VOICE AND GUITAR AUDIO EFFECTS PROCESSOR IMPLEMENTED ON A XILINX SPARTAN FPGA

Alejandro Valdés, Esteban Aristizábal, Hernán Benítez IEEE Member, Michael Martinez

Abstract— **This article describes the implementation of a digital signal processor on an FPGA (Field Programmable Gate Array) including a graphic monitoring system that allows students of the BSc in electronics of the Xavierian University to understand and research further on the field of audio processing.**

The project has recording and visualization modules to analyze signals and to observe how the sonic effects are produced.

The device consists of two inputs in parallel, one for guitar and one for vocals. Each input contains the required signal adequacy. After the signal is received, there is a 12 bits analog to digital conversion for each input, summing a total of 24 bits to be processed by the FPGA. Finally, the signal is converted back to analog ready to be amplified.

***Keywords*— Audio, Guitar Effects, FPGA, Digital Signal Processing.**

I. INTRODUCTION

Sound engineering is the discipline that focuses on sound phenomena in every field and its applications such as recording, acoustics, electroacoustic, live sound and systems design. Sound engineering is based on electronics theory and musical appreciation [1].

The demand of sound engineers in Colombia is increasing continuously due to:

- The continuous growth on sound technology in broadcast and television channels, radio stations and the high demand on live sound.
- Evolution of arts in our country and the high quality required producing new artists.
- Colombian cinematography has received more attention locally and around the world.

This situation encourages the research on digital signal processing to facilitate education for students of electronic engineering who want to study further how to process digital audio. This article describes the design process, results and conclusions after implementing a digital system with audio effects such as distortion, delay, low pass filter and panning on an FPGA from Xilinx.

II. METHODS

The project consists on signal processing using an FPGA. The final device has an input for voice signals and another one for guitar signals. The predefined effects controlled through a graphic interface will affect these signals. The interface contains the following modules that allow students to analyze each effect:

- Effects Selection
- Signal Monitoring
- Recording and Synthesis

These modules turn this project into a very useful tool to study audio.

A. ANALOG TO DIGITAL CONVERTER AD 1[2]

The Analog to Digital converter AD1 [2] is able to convert a signal with a maximum of one million samples per second, fast enough to perform the most demanding audio applications making it suitable for this project. This module is composed by two ADCS7476MSPS analog to digital integrated circuits, each one with 12 bits to perform simultaneous operations fulfilling the requirements of two independent processing channels. Also each IC has an anti - alias filter as shown in Figure 1. A/D 1 Module Diagram.

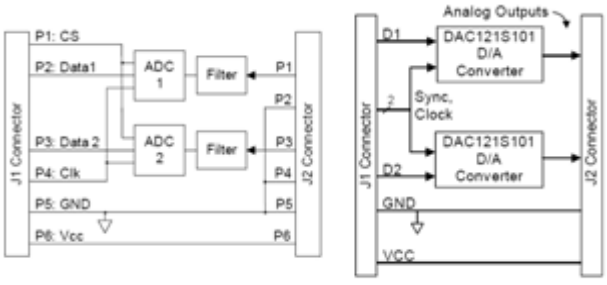

Figure 1. A/D 1 Module Diagram.

B. DIGITAL TO ANALOG CONVERTER DA2 [3]

The DA2 Module converts digital signals into analog signals simultaneously in two channels with a 12 bits resolution [3]. It consists of two DAC121S101 integrated circuits from National Semiconductor.

C. GRAPHIC INTERFACE

The graphic interface was designed in Matlab's GUIDE (Graphical User Interface Development Environment), which consists of a visual programming environment to build and execute programs that require a continuous data input.

1) Main Menu

The main menu shows all the options available. There are four options including Visualization, Recording, Effects Control and Synthesis. To access to each one of these options the user has to click on the corresponding image and automatically a new window will pop up.

2) Visualization

This option opens a new window that contains a "PLAY" button that activates Simulink allowing the observation of the signal's spectrogram and its amplitude. [4].

3) Recording

This option allows the user to select the desired period of time to record the input signals. It has two buttons, one for each input channel.

