# Multi-Effect Guitar

Electric guitar system with practice amplifier, distortion and reverb effects

April 12, 2016

## ABSTRACT

In this project, we designed and implemented a digital signal processor, and a practice amplifier for a guitar. This allowed us to explore both the digital and analog domain of signal processing. Most of this design is hardware intensive. We utilized Altera’s DE2 board to handle the digital side of things, and Texas instrument’s LM384N 5-Watt audio amplifier to handle the analog signal processing. The amplifier boosted the signal coming out of the DE2 to a reasonable playing volume. The guitar included in this design was an ESP LTD EC-400 electric guitar. The signal processor housed two main components, the distortion effect and the reverberation effect. Each effect could be applied by simply asserting the switches responsible for applying the effect. The amount of distortion could also be manipulated by incrementing or decrementing the effect via push buttons onboard the DE2. The audio amplifier received its input from the line out on the DE2 and boosted the signal by an internally fixed gain of 34 [dB]. To acquire the data stream coming from the guitar, we took advantage of the active pickups which output a signal strong enough for the DE2 to process.

## TABLE OF CONTENTS

## FUNCTIONAL REQUIREMENTS

The multi-effect guitar project enables the user to apply audio effects such as distortion and reverb to audio data produced by an electric guitar. The project also includes a practice amplifier to allow the user to listen to the processed audio data. The effects can be turned on and off by using the switches on the DE2. Only one effect can be turned on at a time. The distortion effect is made configurable to allow the user to change the level of distortion applied by utilizing push buttons on the Altera DE2 to increment and decrement the distortion level. Incrementing the distortion level decreases the clipping threshold allowing for a harsher, more prominent distortion. Gain levels are also increased to accommodate for the power loss presented by the clipping technique. On the other hand, reverb has one set configuration (delay of 46 [ms]) with a fixed gain of 0.75. The delay line used for reverb was implemented using the SDRAM off-chip memory.

The audio effects are immediately applied as soon as the switch is turned on. If there is audio data coming in from the guitar, the effect will be noticeable. We implemented these effects in hardware to reduce the latency as much as possible(because hardware is faster). This enables us to perform real-time digital signal processing without introducing audio latency to the output which is important when playing an instrument. The multi-effect guitar uses 44.1 kHz sampling rate with a 16-bit resolution.

All of the functional requirements described above were implemented successfully. On the other hand, the original functional requirements included more features including wireless audio data transmission, live streaming and a guitar tuner.

The wireless audio data transmission was implemented using the Sparkfun Purpletooth Jamboree configured as the master and Bluetooth Audio Breakout configured as the slave which uses the BC127 bluetooth chip. We were successful in configuring the bluetooth boards and getting them to pair with each other. The slave was originally intended to transfer audio data to the DE2 using the I2S protocol but was replaced with tapping to the Tx/Rx lines of the bluetooth slave and soldering it to a 3.5mm audio cable instead which can be inserted into the LINE IN port of the DE2. This was done because there was no clear benefit of using the I2S block to implement the data transfer. The bluetooth feature met its demise when the Bluetooth Audio Breakout (slave) broke causing the audio playback to be cut-off after playing for a couple of seconds. Due to time constraints and product availability, the bluetooth feature did not make it to the final production.

The live audio streaming uses client-server architecture to communicate audio data via TCP sockets. The server uses icecast which is a popular internet radio solution that supports a number of platforms (windows, mac osx, linux, android, ios, etc) and languages (C, Java, Javascript). Icecast can also handle load balancing and timing issues. It acts as a middle-man between one source client (which is broadcasting the data), and one or more sink clients. In our case, the DE2 was meant to act as a source client providing the audio stream; and our node.js web application was to act as the sink.

Our web app worked as intended (it could connect and rebroadcast to any icecast source), however, our source client (the DE2) did not. The source was able to send audio data to the server, but at an incorrect bit-rate (and possibly incorrect endianness).

Our UART was supposed to be used to configure and monitor the status of the receiving bluetooth module. The uart successfully communicated with the bluetooth module, and was able to send commands but not receive them properly (we did however receive data consistently). We tested this by using an arduino to confirm that the command sent from the DE2 actually took. We had to use an arduino as a backup to configure the modules. The automatic pairing worked for brief periods of time. In the end, the receiver would allow a connection, but then suddenly kick out.

## DESIGN AND DESCRIPTION OF OPERATION

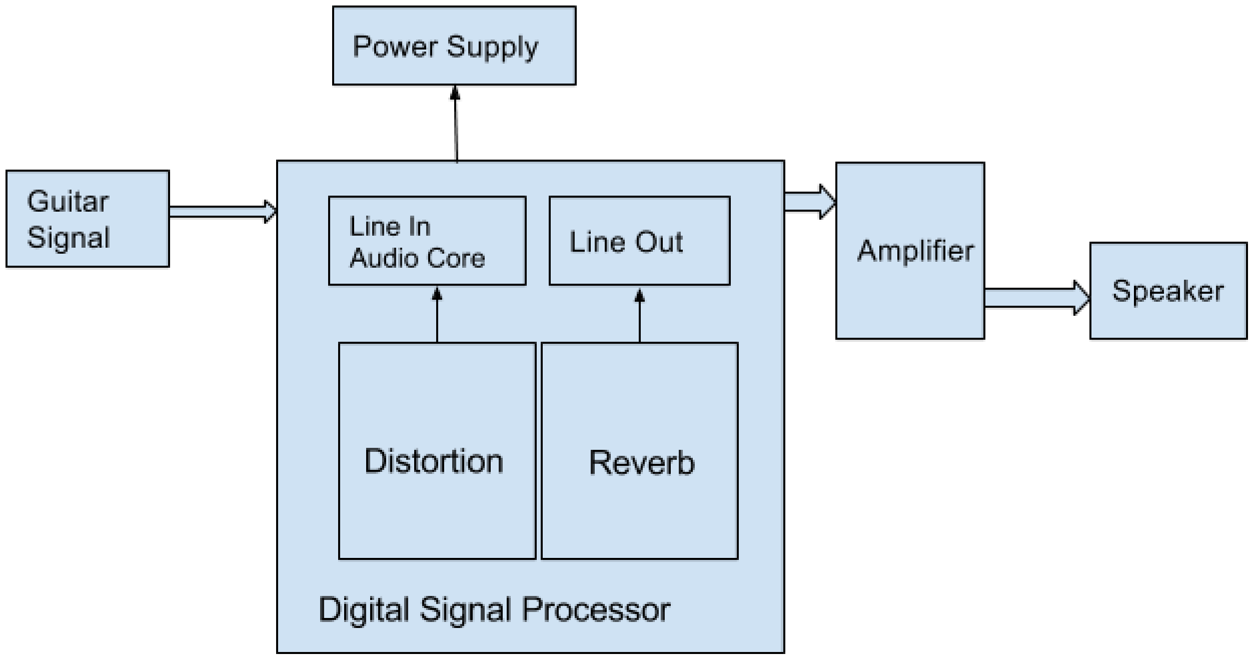


Diagram 1: Hardware Block Diagram of the project

### Design

The most important part of our design is signal handling. In this project, the incoming data source was the audio stream coming from the guitar. This particular guitar uses active pickups which convert the compressional sound waves generated from the guitar to an analog voltage signal. This signal was then fed into the Altera DE2 board via the 3.5 mm auxiliary cable. Once inside the board, the data was processed by both custom and stock hardware components housed on the FPGA. After the signal was processed it was then piped out via the line out on the DE2. The analog signal from the line out was then fed into the amplifier input using a 3.5 mm jack. The amplifier linearly amplified the input signal. This signal is then sent to the speakers to convert the voltage signal back into compressional (audio) sound waves.

### Multi-Effect Guitar

Digital Signal Processor

- **Distortion** - Distortion was implemented using hard clipping. The audio samples are clipped at a constant value. This produces a clamped waveform which closely resembles a square wave. This effect introduces higher frequency components, and is heard by the listener as distorted sound. The implementation of the distortion component was coded in VHDL. The component compares the absolute value of each audio sample (which represents an amplitude value) with some constant clipping threshold. If the absolute value of the audio sample is larger than the clipping threshold, the samples is clipped to that threshold value. Distortion is enabled by switching SW(13) ON. It is also made configurable by giving the user the ability to increase how much distortion is applied using KEY(1) to increment and KEY(3) to decrement. The configuration values can be found in the datasheet below.

