PADAWAN: Parallel Accelerator for Digitising Audio with Attenuation of Noise

Ngonidzashe Mombeshora[†] and Mark Njoroge[‡]
EEE4120F Class of 2020
University of Cape Town
South Africa
[†]MMBNGO003 [‡]NJRMAR003

***Abstract*—Putting effects and filters on audio is a desirable component in any music system. The ability to choose a combination of different effects simultaneously and hear the effects in real time is even more desirable. However when doing this in software, the sequential nature of the processing would produce lag and latencies which make real time processing impossible. The PADAWAN accelerator is an FPGA based implementation that aims to achieve near-real time application of multiple effects and filters. The hardware used for the project is the Nexys A7 Board produced by Diligent. The goal of the project is to investigate the speed-up achieved when said effects are applied simultaneously to an input audio stream. A Golden measure will be implemented which will be the base of all comparisons with regards to execution time. The audio stream will be input via the XADC and shall be passed through the relevant filters, based on the user input. The data is then passed through the audio jack via a PWM signal. Comparisons are made on the execution and accuracy of both implementations with the goal of observing improved execution times on PADAWAN. Ultimately, this is an investigation into the possibilities of parallel processing such that processes can be done faster. This report outlines the methodologies and design schemes used in the development of the prototype.**

I. Introduction

PADAWAN is a digital accelerator for the processing of audio signals. The acronym standing for Parallel Digital Accelerator With Attenuation of Noise. The main reason behind its conception is that audio processing such as filtering and audio effect implementation done sequentially leads to output audio that is delayed. There is little influence the architecture(cascaded or parallel) used in implementing the filters and effects has on the delay observed when applied in software. The need for real-time output has brought about the motivation for producing a system which performs said operations at a faster execution time and hence parallelisation via an FPGA.

The main objective of this investigation is to create real-time audio processing on an audio live stream from a music video. The term audio processing in this report shall be referring to the application of filters and audio effects such as echo, reverberation, chorus, distortion and noise shaping. The creation of real-time output is to be achieved by the PADAWAN through parallelisation. Running audio processing tasks in parallel at the same time should significantly reduce the execution time as opposed to running the same processing tasks in cascade. Another objective is to implement noise shaping within the PADAWAN system, this is to reduce quantization effects and bit-depth reduction on the digital signal hence improving the quality of perceived output sound. [1] A further explanation and justification for noise shaping is explored further in section II.

II. Background

This project focuses on digital processing on audio signals in order to produce various desirable effects. In order to understand and implement such a process, the first step would be to review existing works. There are various sources available on the internet regarding digital and audio signal processing which all prove a strong starting foundation to work from.

A. Digital filtering

Filtering in the digital domain largely falls under two different categories. These are Finite Impulse Response(FIR) and Infinite Impulse Response(IIR) filters. As the name suggests, the difference between these type of filters lie in the behaviour observed at the input of an impulse [2]. IIR Filters have a recursive response that last longer than FIR filters [3]. Mathematically, the difference between the two types is in what is needed for the current output sample. With FIR filtering, the output is solely dependent on the previous inputs, whereas with IIR filtering, the output is dependent on both previous inputs and previous outputs [4]. Both filtering types have advantages over the other, and ultimately the choice on which filter type to use rests in the requirements of the system. For the purpose of this project, the FIR filtering scheme was chosen due to its stability and linear response which is as a result of the output relying solely on previous inputs.

B. Digital audio effects

Effects form a large part of audio signal processing. When they occur naturally, they usually cause unwanted audible interference with the desired sound. However, when used carefully and pointedly, they can create new sounds that would otherwise have been unachievable. The types of effects that can be employed range a wide span and are categorized by what the effect does on the audio signal [5]. These categories include Modulation effects, Time-based effects,

Spectral effects and Dynamic Audio effects [6]. Filters too are considered audio effects as they produce an audible change in the audio being produced. The categories all differ in the way they affect the audio signal with effects that are similar falling under the same category. An example of this would be a delay/echo and reverb which all fall under time-based audio effect.