After the signals have been recorded, a button named "FFT" can be pressed to obtain the frequency response of each one the two signals recorded.

4) Effects

This option contains all the audio effects programmed on the FPGA. When any of the effects are selected, the program sends a signal through the serial port to activate the corresponding algorithm in the FPGA.

5) Synthesis

In this section there are 8 buttons representing each one of the musical notes (DO, RE, MI, FA, SOL, LA, SI). Each button activates a filter that is applied to a white noise sample to produce the corresponding frequency of the selected note.

Also there is a "time" button that generates an array of the size specified by the user. When the musical notes buttons are pressed, each note is being recorded in the current position of the array. When the array is full, user can hit "play" and listen to all the generated sounds through the white noise filtering method.

D. AUDIO EFFECTS

The effects were programmed and simulated in Matlab using pre recorded sound samples.

Five effects were implemented :

- ***Distorsion*** (Modifies the dynamic range of the signal)
- ***Delay*** (records a signal and then plays it back after an specific period of time)
- ***Reverse*** (Inverts a delayed signal on time)
- ***Low Pass Filter*** (attenuates high frequencies)
- ***Panning*** (Locates a mono signal in a stereo field)

1) Distortion

There are two well-known methods to distort an audio signal, saturation and absolute value. Saturation consists on establishing a threshold lower than the signal's maximum peak that samples can't exceed, producing a distorted signal.

Distortion by saturation main disadvantage is that if the audio signal falls under the specified threshold, it doesn't get saturated reason why the effect had to be designed to change threshold's value dynamically depending on the incoming signal's amplitude. This dynamic design would compromise many clock cycles due to the amount of comparisons required. In contrast, the absolute value method doesn't depend on the signal's amplitude, thus it was implemented. **Figure 5** presents the simulation of distortion caused by saturation.

Simulations were developed in Matlab, and they were tested using sinusoidal and prerecorded guitar signals.
Input and output graphics of a sinusoidal signal are shown with amplitude of 6 Volts and a frequency of 1 Khz. The signal is generated in Matlab.

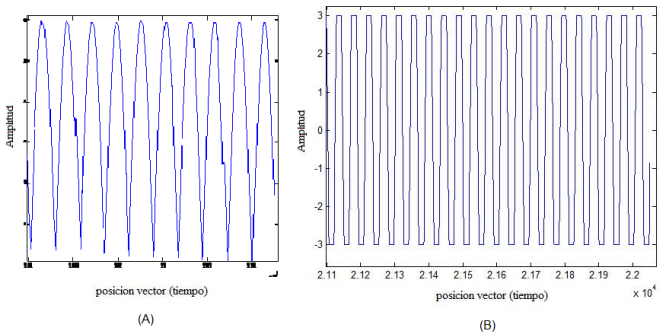

Figure 2. Output Distortion by Saturation

2) Delay

Delay is one of the simplest audio effects to implement and it is widely used in the music industry. It is created summing a signal with a copy of itself delayed a constant period of time (**Figure 3**). This is achieved saving a copy of the signal in memory and playing it back after the *delay time* along with the current signal.

The effect is implemented using a data memory as a circular buffer and making calculations in the time domain.

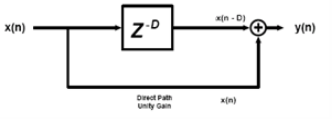

Figure 3. Delay Blocks Diagram.

This project has a delay time of 520ms

The simulation of this effect was tested in Matlab. The results showed that if the delay time was smaller than 100ms, it is not possible to differentiate the original signal against the delayed one, and if the delay time is bigger than 1 second, it makes very difficult to play an instrument or sing.

3) Reverse

Reverse uses the same principle of the Delay effect, but in this case the delayed signal is not played back normally but inverted, producing an interesting sonic effect.
To achieve this a copy of the original signal is stored in memory, and after a specified time defined by the number of samples, the audio that has been saved is played back over the current sound or original (**Figure 4**).