- **Reverb** - The design for reverb was implemented by using an infinite impulse response filter along with attenuation and delay lines. It's enabled by toggling SW(14). Essentially, the input audio passes through a Comb Filter which attenuates and delays the signal. The output of the filter is then added back to its input which results in a decaying and delayed stream of echoes. In order to perform delay, the incoming audio is stored in memory using the SDRAM. More specifically, the SDRAM was implemented using a circular buffer with multiple taps (read pointers). The current implementation leverages the entire 8 [MB] of SDRAM.

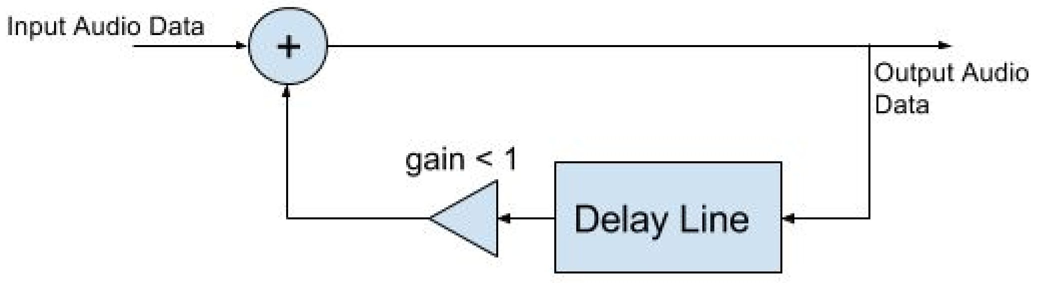


Diagram 2: Comb Filter

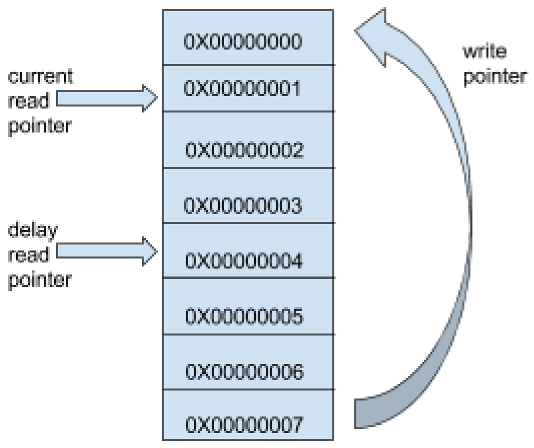


Diagram 3: Circular Buffer

Each audio sample takes up two address spaces(byte-addressable memory). For our implementation of reverb, we utilized two read pointers, one for the original signal and the other for the delay. For a 46 ms delay, it was found that the delay read pointer should be incremented by 4096 after every read cycle. This is described in the calculation below:

Delay[s] = (Buffer Size[bytes] / Sampling Rate[hz] ) / Increment [bytes]

= ( 8388608 / 44100 ) /4096

= 46 [ms]

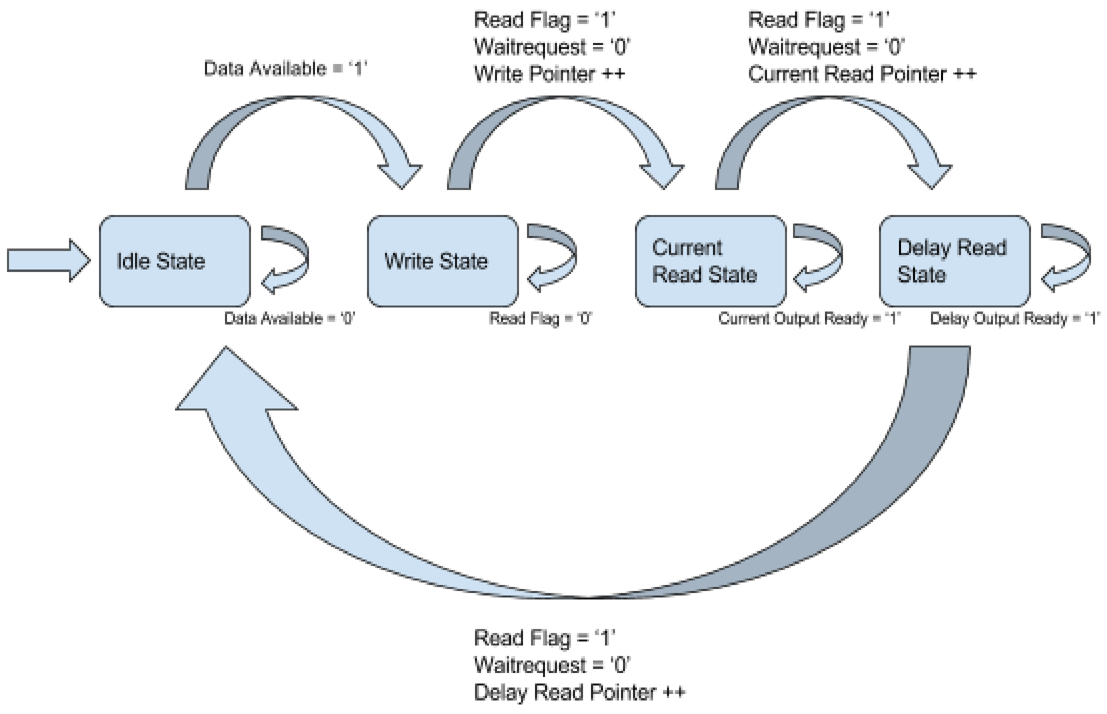
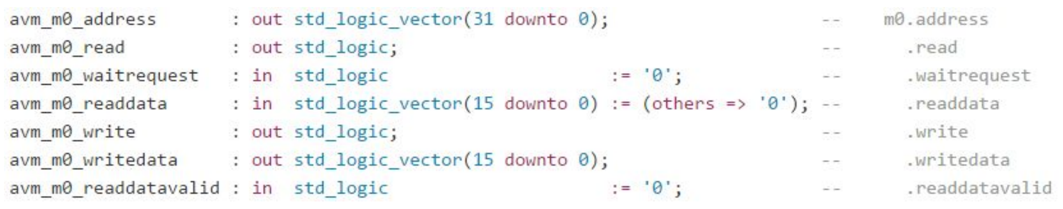


Diagram 4: SDRAM State Machine

A custom component was created in order to interface with the SDRAM along with the SDRAM controller in Qsys. The following signals were included in the custom component:



```
avm_m0_address       : out std_logic_vector(31 downto 0);                       --    m0.address
avm_m0_read          : out std_logic;                                           --      .read
avm_m0_waitrequest   : in  std_logic                     := '0';                --      .waitrequest
avm_m0_readdata      : in  std_logic_vector(15 downto 0) := (others => '0'); --      .readdata
avm_m0_write         : out std_logic;                                           --      .write
avm_m0_writedata     : out std_logic_vector(15 downto 0);                       --      .writedata
avm_m0_readdatavalid : in  std_logic                     := '0';                --      .readdatavalid
```

Diagram 5: Required SDRAM signals

## Analog Amplifier

The design around the amplifier used the LM384N Texas instruments audio analog chip. This particular IC has several desirable characteristics. Firstly, the input impedance of the LM384N is 150 [kΩ]. A high input impedance places very little loading on the input voltage signal. Thus, the guitar signal retains its signal integrity. Secondly, the LM384N supports a wide operating voltage range ( between 12 and 26 [V] ) making it compatible with all types of power supplies. Thirdly, the chip's output quiescent voltage is fixed to half of the supply rail (no need to bias the circuit). This allows the output AC signal to take advantage of the full swing without introducing too much distortion (clipping). Finally, the LM384N's gain is internally fixed at 34 [dB] (which is roughly equal to 50 [V/V]).

