Advanced MIDI Guitar Effects System

Ronan O'Malley
Final Year Project, Supervised by Dr Martin Glavin.
Dept. of Electronic Engineering, National University of Ireland, Galway

***Abstract*-This paper describes a unique digital effects system for guitarists, which is suitable for use in live performances. The project will allow a guitar player to easily configure effects in advance from a PC based graphical user interface, a feature that sets this system aside from conventional effects systems that use cumbersome dials, buttons and switches. This feature also allows the professional user to fine-tune specific parameters, inaccessible in traditional multi-effects units.**

A pedal board device provides real-time adjustment and selection of effects from a custom built effect database. Both means of control adopt the MIDI protocol, and therefore they may be complemented with any MIDI compatible device such as a sequencer or programme changer.

I. Introduction

The aim of this project is to research and develop a digital effects system with real-time MIDI (Music Instrument Digital Interface) control and PC configuration for the electric guitar.

Section II of this paper outlines the initial research and development conducted to confirm existing effects theory and develop unique algorithms that port easily to hardware. A set of variable parameters with boundaries is established for each effect to ensure straightforward manipulation for users who have no underlying knowledge of the algorithm, while allowing professional users advanced algorithm configuration.

The project proceeds with development of pedal board for real-time control, as outlined in Section III. The 8-bit micro-controller based control system utilises existing MIDI protocol. The signal processing functionality is controlled in two ways using this technology:

- Static on/off data sent from foot-switches, indicating when an effect should be enabled or disabled in real-time.
- Dynamic positional data sent from a variable pedal, this can alter an effect parameter during guitar play and provides a versatile performing environment.

A separate micro-controller is employed to perform switching and parameter change functions to accommodate the possibility of wireless communication between pedal-board and central signal processor. This link is simulated in this project with a wired asynchronous serial link to facilitate MIDI communication.

Effects developed in Matlab are re-coded in embedded in signal processing hardware. Section IV outlines this as well as the operating system and MIDI decoder control layers that are implemented on top of this audio processing data layer.

Section V concludes with a walkthrough of the MIDI connection from a PC to the DSP which provides a transmission path for a Java GUI to enable natural and flexible reconfiguration of effects, and storage of set "patches" which can be recalled at the push of a footswitch.

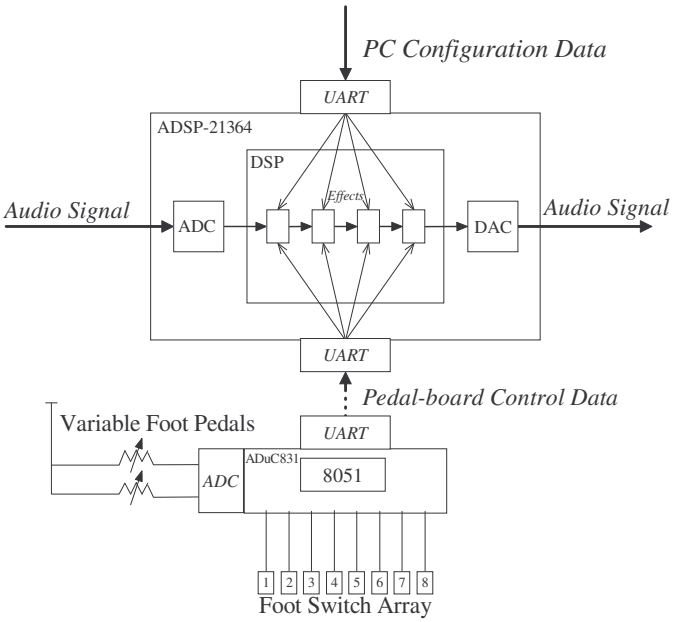

Fig. 1. System overview

II. *Effect Simulation*

A number of popular effects were simulated to prove the theory behind them, and establish a reliable and reproducible algorithm. A set number of variable parameters with clear boundaries were established for each effect, to aid in the process of embedding them.

A. Fuzz Distortion

This effect is also known as symmetrical clipping or the *Hendrix Effect*. It is a harsh distortion effect, any sample above or below a threshold is limited to that amplitude. The signal is then amplified to compensate for the loss of average signal magnitude.