C. Dither

Dither is low volume noise introduced into a digital audio signal when converting from a higher bit resolution to a lower bit resolution.This reduction in bit resolution introduces quantization errors rendering the output audio sound unpleasant to the human ear. [7] In the specific case of PADAWAN, the input audio signal is a 24bit signal, truncated, audio processed and output as a 16bit audio signal. To improve the apparent sound quality adding dither is required, this reduces the quantization effect.

III. Methodology

There are various methodology structures that could have been used in the design and development of the system, each with their own benefits and pitfalls. The methodology chosen for this project is the spiral technique. This would involve iteratively developing the accelerator and reviewing the progress made and how it matches up to the desired target. Figure 1 shows general application of the spiral technique. Using this technique, we would ensure that the final iteration of the accelerator would be the best version as it went through various testing and review phases.

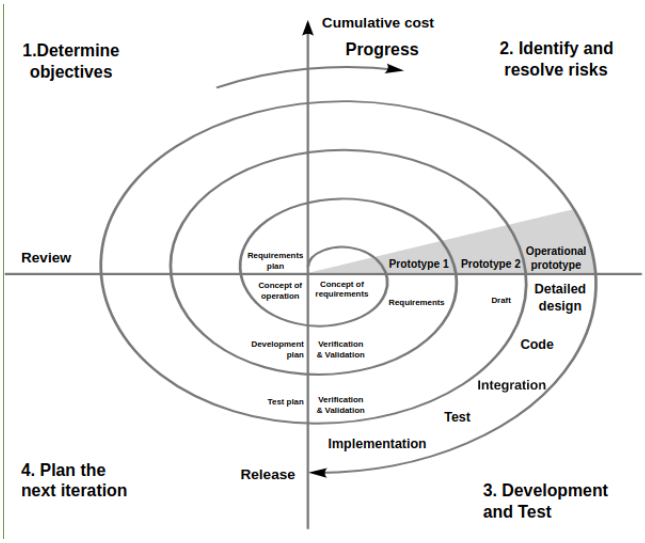

Fig. 1: Spiral Design Technique [8]

Developing the accelerator began from research as outlined in section II and understanding the requirements needed for the system.The following requirements were settled upon:

1) Successful implementation of at least one type of filter and one type of effect. e.g Low pass filter and an echo.
2) FPGA be able to implement these in real-time or with non-noticeable time delay.
3) FPGA should be able to take in an audio stream
4) FPGA should also be able to output an audio signal via PWM to the audio jack
5) Output must pass through a noise shaper in order to increase the apparent sound output resolution.
6) Input switches will be used to select filter and effect modes

As the focus is on speeding up audio processing, the first step was to produce a golden measure in software. This would finalise the algorithms that would then be converted to the hardware implementation. As the hardware being used was Xilinx's Nexys A7, there were resources and tutorials available online provided by Xilinx that formed a starting point in developing the hardware implementation. The details of the important sub sections in the methodology of the project are enumerated below.

A. Hardware

As the project focuses on audio processing, specific peripherals on the FPGA were inevitably going to be utilised. Most notably, this involved using the XADC to read the input data stream, and using the audio connector to output the processed sound to a speaker. The sound to be output would be produced via Pulse Width Modulation. The on board microphone could have also been used as an input for the audio stream. The system would also make use of switches to select both the input channel from the XADC and the filters and effects that would be desired by the user.

B. Filters and Effects

As previously mentioned, the initial process in developing the filters and effects to be used was to develop a golden measure. This was to be done in Matlab. Once the Matlab golden measure was implemented and met requirements, the functions used would be converted to Verilog modules. The modules would be developed one after the other, only moving onto a new one once a working version of the module had been produced. The effects and filters to be used implemented were chosen in such a way that the different categories would be represented. As such, the effects to be implemented were Echo, Chorus, Distortion and Reverb. The filters chosen were a low pass FIR filter and a high pass FIR filter. A dither filter was also planned to be included, provided time permitted.

