

FPGA based

Audio Effect System

Submitted by: Dean Devereaux

Student number: R00116198

Date: 28th May 2018

Supervisor: Patrick Collins

Assessor: John Horan

Department of Electrical &Electronic Engineering

## Declaration

## 

I declare that this submission is my own work and writing.

Any text, diagrams or other material copied from other sources (including, but not limited to, books, journals and the internet) have been clearly acknowledged and referenced as such in the text by the use of ‘quotation marks’ (or indented *italics* for longer quotations) followed by the author’s name and date [e.g. (Byrne, 2008)] either in the text or in a footnote/endnote. These details are then confirmed by a fuller reference in the bibliography section.

I acknowledge that I understand the concept of plagiarism. I have studied CIT’s guide to referencing (Section 3 Referencing at http://www.mycit.ie/libraryguide/ ) and I have read the section on Plagiarism and Cheating (pages 46 – 49) in CIT’s Student Regulations handbook (available at [http://www.mycit.ie/](http://www.mycit.ie/contentFiles/PDF/CIT%20regulations%20update%20May%202015.pdf) ). I understand that only assignments which are free of plagiarism will be awarded marks. By submitting this assignment, I agree to these terms. I further understand that CIT has a plagiarism policy which can lead to the suspension or permanent expulsion of students in serious cases.

I understand that my project report will be evaluated by the Safe Assignment/TurnItIn tool in Blackboard to detect instances of plagiarism.

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Table of Contents

[Declaration i](#_Toc515300597)

[Table of Contents ii](#_Toc515300598)

[List of Figures iv](#_Toc515300599)

[List of Tables v](#_Toc515300600)

[Abstract vi](#_Toc515300601)

[Acknowledgements vii](#_Toc515300602)

[1. Introduction 1](#_Toc515300603)

[2. Literature Review/Theoretical 2](#_Toc515300604)

[a. Basys3 2](#_Toc515300605)

[b. Pmod I2S 2](#_Toc515300606)

[c. Comb Filters 3](#_Toc515300607)

[d. Reverberators 5](#_Toc515300608)

[e. Non-Linear Processing 6](#_Toc515300609)

[a. Vivado Design Suite 6](#_Toc515300610)

[3. System Design 7](#_Toc515300611)

[a. XADC 7](#_Toc515300612)

[b. Pmod I2S 8](#_Toc515300613)

[c. Effects 8](#_Toc515300614)

[d. Slide Switches and Push Buttons 9](#_Toc515300615)

[4. Simulation and Implementation 9](#_Toc515300616)

[a. XADC Simulation 9](#_Toc515300617)

[b. MATLAB Simulation 10](#_Toc515300618)

[c. Vivado Simulation 11](#_Toc515300619)

[5. Results 12](#_Toc515300620)

[6. Analysis 15](#_Toc515300621)

[7. Conclusion 15](#_Toc515300622)

[References 16](#_Toc515300623)

[Appendix A 17](#_Toc515300624)

[Sine Wave Generator 17](#_Toc515300625)

[IIR MATLAB Code 19](#_Toc515300626)

[IIR Vivado Code 19](#_Toc515300627)

[FIR MATLAB Code 20](#_Toc515300628)

[FIR Vivado Code 21](#_Toc515300629)

[Overdrive MATLAB Code 22](#_Toc515300630)

[Overdrive Vivado Code 23](#_Toc515300631)

## List of Figures

[Figure 1 IIR Comb Filter 3](file:///D:\ELS4%20Semester%202%202018\R00116198_DeanDevereaux_FPGABasedAudioEffectSystem_FYP.docx#_Toc515297236)

[Figure 2 FIR Comb Filter 3](file:///D:\ELS4%20Semester%202%202018\R00116198_DeanDevereaux_FPGABasedAudioEffectSystem_FYP.docx#_Toc515297237)

[Figure 3 Universal Comb Filter 4](file:///D:\ELS4%20Semester%202%202018\R00116198_DeanDevereaux_FPGABasedAudioEffectSystem_FYP.docx#_Toc515297238)

[Figure 4 Schroeder’s Reverberator 5](file:///D:\ELS4%20Semester%202%202018\R00116198_DeanDevereaux_FPGABasedAudioEffectSystem_FYP.docx#_Toc515297239)

[Figure 5 Moorer’s Reverberator 5](file:///D:\ELS4%20Semester%202%202018\R00116198_DeanDevereaux_FPGABasedAudioEffectSystem_FYP.docx#_Toc515297240)

[Figure 6 System Design Block Diagram 7](#_Toc515297241)

[Figure 7 Pmod I2S 8](file:///D:\ELS4%20Semester%202%202018\R00116198_DeanDevereaux_FPGABasedAudioEffectSystem_FYP.docx#_Toc515297242)

[Figure 8 Overdrive soft clipping formula 10](#_Toc515297243)

[Figure 9 IIR Filter: RTL Schematic 11](#_Toc515297244)

[Figure 10 FIR Filter: RTL Schematic 11](#_Toc515297245)

[Figure 11 FIR MATLAB Results 12](#_Toc515297246)

[Figure 12 FIR Vivado Results 12](#_Toc515297247)