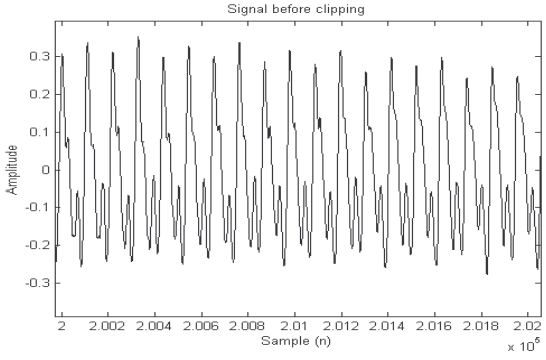

Fig. 2. Audio signal before application of fuzz algorithm

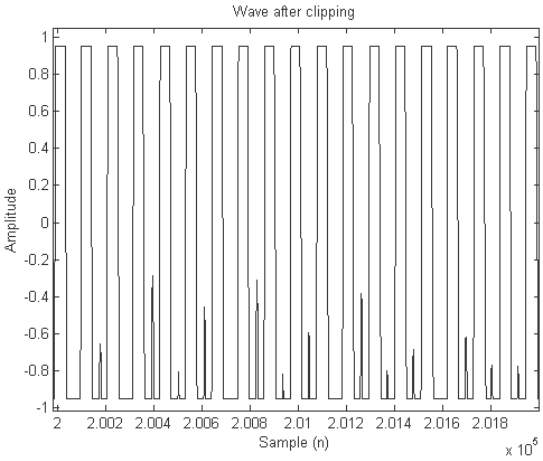

Fig. 3. Audio signal after application of fuzz algorithm

This alteration creates signal discontinuities, producing high frequency components that give the sound its distinct edge.

B. *Tremolo*

A tremolo effect is produced by applying a low frequency oscillating mask to the incoming signal. Triangle-wave oscillator and sine-wave oscillator variants were coded, however the triangle variant produced superior results.
The following parameters and boundaries were established through experimentation with Matlab:

- Frequency of oscillator – 2.4Hz to 24Hz.
- Wave shape of oscillator – Triangle is superior.
- Output may need to be amplified to compensate for overall loss of signal magnitude.

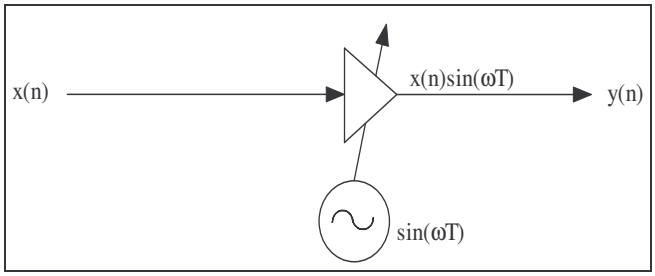

Fig. 4. Block diagram of tremolo effect

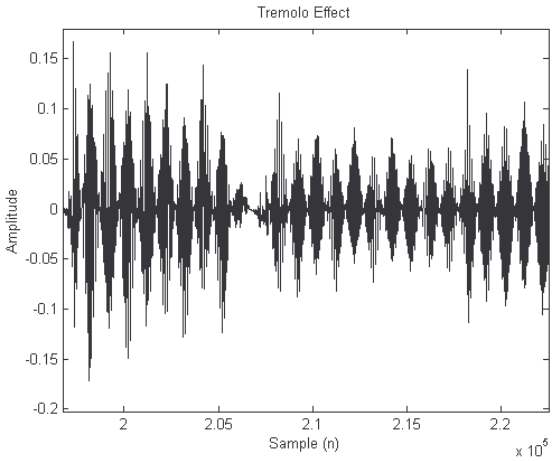

Fig. 5. Audio signal after application of triangular mask

C. *Delay*

This effect is created by adding delayed, diminished samples to the incoming signal. The following parameters were established to vary the effect:

- Delay period.
- Amplitude of first delay.
- Diminishing rate - this determines the amplitude of the next delay as a function of the current sample.
- No of delays – this is an optional constraint. Generally this would be determined by the diminishing rate reducing the delay amplitude to zero. However this parameter becomes important when embedding this effect.

Varying these parameters can produce a number of different effects. For example, one set of parameter boundaries can produce a traditional *delay* effect; another will produce an *echo* effect.