C. Analysis of results

Results would be obtained from a structured experimental process. The golden measure would be implemented and timed. This would provide a basis for the accelerated implementation. The important aspect of this project would to be see how much faster the hardware implementation of the project would be than the golden measure. Calculating the speed-up would provide a good insight into the achieved

results. Using the following formula, the speed up would be calculated:

$$Speed - up = \frac{T_{goldenmeasure}}{T_{PADAWAN}} \quad (1)$$

IV. Design

This section deals with the design of the accelerator system, detailing the approach taken from handling the hardware and software interaction required for the PADAWAN to function as specified in sections III-A and III-B.

A. User input

1) Audio Stream: Input required for the PADAWAN includes a live audio stream fed into the FPGAs' 12 bit ADC. Some design considerations are to be made with regards to operating the ADC. The ADC has mainly two modes, unipolar mode and bipolar mode. When unipolar operation is enabled, the differential analog inputs (VP and VN) have an input range of 0V to 1.0V. In this mode, the voltage on VP (measured with respect to VN) must always be positive. When bipolar operation is enabled, the differential analog input (VP–VN) can have a maximum input range of ±0.5V. [9] The common mode or reference voltage should not exceed 0.5V in this case. Considering that the audio output from a host PC via an auxillary cable ranges from (by measurement)-2.0V to +2.0V centred around 0V, non of the above modes of operation can be directly implemented straight away. The solution to mitigate this problem it to introduce outside circuitry, more precisely introduce a voltage level shifter circuit. One can be easily created using some resistors and a BJT. Shifting the voltage by 2Volts and then attenuating it by a factor of 4, will results in a voltage ranging from 0V to 1V making it suitable input to the ADC for uni-polar operation mode.
The ADC has two sampling modes, mainly continuous sampling and event driven sampling, their names appropriately explaining the difference between the two. For the use case at hand continuous sampling mode is the chosen mode for the simple reason that the input audio stream is a continuous analog stream that needs to be continuously converted to digital form without the need for an event to have occurred.
It is to be noted that the on board ADC has maximum sampling rate capability of 1MSPS (1 Mega samples per second) and shall be run on a separate clock, (ADCCLK).

2) Effect Selection: The Nexys A7 is equipped with 15 slider switches, which are more than to enough for use as filter selectors. Switches SW[2:5] are going to be used as effect selectors for distortion, chorus, echo and reverberation in that order.

B. Software design

Upon receiving required data and input from the outside world the PADAWAN thereafter handles the major part of this investigation in software. This includes filter and effect implementation.

1) Filters and Effects: To achieve parallelisation, filters will be implemented as separate Verilog modules, to be instantiated within the top level module. As soon as data is sampled by the ADC its output will be continuously fed into the filter modules. Due to time constraints only a low pass filter shall be implemented as this is the most important one, serving the much needed purpose of noise attenuation.If time permits the noise shaping/dither filter would be of next priority. Details of the approach taken into the choice of filter design are outlined in III-B.

2) Input data buffer: To implement the effects and filters for this investigation, past samples read by the ADC are required. The idea of writing these to the FPGAs BRAM (Block Ram) is tempting. But considering the main goal of this investigation being to create a near real time audio output and execution of audio processing the drawback of using BRAM is the time delay it would introduce, caused by the writing storing and reading processes. This would introduce some uncertainy in the comparison of execution time with the golden measure, as the timing of the processes involving BRAM would be unknown.
The approach taken to reduce unwanted delays in the audio processing is to use an input data buffer. The buffer shall be made up of 21 registers and update themselves on every ADCCLK cycle. (This way buffer register 21 holds the sampled value from 21 ADCCLK cycles before) Using more registers would improve the apparent sound output for effects such as the echo.

C. Software implementation of filters and effects

1) Low Pass filter: The general mathematical equation for an FIR filter is

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

This can be better visualised by looking at the block diagram.