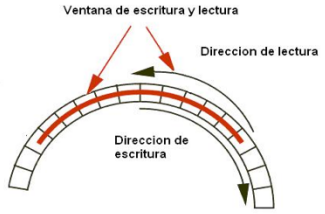

Figure 4. Reverse Effect Implementation

4) Low Pass FIR Filter

Choosing between a FIR and IIR filters depends on the nature of the problem. FIR filters are used in situations where a lineal phase is required into the pass band of the filter. Without this requirement both FIR and IIR can be implemented. Despite IIR filters are computationally less complex, it was decided to work with FIR filters to maintain phase lineal. Some audio applications use IIR filters resulting in unwanted phase distortions.

To design the filter, FDATOOL was used. The implemented method is *least-squares* and the filter parameters were Filter Order, Sampling Frequency (Fs), Band Pass Frequency (Fpass), Stop Band Frequency (Fstop), Band Pass magnitude (Wpass) and Stop Band magnitude (Wstop).

A low pass filter was designed with a sampling frequency of 100Khz, band pass frequency of 2KHz and a reject band frequency of 10Khz

5) Panning

Panning is achieved placing a mono channel in a stereo field. The effect is created manipulating the amplitude of the signal, when the amplitude of the left channel is attenuated, the virtual sound source moves to the right side proportionally to the attenuation.

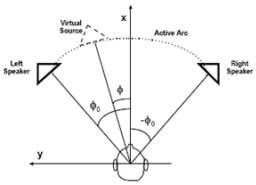

Figure 5. Panning Diagram

E. PC – FPGA Communication

The communication between computer and FPGA was implemented though the serial RS - 232 transmission protocol. The FPGA offers a serial RS – 232 port, however, the computer from which control data is sent and signals are monitored doesn't have this port.

To connect both devices, a connector cable TU-S9 was used as an adaptor from USB to RS – 232. With the required installation drivers, after the TU-S9 is connected, a virtual COM port is generated that allows a common connection with Matlab.

To set up the communication protocol with Matlab, a serial object has to be created.

```
s1 = serial ('COM1');
```

Then the serial object is associated with a physical port, which is used to set up the communication. In this case the synchronization speed between FPGA and the computer is 2400 bauds.

The computer sends one bit to start the communication protocol, the ASCII code of the predetermined characters and one stop bit. In total the FPGA receives and processes ten bits.

F. SYNTHESIS IMPLEMENTATION

The sound synthesis process is conformed by three fundamental blocks; Generator, process and output.

- A generator or source is something that produces a basic sound that can be molded later.
- A process is something that molds that basic sound to get an specific result.
- An Output is what interprets the result in the previous block to translate it in mechanical waves (air pressure).

The generator or source and the process might change depending on the type of synthesis used. For this dissertation project the basis of one Spectral Technique method was implemented. [5]

The source is a white noise sample. One of the main characteristics of white noise is that it has a very wide bandwidth and its frequencies have the same power.

To mold the sound coming from white noise, eight FIR filters were implemented, each one passing one of the frequencies corresponding to the natural musical notes. The following are the corresponding frequencies to each one of the musical notes.

DO = 261Hz, RE= 294Hz, MI = 330Hz, FA = 349Hz, SOL = 392Hz, LA = 440Hz, SI = 492Hz, DO = 522Hz.

Implemented filters are selective with an *order* of 5000. The next figure shows the response of the first filter with a pass frequency of 261 HZ.

DO

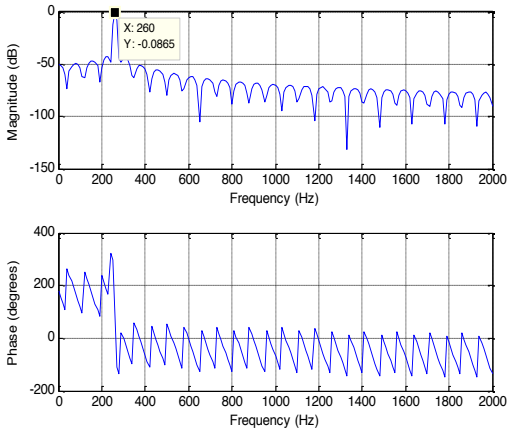

Figure 6. 261 Hz Filter (DO).

