A Guide on Using Xilinx System Generator to Design and Implement Real-Time Audio Effects on FPGA

L. Merah, P. Lorenz, A. Ali-Pacha, and N. Hadj-Said

***Abstract*—The enormous progress in communication technology has led to a tremendous need to provide an ideal environment for the transmission, storing, and processing of digital multimedia content, where the audio signal takes the lion's share of it. Audio processing covers many diverse fields, its main aim is presenting sound to human listeners. Recently, digital audio processing became an active research area, it covers everything from theory to practice in relation to transmission, compression, filtering, and adding special effects to an audio signal. The aim of this work is to present the real-time implementation steps of some audio effects namely, the echo and Flanger effects on Field Programmable Gate Array (FPGA). Today, FPGAs are the best choice in data processing because they provide more flexibility, performance, and huge processing capabilities with great power efficiency. Designs are achieved using the XSG tool (Xilinx System Generator), which makes complex designs easier without prior knowledge of hardware description languages. The paper is presented as a guide with deep technical details about designing and real-time implementation steps. We decided to transfer some experience to designers who want to rapidly prototype their ideas using tools such as XSG. All the designs have been simulated and verified under Simulink/Matlab environment, then exported to Xilinx ISE (Integrated Synthesis Environment) tool for the rest of the implementation steps. The paper also gives an idea of interfacing the FPGA with the LM4550 AC'97 codec using VHDL coding. The ATLYS development board based on Xilinx Spartan-6 LX45 FPGA is used for the real-time implementation.**

***Index Terms*—Digital audio processing, echo, flanger, FPGA, XSG, ISE, VHDL, ATLYS, xilinx, digilent, real-time.**

I. Introduction

Sound is an integral part of our lives, it is one of our most important ways of sensing the world around us. Moreover, it is fundamental to our communication with our surroundings and fellow human beings [1]. s communication technology evolves; digital audio processing has gained great importance recently, it covers many diverse fields, its main aim is presenting sound to the human listeners and human-machine communication, it is a branch of digital signal processing which consists of applying mathematical models to an audio signal. Many daily applications are based on digital audio signal processing, such as:

- *Audio broadcasting*: Broadcasting audio signals over satellites and cellular networks need audio processing to enhance their fidelity or optimize for bandwidth, as an example audio compression plays an important role in mobile multimedia broadcasting [2].
- Audio synthesis: Digital audio synthesis is a very broad topic with a great deal of theory, mathematics, and engineering [3]. It is the process of generating audio signals electronically (such as music and human speech), music is the big area using audio synthesis, which is the heart of most musical instruments (synthesizers). Speech recognition is also a part of an audio synthesis, It consists of algorithms that have the ability to identify words and phrases, it has a large usage (crime investigation, search for reports or documents on the computer, give commands to a machine, authorizing access, etc.).
- Audio effects: This is the subject of this work, audio effects are algorithms that used for improving, enhancing, filtering sounds using some control parameters. Today these algorithms are widely used in professional or home music production studios, electronic or virtual musical instruments, and all kinds of consumer devices, including video game consoles, portable audio players, smartphones, or appliances [4]. Audio effects can be classified by the way do their processing [5]; basic Filtering (Lowpass, Highpass filter etc, Equalizer), Time-Varying Filters (Wah-wah, Phaser), Delays (Vibrato, Flanger, Chorus, Echo), Modulators (Ring modulation, Tremolo, Vibrato), Non-linear Processing (Compression, Limiters, Distortion, Exciters/Enhancers), and some special effects (Panning, Reverb, Surround Sound).

In the last decade, it has become apparent that digital electronics are integral parts of our everyday lives. Today, computing power is dramatically evolved. A lot of dedicated algorithms and mathematical systems are ready to use without prior knowledge of programming, simple commands are ready to use while hiding behind a big number of instructions on nowadays high-level synthesis tools. Computation in digital electronics is performed using two main ways; hardware and software. Computer software could support performing different and very complex tasks at the same time with a high degree of flexibility to modify. These features serve digital audio processing; most audio processing systems are software-based implementation. On the other hand, hardware-based implementation is better in terms of performance, real-time audio processing, especially online broadcasting that requires this kind of implementation.

The share of Programmable Logic Devices (PLD),

Manuscript received June 12, 2021; revised August 12, 2021.

L. Merah is with the Department of Electronics, Faculty of Technology, University of Laghouat, Laghouat 03000, Algeria (e-mail: l.merah@lagh-univ.dz).

P. Lorenz is with the Institut de Recherche en Informatique, Mathématiques, Automatique et Signal (IRIMAS), University of Haute Alsace IUT, 68008 Colmar, France (e-mail: pascal.lorenz@uha.fr).