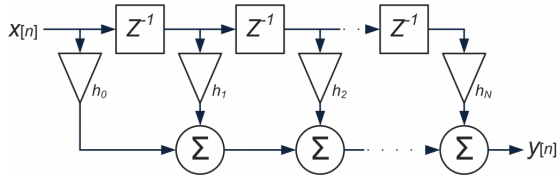

Fig. 2: FIR Filter Block Diagram [10]

The difference equation and diagram show that the output of the filtering algorithm is based on the sum of weighted previous signals, the weight is described by $h[k]$ and each sample has a specified weight based on the chosen cutoff frequency, filter order and other parameters. There exist Matlab functions that produce the required weights based on input parameters [11]. The functions would produce the desired coefficients to be multiplied by previous samples which would achieve the desired filter. The following code snippet shows the generation of the lowpass filter.

```
Fs = 48000;  % Sampling Frequency

N     = 10;     % Order
Fpass = 4000;   % Passband Frequency
Fstop = 10000;  % Stopband Frequency
Wpass = 1;      % Passband Weight
Wstop = 1;      % Stopband Weight
dens  = 20;     % Density Factor

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass Wstop], ...
           {dens});
```

The coefficients produced range from -1 to 1 so in order to implement this in hardware, they would be scaled. The golden measure for this filter was a matlab function. It was then converted to a verilog module and the following code shows the implementation of the filter:

```
module lpf(
    input clk,
    input [10:0] sample1,
    input [10:0] sample2,
    input [10:0] sample3,
    input [10:0] sample4,
    input [10:0] sample5,
    input [10:0] sample6,
    input [10:0] sample7,
    input [10:0] sample8,
    input [10:0] sample9,
    input [10:0] sample10,
    output reg [10:0]  output_sample
    );

    reg h1 = 16;
    reg h2 = 0;
    reg h3 = 23;
    reg h4 = 33;
    reg h5 = 290;
    reg h6 = 378;
    reg h7 = 290;
    reg h8 = 119;
    reg h9 = 23;
    reg h10 = 0;

    always @(posedge clk) begin
        output_sample <= ((sample1 * h1)>>10
                          + (sample2 * h2)>>10
                          + (sample3 * h3)>>10
                          + (sample4 * h4)>>10
                          + (sample5 * h5)>>10
                          + (sample6 * h6)>>10
                          + (sample7 * h7)>>10
                          + (sample8 * h8)>>!0
                          + (sample9 * h9)>>10
                          + (sample10 * h10)>>10)>>2;


    end
```

2) Distortion: Distortion is a dynamic effect that alters the waveform of the sound signal [12]. It can be achieved in various ways, for example, one can manipulate the amplitude or frequency of the original signal. A common amplitude distortion method is via signal clipping. This can be done by amplifying the input signal to the point that certain portions saturate, which is overdrive. The application of distortion in this project was a simple signal shift of the amplitude, provided a sample had crossed a specific threshold. This was chosen to amplify the perceived distortion to the human ear. This is done on each current sample. The module for this effect is shown as follows:

```
module distortion(
    input ready,
    input [10:0] sample_in,
    output reg [10:0] sample_out
    );


    always @(posedge ready) begin
        if (sample_in > 5000) begin
            sample_out<= sample_in - 10000;
        end
        else begin
            sample_out<=sample_in;
        end

    end


endmodule
```

3) Echo: An Echo is a time based effect that is perceived as a repeated signal. In order to achieve an echo, a previous sample of the signal would need to be added to the current sample. This is essentially delaying a signal. In order for the echo to be perceived by the human ear, the required delay would need to be $> 30ms$ [13]. A simplified block diagram of an echo is shown in figure 3.