[Figure 13 IIR MATLAB Result 13](#_Toc515297248)

[Figure 14 IIR Vivado Results 13](#_Toc515297249)

[Figure 15 Overdrive MATLAB Results 14](#_Toc515297250)

[Figure 16 Overdrive Vivado Results 14](#_Toc515297251)

## List of Tables

[Table 1 LRCK and MCLK ratio’s for Bit rate 3](#_Toc515297252)

[Table 2 UCF Parameters 4](#_Toc515297253)

[Table 3 Delay Types 8](#_Toc515297254)

## Abstract

For this project the objective will be to apply an effect to an audio signal using a field programmable gate array on the basys3 development board which will be programmed using HDL on Vivado design suite. This audio signal will go through the development board and produce an altered signal outputted to a speaker. Three effects will be created and applied to the signal using slide switches to choose the chosen effect and push buttons to increment and decrement the amplitude.

The research of the hardware from last semesters report will be revised and the effects will be explained including the filters associated with these effects and the design process. The problems which arose with the ADC through simulation will also be examined and how this lead to the implementation and results.

## Acknowledgements

I would like to express my sincere thanks and tremendous gratitude to my supervisor Paddy Collins and assessor John Horan for their guidance and patience throughout the year.

Dean Devereaux

28th May 2018

# Introduction

This report will be going through the research and work completed in semester two. The areas at which this project will be focusing on are digital system design and digital signal processing. The end product should show that the Artix-7 FPGA can be used to create an audio effect system similar to a pedal board used by musicians. This idea has already been done by a music industry company by the name of Antelope Audio who sell a Hardware FPGA Audio effects commercially. (“Antelope Audio Hardware-based FX,” n.d.)

The main goal of this project is to program the basys3 development board to digitize any analogue audio sent through the JXADC Pmod I/O and apply any three of the following effects

* Delay
* Non-Linear Processing
* Spatial

The digitized audio will then be sent out through a second Pmod I/O port and converted back to analogue audio using the external Pmod I2S which can then be connected to a speaker to output the audio with the applied effect. For the three effects, slide switches from the basys3 will be used to control which effects gets applied to the audio passing through and the push buttons to increment and decrement the gain of such effects.

(“DeanDevereaux\_FPGA\_Based\_AudioEffectsSystem” 5th January 2018)

In the previous report for semester one the research of the Basys3, Vivado and the XADC was looked over. For this report the comb filters used to alter a signal will be mentioned and the work done to try input audio through the internal XADC and output through the DAC.

# Literature Review/Theoretical

# Basys3

The basys3 development board uses the artix-7 FPGA chip which allows for high functionality at low power consumption. The basys3 is equipped with an internal ADC which interfaces with the fpga through a dynamic reconfiguration port and can receive an input signal through the I/O JXADC Pmod. The internal ADC can convert up to 12 bits and samples up to 1 MHz

(“DeanDevereaux\_FPGA\_Based\_AudioEffectsSystem” 5th January 2018)

(“Basys 3 Reference Manual” 2016.)

# Pmod I2S

This Diligent product uses the Cirrus logic CS4344 DAC and can interface with the Basys3 through GPIO protocol to convert digital signals back into analogue which will get output through an audio headphone jack. This Pmod has three clock pins and a single input pin. MCLK is the master clock and takes the clock signal from the host board to allow for synchronization however this clock can only take a clock signal anywhere from 512 kHz to 50 MHz LRCK stands for left right clock and will be used for sample rate. The max sample rate this clock can take is 192 kHz. Serial clock (SCK) which is the bit clock has two methods of applying a clock signal. It can use the ratio of MCLK and LRCK to calculate a bit rate for the signal for take a clock signal from the host board itself. Table 1 below shows the sample and Clock rate commonly used to create a SCK value.

(“DeanDevereaux\_FPGA\_Based\_AudioEffectsSystem” 5th January 2018)

(“Pmod I2S.” Digilent, April 12, 2016.)

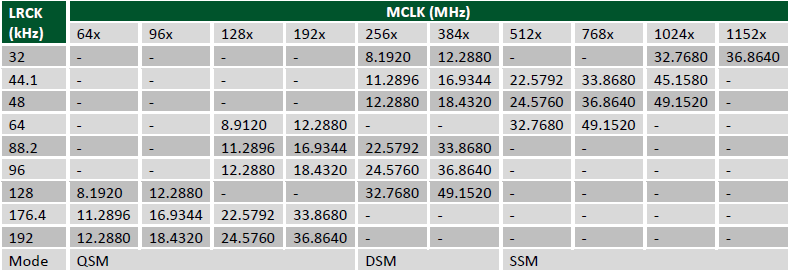


Table 1 LRCK and MCLK ratio’s for Bit rate

(“Pmod I2S.” Digilent, April 12, 2016.)