A. Ali-Pacha and N. Hadj-Said is with the Department of Electronics, University of Science and Technology of Oran (USTO), Oran 31036, Algeria (e-mail: a.alipacha@gmail.com,nim_hadj@yahoo.fr).

doi: 10.18178/ijfcc.2021.10.3.577

especially FPGAs, in the semiconductor logic market is tremendously growing year-on-year [6]. FPGAs are truly revolutionary devices that blend the benefits of both hardware and software. They implement circuits just like hardware, providing huge power, area, and performance benefits over software, yet can be reprogrammed cheaply and easily to implement a wide range of tasks [7]. Initially, FPGAs are integrated circuits with no mission, contain a huge number of reconfigurable logic blocks and interconnects that can be programmed by a Hardware Description Language (HDL) such as Verilog and VHDL to perform a specific function. Each FPGA vendor has its own synthesis tool enabling the developer to synthesize ("compile") their designs through a number of steps until generating programming file. Synthesis tools have recently evolved dramatically, they permit today achieving complicated designs even without prior knowledge of HDL programming.

The aim of this work is not to propose new ideas in the context of the audio processing domain, but to present a design guide helping developers to benefit from the Xilinx design tools for achieving their designs. For a design guide to be more practical and helpful, it should contain all the designing steps; design, simulation, synthesis, and real-time implementation. Choosing the audio processing subject as an example of implementation on FPGA is due to the limited number of works in this context.

The paper is organized as follows: section II is devoted to presenting the design tools and hardware used for the implementation, it is the initialization phase in which, the most important electronic parts are tested before any other step. We present the LM4550 AC'97 audio codec chip, and how using VHDL code to interface it with the FPGA chip, this part is the important one because it converts the incoming audio signal from outside from analog to digital and sends it to the FPGA for processing. At the same time, it ensures the conversion from digital to analog if the audio signal after processing step and send it to the outside. We create a project under the ISE tool and add the VHDL component to drive the LM4550 chip, then after some steps we configure the ATLYS board and verify the correct working of the hardware by sending and hearing back an audio signal.

Section III is devoted to presenting the implementation steps of some audio effects' systems, namely the Echo and Flanger effects using XSG. The theoretical background of these effects and details of implementation with simulation results is presented in this section.

In section IV, the final design steps are presented, we assembled the two designed systems in one system, then generated the equivalent VHDL code automatically from the XSG blocks. The generated code is exported as a component to the previously created project under ISE. After the configuration of the ATLYS board using the generated programming file; the real-time hardware evaluation is done by hearing the results on speakers.

II. Software and Hardware Initialization

This section is devoted to presenting the design tools and hardware used for the implementation. More precisely, it presents the preparation phase of hardware and software tools. As mentioned previously; designing audio effects systems on this work are based on XSG tool.

XSG is designed to be a part of the Simulink library that addressed to hardware developments. It is a system-level modeling tool that facilitates the FPGA hardware design. It extends Simulink in many ways to provide a modeling environment that is well suited to hardware design. With the advent of the XSG, system architects and FPGA developers could collaborate in the Simulink environment, using two major components as follows: The library of dedicated blocks, Xilinx Block set, for model building in the Simulink environment; and HDL generator that uses the Xilinx optimized IP algorithms and generates the synthesized HDL code with consequent physical implementation of the project as FPGA, using the Smart-IP [8].

The design steps are as following : Designing audio processing (audio effects) systems using XSG blocks library, simulation of the designs under Simulink for functional verification, exporting the designs as synthesized VHDL codes to the ISE tool, adding some additional VHDL codes (LM4550 chip driving, components instantiation, ..) and constrains files, generating the *bitstream* for programming the FPGA, and finally the real-time verification by loading an audio signal and hearing the result on speakers.

A. The LM4550 Chip Driving

The ATLYS board includes a LM4550 AC'97 audio codec chip (Fig. 1) with four audio jacks for *line-out*, *headphone-out*, *line-in*, and *microphone-in*. Audio data at up to 18 bits and 48 KHz sampling is supported, and the audio in (record) and audio out (playback) sampling rates can be different [9]. The LM4550 was designed specifically to provide a high quality audio path and provide all analog functionality in a PC audio system. It features full duplex stereo A/D's and D/A's and an analog mixer with 4 stereo and 3 mono inputs [10].

The first thing that we should to do is to ensure the driving of the LM4550 AC'97 codec chip using VHDL, it is the main part in the design, because it ensures the conversion of the audio signal from analog to digital and from digital to analog after processing, with additional operations such as amplification. In this work, the driving of the LM4550 AC'97 is based on the VHDL code written by Tony Storey and Scott Larson [11]. The principle of working of this code is summarized in Fig.2, The controller presented in this figure has the following inputs and outputs :

- *Clk*: The main clock input comes from the 100 MHZ ATLYS on-board oscillator.
- *Reset*: The initialization signal comes from a push-button on the board.

