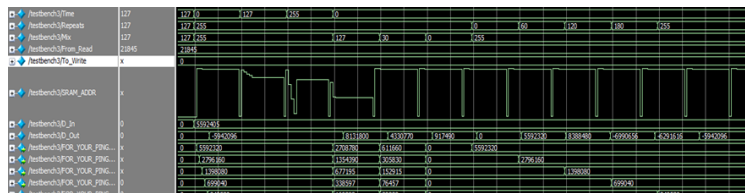


Figure 22: Digital Delay Simulation, ModelSim

From the figure, it is evident that as the Time parameter increases, the address value requested for read/write operations steps in larger decrements. When the Mix parameter is set to maximum, the first repeat is approximately the same amplitude as the input signal. It is not exactly the same, as the memory can only take 16 bits of the 24-bit sample. The workaround to this deficiency in the SRAM was to only send and receive the 16 most significant bits of the sample to memory. Since almost digital audio effects units use 16-bit resolution or greater, this should not be a noticeable trait to the user. If anything, the echoes will sound slightly quieter than expected. One can also observe from the simulation that as the Repeats parameter is brought to 0, all of the repeats except the first one are muted. As that parameter is increased, the repeats are unmuted one by one.

3.5 Digital Patch Bay

For the router, or Digital Patch Bay, as it is difficult to get practical values that could show the digital routing is working in real life, a simulation was used to show it works. For the simulation a constant value was sent in, the routing parameters were changed, and as such it propagated correctly through the routing options. As seen in Figure 23, this was done in a gate level simulation, hence the red uncertainties, to ensure that the device would work properly.

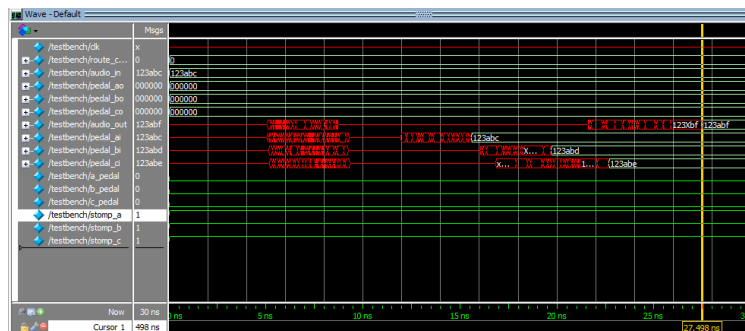


Figure 23: Gate level simulation of the router

3.6 Power

3.6.1 12 Volt Power Supply

The testing for this was fairly trivial. An adequately small network of power resistors was put across the terminals, and we observed the voltage. For a 4.24Ω resistor, the voltage out of the power supply stayed at a constant value of around 11.73V, as such it met design verification requirements. The value of 4.24Ω was calculated by finding the maximum theoretical power draw of the full circuit, which, summing up the various components maximum current values and adjusting accordingly, was found to be a maximum of 2.833A. Using Ohm's law, $12V / 2.833A$ yields a value of 4.24Ω .

3.6.2 9 Volt Regulator

The testing for the 9 Volt regulator was similar to that of the 12 Volt power supply. When a 10.8Ω resistor was put across the output terminals, the voltage fell to 8.84V, and as such it met the design verification requirements.

Again similar to the 12V power supply calculations, 10.8Ω was calculated using Ohm's law with a maximum current estimated (in the event of powering 3 pedals) being 0.833 Amperes. As such $9V / 0.833A = 10.8\Omega$.