# Comb Filters

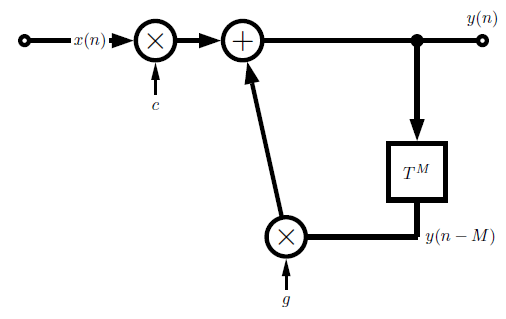
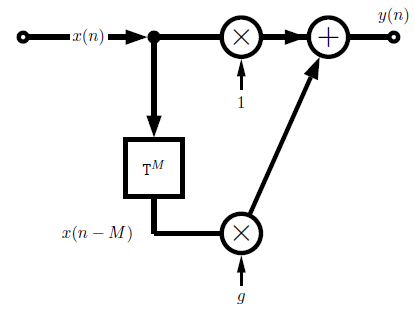
IIR filter which stands for infinite impulse response involves feedback for a delay effect. Fig.1 shows the design for an IIR filter. The feedback line uses a transport delay to delay the signal by a chosen amount and the gain decays the amplitude of the signal. The feedback line is then added back to the input and creates an endless reflection of delays. Eq.1 shows the differential equation which is associated with the IIR filter.

Figure 1 IIR Comb Filter

(“Basic Digital Audio Effects”, n.d.)

(“Basic Digital Audio Effects”, n.d.)

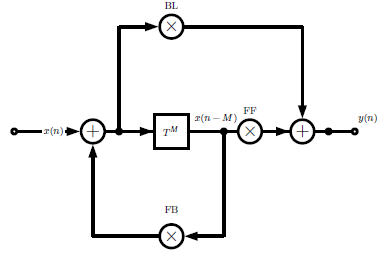
The Finite impulse response filter applies a feedforward line which uses similar method to the IIR filter. The feedforward line delays by a chosen amount and the gain can decay or intensify the signal which will be added back to the input signal and output through y(n). Fig.2 shows the layout of a simple FIR filter and Eq.2 shows its differential equation.

Figure 2 FIR Comb Filter

(“Basic Digital Audio Effects”, n.d.)

(“Basic Digital Audio Effects”, n.d.)

The Universal comb filter is the combination of both the IIR and FIR comb filters. Fig. 3 shows this filter which can function as

Figure 3 Universal Comb Filter

(“Basic Digital Audio Effects”, n.d.)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **BL** | **FB** | **FF** |
| **FIR Comb** | 1 | 0 | g |
| **IIR Comb** | 1 | g | 0 |
| **All-Pass** | α | -α | 1 |
| **Delay** | 0 | 0 | 1 |

Both IIR and FIR separately but also could be used as an All-Pass filter. This filter allows all frequencies to pass through without a change in amplitude however the phase gets shifted. The universal comb filter can also be used as a delay line if the blend and feedback don’t get used and the feedforward remains at 1.

(“Basic Digital Audio Effects”, n.d.)

(“All pass Filter”, 2010)

Table 2 UCF Parameters

(“Basic Digital Audio Effects”, n.d.)

# Reverberators

Reverb is a natural effect and occurs within rooms when sound reflects off walls and reaches a certain destination in nano or milliseconds after the original sound. All reflections wont’ reach the destination at the same time and this causes the reverb sound.

Figure 4 Schroeder’s Reverberator

(“Basic Digital Audio Effects”, n.d.)

The most basic and primitive reverberator is the Schroeder’s Reverberator.

This uses IIR comb filters in parallel with random delays to give a natural reverb effect of sound bounding off walls and returning at different times. These signals then add together to be input through two or more All-Pass filters which will increase reflection density of the signal and output the reverbed signal. Fig. 4 shows the Schroeder method.

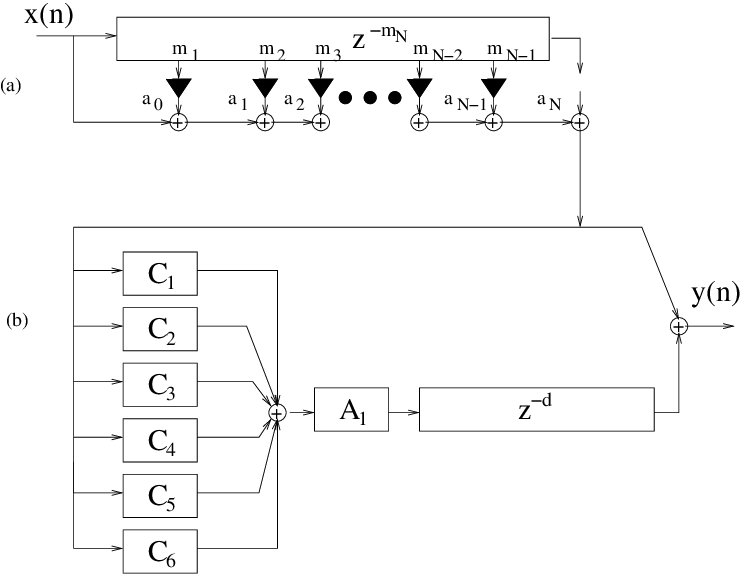
Moorer’s Reverberator is an improvement and extension of the Schroeder. Moorer adds on a FIR filter which is used for early reflection of reverb seen in fig. 5 as part (a). A low pass filter is used with (b) to delay the signal for late reflections of reverb. (“Basic Digital Audio Effects”, n.d.)