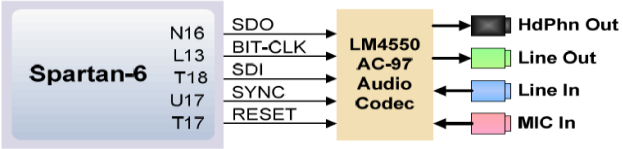

Fig. 1. The ATLYS board LM4550 AC'97 codec connectivity [9].

- *SDATA_IN*: The serial data input signal representing the input audio signal after the Analog/Digital conversion process over the LM4550.
- *12.88 clock*: A 12.288 MHz clock comes from the LM4550 chip.

- *Source*: a 3 bit signal to select the audio signal source (microphone line, input line, etc.).
- *Volume*: a 5-bit signal to control the sound level.
- *SYNC*: a synchronization signal, this signal is normally a 48 kHz positive pulse with a duty cycle of 6.25 % (16/256). *SYNC* is sampled on the rising edge of *AUDBIT-CLK*, and the codec takes the first positive sample of SYNC as defining the start of a new AC Link frame (see [9]).
- *SDATA_OUT*: The serial data representing the audio output signal sent to the LM4550 chip for the Digital/Analog conversion after the processing stage.
- *RESET*: Initialization signal to reset the conversion operation.

There are two internal signals to synchronize the main AC'97 controller with the command state machine AC-97_CMD" (a small FSM (Finite State Machine) to set up codec's registers). One of these signals pulses every 20us and the other is a signal used for error checking during the tag phase, for more details on the serial frame input/output see [10], [12].

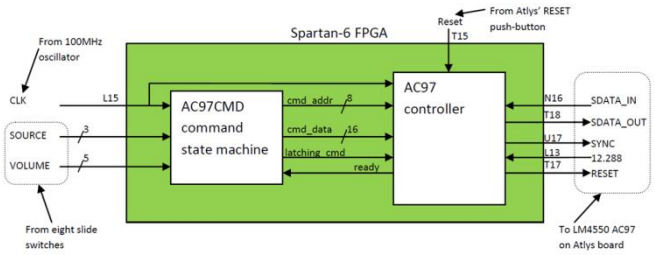

Fig. 2. The VHDL driving code for the LM4550 AC'97 codec working principle designed by Tony Storey and Scott Larson [11].

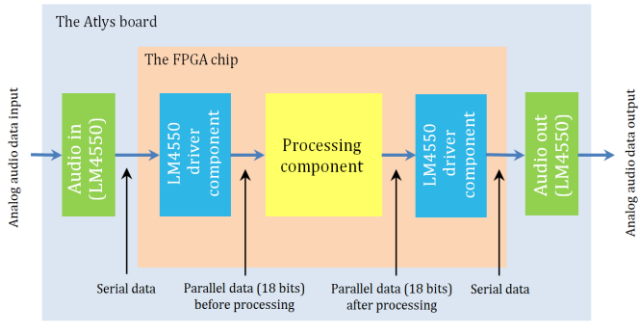

Fig. 3. Basic diagram of the FPGA-based audio processing system.

Before starting the implementation of the desired algorithms, it is very important to practically check the proper working of the hardware (the ATLYS board and the LM4550 chip) as well as the controller of the LM4550 driver written in VHDL. To carry out this operation, we used the ISE compiler of XILINX according to the following steps:

1) Creation of a new project under ISE.
2) Addition of new VHDL source *top.vhd* to collect the different components.
3) Addition of the LM4550 driver VHDL component written by Tony Storey and Scott Larson [11] to the created project.

In this phase, the aim is to test the LM4550 chip before any other step, so we drive the data issued from the LM4550 ADC directly to the LM4550 DAC without any changes. The LM4550 can convert 18-bit of parallel data to serial data to interface the LM4550 chip with the FPGA and recover the 18-bit data serially after the processing phase, in fact, there is no processing in this stage; the 18-bit audio signal is driven directly to the output Fig. 3.

Regarding the sound level control, we avoided using the switches for that as done in [11], we proposed a more practical method where we invoked two push buttons on the

ATLYS board, the following VHDL code is proposed to control the volume with the buttons; where *clk2h* is a clock which has a frequency of 2 Hz (0.5 seconds), this clock is used to control the rate of changing of the sound level if we continuously pushed the buttons *btn1* and *btn0* (see the following listing).

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sound_vol_proc: process(clk2h, btn1, btn0,
                max_vol, min_vol)
begin
 if rising_edge(clk2h) then
       if (btn1 = '1' and max_vol ='0') then
       Vol <= Vol + 1;
       elsif (btn0 = '1'and min_vol ='0') then
       Vol <= Vol - 1;
       end if;
end if:
end process Sound_Vol_proc;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
max_vol <= '1' when Vol = "11111" else
                '0';
min_vol <= '1' when Vol = "00000" else
                '0';
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

After instantiating different components, we added the constraints file (\textit{.ucf}) that must be utilized to assign FPGA I/O pins correctly. After the synthesis process, the final step is the generation of the programming file (bitstream) to configure the FPGA. After the configuration; we use a laptop as an audio source and a speaker to playback the audio signal. We also used an oscilloscope to visualize the audio signal in real-time as shown in Fig. 4.