4 Cost

4.1 Parts

Table 1: Parts Costs

Part	Manufacturer	Quantity	Cost Per Unit (\$)	Total Cost (\$)
DE2-115	Altera	1	309.00	309.00
PCM1748	Texas Instruments	2	1.51	3.02
PCM1808	Texas Instruments	2	0.90	1.80
ADG704	Analog Devices	1	1.00	1.00
ADC0804	Texas Instruments	1	4.00	4.00
P-3PDT-1	Tube Depot	3	6.95	20.85
RS-35-12	Mean Well	1	12.95	12.95
PTN78060W	Texas Instruments	1	17.16	17.16
1k Ω Trimpots	Suntan	3	0.95	2.85
1/4" Connectors	Seismic Audio	6	4.99	29.94
Mono 1/4" Jacks	Amphenol	2	0.99	1.98
Total				404.55

4.2 Labor

Table 2: Labor Costs

Name	Hourly Rate	Total Hours Invested	Total Cost
Connor Gooding	\$30.00 /hr	160	\$12,000.00
Justin Wilson	\$30.00 /hr	160	\$12,000.00
TOTAL	\$30.00 /hr	160	\$24,000.00

$$Cost = \$24,000 + \$404.55 = \$24,404.55$$

5 Conclusion

When viewing the individual parts of the project, many things did work, however there were issues in final system integration, and as such much of the testing of the digital circuits had to be solely done in simulations, and real data for audio waveforms could not be generated.

5.1 Accomplishments

While not everything worked perfectly there were many things that did work to specifications. For example, the ADC was able to turn analog data into 24 bit left-justified serial data, which the DAC was then able to turn back into analog data. In addition, the power system of the whole unit worked. The off the shelf 12V power supply was able to supply ample current, and the 9V regulator was able to supply 9V at its maximum current. Furthermore, the serialization and deserialization modules both worked in testbenches, so the pulse code from the ADC could be converted to parallel data for the effects, then the parallel data could then be turned back to pulse code for the DAC. Lastly, the effects modules all worked in simulation, and even in gate-level simulation which gives a more accurate representation of what would happen on a physical chip.

5.2 Uncertainties

Despite the accomplishments, there were a few uncertainties along the way. The largest issue that presented itself, which prevented physical data from being able to be processed, was with the analog to digital converter. The GPIO that was used on the development board was drawing too much current from the ADC data pin, and as such the voltage from the ADC data out was too low to register anything via the GPIO. This issue could not be remedied by any passive approaches that were taken, so an active approach such as using a high speed buffer, if this development board were to be used in the future, could be used. Another issue that arose was that the software menu was not fully realized in the design. The user could navigate through the menu, but the data was not displaying on the LCD. Likely this issue could be solved with a bit more testing and editing of the state machines that control the LCD. One final thing that was not accounted for was input/output protection to the circuit. There is no form of protection from voltages that are too high, too low, reversed, etc. This could be remedied via a simple protection circuit consisting of two diodes and a resistor.

5.3 Ethical considerations

This project conforms to the IEEE Code of Ethics as follows:

- [1] to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;
- [3] to be honest and realistic in stating claims or estimates based on available data;
- [5] to improve the understanding of technology; its appropriate application, and potential consequences;
- [6] to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
- [7] to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
- [9] to avoid injuring others, their property, reputation, or employment by false or malicious action;
- [10] to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

5.4 Future work

For this to be a viable product to go on the market there would need to be some work done on the project. First and foremost, the aforementioned issues should be fixed. Once the issues are fixed, a switch from an FPGA development kit to a stand-alone FPGA chip or a microprocessor (such as a SHARC) would be a logical leap. There are more functions on the development kit than truly needed, and as such one would be able to skim down to just the bare minimum of components used in the design, such as the FPGA itself, and SDRAM. Increasing the amount of routing options would also be a step in the right direction. This project was limited to three options, however in real life situations many musicians have five or more pedals or effects they would like to implement. A next logical step to take for future work would be to get rid of the 2 line 16 character LCD and three button menu system and switch to a more modern touch screen interface with a GUI. This would greatly increase ease of use, especially for an increased number of routing options, since, if there were eight routing options, which is $8!$ or 40,320 options that the user could select. If this was in a linear list it would take far too long to get to the routing the user might desire.