Figure 5 Moorer’s Reverberator

(“Basic Digital Audio Effects”, n.d.)

# Non-Linear Processing

Overdrive and Distortion are the two intentional non-linear processing effects which were reviewed for this project. Overdrive involves “Audio at a low input level driven by higher input levels in a non-linear curve characteristic”. This involves symmetrical soft clipping which saturates the input signal when it reaches a certain threshold value. With distortion then it is “a wider tonal area than overdrive operating at a higher non-linear region of a curve”. The original signal will have a gain applied to it which controls level of distortion in the signal. This distortion then gets mixed with the original to form an output signal. (“Basic Digital Audio Effects”, n.d.)

# Vivado Design Suite

Vivado is a Xilinx design suite which programs with hardware descriptive languages Verilog and VHDL. Digilent® include many tutorials on their website for programming development boards that specifically use Vivado. Vivado is currently on version 2018.1 as of April 12th, 2018

(“DeanDevereaux\_FPGA\_Based\_AudioEffectsSystem” 5th January 2018)

# System Design

The Basys3 FPGA development board which uses the Artix-7 chip will input an analogue audio signal and converts to a digitized audio signal using an internal ADC. The board will then apply any three effects being controlled by slide switches before the signal is output to the external Pmod I2S, through the headphone jack to a speaker. Fig.6 shows the design layout for the proposed system.

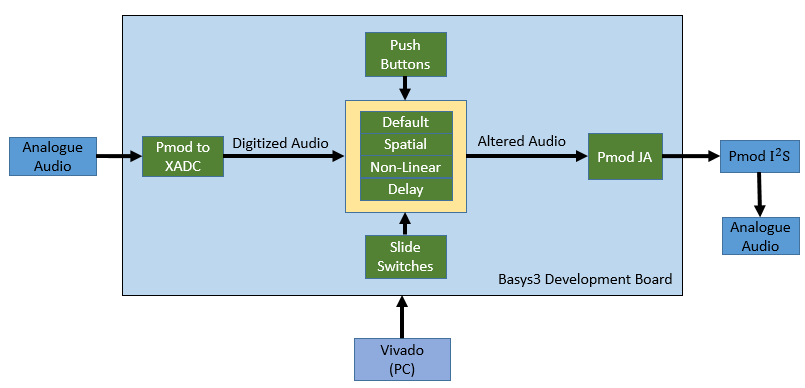


Figure 6 System Design Block Diagram

# XADC

The internal ADC would be used with I/O JXADC Pmod to input audio. The Pmod can send 8 bits of data at a time in parallel using the bipolar auxiliary signal ports vauxp 6, vauxn 6, vauxp 14, vauxn 14, vauxp 7, vauxn 7 and vauxp 15, vauxn 15. The ADC will have a sample frequency of 44.1 kHz which is commonly used with audio files today.

(“DeanDevereaux\_FPGA\_Based\_AudioEffectsSystem” 5th January 2018)

# Pmod I2S

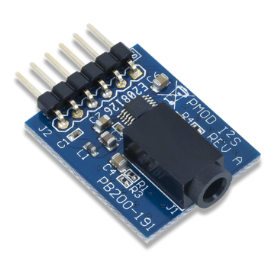
Due to the basys3 not having an internal DAC the Pmod i2s was chosen to interface with the Basys3 and convert the digitized altered signal. The Pmod can convert up to 24 bits and has a max sampling frequency of 192 kHz which is more than adequate for the conditions needed. The I2S has the advantage of outputting the signal through an audio jack therefore it could be used to output the audio into a speaker to observe the effects.

Figure 7 Pmod I2S

(“Pmod I2S.” Digilent, April 12, 2016.)

# Effects

A simple IIR and FIR comb filter would be implemented separately to create a delayed effect. Based on the timing a delay effect can change the outputs nature. Table 2 shows the different delays that can be applied based on the timing of the delay.

|  |  |
| --- | --- |
| **Delay Type** | **Time** |
| Resonator | 0-20ms |
| Slap back | 25-50ms |
| Echo | >50ms |

Table 3 Delay Types (“Basic Digital Audio Effects”, n.d.)

Reverb is the preferred spatial effect to be applied to the Basys3. This effect will be created by programming the Schroeder and Moorer reverberator into the basys3 development board which uses IIR, FIR and Universal comb filters along with a low pass filter. Overdrive and Distortion will be tested to see which can be used on the Basys3 to provide an intentional Non-linear effect. This will be done through symmetrical soft clipping for overdrive and applying a gained to the original signal and mixing back into the original for distortion such as is explained in section 2.

# Slide Switches and Push Buttons

The basys3 has 15 slide switches attached to it. These slide switches will be programmed to choose an effect and will also allow multiple effects to be used simultaneously. The push buttons would then be used to increment or decrement the gain values. Due to the limited amount of push buttons only the delay and non-linear will implement this. The up and down push button will be used for the delay and left and right for non-linear.