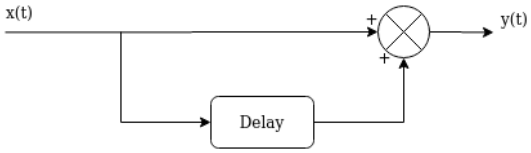

Fig. 3: Block Diagram of an Echo

Implementing the echo in a verilog module is shown below:

```
module echo(
    input clk,
    input [10:0] current_sample,
    input [10:0] previous_sample,
    output reg [10:0] echo_out

    );


    always @(posedge clk) begin
        echo_out <= (current_sample + previous_sample)>>1;
    end

endmodule
```

4) Chorus: The chorus effect is very similar to an echo. It can be described as a delay of the input signal with pitch modulation [5]. The intended effect to the ear is a range of different sounds sounding together, similar to that of a choir. The delay length needs to be long enough that the difference between the two signals is heard as a combination of similar sounds, but not too long such that it is perceived as an echo. The following verilog code shows the implementation of a chorus:

```
module chorus(
    input clk,
    input [10:0] current_sample,
    input [10:0] previous_sample,
    output reg [10:0] chorus_out

    );


    always @(posedge clk) begin
        chorus_out <= (current_sample + previous_sample)>>1;
    end

endmodule
```

D. Golden Measure

The following Matlab function shows how the golden measure was calculated. The distortion, echo, chorus and lpf are implemented the same way as the verilog modules described earlier. The function performs the desired effects and filter cascaded one after the other and measures the execution time for all the effects to be implemented on one sample.

```
function time = golden_measure()
    %generate random samples
    samples = rand(11, 1)*2 - 1;
    %filter coefficients
    h = [-0.0066 -0.0229 -0.0100 0.0957 0.2667 0.3548 0.2667 0.0957 -0.0100 -0.0229 -0.0066];
    tic;
    %distortion
    if samples(11) > 0.2
        out = samples(11)-0.7;
    else
        out = samples(1);
    end
    %echo
    out = (out + samples(1))/2;
    %chorus
    out = (out + samples(5))/2;
    %lpf
    out = (out + (h * samples))/2;
    time = toc();
end
```

E. Pulse width modulation

The PWM module is the last stop for the audio signal in PADAWAN's software. The pulse width modulation verilog module shown below converts the digital signal that has undergone the required audio processing to an analogue signal. This is then fed to an aux cable from the audio jack and fed into a speaker. It is to be noted that the on-board audio jack (J8) is driven by a Sallen-Key Butterworth Low-pass 4th Order Filter that provides mono audio output.The low-pass filter on the input will act as a reconstruction filter to convert the pulse-width modulated digital signal into an analog voltage on the audio jack output. [14]

```
module pwm_module(
input clk,
input [10:0] PWM_in,
output reg PWM_out
);

reg [10:0]new_pwm=0;
reg [10:0] PWM_ramp=0;
always @(posedge clk)
begin
    if (PWM_ramp==0)new_pwm<=PWM_in;
      PWM_ramp <= PWM_ramp + 1'b1;
      PWM_out<=(new_pwm>PWM_ramp);
end

endmodule
```

F. PADAWAN system

Figure 4 show the holistic view of the PADAWAN system as described by the subsections above. The shows the input audio stream and how it interacts with the aforementioned modules. As can be seen some modules and paths have a blue highlight, these ones were successfully implemented and performed as desired. The unhighlighted paths were unfortunately not implemented due to time constraints and these are dealt with in the Conclusion and recommendations section, VIII.

V. Proposed Development Strategy

The PADAWAN system as it is now is a basic level prototype for audio signal processing. However, it holds the potential to be groundbreaking if developed further to a point at which it would be a viable commercial product. A device that produces real-time multiple effects upon command would be highly appreciated by the performance and artistic industries. The prototype's main design could as well be expanded to incorporate a larger variety of more complex effects and filters. If developed right, the commercial imprint it would have would be highly distinguished amongst the existing competition. The goal in this scenario would be to