Appendix A Requirement and Verification Table

Table 3: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Power Supply (a) $V_{out} = 12\text{ V} \pm 0.5\text{ V}$ at 2.833 A	1. Verification process for item 1: (a) Attach 4.24 Ω resistor as a load (b) Attach oscilloscope across load (c) Plug power supply into wall (d) Ensure output remains between 11.5 and 12.5 V	1. Y
1. 9 Volt Regulator (a) $V_{out} = 9.0\text{ V} \pm 0.6\text{ V}$ at 0.833 A	1. Verification process for item 1: (a) Attach 10.8 Ω resistor as a load (b) Attach oscilloscope across load (c) Apply 12V to the voltage in pin, and ground to the GND pin (d) Ensure output remains between 8.4 and 9.6 V	1. Y
1. FPGA Deserializer/Serializer (a) Test value input using 24-bit PCM Left-Justified serial format is converted to a 24-bit parallel signal. (b) Test value input in 24-bit parallel format is transmitted out in 24-bit PCM Left-Justified serial format.	1. Verification process for item 1: (a) Using either testbench or a test signal-generating module in SystemVerilog, generate the 24-bit PCM Left-Justified signal 24h555555 and feed it into the input to the Deserializer over 24 bit-clock cycles. (b) Verify that the 24-bit parallel output of the Deserializer is 24h555555. 2. Verification process for item 2: (a) Using either testbench or a register module in SystemVerilog, feed the 24-bit parallel input 24hFFFFFF into the input of the Serializer. (b) Verify that for 24 bit-clock cycles, the serializer is outputting a logic high (3.3V herein is considered logic high).	1. Y 2. Y
Continued on next page		

Table 3 – continued from previous page

Requirement	Verification	Verification status (Y or N)
<p>1. FPGA Distortion Unit</p> <p>(a) Setting Enable to logic-low (herein known as 0 V relative to GND) bypasses the unit and setting Enable to logic-high sends the input signal through the unit.</p> <p>(b) The magnitude of both positive and negative values exceeding the potentiometer-set clipping threshold, Drive, are clipped to the magnitude of the threshold.</p> <p>(c) Volume parameter effectively adjusts the gain of the unit, with unity gain at the center of the input spectrum (8d127) and complete mute ($-\infty$ dB) at Volume = 8'd0.</p>	<p>1. Verification process for item 1:</p> <p>(a) Set Enable to 1b0.</p> <p>(b) Set D_{In} to 24h7FFFFFFF.</p> <p>(c) Verify that $D_{Out} = D_{In}$ using either testbench or onboard hex displays and switching.</p> <p>(d) Set Enable to 1b1.</p> <p>(e) Verify that $D_{Out} \neq D_{In}$ and changes with 2 different sets of parameter values (of the set D_{In}, Drive, Volume) using either testbench or onboard hex displays and switching.</p> <p>2. Verification process for item 2:</p> <p>(a) Set Enable to 1b1.</p> <p>(b) Set Drive to 8d0 and D_{In} to 24h7FFFFFFF.</p> <p>(c) Verify that $D_{Clipped} = 24h7FFFFFFF$ (no clipping).</p> <p>(d) Set Drive to 8h08 and keep D_{In} at 24h7FFFFFFF.</p> <p>(e) Verify that $D_{Clipped} < 24h7FFFFFFF$ (clipped).</p> <p>(f) Repeat a-c with $D_{In} = 24h800000$.</p> <p>(g) Verify that $D_{Clipped} > 24h800000$ (clipped).</p> <p>3. Verification process for item 3:</p> <p>(a) Set Enable to 1b1.</p> <p>(b) Set Volume to 8d0, Drive to 8d0, D_{In} to 24h00FFFF.</p> <p>(c) Verify that $D_{Boosted} = 24h000000$.</p> <p>(d) Verify that $D_{Boosted} < 24h00FFFF$.</p> <p>(e) Set Volume to 8d127, Drive to 8d0, and D_{In} to 24h00FFFF.</p> <p>(f) Verify that $D_{Boosted} = 24h00FFFF$.</p> <p>(g) Set Volume to 8hC0, Drive to 8d0, and D_{In} to 24h00FFFF.</p> <p>(h) Repeat a-h with D_{In} set to 24hFF0001.</p>	<p>1. Y</p> <p>2. Y</p> <p>3. Y</p>
Continued on next page		