# Simulation and Implementation

# XADC Simulation

In Semester one the ADC was tested with the demo found on Digilent and worked perfectly were the LED’s and seven segments displayed the voltage digitally. For this semester I started off by examining the ADC’s output through an oscilloscope. There was too much data being output in real time therefore simulation was the solution. The end of conversion line could be viewed working correctly at 44.1 kHz giving a high impulse every 22.6us however the ADC wasn’t outputting an actual value. A design text file was generated and used to input simulation values into the ADC. When simulated these values weren’t being applied. A solution was sought online, and it was found that other individuals online had the same problem and solutions suggested did not work in for this this case.

With the XADC not working correctly in simulation there was no way to know if the implementation would work on the hardware/simulate with the DAC and applied effects therefore the effects would be worked on through simulation only on a FPGA board.

# MATLAB Simulation

Before applying any effects to the FPGA, the filters were tested on MATLAB using code found online implementing such effects. All simulations used a sine wave of 1 kHz with a sampling rate of 44.1 kHz. FIR used a delay of 2ms and IIR a delay of 10ns. Both used a gain of 0.5 with the delay line which would attenuate the signal. To get a chosen delay time in MATLAB a value was created for memory allocation by multiplying the time delay desired by the sample frequency. E.g. 44.1 kHz x 2ms = 88.2 🡪 88

Overdrive used symmetrical soft clipping. This method had threshold values to saturate the signal causing the sine wave to become almost square by introducing odd harmonics. Fig. 8 below shows the threshold values with the MATLAB simulation

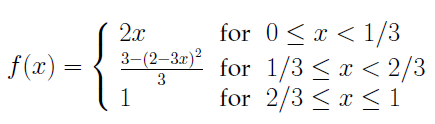


Figure 8 Overdrive soft clipping formula

(“Basic Digital Audio Effects”, n.d.)

To create a distorted signal a gain of 11 was applied to the sine wave which was then mixed back in with the original to

All the effects used were also simulated using an audio file to listen to the effect however the sine wave plots are much easier to understand.

# Vivado Simulation

Verilog hardware descriptive language was used to program the effects within the basys3 development board. To implement the IIR filter shown in fig.9 shift registers were used to hold the data which would be pushed through every clock cycle creating a feedback line. To apply a gain in Verilog an arithmetic shift right had to be used on the data. This would only allow gains of 1/2, 1/4 and 1/8 to be used with the filters. For the input a 1 kHz sine wave was generated with a code and the right clock signal.

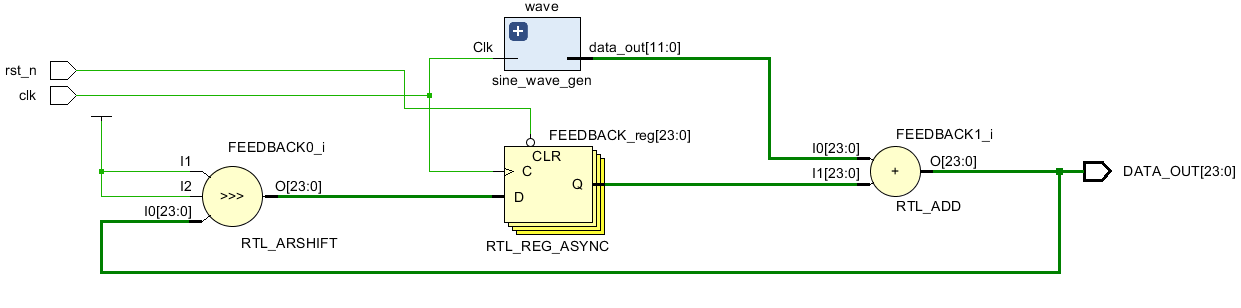


Figure 9 IIR Filter: RTL Schematic

The FIR was implemented similar to the IIR filter by using shift registers for a feedforward delay line with a sine wave generator input. However, each output of the shift registers is multiplied and added together with the input signal. Fig. 10 shows this implementation on Vivado.

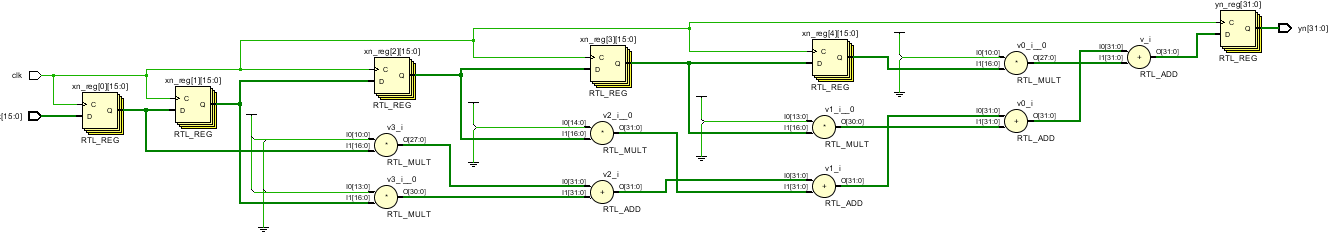


Figure 10 FIR Filter: RTL Schematic

Unfortunately, there was no time remaining to implement the reverberator and distortion onto the FPGA therefore no work on the all-pass filter was completed after researching.