The actual design followed a very similar approach to the application circuit provided in the datasheet [6]. The operational amplifier was configured using the inverting topology. Several of the pins were set to ground, including the plus side of the op-amp. Capacitors were added for different reasons. The capacitor added between the power supply and ground is used as a bypass capacitor to prevent the high frequency components generated by the line voltage from influencing the performance of the amplifier IC. The capacitor at the input of the negative side of the op amp is used as a DC decoupler, making the op-amp and the music source independant and blocked. The final capacitor added to the output of the amplifier is used both as a decoupler (to prevent the speakers from receiving any DC) as well as a high pass filter to help attenuate the line voltage hum. The filter consists of the load (two 8 ohm speakers connected in parallel) and the capacitor described above. The cutoff frequency was determined to be 84 [Hz]. The last component added into the amplifier was a potentiometer, which controls the input signal's power (i. E. volume control). Part of the design included building the circuit on the breadboard to prototype and test to make sure that it worked as desired. The final stage of the design consisted of laying out and routing the components onto a vector board to build a permanent solution. A heat sink was incorporated into this design because the power output to the load caused the chip to heat up significantly. Thermal paste was added between the chip and the aluminum heat sink to help with the transfer of heat.

## BILL OF MATERIALS

| Part | Cost | Quantity | Part Number | Link |
|---|---|---|---|---|
| 3.5mm Male to 3.5mm Male Audio Cable | $0.97 | 2 | DX-DMP354 | http://bit.ly/1ViSlWi |
| 6.3 to 3.5mm Audio Connector | $7.99 | 1 | NS-HZ307-C | http://bit.ly/1qkpM3k |
| Altera/Terasic DE2 development board | $412.97 | 1 | P0301 | http://bit.ly/1PioZbG |
| Speaker Visaton FRS 7 S | $31.46 | 2 | 364-3357 | http://bit.ly/1VkgAcD |
| 1000 microfarad electrolytic capacitor | $0.35 | 1 | COM-08982 | http://bit.ly/1qkrp0V |
| 4.7 microfarad electrolytic capacitor | $0.12 | 1 | R4.7/50 | http://bit.ly/1qLxd4i |
| 470 microfarad electrolytic capacitor | $0.56 | 1 | EKMG250EMC471MJC5S | http://bit.ly/1qEYeWM |
| Vector Board | $2.00 | 1 | N/A | N/A |
| Wire | $0.11/ft | 3 ft | N/A | http://bit.ly/1yY46fV |
| Electric (Active) guitar ESP LTD EC-400 | $600 | 1 | N/A | http://bit.ly/22qldR7 |

## AVAILABLE SOURCE

### Altera Audio Core

Altera's audio core provides a simple interface to interact with the audio codec on the Altera DE2 board [5].

- source size:  507 KB [6]
- performance: this core should have acceptable performance (it specifically designed for this board) [5]
- resource requirements:
    - 235 Cyclone II FPGA logic elements [5]
    - extra clock pll [5]

## DATASHEET

### Audio Properties

| | Bit-depth |
|---|---|
| Input Audio (LINE IN) | 16-bit |
| Output Audio (LINE OUT) | 16-bit |

| | |
|---|---|
| Audio Sampling Rate | 44.1 kHz |
| Guitar Output Voltage (clean bass E-string) | ~ 100 mV |
| Guitar Output Voltage (through DE2, bass E) | ~ 1.30 V |

Distortion Configurations

| Level | Clipping Threshold* | Gain* | Output Voltage [mv] $^{+}$ |
|---|---|---|---|
| 1 | 3000 | 1 | 740 |
| 2 | 1900 | 2 | 640 |
| 3 | 1300 | 2 | 520 |
| 4 | 700 | 3 | 440 |

| | | | |
|---|---|---|---|
| 5 | 300 | 7 | 440 |

* As the clipping threshold gets lower, the distortion effect produces harsher audio output. The gain multiplier is required as the distortion level increases due to the power loss from clipping the signal amplitudes in the process which helps mitigate the volume from dropping.

+ The output voltage for distortion was measured by using the bass E-string.

### Delay Configurations (Reverb)

| Output Voltage | Delay (ms) | Gain* |
|---|---|---|
| ~ 1.10 V | 46 | 0.75 |

* The gain is set to a value less than 1 in order to slowly dampen the delayed echoes generated.

## Electric Specifications

### DE2

| | Voltage [V] DC | Current [A] DC |
|---|---|---|
| Peak | 8.89 | 0.510 |
| Idle | 8.89 | 0.510 |

### Practice Amplifier

| Description | Datasheet |
|---|---|
| Input Impedance | 150 [kΩ] |
| Input Voltage | 12-26 [V] |
| Voltage pk-pk (AC) | 5.6 [V] |
| Idle Current Draw | 0.01 [A] |
| Typical Current Draw | 0.210 [A] |
| Power Draw | 1.176 [W] |

### Operational Temperature

| | Temperature [Degrees Celsius] |
|---|---|

| | |
|---|---|
| LM384N IC Chip | -40 to 85 |

## I/O Signals

| | |
|---|---|
| SW(13) | Enables Distortion |
| SW(14) | Enables Reverb |
| KEY(1) | Distortion Level Increment |
| KEY(2) | Distortion Level Decrement |
| AUDIO_IN | Analog audio input from the guitar to the audio codec in the DE2 (Audio codec performs ADC) |
| AUDIO_CORE_IN | Digital audio input from the audio codec to the audio core. |
| DSP_IN | Digital audio input from the audio core to the DSP (Utilizes Avalon ST) |
| MUX3X1_OUTPUT | Output data from the MUX3x1 depending on the effect selected. MUX selects from DISTORTION_OUT or REVERB_OUT to output to DSP_OUT |
| MEMORY_SINK_DATA | Output audio to the SDRAM from the DSP (Used as RAM storage for the reverb component) |
| MEMORY_SOURCE_DATA | Input audio from the SDRAM to the DSP (Used by the reverb component for delay) |
| REVERB_BUFFER_OUT | Output audio into the DSP after processing from the reverb component |
| REVERB_OUT | Output after adding REVERB_BUFFER_OUT and DSP_IN to mix audio samples for reverb effect |
| DISTORTION_OUT | Output audio in the DSP after processing from the Distortion component |
| DSP_OUT | Output from the DSP to the Audio Core (Effects applied may vary depending on the effect selected using the MUX3x1. Utilizes Avalon ST) |

| | |
|---|---|
| AUDIO_CORE_OUT | Output from the audio core to the audio codec |
| AUDIO_CODEC_OUT | Output from the audio codec after performing DAC |
| AMP_OUT | Output analog audio from the practice amplifier |

The measurements described in this section were performed by using the DE2 power measurement wiring harness provided by Nancy.

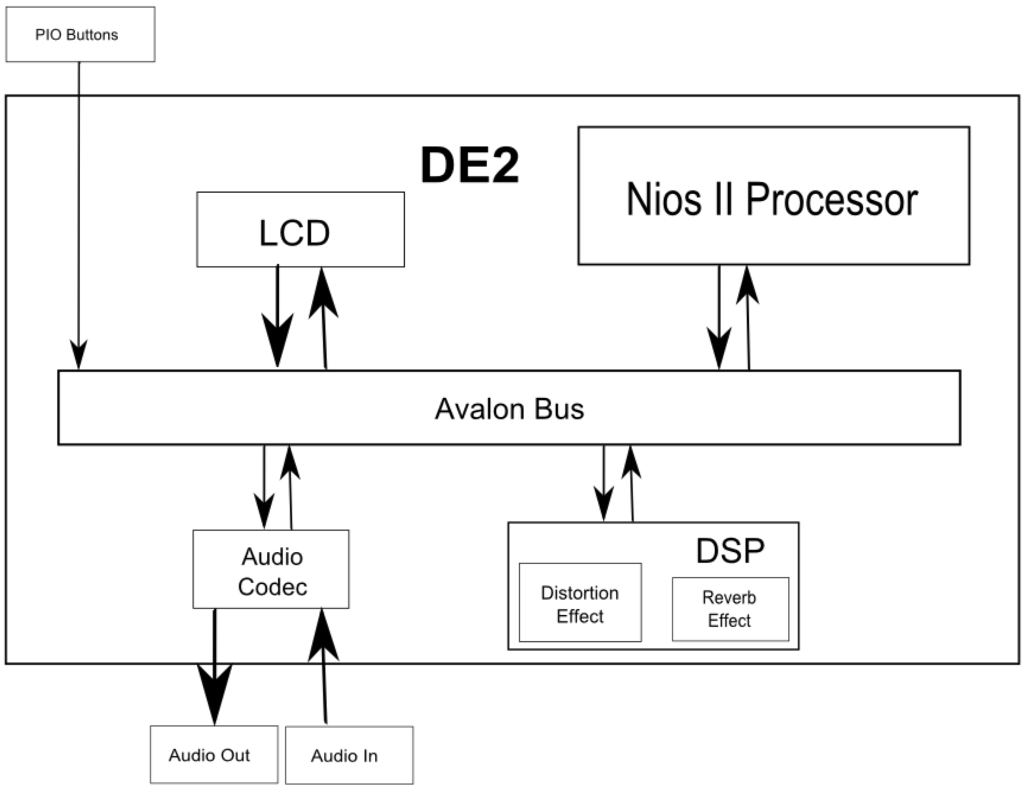


Diagram 6: Hardware Block Diagram

## BACKGROUND RESEARCH

### Implementation of Tuner

The research on the tuner component used the “A digital guitar tuner” article written by Mary Lourde and Anjali Kuppayil Saji at the department of electrical and electronics engineering UAE.[29] This article encompasses the entire design of the guitar tuner. It discusses how to relate frequency to the corresponding notes, and gives suggestions to improve the accuracy of the tuner.[29]