Table 3 – continued from previous page

Requirement	Verification	Verification status (Y or N)
<p>1. FPGA Tremolo Unit</p> <p>(a) Setting Enable to logic-low bypasses the unit and setting Enable to logic-high sends the input signal through the unit.</p> <p>(b) The potentiometer-controlled parameter Rate adjusts the frequency of the modulation on the input signal, never entering into the audible frequency range.</p> <p>(c) The potentiometer-controlled parameter Depth adjusts the attenuation multiplier on the signal during the off cycle of the modulation, completely muting the signal when set to max (8hFF).</p> <p>(d) The potentiometer-controlled parameter Duty Cycle adjusts the time ratio between the on and off cycles of the modulation, and the ratio sits at unity when set to 8d127.</p>	<p>1. Verification process for item 1:</p> <p>(a) Set Enable to 1b0.</p> <p>(b) Set D_{In} to 24h7FFFFFFF.</p> <p>(c) Verify that $D_{Out} = D_{In}$ using either testbench or onboard hex displays and switching.</p> <p>(d) Set Enable to 1b1.</p> <p>(e) Verify that $D_{Out} \neq D_{In}$ and changes with 2 different sets of parameter values (of the set D_{In}, Drive, Volume) using either testbench or onboard hex displays and switching.</p> <p>2. Verification process for items 2-4:</p> <p>(a) Set Enable to 1b1.</p> <p>(b) Set the following: parameters Depth to 8hFF, Duty Cycle to 8d127, D_{In} to 24h7FFFFFFF.</p> <p>(c) Set the Rate parameter to 8d0.</p> <p>(d) Using testbench, verify that the pattern occurs in which $D_{Out} = D_{In}$ for 250 ms, and for 250 ms immediately afterward $D_{Out} = 24h000000$, repeating periodically.</p> <p>(e) Set the Rate parameter to 8hFF.</p> <p>(f) Verify using testbench that the modulation patterns frequency does not exceed 15 Hz (a period of 67 ms).</p> <p>(g) Set the Rate parameter back to 8d0, then adjust the Depth parameter to 8d127, keeping D_{In} and Duty Cycle at their current values.</p> <p>(h) Using testbench, verify that the pattern occurs in which $D_{Out} = D_{In}$ for 0.25 s, and for 0.25 s immediately afterward $D_{Out} = 24h3FFFFFFF$, repeating periodically.</p> <p>(i) Adjust the Duty Cycle parameter to 8h00.</p> <p>(j) Using testbench, verify that the pattern occurs in which $D_{Out} = D_{In}$ for 50 ms, and for 450 ms immediately afterward $D_{Out} = 24h000000$, repeating periodically.</p>	<p>1. Y</p> <p>2. Y</p> <p>3. Y</p> <p>4. Y</p>