# Results

Fig.11 shows the results of FIR when implemented on MATLAB. What can be seen is the 1 kHz Input sine wave staying constant at 1V peak to peak. At 2ms the feedforward delay gets added to the input which increases the amplitude of the output signal to 1.5v since the feedback has a gain of 0.5. This is in fact the 2ms of input signal before feedforward gets applied.



Figure 11 FIR MATLAB Results

Fig.12 is then the FIR implemented in Vivado using Verilog. This version of the FIR used 10 shift registers for a delay of 10 clock cycles. The top sine wave represents the input, middle the feedforward line and bottom, the output.

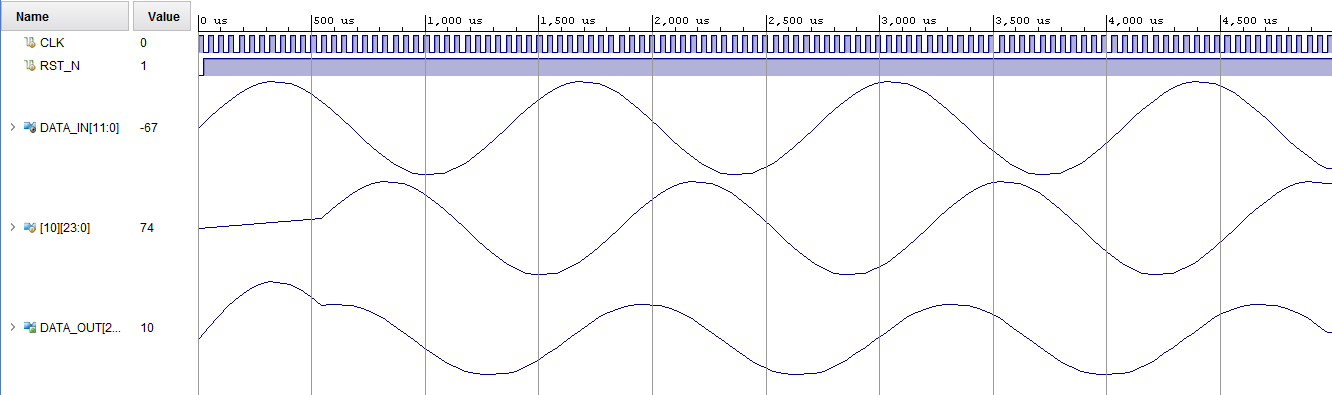


Figure 12 FIR Vivado Results

Fig.13 is the IIR result with a 266us delay applied to the feedforward line. The output increases in amplitude when the feedforward is applied however it settles back to normal after.



Figure 13 IIR MATLAB Result

Implemented on Vivado fig.14 shows the input signal, top and the feedback in the middle which added together after 10 clock cycles to give a delayed output response.

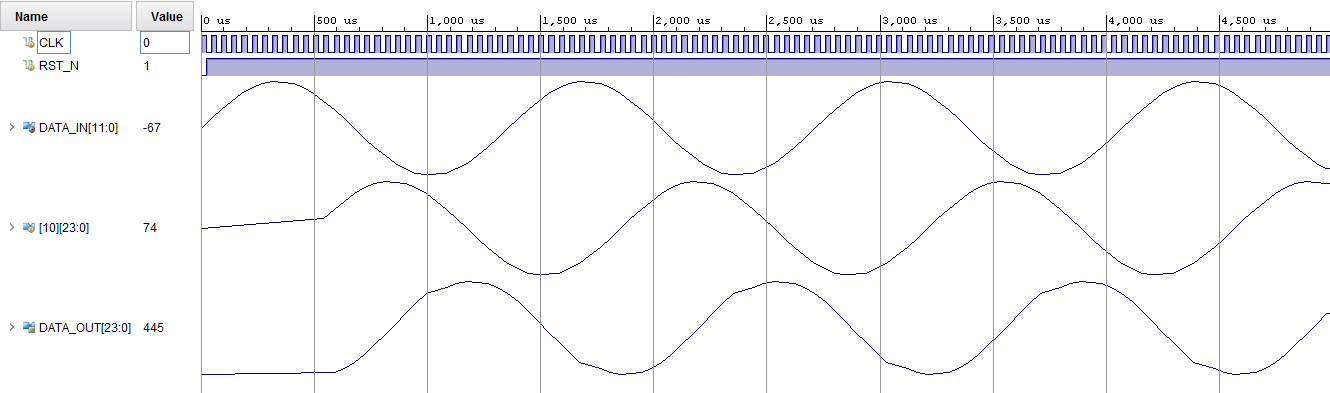


Figure 14 IIR Vivado Results

On MATLAB the overdrive results in fig.15 almost gave a square wave output response. Fig.16 is very similar to the MATLAB results with the saturated output showing a distorted signal.