### Pitch Determination of human speech

The research on the tuner also utilized a peer reviewed article which was presented at the symposium of computer processing in communications, polytechnic institute of Brooklyn, April 8, 1969. The article entitled “Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate.” written by A. Michael Noll at bell telephone laboratories helped with some of the mathematics in determining pitch.[34] It provided three methods to accomplish the task of pitch determination.

### Implementation of Distortion, Echo and Reverb

The peer-reviewed article by Rulph CHassaing and Donald Reay titled “Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK” was utilized the implement the current distortion and reverb effects that the project has. Essentially, it was mentioned that distortion was one of the easiest to implement which can be simply done by overamplifying each sample and clipping it at maximum and minimum values [26]. As for reverb, it was mentioned that the buffer size is responsible for providing the length of the delay in which a dynamic change of the echo length leads to a reverb effect [26].

## SOFTWARE DESIGN

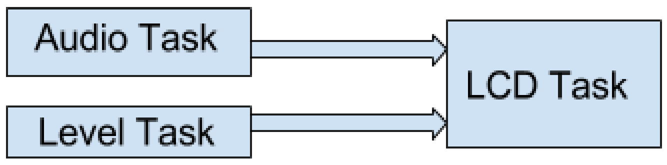


Diagram 7: Software processes

The software consists of an LCD task, a level task and an audio task. The audio task monitors the states of the switches on the DE2 and posts that information to the LCD task (via a queue). For example, if switch 4 is asserted, reverb will be displayed on the LCD screen.

The Level tasks adjusts the clipping level (when distortion is enabled) based on user input received via pushbuttons. The Level Task is interrupt-driven (interrupts are triggered when a button is pushed).

## TEST PLAN

### Hardware:

Testing begins at the source. The output of the guitar's analog signal is to be captured by the oscilloscope to verify its integrity. The ways planned to test the signal is by changing the style of play, (strumming, finger picking, tapping, etc) and changing its frequency and selecting the signal that outputs the weakest signal and work with that. In the event that the signal is too degraded or weak for the DE2 to handle, a preamplifier may be incorporated into the design. The next hardware component on the list to test is the audio core. This will be done by connecting up both the line in and line out to an audio source and speakers respectively to verify that the audio core works as a standalone unit. At this point the next piece of hardware to test is the FPGA implementation of the DSP. The testing will include testbench simulations both in modelsim and matlab.

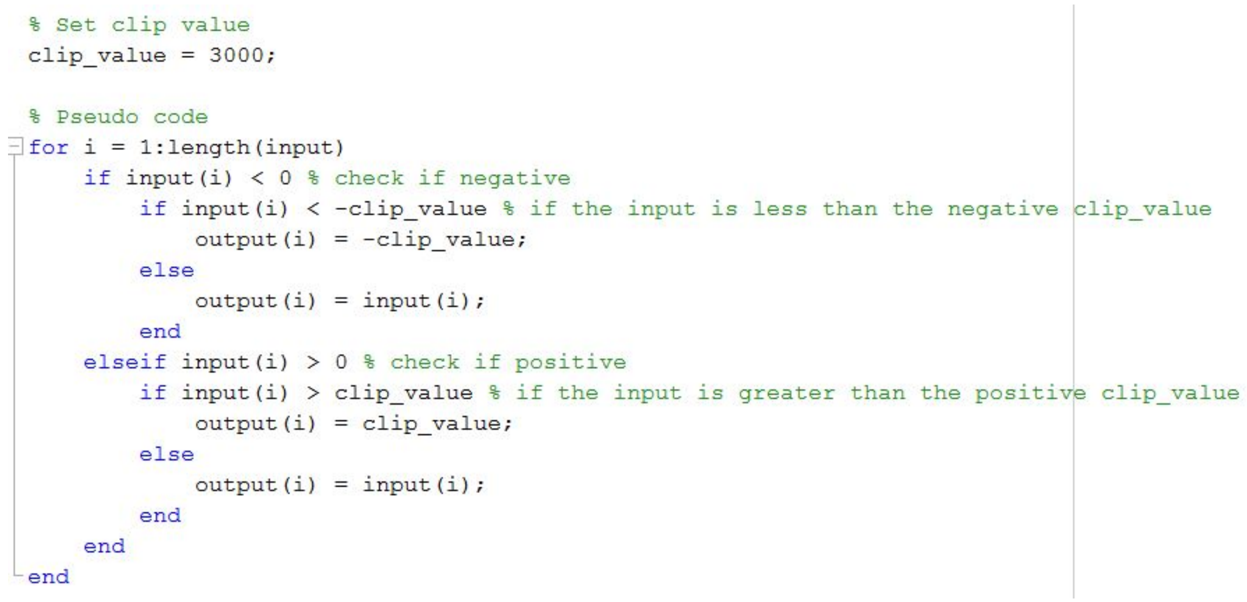
### Software:

This project is heavily dependant on the hardware side. The most important hardware dependant software tasks includes a level task and an audio task. As level task depends on the pushbuttons, we first made sure that the interrupts fired correctly when the buttons were pressed. Next we made sure that the gain was incremented or decremented according to which button was pressed. Finally, we made sure that the gain value was properly bounded keeping the value with 1-5. The audio task was tested to make sure that it responded quickly enough to update the LCD screen with no noticeable lag.

## RESULTS OF EXPERIMENTS AND CHARACTERIZATION

### Distortion Experiment

- The clipping method performed in this project is hard-clipping in which odd-ordered harmonics pushes the input signal into a more square-shaped wave. Since we're clipping the signal at a certain threshold level, producing flat peaks in the signal, we're introducing odd-ordered harmonics.
- This way, we're producing low and high frequency distortion causing a harsher produced sound.
- This shows the simulation of the hard signal clipping in order to produce a distorted waveform.
- In this simulation, a sample wav file was read in Matlab. The sample rate is 44.1KHz, 16-bit resolution using 88200 samples.
    - The amplitude clipping threshold was set to 3000. This produces a less harsh distortion effect. This was done in order to distinguish the clipping in the plots.
    - The MatLab algorithm used is displayed below



```
% Set clip value
clip_value = 3000;

% Pseudo code
for i = 1:length(input)
    if input(i) < 0 % check if negative
        if input(i) < -clip_value % if the input is less than the negative clip_value
            output(i) = -clip_value;
        else
            output(i) = input(i);
        end
    elseif input(i) > 0 % check if positive
        if input(i) > clip_value % if the input is greater than the positive clip_value
            output(i) = clip_value;
        else
            output(i) = input(i);
        end
    end
end
```