III. RESULTS

The effects that modify the amplitude or frequency of the signal are shown with pictures form a digital oscilloscope while the ones that modify the time are shown through pictures taken from audio software Sound Forge.

A. DELAY

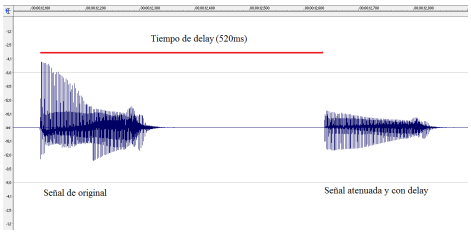

Figure 7. Delay implemented on an FPGA

For observation purposes on the delay effect, a musical note was played and then muted immediately to avoid the sum of the original signal with the retarded one.

Figure 7 shows the output signal when one note is played and after the specified time in the effect, it is reproduced again but with a lower gain.

The result of the implementation was a delay time between the original and the retarded signals of 520ms with 50% attenuation on the retarded signal, achieving the project's

expectative. These results are comparable with those obtained with the commercial effects processor Digitech RP100 as shown in **Figure 8** where the delay between the original and retarded signals was 400ms.

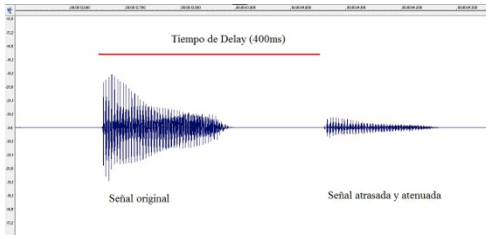

Figure 8. Delay on a Digitech RP100 processor.

B. REVERSE

Reverse effect involves a delay, reason why it was observed using audio software Sound Forge. Once again a musical note was played and immediately muted to appreciate the output.

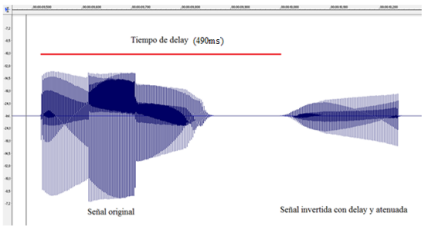

Figure 9. Reverse Effect

Figure 9 shows the output of the reverse effect, consisting of one original signal and a retarded copy of the original attenuated and inverted. Also it is observable how the retarded signal starts with small amplitude and then grows, confirming that it is inverted. In this effect, a delay of 490ms was implemented.

C. LOW PASS FILTER

Following, the data of the low pass filter implementation is presented. The table shows the values of simulation, implementation, absolute error and relative error in different points.

F (KHz)	Vin	Vout	Imp dB	Simulation dB	Absolut Error	Rel. Error
2	500	352	-3,04	-1,63	1,41	86%
2,5	500	345	-3,22	-1,88	1,34	71%
3	500	340	-3,34	-2,27	1,07	47%
3,5	500	328	-3,66	-2,63	1,03	39%
4	500	320	-3,87	-3,08	0,79	25%
4,5	500	305	-4,29	-3,6	0,69	19%
5	500	296	-4,55	-4,22	0,33	7%
5,5	500	272	-5,28	-4,9	0,38	7%
6	500	256	-5,81	-5,64	0,17	3%
6,5	500	240	-6,37	-6,45	0,08	1%
7	500	220	-7,13	-7,39	-0,26	3%
7,5	500	215	-7,33	-8,46	-1,13	13%
8	500	200	-7,95	-9,43	-1,48	15%
8,5	500	195	-8,4	-10,9	-2,5	22%
9	500	190	-8,4	-12,4	-4	32%
9,5	500	184	-8,68	-14,2	-5,5	38%
10	500	175	-9,11	-16,3	-7,19	44%
				Avg. Relative Error		28%
				Relative Error standard deviation		24,29 %

Chart 1. Results from filter designed with FDATOOL

Side frequencies in the filter have a bigger error value than mid frequencies. Lower frequencies suffer from saturation while higher ones are affected by noise caused by pre amplifiers, active filters, A/D and D/A converters.