Figure 15 Overdrive MATLAB Results

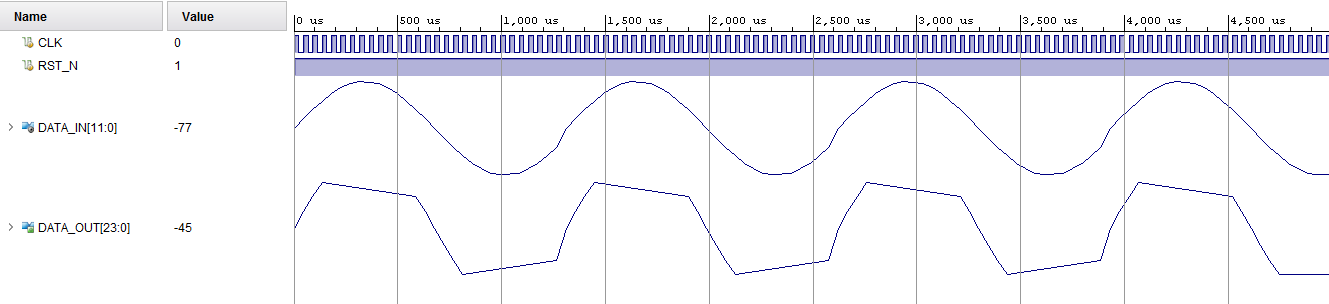


Figure 16 Overdrive Vivado Results

# Analysis

Comparing the Vivado results with MATLAB simulation, FIR seems to be reducing the amplitude when applied rather than increasing. There was no way to examine the output waveform besides the plot. The MATLAB code which implemented the same way as Vivado used an audio file and could be heard with an echo of 60ms.

IIR on Vivado has output a sine wave out of phase with the input due to the feedback line. These results are almost accurate with MATLAB results of the sine wave. The IIR was tested with the same audio file and had a much better echo effect to FIR due to its endless reflections.

Overdrive Vivado results are very accurate when compared with MATLAB. You can see the cut-off point on the output signal that almost converts the signal to a square wave.

# Conclusion

Unfortunately, the basys3 was not used for Audio Effects but the Delay and Non-linear effects were simulated onto the FPGA with Vivado. This failure in operating the hardware was due to the very little knowledge known about the specific ADC and DAC being used. Too much time was also taken up by focusing on trying to implement the hardware first when simulations should have been the only option before any hardware was configured. If more time was allocated the ADC could be examined thoroughly for simulation to pin point the problem unsolved in this report.

From this project a better understanding of Verilog HDL and the processing of Delay, Spatial and Non-Linear audio effects have been acquired along with the operation of the basys3 board. The Artix-7 FPGA does have the capability to be used as an audio effect system in theory and simulation. How much is it capable of is the next question.

## References

“DeanDevereaux\_FPGA\_Based\_AudioEffectsSystem” 5th January 2018

[Accessed 30th April 2018]

“Basys 3 Reference Manual” 2016.

<https://reference.digilentinc.com/basys3/refmanual>

[Accessed 24th September 2017]

“Pmod I2S.” Digilent, April 12, 2016.

<https://reference.digilentinc.com/reference/pmod/pmodi2s/reference-manual> [Accessed 6th October 2017]

“Basic Digital Audio Effects”, n.d.

<http://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf> [Accessed 4th October 2017]

“All pass Filter”, 2010

<https://ccrma.stanford.edu/~jos/pasp/Allpass_Filters.html>

[Accessed 2nd April 2018]

“verilogcodes”, 17th November 2015

<http://verilogcodes.blogspot.ie/2015/11/verilog-code-for-simple-sine-wave.html> [Accessed 30th April 2018]

## Appendix A

## Sine Wave Generator

module sine\_wave\_gen(Clk, data\_out);

//declare input and output

input Clk;

output [11:0] data\_out;

//declare the sine ROM - 30 registers each 8-bit wide.

reg [11:0] sine [0:29];

//Internal signals

integer i;

reg [11:0] data\_out;

//Initialize the sine rom with samples.

initial begin

i = 0;

sine[0] = 0;

sine[1] = 16;

sine[2] = 31;

sine[3] = 45;

sine[4] = 58;

sine[5] = 67;

sine[6] = 74;

sine[7] = 77;

sine[8] = 77;

sine[9] = 74;

sine[10] = 67;

sine[11] = 58;

sine[12] = 45;

sine[13] = 31;

sine[14] = 16;

sine[15] = 0;

sine[16] = -16;

sine[17] = -31;

sine[18] = -45;

sine[19] = -58;

sine[20] = -67;

sine[21] = -74;

sine[22] = -77;

sine[23] = -77;

sine[24] = -74;

sine[25] = -67;

sine[26] = -58;

sine[27] = -45;

sine[28] = -31;

sine[29] = -16;

end

//At every positive edge of the clock, output a sine wave sample.

always@ (posedge(Clk))

begin

data\_out = sine[i];

i = i+ 1;

if(i == 30)

i = 0;

end

endmodule

(“verilogcodes”, 17th November 2015)

## IIR MATLAB Code

F = 1000; % Sine wave frequency(herts)

fs= 44100; % Sampling frequency(samples per second)

dt = 1/fs; % seconds per sample(22.6u)

StopTime = 0.1; % seconds

t = (0:dt:StopTime)'; % seconds

data = sin(2\*pi\*F\*t);

g=0.5; %Gain

M=2210; %Delay

Delayline=zeros(M,1);

for n=1:length(data);

y(n)=data(n)+g\*Delayline(M);

Delayline=[y(n);Delayline(1:M-1)];

End

(“Basic Digital Audio Effects”, n.d.)