Diagram 8: MatLab distortion algorithm

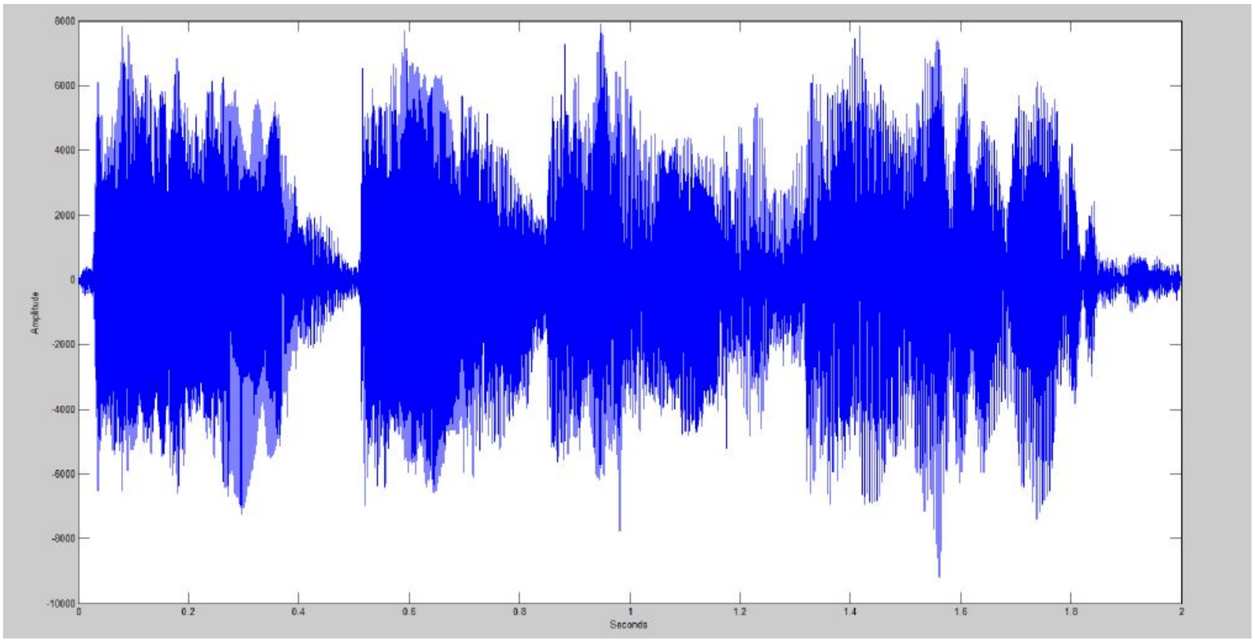


Diagram 9: Clean Audio Wav File [18]

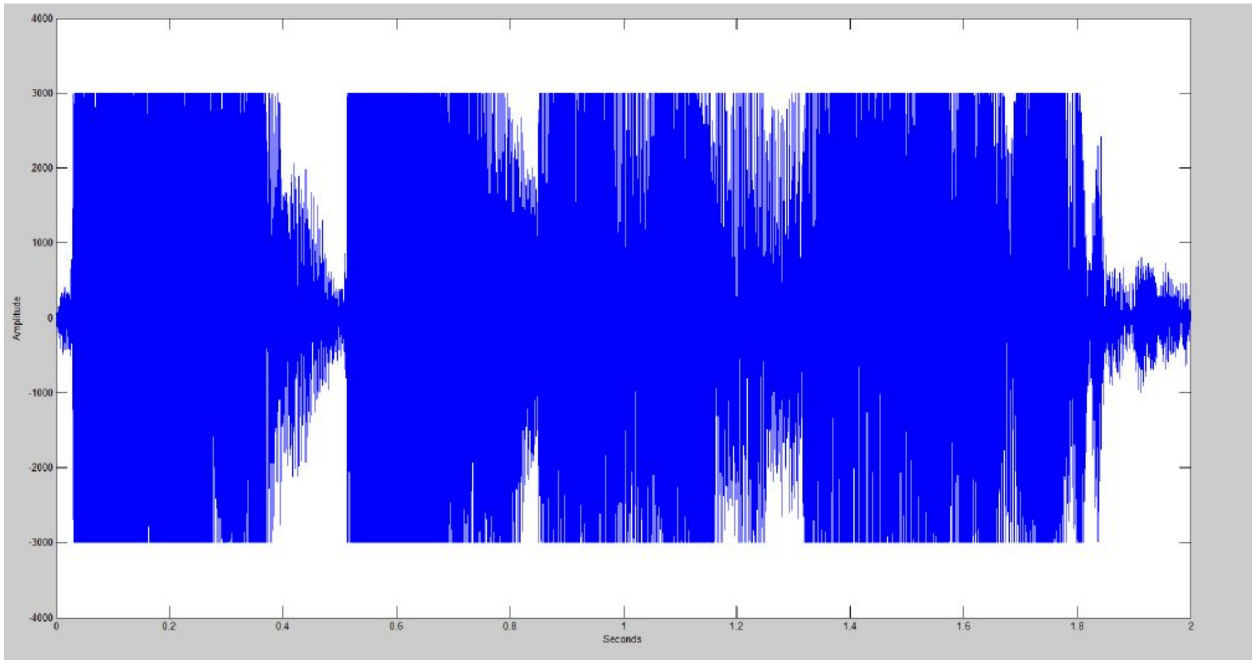


Diagram 10: Distorted Audio Wav File, clip threshold set at 3000

- The hardware simulation was performed in ModelSim using a VHDL component and testbench.

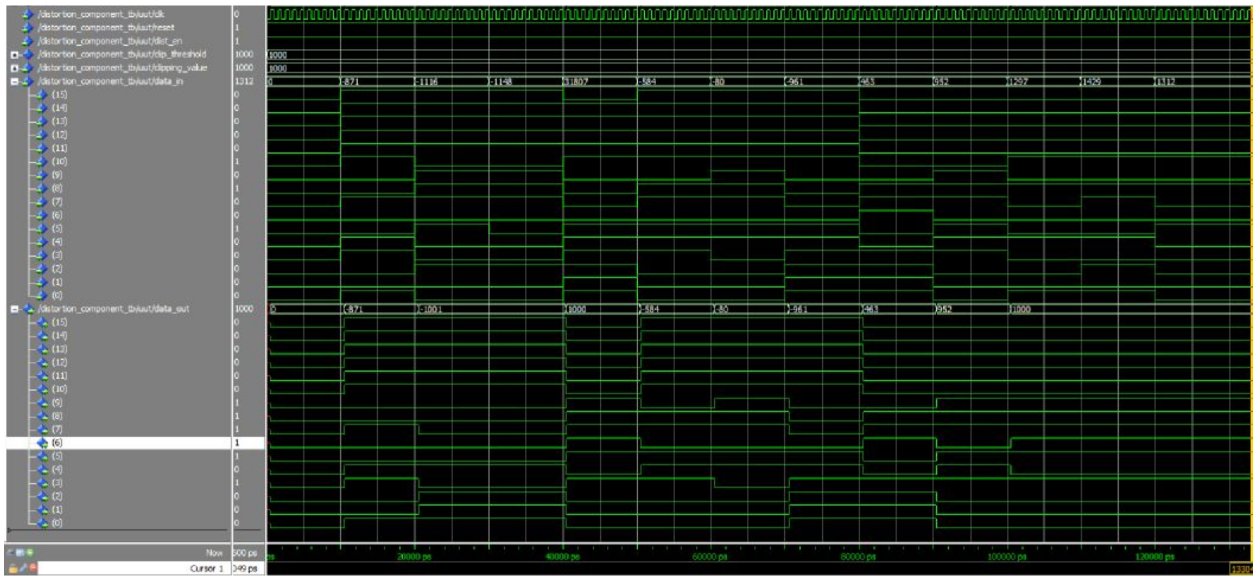

/distortion_component_tb/uut/clk
/distortion_component_tb/uut/reset
/distortion_component_tb/uut/dist_en
/distortion_component_tb/uut/clip_threshold
/distortion_component_tb/uut/clipping_value
/distortion_component_tb/uut/data_in
/distortion_component_tb/uut/data_out
Now
Cursor 1

Diagram 11: Distortion VHDL Testbench and Simulation

## Reverb Experiment

- The implementation of reverb utilizes an infinite impulse response filter that includes delay and gain.
- The audio sample must be multiplied by the gain at a value less than 1 and delayed in order to produce dampened echoes.
- The echoes generated are added to the current input signal.
- The experiment illustrated below shows how an impulse generate its impulse response when passed through the filter.
- We chose delays less than 50 ms because delays greater than 50 ms can be heard as actual distinct echoes. [24]
- Algorithm used for Reverb simulation in MatLab, where “test1” is the input signal and output is the output signal.

```
output(n) = 0.8 * test1(n) + test1(n - N) - 0.8 * output(n - N);
```