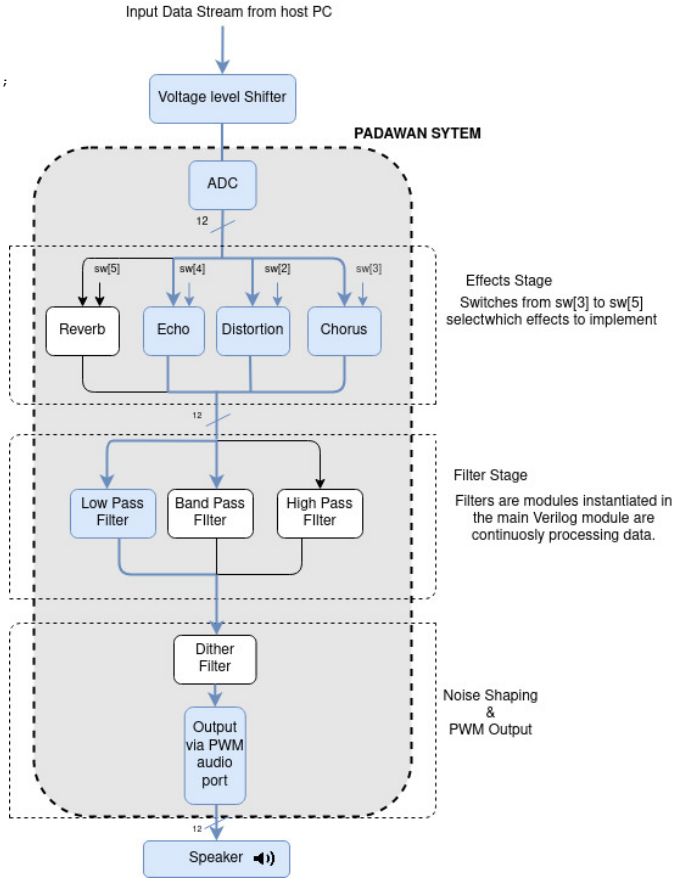

Fig. 4: PADAWAN block diagram

implement a digital sound mixing console that is FPGA based to be used by sound engineers.

A. Commercial design

As stated, the design would be upscaled massively to incorporate multiple filters and effects. This would require more input options. Moving away from the binary switches, rotary potentiometers and slider potentiometers would be used to increase flexibility in the degree of filters and effects. This would give the user higher control in the sound processing they require. The system could also be expanded to use a microphone as a source of input for the audio stream. Different output audio channels could also be incorporated such that different sounds could be played back and compared.

B. Commercial Usage

With the improved commercial design and direction taken, the PADAWAN system could be marketted as faster alternative to traditional mixers. The target market would be individuals who have a need for sound processing. The inclusion of a microphone and multiple channels aid in performance of audio processing on a live stage performance. The sound would be altered in real time with minimal latencies wwhich would appease the crowd. In a studio setting, the PADAWAN system would be used when producing audio signals and would create an overall higher efficieny in energy and time usage.

VI. Planned Experimentation

The section continues from on from section III, specifically dealing with the proposed setup for experimentation.

A. Experimental setup

First the golden measure is obtained. The sequential code for the golden measure was written and run in Matlab. The run-time required to perfom and complete the required audio processing is to be recorded. The same is done for the parallelised implementation in PADAWAN. The same data would be run on both systems. Comparisons on these obtained results is done by techniques such as speed-up calculations.

For the golden measure, data would be loaded into the Matlab Workspace and the *golden measure* function would be run. This would be to see execution time of the filters on a sample. The test data is a 12 second .wav file. The result from the operations can be played to hear the difference the effects made. Furthermore, the sound signal could be plotted in order to visualise the results. It is also vital to run the function multiple times in order to obtain a large sample of timings that would be produce of reliable average of execution times.

A similar procedure would be used to test the output in the accelerated implementation. The same sound signal would be played to the FPGA and the same effects would be applied.

To compare the accuracy of the prototyped PADAWAN system, an oscilloscope would come in handy. This would enable comparison of the audio stream to the PADAWANs output(Without any audio processing taking place). An oscilloscope will further provide timing information such the input to output delay. Unfortunately without access to the Oscilloscope this was not possible and an alternative approach was using an online tone generator to compare PADAWANs output to the original generated tone, this method is high dependant on the accuracy of the human ear.

The Oscilloscope would also be used to see how the output signal compares with the input signal after going through the various effects and filters. This visualisation would be compared the golden measure achieved to validate the same processes are being performed.