D. DISTORTION

1) Absolute Value Distortion.

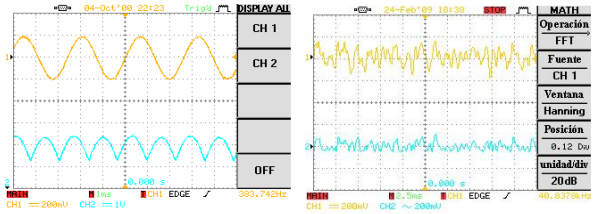

Figure 10 Distortion with an input signal of 400mVpp

Figure 10 Shows an input signal of 400 mVpp at 400 Hz (Top signal). The output signal (Bottom) is the absolute value of the input.

a) Spectral Analysis

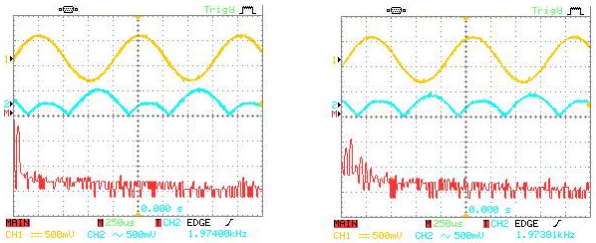

Figure 11. (a) Input signal spectrum y (b) Output signal spectrum.

Figure 11 (a) shows a sinusoidal input signal of 1Vpp at 1Khz. Using oscilloscope's FFT function it is observable how harmonics are generated with distortion effect (**Figure 11 (b)).**

2) Distortion by Saturation.

In this type of effect, input signal is trimmed to obtain a distorted sound.

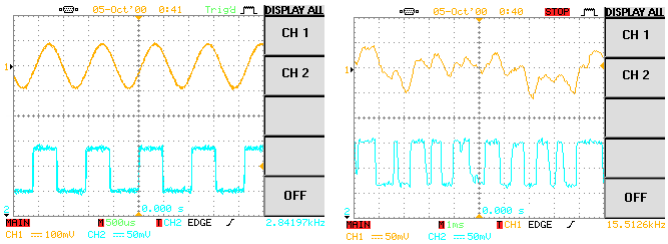

Figure 12. Distortion by saturation.

The more trimmed the signal is, more distortion is generated. In **Figure 12** input signal is 500mVpp and output signal is 150mV at a frequency of 1Khz.

a) Spectral Analysis

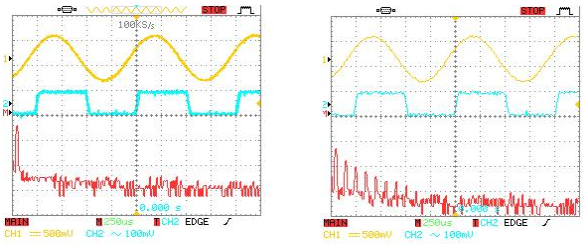

Figure 13 (a) Input Signal frequency Spectrum, **(b)** Output Signal frequency Spectrum.

Input and output signal frequency spectrum is shown in **Figure 13** (a) Input Signal frequency Spectrum, (b). Output graphic evidences how trimming the signal generates harmonics.

E. PANNING

1) Right Panning.

Figure 20 shows the two output channels of panning effect. The signal in the top of the graphic corresponds to the left channel and the bottom one to the right. Panning is expressed as percentage. To obtain a panning effect to the right, the signal of the left channel has to be attenuated. Attenuation percentage is proportional to signals' shifting to the opposite side of the channel that it is being panned.

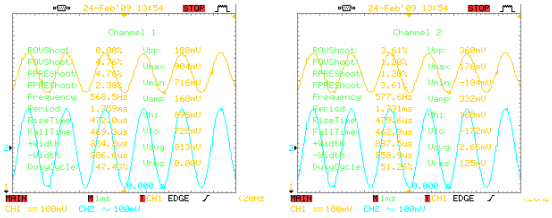

Figure 14. Panning 50% to the right

As a result of panning 50% to the right, there is a signal of 188mVpp in the left channel and a signal of 360mVpp in the right channel shown in Figure 14.

IV. CONCLUSIONS

- A digital signal processing system was created that serves as a starting point for future projects in audio in the Xavierian University. Regarding previous electronic engineering dissertation projects that focus in the same field, it is possible to think in the opening of a new research department on sound due to its growing relevance not just in music but also in electronic devices used daily around the globe like mobile phones and electronic agendas.

- When implementing the projects' FIR filter, two tools were tested to generate its VHDL code; the first one was Core Generator from Xilinx and the second one was FDATOOL from Matlab. Core generator has a very practical interface that allows a fast design for filters with few parameters and also generates an optimized output for the target FPGA. However due to the few parameters required to design filters it can't be specified the quantization of data resulting on filters that behave differently than the simulation ones. FDATOOL has an easy interface that allows the user to completely customize filters but its output is not in VHDL thus requires to be converted resulting in an output not optimized that consumes more data space in the FPGA. Finally the second option was implemented with few coefficients to avoid FPGA's resources saturation.

- FPGAs use in audio applications is becoming a very common practice because they provide excellent results. It could be expected that FPGAs get into the field of professional audio production **Error! Reference source not found.**].

- Designing tools like ChipScope Pro from Xilinx, Simulink and GUI from Matlab were a key factor to develop this project. ChipScope Pro allows the user to analyze data directly from the FPGA, Simulink offers graphic programming and GUI eases the graphical user interface design.

- It was observed that audio effects with delay times less than 100ms don't offer a clear differentiation between the original signal and the retarded one. Audio effects with times greater than 1 second make it hard to perform an instrument, so it's very important to manage a controlled range of time to produce Delay and Reverse effects.

- The processor contains two channels. When implementing the second channel it became evident the benefits of working on an FPGA; several VHDL predesigned blocks for channel one were used for building it.

- The combination of two or more effects generates new effects. This result can be observed in the Reverse effect, which consists of the backwards reading of a delayed signal.

V. REFERENCES

[1] Ingeniería De Sonido. (2007). Consultada en Enero de 2007. Universidad de San Buenaventura Bogota D.C. Página de la Facultad de Ingeniería. http://www.usbbog.edu.co/index.php?option=com_content&task=view&id=243&Itemid=230

[2] Digilent (2005). Digilent PmodAD1 Analog To Digital Module Converter Board Reference Manual. Consultado en junio de 2008.http://www.digilentinc.com/Data/Products/PMOD-AD1/Pmod%20AD1_rm.pdf

[3] Digilent (2006). Digilent PmodDA2 Digital To Analog Module Converter Board Reference Manual. Consultado en junio de 2008. http://www.digilentinc.com/Data/Products/PMOD-DA2/PMod%20DA2_rm.pdf

[4] Bemis, R., (2004). Audio Spectrum Analyzer. Obtenido en Marzo de 2007 de http://www.mathworks.com/matlabcentral/fileexchange/5572/

[5] , C. (2008). Sintetizadores: Una aproximación a la recreación del sonido. Consultado en Enero de 2008 de http://synth.claudiomomberg.com/sintesis.html/

[6] Villazon, Andrés F., Certuche, Leonardo. (2002). Procesador programable de efectos para guitarra en tiempo real PRO2GUIFECT-RT. Tesis de pregrado, Pontificia Universidad Javeriana, Cali, Colombia.

Authors:

Alejandro Valdés – Electronic Engineering Student Pontificia Universidad Javeriana.
alejandrovaldesalvarez@gmail.com

Esteban Aristizábal – Electronic Engineering Student Pontificia Universidad Javeriana.
earistizabal@puj.edu,.co

Hernan Benítez – Assistant Professor at Pontificia Universiada Javeriana Cali, hbenitez@javerianacali.edu.co

Michael Fernando Martinez – Assistant Professor Pontificia Universiada Javeriana Cali, mfromero@javerianacali.edu.co

Special thanks to **Leonardo Jaramillo**