Diagram 12: Reverb algorithm in MatLab

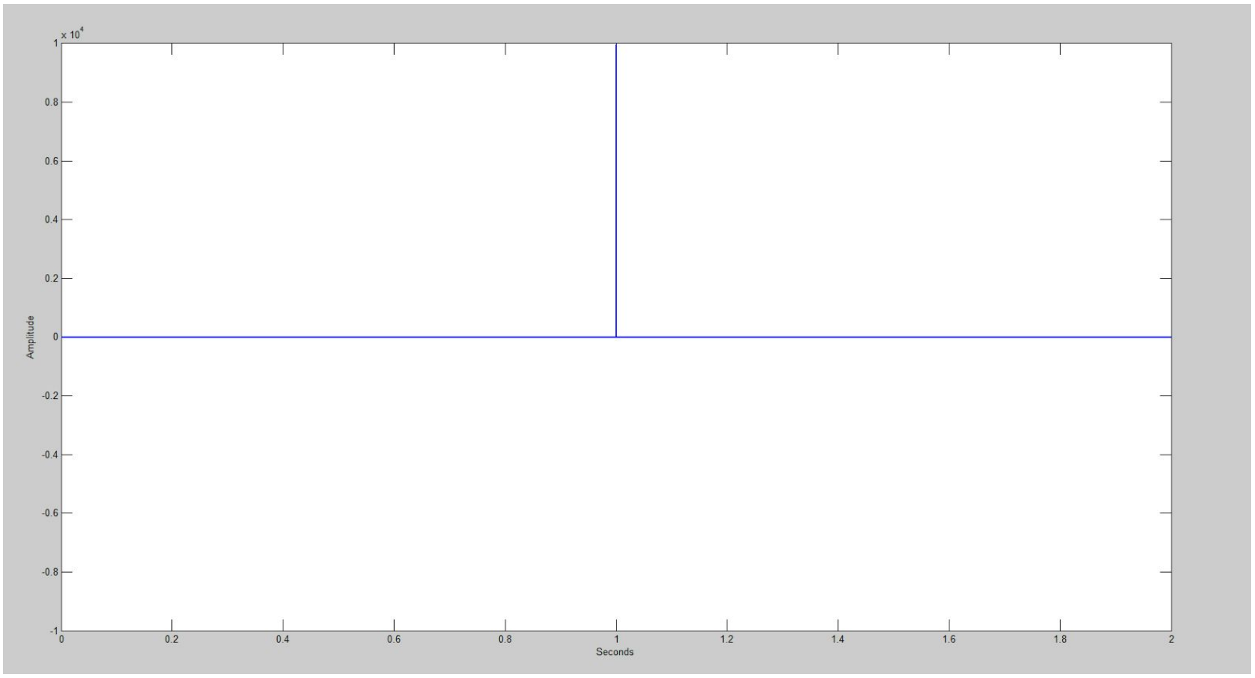


Diagram 13: Clean impulse [25]

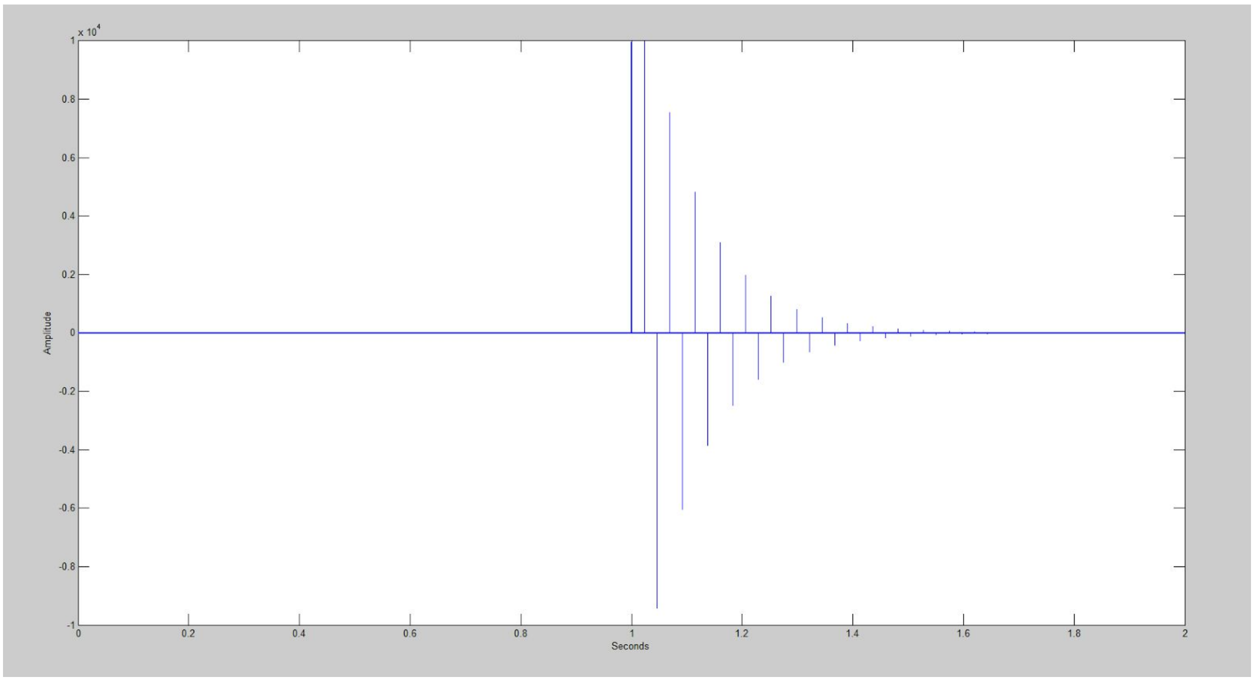


Diagram 14: Reverb impulse response

- Reverb was also simulated in VHDL, the “buffer_out” signal is multiplied and divided to implement a 0.75 gain. “Div_result” is the end result of the operation which is then added to “data_in” to produce “data_out”. The delay is caused by waiting for the RAM to be filled up before it start outputting what’s stored in it.
  - The RAM (buffer) size used here is 4 and the bit-depth is 16.
  - We can see that “data_out” becomes much larger by the end due to the addition of signals performed. The result of this addition implements the feedback required to produce reverb.

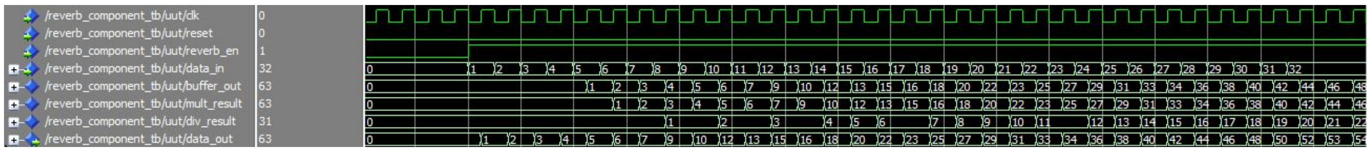

/reverb_component_tb/uut/clk 0
/reverb_component_tb/uut/reset 0
/reverb_component_tb/uut/reverb_en 1
/reverb_component_tb/uut/data_in 32
/reverb_component_tb/uut/buffer_out 63
/reverb_component_tb/uut/mult_result 63
/reverb_component_tb/uut/div_result 31
/reverb_component_tb/uut/data_out 63

Diagram 15: VHDL simulation of reverb

## SAFETY

The voltage required to power the components of this project vary. The DE2 board uses a wall output DC voltage of 9 [V] and the amplifier uses the laboratory DC power supply with input voltage of 12 [V]. Both of these supplies are considered to be low voltage, leaving minimal risk to the user. There is minimal risk of damaging the listener’s eardrums, because the pain threshold is above the possible output power. In terms of failure modes, they aren’t needed because if the project fails nothing will cause damage of property or lives. The intended operating temperatures are between 0-70 degrees celsius as per the datasheet.

## REGULATORY AND SOCIETY

For the BC127 bluetooth module use is allowed as long as a radiator is not incorporated into the design. It has been tested and granted approval under industry Canada (IC) radio standard specification (RSS). This certification is applicable in Canada. [28] As long as the part does not interfere with the other radio signals, and accepts any interference use is allowed.