## IIR Vivado Code

module IIR\_COMB\_FILTER(

input clk, rst\_n,

output reg signed[23:0]DATA\_OUT

);

wire signed[11:0]DATA\_IN;

reg signed[23:0]DELAY[10:0];

initial begin

DATA\_OUT = 24'h000000;

DELAY[0] = 24'h000000;

DELAY[1] = 24'h000000;

DELAY[2] = 24'h000000;

DELAY[3] = 24'h000000;

DELAY[4] = 24'h000000;

DELAY[5] = 24'h000000;

DELAY[6] = 24'h000000;

DELAY[7] = 24'h000000;

DELAY[8] = 24'h000000;

DELAY[9] = 24'h000000;

DELAY[10] = 24'h000000;

end

always @(DATA\_IN)begin

DATA\_OUT = DATA\_IN+DELAY[10];

end

always @(posedge clk or negedge rst\_n)begin

if(rst\_n == 0)

DELAY[0]<=24'h000000;

else

DELAY[0]<=DATA\_IN; //DATA\_IN(FIR), DATA\_OUT(IIR)

DELAY[1]<=DELAY[0];

DELAY[2]<=DELAY[1];

DELAY[3]<=DELAY[2];

DELAY[4]<=DELAY[3];

DELAY[5]<=DELAY[4];

DELAY[6]<=DELAY[5];

DELAY[7]<=DELAY[6];

DELAY[8]<=DELAY[7];

DELAY[9]<=DELAY[8];

DELAY[10]<=DELAY[9];

end

## FIR MATLAB Code

F = 1000; % Sine wave frequency(Hertz)

fs= 44100; % Sampling frequency (samples per second)

dt = 1/fs; % seconds per sample(22.6u)

StopTime = 0.004; % seconds

t = (0:dt:StopTime)'; % seconds

data = sin(2\*pi\*F\*t);

g=0.5; %Gain

M=88; %M=Fs\*DelayTimeDesired

Delayline=zeros(M,1); %Feedforward

for n=1:length(data);

y(n)=data(n)+g\*Delayline(M);

Delayline=[data(n);Delayline(1:M-1)];

end;

(“Basic Digital Audio Effects”, n.d.)

## FIR Vivado Code

module IIR\_COMB\_FILTER(

input clk, rst\_n,

output reg signed[23:0]DATA\_OUT

);

wire signed[11:0]DATA\_IN;

reg signed[23:0]DELAY[10:0];

initial begin

DATA\_OUT = 24'h000000;

DELAY[0] = 24'h000000;

DELAY[1] = 24'h000000;

DELAY[2] = 24'h000000;

DELAY[3] = 24'h000000;

DELAY[4] = 24'h000000;

DELAY[5] = 24'h000000;

DELAY[6] = 24'h000000;

DELAY[7] = 24'h000000;

DELAY[8] = 24'h000000;

DELAY[9] = 24'h000000;

DELAY[10] = 24'h000000;

end

always @(DATA\_IN)begin

DATA\_OUT = DATA\_OUT+DELAY[10];

end

always @(posedge clk or negedge rst\_n)begin

if(rst\_n == 0)

DELAY[0]<=24'h000000;

else

DELAY[0]<=DATA\_IN; //DATA\_IN(FIR), DATA\_OUT(IIR)

DELAY[1]<=DELAY[0];

DELAY[2]<=DELAY[1];

DELAY[3]<=DELAY[2];

DELAY[4]<=DELAY[3];

DELAY[5]<=DELAY[4];

DELAY[6]<=DELAY[5];

DELAY[7]<=DELAY[6];

DELAY[8]<=DELAY[7];

DELAY[9]<=DELAY[8];

DELAY[10]<=DELAY[9];

end

## Overdrive MATLAB Code

N=length(x);

y=zeros(1,N); % Preallocate y

th=1/3; % threshold for symmetrical soft clipping

% by Schetzen Formula

for i=1:N

if abs(x(i))< th, y(i)=2\*x(i);end;

if abs(x(i))>=th,

if x(i)> 0,

y(i)=(3-(2-x(i)\*3).^2)/3;

end;

if x(i)< 0,

y(i)=-(3-(2-abs(x(i))\*3).^2)/3;

end;

end;

if abs(x(i))>2\*th,

if x(i)> 0,

y(i)=1;

end;

if x(i)< 0,

y(i)=-1;

end;

end;

end;

(“Basic Digital Audio Effects”, n.d.)

## Overdrive Vivado Code

module OVERDRIVE(

input clk, rst\_n,

output reg signed[23:0]DATA\_OUT

);

wire signed[11:0]MAG;

wire signed[11:0]DATA\_IN;

reg signed[11:0]THRES;

always @(DATA\_IN)begin

THRES = MAG/1.71;

if(DATA\_IN > THRES || DATA\_IN < -THRES)

DATA\_OUT = DATA\_OUT;

else

DATA\_OUT = DATA\_IN;

end