Table 3 – continued from previous page

Requirement	Verification	Verification status (Y or N)
<p>1. FPGA Delay Unit</p> <p>(a) Setting Enable to logic-low bypasses the unit and setting Enable to logic-high sends the input signal through the unit.</p> <p>(b) The potentiometer-controlled parameter Time adjusts the amount of time delay (s) on the echoed signal, with a maximum of 3 s (20 beats per minute) delay time and a minimum of 50 ms (1200 beats per minute) delay time.</p> <p>(c) The potentiometer-controlled parameter Repeats adjusts the amount of times the original input signal is echoed, with a maximum of five repeats and a minimum of one repeat. If there are multiple repeats, each repeat following the first repeat is half the amplitude of the previous repeat.</p> <p>(d) The potentiometer-controlled parameter Mix adjusts the amplitude of the echoed signal(s) with respect to the original input signal, with a maximum of $Echo = D_{In}$ and a minimum of $Echo = 0$ (completely muted echoes). At $Mix = 8d127$, Echo has half the amplitude of D_{In}.</p>	<p>1. Verification process for item 1:</p> <p>(a) Set Enable to 1b0.</p> <p>(b) Set D_{In} to 24h7FFFFFFF.</p> <p>(c) Verify that $D_{Out} = D_{In}$ using either testbench or onboard hex displays and switching.</p> <p>(d) Set Enable to 1b1.</p> <p>(e) Verify that $D_{Out} \neq D_{In}$ and changes with 2 different sets of parameter values (of the set D_{In}, Drive, Volume) using either testbench or onboard hex displays and switching.</p> <p>2. Verification process for items 2-4:</p> <p>(a) Initialize Time to 8hFF, Repeats to 8h00, Mix to 8dFF and Enable to 1b1.</p> <p>(b) Send a 25 ms pulse of 24h7FFFFFFF to D_{In}, followed by 10 s of 24h000000.</p> <p>(c) Verify that a pulse of 25 ms length and amplitude 24h7FFFFFFF is echoed 5 seconds after the start of the initial pulse.</p> <p>(d) Adjust Time to 8h00 and rerun the simulation.</p> <p>(e) Verify that a pulse of 25 ms length and amplitude 24h7FFFFFFF is echoed 50 ms after the start of the initial pulse.</p> <p>(f) Adjust Repeats to 8hFF and rerun the simulation.</p> <p>(g) Verify that there a total of six nonzero pulses appear, each of 25 ms length, spaced 50 ms apart (positive edge to positive edge), with the first two pulses of an amplitude of 24h7FFFFFFF, and the four pulses that follow each half the amplitude of the one before it.</p> <p>(h) Adjust Repeats to 8h00, adjust Mix to 8d127, and rerun the simulation.</p> <p>(i) Verify that a pulse of 25 ms length and amplitude 24h3FFFFFFF is echoed 50 ms after the start of the initial pulse.</p> <p>21(j) Adjust Mix to 8d00, and rerun the simulation.</p> <p>(k) Verify that the initial pulse is not echoed.</p>	<p>1. Y</p> <p>2. Y</p> <p>3. Y</p> <p>4. Y</p>

Table 3 – continued from previous page

Requirement	Verification	Verification status (Y or N)
<p>1. FPGA Router and Software</p> <p>(a) The router can arrange the three digital pedals in all six possible permutations in the signal chain.</p> <p>(b) The router can bypass any of the three digital pedals and replace it with an analog export placeholder.</p> <p>(c) The user can operate the router and fulfill the two aforementioned requirements using the software menu.</p> <p>(d) The user can navigate to each digital pedal currently installed in the signal chain and adjust each pedals parameters using the three potentiometers.</p> <p>(e) When the user navigates away from a digital pedal in the menu, the pedal keeps its most recently set values for its parameters.</p> <p>(f) All of these requirements are visually observable on the FPGAs LCD display.</p>	<p>1. Verification process for item 1-3, 6:</p> <p>(a) Set D_{In} to 24h555555.</p> <p>(b) Navigate to the router using the software menu.</p> <p>(c) Arrange the three digital pedals in all six permutations. For each permutation, verify that each pedal is now in the correct place by using the footswitches to isolate each pedal in the chain and monitor its effect on the input signal.</p> <p>(d) Activate third-party export mode for one of the pedals using the software menu.</p> <p>(e) Verify that the router sends 24h555555 to the Data Out wire of the specified analog pedal slot in digital hardware.</p> <p>(f) Send 24h555555 into the Data In wire of the specified analog pedal slot in digital hardware.</p> <p>(g) Verify that the router receives 24h555555 and sends it along to the next digital pedal in the signal chain.</p> <p>(h) Repeat steps d-g for the other two analog pedal slots.</p> <p>2. Verification process for items 4-6:</p> <p>(a) Navigate to one of the digital pedals parameters using the software menu.</p> <p>(b) Set each parameter to a specific value using the potentiometers.</p> <p>(c) Exit the parameter screen in the software menu.</p> <p>(d) Verify that the values entered in the parameter screen are equal to the parameters currently passed to the parameter inputs on the pedal in digital hardware.</p> <p>(e) Repeat steps a-d for the other two digital pedals.</p>	<p>1. Y</p> <p>2. Y</p> <p>3. N</p> <p>4. N</p> <p>5. N</p> <p>6. N</p>

Continued on next page

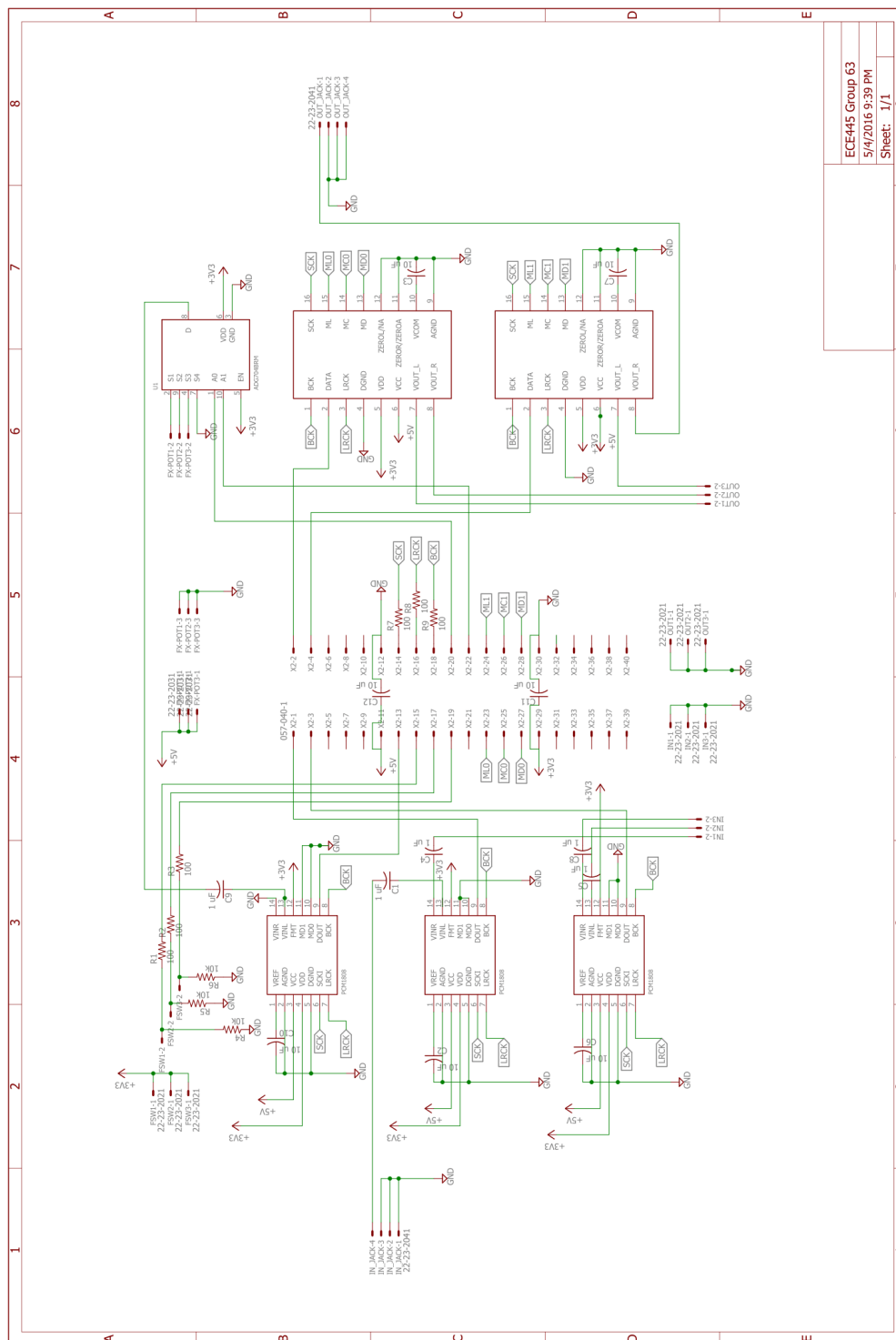
Table 3 – continued from previous page

Requirement	Verification	Verification status (Y or N)
<p>1. ADCs</p> <p>(a) Ensure that a 1 kHz, 2V peak to peak sine wave is encoded in PCM Left Justified format as digital highs of greater than 3.0V and digital lows of less than 0.2 V.</p> <p>(b) Ensure the latency of an audio signal is below 10 ms.</p>	<p>1. Verification process for item 1:</p> <p>(a) Hook the ADC to a function generator.</p> <p>(b) Hook the ADC up to a logic analyzer.</p> <p>(c) Give 3.3V into the V_{DD} input.</p> <p>(d) Give 5.0V into the V_{CC} input.</p> <p>(e) Turn the function generator on, and select a 1 kHz sine wave with 2V peak to peak.</p> <p>(f) Turn the logic analyzer and oscilloscope on.</p> <p>(g) Ensure that the output to the logic analyzer is that of a 20 Hz sine wave in PCM Left Justified format with high values of greater than 3.0V and low values of less than 0.2V.</p> <p>2. Verification process for items 2:</p> <p>(a) Hook the ADC to a function generator.</p> <p>(b) Hook the the function generator to the oscilloscope.</p> <p>(c) Hook the ADC output to the oscilloscope.</p> <p>(d) Give 3.3V to the V_{DD} input</p> <p>(e) Give 5.0V to the V_{CC} input.</p> <p>(f) Turn on the function generator, and select a 1 kHz sine wave with 2V peak to peak.</p> <p>(g) Turn the oscilloscope on.</p> <p>(h) Measure the delay between a peak on the sine wave and a packet with all logic high, and ensure it is below 10 ms.</p>	<p>1. Y</p> <p>2. Y</p>
Continued on next page		

Table 3 – continued from previous page