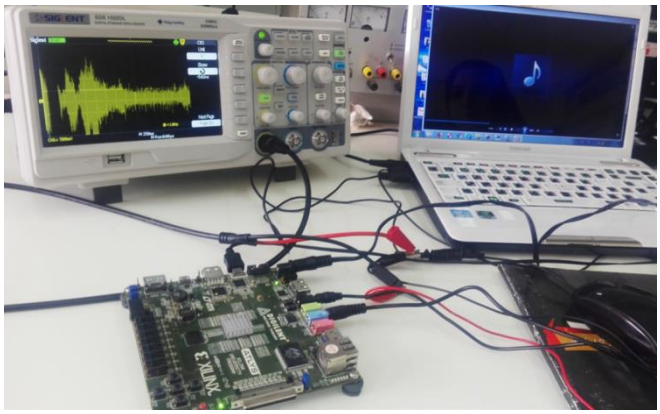

Fig. 4. Real-time verification of the hardware used for the implementation.

III. Audio Effects Implementation Using XSG

This section is devoted to presenting the implementation steps of some audio effects' systems, namely the Echo and Flanger effects using XSG. The systems are designed and simulated separately under Simulink, then collected in one system for implementation later. After simulation, the whole system is then exported as one component to the previously created project under ISE.

A. The Echo Effect System's Designing

One of the simplest effects yet most ubiquitous is the Echo effect, which is sometimes also called the delay effect. It can be found in many different applications. In this section, we will present the steps for implementing the system which adds the echo effect to an audio signal on FPGA.

Let us first recall the operating principle of the Echo effect, this effect can simply be considered as a delay of the audio signal Fig.5. It is produced by repeating the original audio signal after a fixed period of time. This effect is extremely applied in microphones and stereos. A FIR filter with a single delay will achieve this effect. The difference equation for the FIR filter can be written as follows:

$$Y[n] = X[n] + a \times X[n - D] \quad (1)$$

where $Y[n]$ is the output audio signal, $X[n]$ the input audio signal, D the delay applied, and a the gain of the echo effect.

The key to implementing the Echo effect is memory; we need to have a memory to save a version of the audio signal (delay) to add it to the original signal after a certain time. Note that the memory size represents the Echo delay, in other words; it represents the D samples. Indeed, for memories of small sizes, one does not need an external memory, FPGAs provide embedded memories called Block-RAMs or B-RAMs. The XC6SLX45 FPGA has 2.1 Mbits of fast block-RAM. The Fig.6 shows the XSG-based design of the echo effect system. Details of some blocks are given as follows:

- *From Multimedia File*: The role of this block is to import the audio signal from a file, the audio signal is mono, with 44100 Hz sampling frequency and 16 bits of size.
- *Sound_in*: This block has the role of converting the 16-bit audio data of double-precision to an 18 bit fixed point precision.
- *Gain a*: Represents the gain in the attenuation of the Echo effect.
- *Single Port RAM* : This block represents the configuration interface of the used memory, this block can be configured as shown in the Fig.7; where *depth* represents the memory size, *D = 10000*. The delay r can be calculated as follows: $r = D/44100 = 226\,ms$. For more information about the configuration of the BRAM, see [13].
- *Counter* : It is a counter to control the addresses of memory (memory reading).

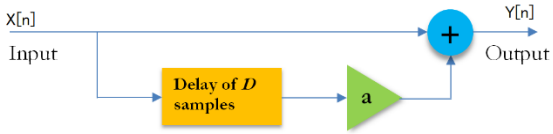

Fig. 5. Basic diagram of the echo effect system.

- *Sound_out* : This block has the role of converting back the 18-bit fixed-point precision to double precision.
- *To Audio Device*: This block has the role of exporting the processed signal to the PC sound equipment (sound card) to listen to the result in real time.

The obtained result of simulation was sent to the speaker and showed clearly the Echo effect on the original audio signal. A portion of the obtained result was also presented on the Fig. 8.

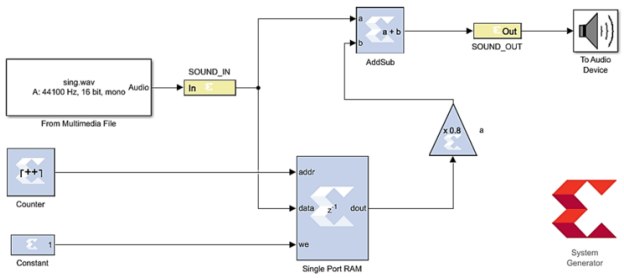

Fig. 6. The XSG-based design of the Echo effect system.

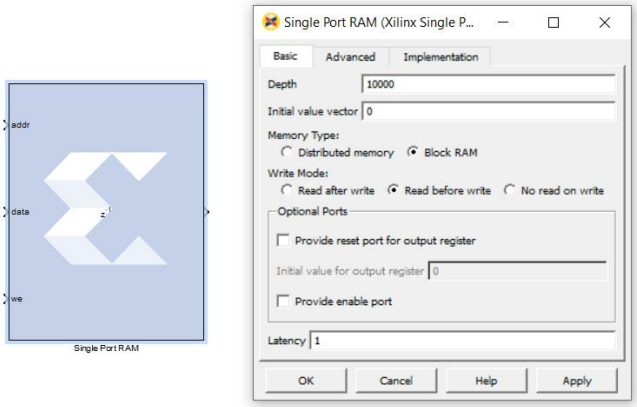

Fig. 7. The configuration interface of the BRAM.

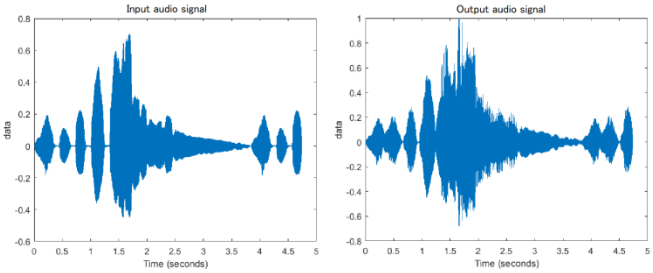

Fig. 8. The Echo effect system simulation result.

B. The Flanger Effect System's Designing

The Flanger effect is mathematically similar to the Chorus effect and the Echo effect, but it sounds quite different. The Flanger effect is similar to what we experience when a jet plane flies over our heads. The direct sound from the jet is mixed with the sound reflecting off the ground, which has a time-varying delay due to the jet's movement. Like the Chorus effect, the Flanger effect is quite popular with guitar players and it can frequently be found in their pedalboards and in their racks [1].

Contrary to the conception of the echo effect system where the value of delay D is fixed, the Flanger filter has a variable delay, it changes periodically. The difference equation for the Flanger filter can be written as follows:

$$Y[n] = X[n] + a \times X[n - D(n)] \quad (2)$$

The delay $D[n]$ is no longer fixed, it is periodic, in general, a sinusoidal oscillator is used for this. Flanger emulation is performed using a variable delay line controlled

by the oscillator. Fig.9 shows the basic structure Flanger effect system.

The configuration of the Flanger effect system presented in Fig. 9 has several variable parameters. The depth values for the Flanger effect generally vary from 1 to 10 ms with a delay center of 5 ms or less and a very slow scan rate (0.15 Hz). The delay line output is mixed with the input signal and the relative mix level (gain) value controls the strength of the Flanger effect. Decreasing this value creates a more subtle Flanger effect. The sweep oscillator is used to move the variable tap back and forth around a nominal delay time (center). A typical sweep rate is slow, around 0.1 to 0.5 Hz, but can vary from 0.01 Hz up to around 10 Hz at the maximum [14].

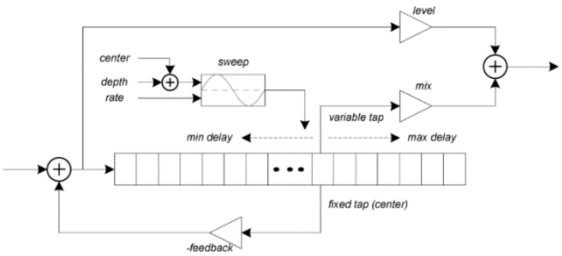

Fig. 9. The Flanger effect system's basic structure [14].

The implementation of the Flanger effect system using XSG is almost similar to the Echo effect except the delay is now variable and controlled by a sinusoidal oscillator. There are several methods to implement trigonometric functions in XSG, one of these methods is using a Direct Digital Synthesizer (DDS).

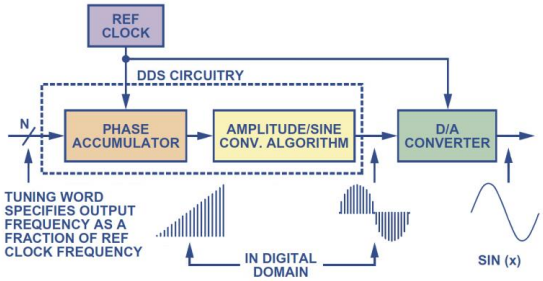

Fig. 10. Signal flow through the DDS architecture [15].

A DDS is a method used to create arbitrary waveforms from a single fixed frequency reference clock. It is used in applications such as signal generation, local oscillators in communication systems, function generators, mixers, modulators, sound synthesizers [15]. A major advantage of a DDS is that its output frequency, its phase, and its amplitude can be manipulated precisely and quickly under a digital control of a processor. Other inherent attributes of DDS include the ability to tune into the extremely fine frequency and phase resolution, and to "jump" quickly between frequencies. These combined features have made this technology popular in military radar and communication systems. The basic principle of a DDS is shown in Fig. 10.

The phase accumulator is actually a module-M counter that increments its stored number each time it receives a clock pulse. The magnitude of the increment is determined by the binary coded input word (M). This word forms the phase step size between reference-clock updates; it effectively sets how many points to skip around the phase wheel. The larger the jump size is, the faster the phase accumulator overflows and completes its equivalent of a sine-wave cycle [15]. The sinusoidal wave frequency can be calculated as follows:

$$F_{out} = \frac{\Delta\theta \times F_c}{2^N} \tag{3}$$

where:

F_{out} : The output frequency of the sine wave.
$\Delta\theta$: Phase increment value.
F_c : The reference clock frequency.
N : Length of the phase accumulator in bits.

Any changes in the value of $\Delta\theta$ leads to immediate and continuous changes in the output frequency.

In fact, XSG contains a very rich DDS block, the Fig.11 presents the block as well as the main window for its configuration, where we can define the phase and the amplitude sizes of the output in binary, for more information about XSG DDS compiler, see [16].

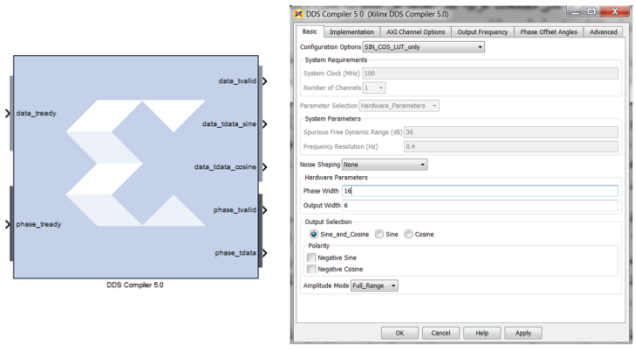

Fig. 11. XSG DDS block with its main configuration window.

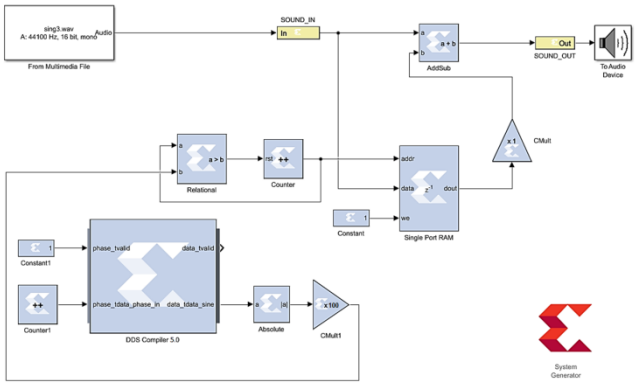

Fig. 12. XSG-based Flanger effect system.

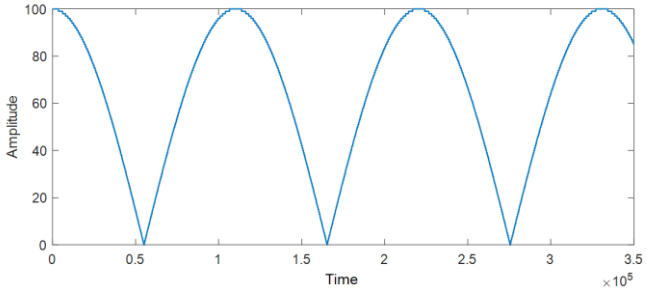

Fig. 13. The final DDS block output.

The whole Flanger effect system designed using XSG is presented in Fig.12. The DDS block compiler generates the sine wave with an amplitude that varies between 1 and -1, the output is then passed through the *Absolute* block to convert

the negative part of the signal into positive, then amplified and rounded off by the *Cmult1* block. In this case, we will obtain a sinusoidal wave with an amplitude that varies from 0 to 100 (after amplification) with a single alternation as shown in the Fig. 13.

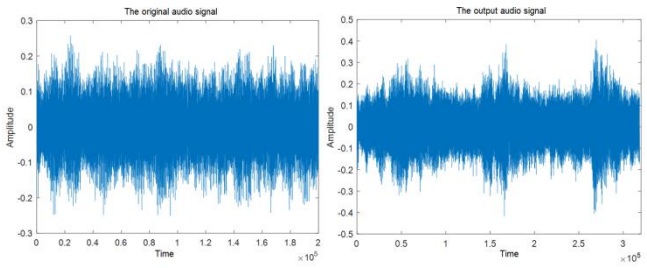

Fig. 14. Simulation result of the anger effect.

From the defined configuration, the delay scan interval varies from $0\,ms$ to $2.2\,ms$ $(100 \times (1/44100) = 2.2\ ms)$, with a center of 1.1 ms. The final output of the DDS block controls a counter, then the maximum value that the counter can reach is defined by the sinusoidal wave phase value (sweep frequency), thereafter the counter will indicate the delay value. This delay varies periodically due to the sinusoidal function. The simulation result of the Flanger effect system applied to an audio signal is presented in Fig. 14.

IV. Hardware Implementation of the Echo and Flanger Effects Systems

In this section the two designs are assembled in a one (Fig. 15) and implemented on the FPGA, the same memory will be shared between the two systems using a multiplexer, this multiplexer will be controlled later by a switch on the ATLYS board. Afterward, the system is exported to the project created previously under ISE. In this step, the compiler will generate the equivalent VHDL code of the whole system from XSG blocks, we can use the XILINX token to do this. The exported files will be added as a component named *SYS_CP* (Fig.16) to the *top.vhd* source, the component has six inputs and one output:

- *clk44100* : A 44.1 kHz clock input incoming from a clock divider subprogram.
- *SOUND_IN* : The audio input vector of 18 bits of size (original audio signal) incoming from the LM4550 driver component (*LM4550_CP*).
- *ce* : An activation input always fixed to 1.
- *SW* : A switch to control the MUX.
- *delta_plus*, *delta_minus* : signals came from the ATLYS on board buttons to control the phase increment value $\Delta\theta$ and therefore control the scanning frequency amount.
- *SOUND_OUT* : The audio output vector of 18 bits of size (audio signal after processing) routed to the LM4550 driver component.

After the generation of the programming file and configuring the board, we can confirm the good results obtained by hearing the output audio on the speakers.

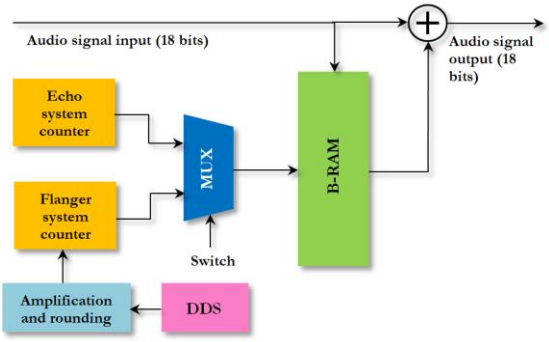

Fig. 15. Designs assembling and memory sharing.

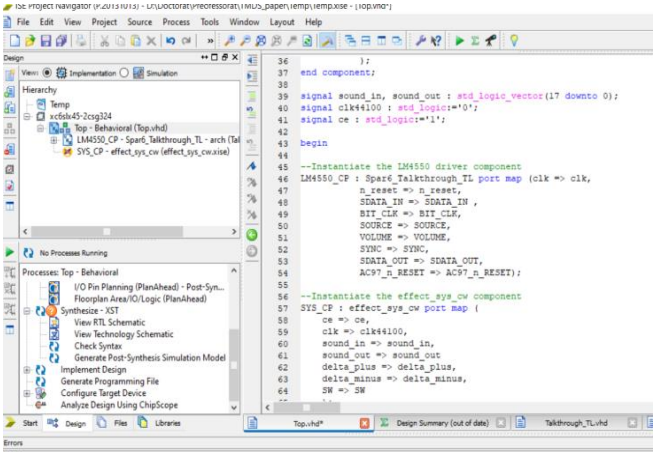

Fig. 16. Adding the whole design as a component on the ISE created project.

V. Conclusion

Through this work, we have transferred some experience in designing digital systems in general, and more specifically audio processing systems using XSG and ISE tools and presenting their FPGA-based hardware implementation steps. It is a good opportunity to understand how the widely used LM4550 AC'97 audio codec works and how to interface it with FPGA using VHDL. Two examples of audio processing systems have been studied and designed in this work, namely the Echo effect and the Flanger effect systems. In fact, giving an idea of how these two systems can be designed and implemented in real-time is sufficient, because many other audio processing systems can be easily derived from these two ones. Chorus, Vibrato, Phaser, Time-Varying and Fractional Delays effects, and other special effects such as; Reverberation and Schroeder's Reverb have in fact the same concept as the implemented systems, of course, with some modifications. What should be taken into account is that the XSG library is richer by a lot of blocks of different arithmetical operation and processing components, the designer can follow the steps presented in this work, and easily design any processing audio system using XSG, generate its equivalent VHDL and put the designed system as a component on the same project, and follow the same steps to program the FPGA (any development board based on Xilinx FPGAs and has the same AC'97 codec can be used with minor modifications on this project).

Conflict of Interest

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

all authors conducted the research, wrote the paper and had approved the final version.

REFERENCES

[1] M. G. Christensen, *Introduction to Audio Processing*, Springer, 2019.
[2] F. L. Luo, *Mobile Multimedia Broadcasting Standards*, New York, NY, US: springer, 2009.
[3] V. E. Shawn, "Pro android media: Developing graphics, music, video, and rich media apps for smartphones and tablets," 2011.
[4] S. Augusto *et al*., *Digital audio Effects*, Springer, 2011, pp. 1-2.
[5] D. Marshall. (2011). Digital audio effects. [Online]. Available: http://users.cs.cf.ac.uk/Dave.Marshall/C\\M0268/PDF/10_CM0268_Audio_FX.pdf
[6] P. Muthukumar *et al*., "FPGA performance optimization plan for high power conversion," in *Proc. International Conference on Soft Computing Systems*, Springer, Singapore, 2018.
[7] H. Scott and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Elsevier, 2010.
[8] P. Evgeni, *Digital Integrated Circuits: Design-For-Test Using Simulink and Stateflow*, CRC Press, 2006.
[9] Atlys Board Reference Manual REV C. Digilent, 2011.
[10] Datasheet. LM4550 AC'97 Rev 2.1 Multi-Channel Audio Codec with Stereo Headphone Amplifier, Sample Rate Conversion and National 3D Sound, National Instruments, DS100972, Rev 2.1, 2004.
[11] T. Storey and S. Larson. (2017). AC'97 Codec Hardware Driver Example. [Online]. Available: https://www.digikey.com/eewiki/display/LOGIC
[12] C. Ababei, *Lab 7: Interfacing FPGA Spartan-6 with AC'97 Codec*, Electrical Engineering Department, University at Buffalo, 2012.
[13] LogiCORE IP Block Memory Generator v6.3, DS512. Xilinx, 2012.
[14] D. Mitchell, *Basic-Synth*, Lulu Press, Inc. 2009.
[15] *Eva Murphy and Colm Slattery, All About Direct Digital Synthesis, Analog Dialogue 38-08*, 2004.
[16] LogiCORE IP DDS Compiler v5.0. Xilinx, DS794, 2011.

Copyright © 2021 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).

Merah Lahcene received the engineering degree in Electronics (Instrumentation) from University Amar Telidji of Laghouat - Algeria in 2004. He worked as an instrumentation engineer within the national company SONATRACH (2006-2008), where he acquired good experience in the maintenance of gas turbines. He received the M.Sc and Ph.D degrees in telecommunication systems from the University of science and technology of Oran - Algeria in 2010 and 2016 respectively. He is currently a lecturer at the department of electronics, University Amar Telidji of Laghouat. His research interests include information security, chaos-based secure information, Random number generators, and signal processing on reconfigurable hardware. He has a number of published works in these contexts. He joined a number of research laboratories such as Advanced Microsystems Engineering Laboratory - the University of Quebec (Ottawa - Canada), Coding and Information Security Laboratory, Coding and Information Security Laboratory, the university of science and technology of Oran (Algeria), and Signals and Systems Laboratory, the University Amar Telidji of Laghouat (Algeria).

Pascal Lorenz received his M.Sc. (1990) and Ph.D. (1994) from the University of Nancy, France. Between 1990 and 1995 he was a research engineer at WorldFIP Europe and at Alcatel-Alsthom. He is a professor at the University of Haute-Alsace, France, since 1995. His research interests include QoS, wireless networks and high-speed networks. He is the author/co-author of 3 books, 3 patents and 200 international publications in refereed journals and conferences. He is senior member of the IEEE, IARIA fellow and member of many international program committees. He has organized many conferences, chaired several technical sessions and gave tutorials at major international conferences. He was IEEE ComSoc distinguished lecturer Tour during 2013-2014.

Ali-Pacha Adda was born in Algeria. He received the engineering degree in telecommunications from the Institute of Telecommunication of Oran - Algeria in 1986; also, he got university degrees in mathematics in 1986 from university of Oran I- Algeria and a magister in signal processing in November 1993, and later he obtained his Ph.D. in safety data in 2004 from the University of Sciences and Technology of Oran. He worked in the telecommunications administration (PTT Oran) in the position of head of telephone traffic for two years (1986 -1988), He is currently a professor (teacher/researcher) in the Electronics department of the University of Sciences and Technology of Oran (U.S.T.O). His research interests are coding, cryptography and security, and digital signal processing using reconfigurable hardware. He is currently the head of LAboratory of COding and Security of Information (LACOSI laboratory).

Hadj-Said Naima received the engineering degree in telecommunications from the Telecommunications Engineering of Oran (ITO) in 1986, and the magister degree also from ITO in (1992) and a PhD from the University of Sciences and Technology, (USTO) Oran (Algeria) in 2005. She is an associate professor at the computer sciences department of University of Sciences and Technology. Her research interests are in the area of digital communications, and cryptography.