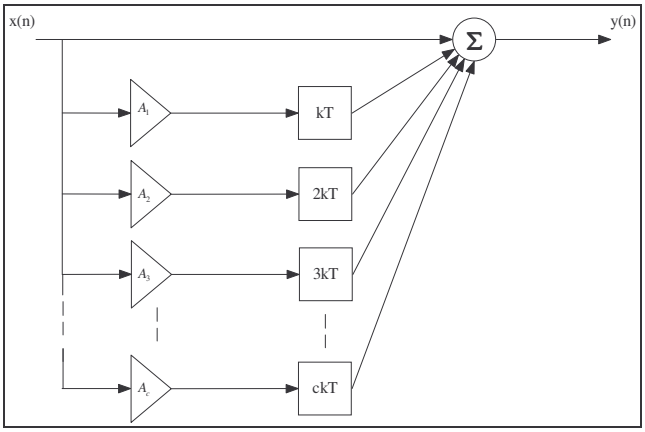

Fig. 6. Delay effect

$$y(n) = x(n) + A_1x(n-k) + A_2x(n-2k) + A_3x(n-3k) + \ldots + A_cx(n-ck)$$

k is the number of samples between delays. This is a function of the delay period *T* and the sampling frequency f_s.

$$k = T f_s \quad (1)$$

Delay range (ms)	Modulation	Effect Name
0 – 20	-	Resonator
0 – 15	Sinusoidal	Flanging
10 – 25	Random	Chorus
25 - 50	-	Slapback
>50	-	Echo

Fig. 7. Typical delay based effects [1]

D. Wah-Wah

A "wah-wah" effect was reproduced in Matlab by constructing a peak filter, and oscillating the filter's centre frequency through a set range. A peak filter is a band-pass filter with a narrow pass band and through experimentation, aided by some knowledge of the frequency content of a guitar sound, a parameter range of 500 Hz – 5 kHz was established for the filter's centre frequency. A state variable digital filter (*Figure 8.*) was employed to produce the band pass filter, and for simulation purposes the filter's centre frequency was oscillated between it's maximum and minimum bounds using a triangle wave as a reference. When implemented in hardware, user pedal input will dictate the filter's centre frequency.

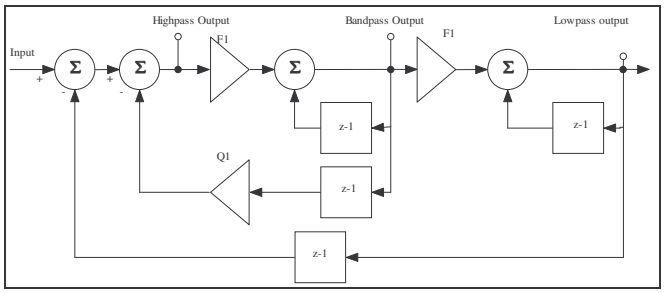

Fig. 8. State variable filter [1]

$$y_{bp}(n) = F_1 y_{hp}(n) + y_{hp}(n-1) \quad (2)$$

$$y_{hp}(n) = F_1 y_{lp}(n) + Q_1 y_{bp}(n-1) \quad (3)$$

$$F_1 = 2\sin\left(\frac{f_c \pi}{f_s}\right) \quad (4)$$

$$Q_1 = 2\varsigma \quad (5)$$

The variables associated with this effect are:

- Damping Coefficient (ζ) of the filter.
- Centre Frequency of the filter.

This is dynamically controlled by a variable foot pedal, creating the "wah" sound that gives the effect its name.

E. Flanger

3.1.5 Flanger

A flanger effect is constructed of a single delay, oscillated within a short range at a low frequency. Effect variables [1],

- Delay range (0 to 3ms)
- Oscillator (Sin wave, triangle wave)
- Oscillating frequency (1 Hz)
- Amplitude of the delay (70% of the input sample)

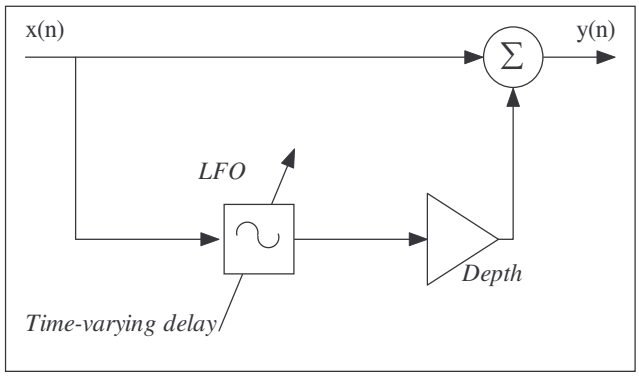

Fig. 9. Flanger effect

III. PEDAL-BOARD

The 8-bit assembly code system was embedded in an 8051 based microcontroller, within an Analog Devices ADuC831 development board. The pedal-board incorporates this microcontroller as it is designed to be a stand-alone entity, which could ultimately also control a wireless transmitter.

An operating system was coded to obtain data from the pedal board input equipment, convert the data into MIDI message format and transmit over an asynchronous connection to the DSP unit.

The real-time operating system firstly polls two variable foot pedals and compares the current value with the last transmitted value in memory. If the pedal's value has changed since the previous transmission, then the new value is converted into MIDI format, and three MIDI messages are sent. These messages indicate that:

- there is a control change
- the channel number
- the controller number of the pedal
- the pedals new value

The ADuC831's on board ADCs produce a twelve bit result upon reading the voltage from the on board potentiometer. To conform to MIDI protocol the seven most significant bits of the ADC reading are used to determine the pedals position, which provides a sufficiently accurate resolution.

A similar functionality is employed for the switches that control effect on/off state. Memory holds the most recent switch value transmissions. If the value of a switch has changed, three MIDI messages are transmitted corresponding to the switch's new value. The DSP board receives these messages, decodes them and applies the corresponding changes to the embedded effects.

IV. DSP System

Guitar effects outlined in Section II were embedded in signal processing hardware. The processor performs audio signal processing concurrently with asynchronous serial reception, MIDI message decoding in addition to control and parameter change.

The first objective in developing the audio processing system was to build a system that sampled audio through the on board ADC and passed it directly out to the DAC.

Reception of an ADC sample triggers a serial port interrupt. Another serial port is configured to transmit to two of the four on board digital to analogue converters. This allows for output of the processed audio to a guitar amplifier, and a set of headphones.

Upon execution of ADC interrupt, the talk-through system reads the current sample, and writes it to the DACs. It is between these two tasks that the effect processing takes place.

Concurrently to the audio processing, a software UART [2] provides asynchronous serial reception of MIDI data using synchronous serial ports, both from the pedal-board and PC GUI. MIDI data is buffered and decoded, after which the corresponding control changes are applied to the effect status and parameters.

V. Java MIDI Programmer

A graphical user interface program was coded in Java. Users make alterations to the effect setup, and these changes are sent in MIDI format over a serial link to the DSP. Java was chosen for ease of use, flexibility and provision of excellent API libraries. This program utilises the *javax.swing* API to provide a user-friendly interface to effect control, ensuring the underlying MIDI protocol is transparent to the user. The *javax.sound.midi* package of the Java API provides the methods and classes for dealing with MIDI device selection, opening ports, establishing connections and message generation. The java code defines how the setup configuration data is serialised into MIDI messages and transmitted.

VI. Conclusions

This paper has outlined a system that provides a unique and dynamic environment for guitarists, both novice and professional. Research has confirmed traditional effects theory and produced robust algorithms for hardware implementation. A standalone pedal-board offers real-time switching, and parameter change. A PC based GUI offers intricate customisation “off the shelf”, without the need to trawl through oceans of manuals. The adherence to MIDI protocol throughout allows the addition of other standard equipment such as synthesisers and programme changers.

VII. Extension Possibilities

It is desirable to have a wireless link between the pedal board and the DSP processor. This would allow a certain amount of freedom when playing, and positioning equipment. The wired serial link has been extensively tested, and RF link would be a beneficial extension to the project. The greatest challenge to establishing a wireless link would be adherence to the MIDI baud rate. Failure to exactly match the 31.25 kbps baud rate means buffering would have to take place.

References

[1] Udo Zölzer, *DAFX Digital Audio Effects*, Chichester, UK, Wiley, 2002.

[2] EE-191, Implementing a Glueless UART Using The SHARC® DSP SPORTs, Dan Ledger, Analog Devices, May 6, 2003. http://www.analog.com/UploadedFiles/Application_Notes/399447663EE191.pdf