Requirement	Verification	Verification status (Y or N)
<p>1. DACs</p> <p>(a) A digitally encoded 1 kHz sine wave in PCM with digital highs of greater than 3.0V and digital lows of less than 0.2V is converted into an analog 1 kHz sine wave with 2V peak to peak.</p> <p>(b) Ensure the latency between input and output is less than 10 ms.</p>	<p>1. Verification process for item 1:</p> <p>(a) Hook the DAC up to an oscilloscope.</p> <p>(b) Hook the DAC up to the FPGA development board.</p> <p>(c) Give the DAC 3.3V in the <i>VDD</i>.</p> <p>(d) Give the DAC 3.3V in the <i>VCC</i>.</p> <p>(e) Power the FPGA development board and write a test program that sends out a digitally encoded 1 kHz sine wave in PCM Left Justified format with digital highs of greater than 3.0V and digital lows of less than 0.2V.</p> <p>(f) Observe the output of the DAC on the oscilloscope and ensure the sine wave is 2V peak to peak and 1 kHz.</p> <p>2. Verification process for items 2:</p> <p>(a) Hook the DAC to an oscilloscope.</p> <p>(b) Hook the DAC up to the FPGA development board.</p> <p>(c) Hook the FPGA development kit to the oscilloscope.</p> <p>(d) Give 3.3V to the DAC <i>VDD</i>, and 5.0V to the DAC <i>VCC</i>.</p> <p>(e) Power the FPGA development board and write a test program that sends out a digitally encoded 1 kHz sine wave in PCM Left Justified format.</p> <p>(f) Measure the delay between the first packet sent by the FPGA and the beginning of the DAC output, and ensure it is below 10 ms.</p>	<p>1. Y</p> <p>2. Y</p>

Appendix B Full Circuit Schematic



ECE445 Group 63
5/4/2016 9:39 PM
Sheet: 1/1