Another alternative solution to handling the delay measurement problem is to use a live stream audio coming from a video. For example a video of a person speaking and checking by eye and ear whether or not PADAWANs output is in sync with the video lip movement.

VII. Results and Discussion

This section outlines the resutls obtained from the designed golden measure and accelerated system.

A. Golden Measure

After multiple runs, the average speed of operation measured by the function is obtained to be

$$T_{goldenmeasure} = 12\mu s$$

. This speed is decent for small datasets and few filters, but once upscaled and implemented in parallel via software, the latencies increase and the result is far less desirable. Figures 5 and 6 show the signal before and after receiving the effects.

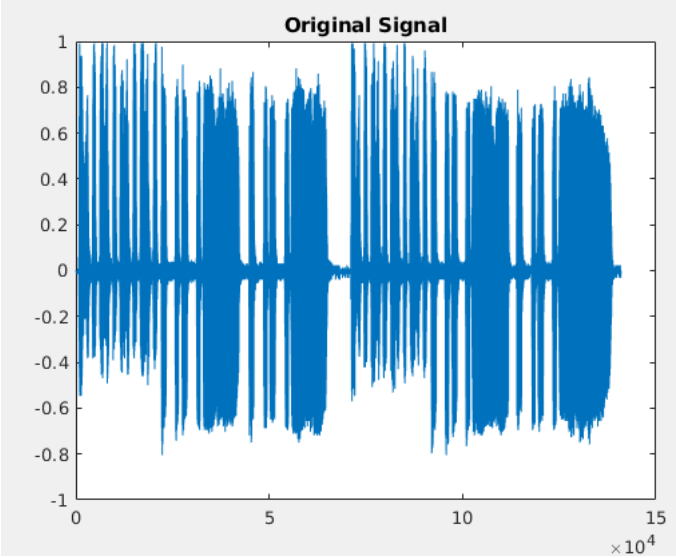

Fig. 5: Unfiltered Signal

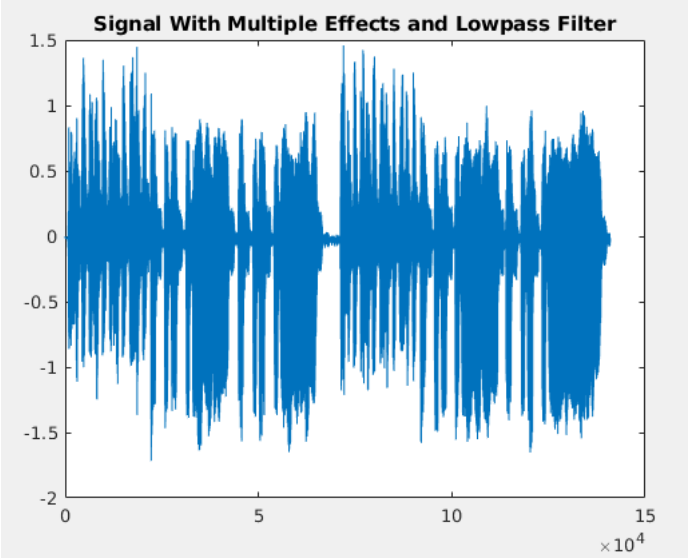

Fig. 6: Filtered Signal

The visual difference is not that evident but if closely analysed, the differences within the waves are seen. The signals above the desired threshold have been shifted down indicating the distortion effect. Echo and Chorus are harder to

spot due because of the high density of the sound samples. If the filtered sound is played out however, all the effects are heard clearly.