This design includes an internet connection to an audio server to stream live music coming from the guitar. By exposing the DE2 to the outside world makes the system insecure. For instance, if a client sends too many bogus requests the DE2 might cause memory leaks, buffer overruns, etc.

## ENVIRONMENTAL IMPACT

The Multi-Effect Guitar may require some components to be soldered using tin-lead solder. This application was performed on the practice amplifier to build the custom circuit on a vector board. However, the application of tin-lead solder used in printed circuit boards are exempted from the requirements of Article 4(1) in the RoHS 1 directive, found in page 5. [20]

The Altera DE2 board falls under the RoHS-compliant devices which is included as part of the Cyclone series FPGAs. Altera is known to maintain the most RoHS-compliant products in the industry. [22]

The only foreseeable environmental impact of this project is its disposal. In order for this project to have negligible negative environmental impact, the components involved in the project including the LEDs, batteries, PCBs, cabling and microchips must be disposed of properly to electronics recycling facilities.

## SUSTAINABILITY

We based our calculations on the power adapter and included a fudge factor of 30 %. Also, approximately 38 % of the FPGA is utilized, so again we multiplied the value by .38 to approximate the power consumption.  The voltage used for the power calculation is 3.3 [V]. The estimates for the bluetooth modules were found by looking through the datasheet.

I (100 % FPGA) = 1.3 A * 0.7 = 0.91 [A]

I (38 % FPGA components) = I(100 % FPGA) * 0.38 = 0.346 [A]

Here is a table listing power consumption for individual components.

Table 1: Bluetooth (TX)

| | | | |
|---|---|---|---|
| Active | 54 [mW] | 75 % | 40.5 [mW] |
| Idle | 3.6 [mW] | 25 % | 0.9 [mW] |
| | | Total | 41.4 [mW] |

Table 2: Bluetooth (RX)

| | | | |
|---|---|---|---|
| Active | 49.5 [mW] | 50 % | 24.75 [mW] |
| Idle | 3.3 [mW] | 50 % | 1.65 [mW] |
| | | Total | 26.4 [mW] |

Table 3: DE2

| | | | |
|---|---|---|---|
| Active | 346 [mW] | 80 % | 276.8 [mW] |
| Idle | 23 [mW] | 20 % | 4.6 [mW] |
| | | Total | 281.4 [mW] |

Total power consumption = sum of all components power is ~ 349.2[mW]. From this very rough estimate, we can determine the Co2 consumption based off calculations provided in the ECE492 Lecture.

CO2 Generated:

0.3492 [W] * 24 h/ day * 365 days/year * 0.989 kg CO2/KWH = 3.03 kg CO2/Year

Overall, when considering the CO2 consumption of an average household, to the project's consumption it is negligible. For instance, the average household uses on average 12800 kg CO2/Year. This figure was taken from [23].

## REFERENCES

[1] S. Hunter, M. Wong and T. Zylstra, "ECE 492 Final Report: Voice Manipulator," 2013. [Online]. Available: https://www.ualberta.ca/~delliott/local/ece492/projects/2013w/G11_VoiceManipulator_FinalReport/G11_VoiceManipulator_FinalReport.pdf. [Accessed 18 January 2016].

[2] Altera, “Embedded Peripherals IP User Guide” [Online]. Available:

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_embedded_ip.pdf [Accessed 1 February 2016]

[3] Altera, “Avalon interface specifications” [Online]. Available:

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec_1_3.pdf [Accessed 10 April 2016]

[4] Altera, "Altera DC characteristics and timing specifications" 2007. [Online]. Available: https://www.ualberta.ca/~delliott/local/ece492/lectures/Pages%20from%20cyc2_cii5v1-IOspecs.pdf. [Accessed 18 January 2016].

[5] Altera, “Audio Core for Altera DE2/DE1 Boards” [Online]. Available:

ftp://ftp.altera.com/up/pub/Altera_Material/11.1/University_Program_IP_Cores/Audio_Video/Audio.pdf [Accessed 1 February 2016]

[6] Texas Instruments, “LM384 5W Audio Power Amplifier” [Online].

Available: http://www.ti.com/lit/ds/symlink/lm384.pdf [Accessed March 14, 2016]

[7] Tarek Kaddoura and Jigar Nahar, “App Notes: DM9000A Ethernet Controller” [Online]. Available:https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/Ethernet_DM9000A/AppNotesEthernet.pdf [Accessed 1 February 2016]

[8] Blue Creation, “Command Set guide manual V5 Rev D” [Online]. Available:

http://cdn.sparkfun.com/datasheets/Wireless/Bluetooth/Melody_5.0_Manual-RevD-RC10-Release.pdf [Accessed 1 February 2016]

[9] University of Southern California,”BSD-inet_aton.c function” http://opensource.apple.com//source/OpenSSH/OpenSSH-7.1/openssh/bsd-inet_aton.c [Accessed 1 March 2016]

[10] Node.js Foundation, “About Node.js” [Online]. Available: https://nodejs.org/en/about/ [Accessed 28 February 2016]

[11] Express.js, “Express home page” [Online]. Available: http://expressjs.com/ [Accessed 28 February 2016]

[12] Xiph.org Foundation, “Icecast home page” [Online]. Available: http://icecast.org/contact/ [Accessed 28 February 2016]

[13] Github TooTallNate(Nathan Rajlich), “Node-icy github repository” [Online]. Available: https://github.com/TooTallNate/node-icy [Accessed 28 February 2016]

[14] Mark Otto & the bootstrap team(https://github.com/orgs/twbs/people), “Twitter bootstrap” [Online]. Available: http://getbootstrap.com/ , getbootstrap.com/about/ [Accessed 28 February 2016]

[15] Google, “Angular JS” [Online]. Available: https://angularjs.org/ [Accessed 28 Februrary 2016]

[16] Future Music, "Distortion, saturation and bitcrushing explained" 2012. [Online]. Available: http://www.musicradar.com/tuition/tech/distortion-saturation-and-bitcrushing-explained-549516 . [Accessed 16 January 2016].

[17] Computer Music Specials, "The ultimate guide to effects: reverb" 2011. [Online]. Available: http://www.musicradar.com/tuition/tech/the-ultimate-guide-to-effects-reverb-461487. [Accessed 16 January 2016].

[18] Sample Guitar Wav File : (clean country blues riff 4) [Online]. Available:

http://www.freemusicloops.co.uk/search/clean [Accessed 1 February 2016]

[19] Cheng-Hao Chang, “A Guitar Overdrive/Distortion Effect of Digital Signal Processing” [Online]. Available:http://ses.library.usyd.edu.au/bitstream/2123/7624/2/DESC9115_DAS_Assign02_310106370.pdf [Accessed 1 February 2016]

[20] European Parliament and the council of the European Union, “DIRECTIVE 2002/95/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 January 2003 on the restriction of the use of certain hazardous substances in electrical and electronic equipment”

[Online]. Available:
http://www.digikey.com/web%20export/environmental/RoHSdirective_2002-95-EC.pdf
[Accessed 23 February 2016]

[21] SparkFun, “What does RoHS Compliant mean?” [Online]. Available:
https://www.sparkfun.com/static/rohs/ [Accessed 23 Februrary 2016]

[22] Altera, “RoHS Compliant, Environmentally Friendly FPGAs, SoCs & CPLDs” [Online].
Available:
https://www.altera.com/support/quality-and-reliability/environmental/lead-free/rel-lead-free.html
[Accessed 23 February 2016]

[23] City of Edmonton, “Towards a sustainable downtown: reducing GHG emissions”
[Online]. Available:
http://www.edmonton.ca/go_downtown/documents/PDF/Downtown_GHG_Emissions.pdf

[Accessed 29 February 2016]

[24] Silicom, “Music and the Human Ear”[Online].

Available: http://www.silcom.com/~aludwig/EARS.htm [Accessed April 10, 2016]

[25] Freesound, “1-sample impulse @ 48kHz (WAV)” [Online].

Available: https://www.freesound.org/people/unfa/sounds/205620/ [Accessed April 10, 2016]

[26] Ieeexplore, “Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK” [Online]. Available:
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5236563&newsearch=true&queryText=Audio%20Reverb [Accessed April 10, 2016]

[27] Sparkfun, "BC127 Bluetooth Hookup Guide" 2015. [Online]. Available:
https://learn.sparkfun.com/tutorials/understanding-the-bc127-bluetooth-module. [Accessed 1 February 2016].

[28] Blue Creation, "BC127 Datasheet" 2015. [Online]. Available:
https://cdn.sparkfun.com/datasheets/Wireless/Bluetooth/DataSheet-BC127.10.pdf. [Accessed 1 February 2016].

[29] Mary Lourde R, Anjali Kuppayil Saji, “A Digital guitar tuner” [Online]. Available:
http://arxiv.org/ftp/arxiv/papers/0912/0912.0745.pdf [Accessed 1 February 2016]

[30] Github codders (Arthur Taylor), “Node-icy github repository” [Online]. Available:
https://github.com/codders/libshout [Accessed 28 February 2016]

[31] Wikipedia, “Transmission Time” [Online]. Available: https://en.wikipedia.org/wiki/Transmission_time [Accessed 28 Februrary 2016]

[32] Wikipedia, “Bluetooth” [Online]. Available: https://en.wikipedia.org/wiki/Bluetooth#L2CAP [Accessed 28 February 2016]

[33] Wikipedia, “List of bluetooth profiles” [Online]. Available: https://en.wikipedia.org/wiki/List_of_Bluetooth_profiles#Advanced_Audio_Distribution_Profile_.28A2DP.29 [Accessed 28 February 2016]

[34] Bell Telephone Laboratories, A Michael Noll, "Pitch determination of human speech by the harmonic product, the harmonic sum spectrum, and a maximum likelihood estimate." 1969. [Online]. Available: http://noll.uscannenberg.org/Papers.htm . [Accessed 29 February 2016].

[35] Nancy Minderman, “NiosII_microc_lab2.vhd” [Online] Available on eClass [Accessed 20 January 2016]

## APPENDICES

### QUICK START MANUAL

Assumptions: User has basic knowledge on how to use Quartus and Qsys.

Amplifier Assembly:

1. Connect pins 2, 3, 4, 5, 7, 10, 11, 12 to ground.
2. Connect pins 14 to Vdd.
3. Attach a capacitor between pins 14 and ground.
4. Attach a capacitor between pins 1 and ground.
5. Connect a capacitor and speaker in series between pins 8 and ground.
6. To add additional speakers, add the speakers in parallel.
7. Attach a potentiometer to the input signal, ground and a capacitor. And pass the output of the capacitor to pin 6.
8. Power the amplifier and introduce an input signal to test the circuit.

DE2 Assembly:

1. Plug in the power source for the DE2 as well as the USB cable to the computer.
2. Assuming that a 3.5mm to a quarter-inch adapter was used for the electric guitar, plug in the 3.5 mm jack from the electric guitar to the LINE IN port of the DE2.
3. Plug in the 3.5 mm jack from the practice amplifier to the LINE OUT port of the DE2.

Programming:

1. Download source files from the course website and untar.
2. Launch Quartus.
3. Restore Archive Project. Look for “niosII_microc_capstone.qar” file from the extracted tar file.
4. Open up Qsys and open “niosII_system.qsys”.
5. Refresh by pressing F5.
6. Go to Generation Tab, click “Generate”
7. Once generation is done, go back to Quartus.
8. Compile the project.
9. Once compilation is complete, program the “niosII_microc_capstone.sof” to the DE2.
10. Open up NIOS II Software Build Tool for Eclipse.
11. Select the “software” folder of the extracted tar file as the working directory when prompted.

12. Import the “WebServer” and “WebServer_BSP” into your workspace.
13. Clean and build the project.
14. Generate BSP. After generating add “`#define ALT_ENHANCED_INTERRUPT_API_PRESENT`” into “system.h” file.
15. Run the project using NIOS II Hardware.

DE2 Switch and Key Instructions:

1. Enable Distortion Effect by toggling SWITCH 13 on the DE2
   a. Increase Distortion Level by pressing KEY 1
   b. Decrease Distortion Level by pressing KEY 3
2. Enable Reverb Effect by toggling SWITCH 14 on the DE2

## FUTURE WORK

RF Transmission

There are several different directions this project could have taken. The most notable being the implementation of wireless audio. One caveat we discovered when implementing a wireless solution was latency. The time that the audio took to transmit (using the bluetooth profile A2DP) created a virtually unplayable atmosphere. One solution could be send audio via an RF transmitter, and do the sampling only at the receiving end.

Web Streaming

The idea here is to stream the audio data coming from the guitar to a remote web server. This is a neat idea because people can listen to the live feed tunes from anywhere. This could be implemented using TCP/IP sockets to stream the data packets from a board that supports the internet protocol stack.

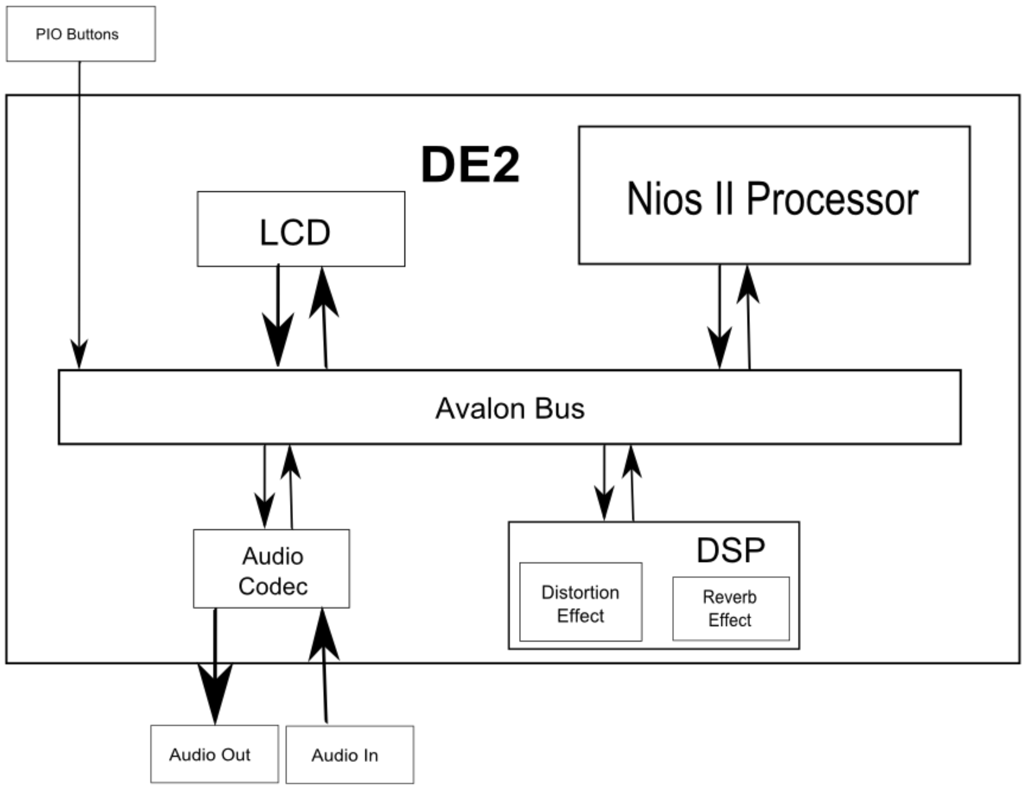
Guitar Tuner

The guitar tuner could be implemented in several different ways. The first that comes to mind is to perform an FFT in hardware, and then use a lookup table in software to determine the values received from the FFT block. At this point some type of feedback would be required for the user to determine when the note being played is the desired one. One thing to consider, especially if the project includes other facets, is the FPGA resources required to implement an FFT.

Infinity Mirror

This is a neat little add on. So the basic layout is you have an infinity mirror where the LEDS are connected to a processing unit. This processing unit acquires data about what strings are being plucked, and which frets are being used. From this information, it’s possible to map the strings/frets to single LEDS inside the infinity mirror.

## HARDWARE DOCUMENTATION


PIO Buttons
DE2
Nios II Processor
LCD
Avalon Bus
Audio Codec
DSP
Distortion Effect
Reverb Effect
Audio Out
Audio In