B. PADAWAN Implementation

PADAWANs most time consuming process involves the ADC. The ADC has two phases, mainly the aquisition phase and the conversion phase, this introduces some undesired latencies, that need to be regarded when obtaining and analyzing the speed-up calculated. [9] Other latencies include PWM during digital to analog conversion. With respect to ADCCLK the aquisitaion phase take 24 ADCCLK cycles and the conversion phase taking 22 ADCCLK cycles. The time to perform the required audio processing and output is only 2ADCCLK cycles. With the ADC sampling at a frequency of 25MHz(T = 40ns), therefore for speed-up calculations we obtain

$$T_{PADAWAN} = 1920ns.$$

Using equation 1 the calculated speed-up from from parallelisation using PADAWAN over the sequencial matlab golden measure is found to be 6.25. Which is remarkable considering this is including the latencies mentioned in above.By playing inputing video sound into the PADAWAN and checking if the output was in sync with video, verification of ”real-time” audio processing was confirmed.

VIII. Conclusion

The goal of this project was to implement audio processing in real time which is a difficult task when done on software. The materials used for this project was Matlab for the golden measure and Nexys A7 FPGA for the hardware implementation. Achieving audio processing on hardware was a target set out and as per the results, this was obtained. The audio stream was input to the system via an XADC and the various effects were implemented in parallel via multiple modules. The timings of both the golden measure and PADAWAN implementation show that a speed up was achieved and hence the main requirement of the system was met.

Additional objectives set out in the need for the user to be able to make their own choice and hear their chosen effect in real time was met via the use of input switches and previously mentioned speed-up. There was not enough time to produce a noise shaping dither filter, however this would be the first point of call if the system were to be improved. Furthermore, as described in section V, the system could be expanded further and include more customisable filters and effects to be used in industry. Another vital improvement that could be made to this prototype is the inclusion of the microphone available on the board

Overall, the project met the requirements that were identified for it and the process of applying filters and effects to an audio signal was accomplished on the FPGA.

Appendix

Link to code: GiThub Repository

References

[1] Wikipedia, “Audio signal processing,” 2020. [Online]. Available: https://en.wikipedia.org/wiki/Audio_signal_processing

[2] ——, “Digital filtering,” 2020. [Online]. Available: https://en.wikipedia.org/wiki/Digital_filter

[3] J. Fertier, “Introduction to filters: Fir vs iir,” 2020. [Online]. Available: https://community.sw.siemens.com/s/article/introduction-to-filters-fir-versus-iir

[4] A. Solutions, “Diference between iir and fir filters,” 2020. [Online]. Available: https://www.advsolned.com/difference-between-iir-and-fir-filters-a-practical-design-guide/

[5] L. Trandafir, “Audio effects: The beginner’s guide to shaping your sound.” [Online]. Available: https://blog.landr.com/audio-effects-plugins-guide/

[6] R. Chng, “How do audio effects shape the sound you create?” 2020. [Online]. Available: https://www.audiomentor.com/audioproduction/how-do-audio-effects-shape-the-sound-you-create

[7] “Dithering explained, what it is,when to use it and why it is important,” 2020. [Online]. Available: http://darkroommastering.com/blog/dithering-explained/

[8] Wikipedia, “Spiral model,” 2020. [Online]. Available: https://en.wikipedia.org/wiki/Spiral_model

[9] Xilinx, “7 series fpgas and zynq-7000 soc xadc dual 12-bit 1 msps analog-to-digital converter,” Xilinx Corporation, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

[10] H. Kibbe, “Finite impulse response using apple’s accelerate framework - part 1,” 2014. [Online]. Available: http://hamiltonkibbe.com/finite-impulse-response-filters-using-apples-accelerate-framework-part-i/

[11] Mathworks, “Lowpass design in matlab,” MathWorks, 2020. [Online]. Available: https://www.mathworks.com/help/dsp/ug/lowpass-filter-design.html

[12] D. Dixon, “What is distortion in music? and how to use it,” 2018. [Online]. Available: https://www.izotope.com/en/learn/what-is-distortion-in-music-when-and-how-to-use-it.html

[13] H. Audio, “Echo effects,” 2018. [Online]. Available: https://www.hackaudio.com/digital-signal-processing/echo-effects

[14] Xilinx, “7 series fpgas and zynq-7000 soc xadc dual 12-bit 1 msps analog-to-digital converter,” Xilinx Corporation, 2018. [Online]. Available: https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual