Volume 52, Number 4, 2011

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

THE IMPLEMENTATION OF SCHROEDER REVERBERATOR ON AN FPGA PLATFORM USING XILINX SYSTEM GENERATOR

Corina BOTA, Botond Sandor KIREI, Albert FAZAKAS and Marina TOPA
Technical University of Cluj Napoca, Faculty of Electronics, Telecommunications and Information Technology
Str. Memorandumului, Nr. 28, Cluj Napoca, Romania
e-mail: botond.kirei@bel.utcluj.ro

Abstract: The paper presents a Simulink model based FPGA implementation method of the Schroeder reverberator. The Simulink model was created with a Xilinx System Generator toolbox, which aids digital circuit synthesis for FPGA platforms. The artificial reverberator can be configured to simulate the acoustics of different enclosures (ordinary room or concert hall), while device utilization on the target FPGA is low and less than 3% of a XC5VLX110T FPGA device slices are occupied by the designed system. The reverberator can model up to 2s reverberation time, which is a typical value of a concert hall.

Keywords: *reverberations, audio signal processing, real-time processing, FPGA, Schroeder reverberator*

I. INTRODUCTION

Sound reverberation is a phenomenon that takes place in any closed space: theatres, cathedrals, concert halls, rooms, even though it may be imperceptible in small enclosures. Reverberation happens as a consequence of the sound wave reflection, when it hits obstacles (e.g. walls, furniture, people and interior decorations) on their way from the source to the sink. Frequently, speech signal propagated through large rooms (e.g. concert halls) loses intelligibility due to the reverberations.

Artificial reverberators (ARs), devices that model reverberations, are important for acoustic enclosure modeling, because they generate responses that are realistic from psychoacoustic point of view. The reproduction of echoes is mandatory in the testing phase of dereverberation algorithms used in hearing aids, car audio systems and speech recognition applications (where dereverberation is carried out as a preprocessing stage). ARs are often used in special effect software packages or recording studios for reproducing the impression of large enclosures. With the help of ARs the acoustics of a certain room can be reproduced in an anechoic chamber.

Different types of ARs were developed to reproduce the acoustics of enclosures. Among the first, Manfred Schroeder developed ARs in the ‘60s [1]. His ARs consist of a comb filter bank followed by allpass filters (APFs). While Schroeder’s ARs offer a simple way to model enclosures, a listener may find the effect “dry”, because it focuses on the reproduction of the dense, late reverberations, avoiding the effect of early ones. Moorer extended Schroeder’s solution by adding a taped delay line before the comb filters, thus early reverberations were reproduced. Gardner further improved the ARs by inserting frequency selectivity on the feedbacks. Better reverberation impression is achieved with Jot ARs where a feedback network not only filters the echoes of different delays, but also mixes them. A recent trend in ARs development is the reproduction of the head related transfer function (HRTF). HTRF models the propagation of incident sound reflected from the ear, shoulder and ears. Recent research showed that the HRTF has a very important role in reproducing the spatiality impression produced by sounds [2]. The latest trend in reverberation modeling is the simulation of binaural room impulse response (BRIR) that models the pairwise nature of sound perception due to two ears [3].

One of the researches made on this subject is digital implementation of ARs algorithms [4]. This paper describes the signal processing algorithms which simulate a concert hall’s natural reverberations. It studies the Schroeder, Moorer and Gardner ARs, going through two steps: checking the algorithm using a Matlab Simulink model; writing and testing the Verilog code. The results prove the feasibility of the Verilog implementation and demonstrate potential future development for improving rooms’ acoustics.

Unlike the microprocessor based systems, dedicated digital systems offer a higher processing speed. For example an AR was implemented using MicroBlaze soft processor core on FPGA board [5]. The disadvantage of the solution is the impossibility of real-time processing. Using a 100 MHz clock frequency and a 24 kHz sampling rate, the processing of a 5 s sound took approximately 60 s. To achieve real time processing, dedicated digital circuit was developed with the help of the Xilinx System Generator (XSG), a toolbox which offers advantages as: it allows signal processing systems implementation, using only Simulink blocks; it does not need complementary knowledge of digital systems design or hardware description language (HDL); it allows direct implementation of systems, without the need of exporting it in vendor specific design environments; it offers the possibility of real-time processing, limited by the system clock frequency.

The work is organized as follows: in Section II. the theoretical fundamentals regarding comb filters and APFs, the building blocks of many ARs, are addressed; Section III. presents the Schroeder AR model created with the use of XSG. The results of simulations are reported in Section IV. Implementation issues regarding the ARs are discussed in Section V; finally future

Manuscript received November 14, 2011; revised November 28, 2011

developments are presented and conclusions are drawn in Section VI.

II. COMB FILTERS – THE BUILDING BLOCKS OF ARTIFICIAL REVERBERATORS

The comb filter is a basic component of ARs. The amplitude response of a comb filter is a series of vertical peaks, similar to the shape of a comb. This type of filtering can also appear in unwanted situations. In closed spaces, the listener hears a combination of the direct sound and the reflected one. The reflected sound crosses a bigger distance than the direct sound, therefore it is a delayed attenuated version of the direct sound. These two signals create a comb filtering effect when they both reach the listener.

There are two types of comb filters: feedforward (FFCF) and feedback (FBCF).

Feedforward comb filters (FFCF)

The FFCF block diagram is depicted in *Figure 1*. The discrete-time model describing the FFCF is:

$$y(n) = x(n) + \alpha \cdot x(n-K) \quad (1)$$

where α is a scaling coefficient applied to the delayed signal (amplification/atenuation) and K is the delay (number of samples). Applying the z transform to the equation (1), we obtain the transfer function (2).

$$H(z) = \frac{Y(z)}{X(z)} = 1 + \alpha \cdot z^{-K} = \frac{z^K + \alpha}{z^K} \quad (2)$$

The FFCF is one of the simplest forms of finite impulse response (FIR) filter, its impulse response is a combination of the initial impulse followed by a delayed and atenuated version of it. It is a simple echo.

Feedback comb filters (FBCF)

The structure of the feedback comb filter is shown in *Figure 2*. Equation (3) represents the discrete time model of the FBCF and equation (4) is the transfer function of the FBCF.

$$y(n) = x(n) + \alpha \cdot y(n-K) \quad (3)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \alpha \cdot z^{-K}} = \frac{z^K}{z^K - \alpha} \quad (4)$$

This is an infinite impulse response filter (IIR). If the filter is stable, its impulse response is a series of equal distanced impulses, with exponential descending amplitudes. This response can be modeled as an ideal plane wave hitting two vertical and parallel walls, back and forth, until it is completely attenuated. The filter is stable only if $|\alpha| < 1$.

Allpass filters (APF)

An APF changes the phase of the audio wave, leaving the frequency unaltered. It can be defined as a filter that has unitary gain regardless of the frequency [1]. An APF can be obtained by connecting a FFCF in series with a FBCF (see *Figure 3*). These two comb filters have to have the same latency and symmetrical scaling factors.

The functions describing this filter are the following:

$$v(n) = x(n) - \alpha \cdot v(n-K) \quad (5)$$

$$y(n) = \alpha \cdot v(n) + v(n-K) \quad (6)$$

Using the commutativity of the linear time invariant filters, the equations become:

$$v(n) = \alpha \cdot x(n) + x(n-K) \quad (7)$$

$$y(n) = v(n) - \alpha \cdot y(n-K) \quad (8)$$

By combining the last two equations one obtains the transfer function (9):

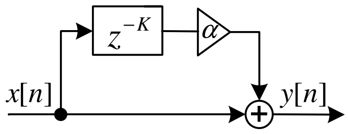

Figure 1. Feedforward comb filter (FFCF)

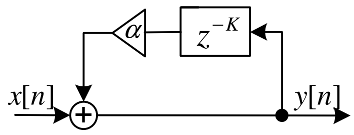

Figure 2. Feedback comb filter (FBCF)

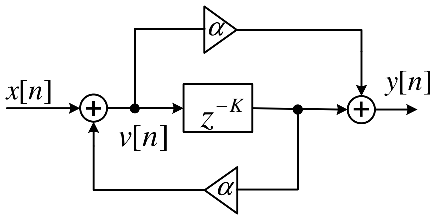

Figure 3. Allpass filter (APF)

$$H(z) = \frac{\alpha + z^{-K}}{1 + \alpha \cdot z^{-K}} \quad (9)$$

III. SCHROEDER REVERBERATOR

An implementation [4], of the first type Schroeder AR comprises four comb filters and two APFs as shown in *Figure 4*.

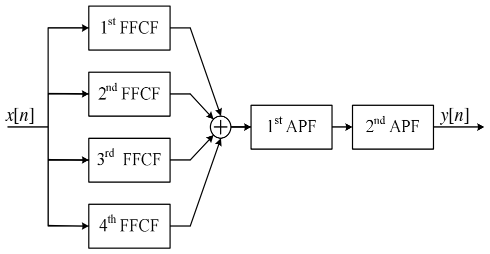

Figure 4. Schroeder Reverberator

Table I. Relevant parameters used in Simulink

Parameter	Value
system clock period	125µs
source file repetition	1
sample time	125µs
sound source output type	double
gateway sample rate	125µs
sample rate for the output audio device	8 kHz

As mentioned in Introduction, we implemented the Schroeder AR using the XSG toolbox in Matlab (see the Simulink model in *Figure 5*). The important sections of the design are marked with dashed lines. The *Signal monitoring* section block allows the user to visualize the input excitation of the AR as well as the response of it. In the schematic, the 4 *FFCF* and 2 *APF* are delimited too. Delay, gain and addition blocks were used for the implementation of the filters.

All Xilinx blocks and systems are digital, therefore they have specific particularities: all blocks are time discrete and have discrete values; all data are integer or fixed-point type because the floating-point representation requires extra resources and computing time. These properties are not implicit in the Simulink environment (i.e. the systems can be discrete or continuous, depending on the solver). Similarly, value continuity or discontinuity depends on the data type used, the implicit data type being *double* (floating point representation). Thus, the use of XSG needs to respect a set of rules:

• any Simulink system containing blocks from the Xilinx toolbox must include the XSG token;

• every Xilinx block have to be interfaced with Simulink system using gateway blocks.

For the analysis and debugging of XSG designs, Xilinx provides a tool named WaveScope. With this tool, the user can visualize any signal in the design, being able to choose the display mode: analog or digital, with binary, decimal or hexadecimal values.

For the simulations, we used a fixed-point data type of 16 bits and 14 fractional bits. Several settings have to be made depending on the sampling rate: system clock period; source file sample rate; gateway sample rate; output audio device sample rate. Their values used in the reported results are summarized in *Table I*.

The advantage of a small sample rate is that, in order to produce reverberations, a small delay is enough (e.g. for a 0.5 seconds delay and a 8 kHz sample rate, the latency parameter of the delay block has to be set for 4000 samples, while for a 44.1 kHz sample rate, the latency should be of 22050 samples).

To verify the functionality of the implemented AR, we simulated each filter separately, and then the whole AR. The response of the four FFCF is added and the result is filtered through two APFs. Each FFCF generates a delayed version of the original impulse. The first APF transforms each echo into a series of echoes and the second one adds another series of echoes. Through this process, the density of impulses grows and the result becomes very similar to natural reverberations.

Table II. summarizes the FFCF and APF parameters (K and α) used in the simulation phase. A test setup was simulated, where delays are set to short values, to verify the behavior of the Simulink model. The second set of parameters are chosen to emulate an ordinary living room, with a reverberation time of 400 ms. Finally, the parameters were set to achieve a reverberation time of *2s,* that is a typical value for concert halls.

IV. SIMULATION RESULTS

Preliminary results are obtained from the simulation of FFCF and FBCF. The delay of the filters are set to *K=4000* and the attenuation to 3dB. For a latency of 4000 samples, the predicted delay is:

$$D = K \cdot T_s = 0.5\sec = 500ms \qquad (10)$$

where T_s = 125µs is the sampling period of the comb filter (or can be interpreted as the clock period of the digital system).

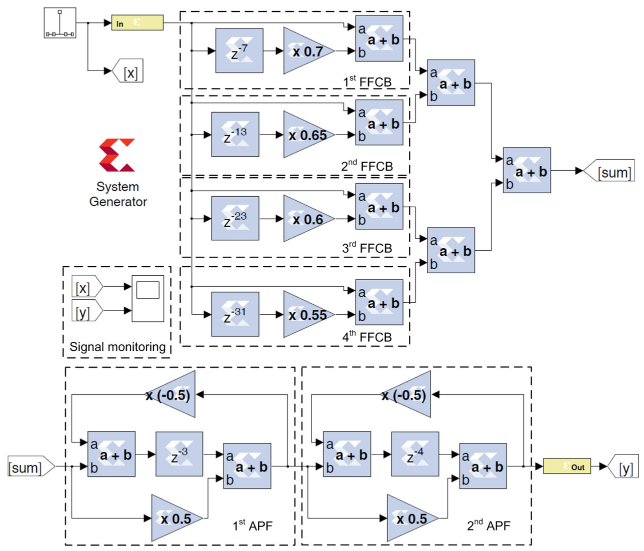

Figure 5. Schroeder reverberator using System Generator blocks

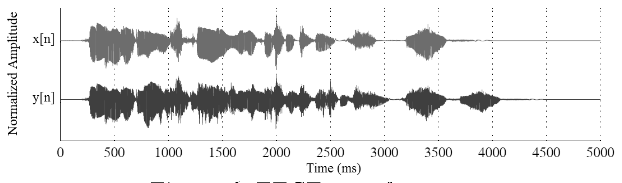

Figure 6. FFCF waveforms

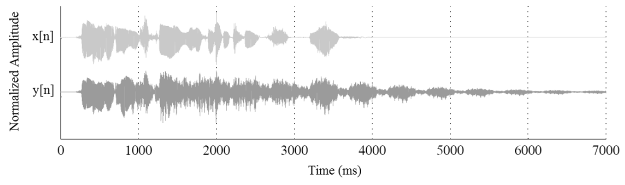

Figure 7. FBCF waveforms

Table II. Delay and gain values used in three scenarios: test setup, ordinary room and concert hall

Filters	Test parameter		Ordinary room		Concert hall	
	K	*α*	*K*	*α*	*K*	*α*
1st FFCF	7	0.7	291	0.7	1500	0.7
2nd FFCF	13	0.65	317	0.65	1820	0.65
3rd FFCF	23	0.6	341	0.6	2010	0.6
4th FFCF	31	0.55	377	0.55	2340	0.55
1st APF	3	± 0.5	41	0.7	120	0.7
2nd APF	4	± 0.5	17	0.7	45	0.7

The responses of these two comb filter to a sound source read from a raw audio file (.wav format) are presented in Figure 6 and Figure 7. It can easily be seen that the FIR filter produces only one echo, while the IIR filter produces several echoes of descending amplitude. In addition, *Figure 8* shows the Schroeder AR's filtering result on the same audio file. The echoes' density is bigger and the AR's effect on the audio signal is smoother than the effect of the IIR filter.

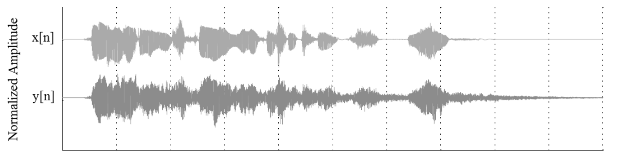

Figure 8. Schroeder reverberator's effect on an audio signal

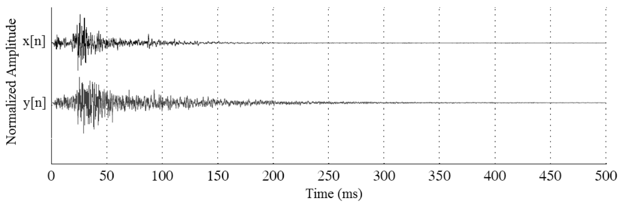

Figure 9. Schroeder reverberator's response to a short-term stimulus (0.4 seconds reverberation time)

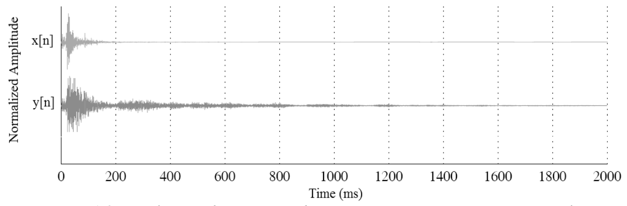

Figure 10. Schroeder reverberator's response to a short-term stimulus (2 seconds reverberation time)

Table III. Area usage on XC5VLX110T device

	Single FFCF	The FFCF and one APF
RAMB36_EXPs	13%	40%
Slices	1%	2%
Slice Registers	1%	1%

In *Figure 9* and *Figure 10*, two different types of response can be seen. In *Figure 9*, the waveforms show the AR's effect on a short-term stimulus, simulating a small room's acoustics. Its reverberation time is around 0.4 s. The settings for this simulation were set according to the specifications in *Table I*. Using the same stimulus, we obtained the waveforms in *Figure 10*, representing a concert hall's acoustics. The parameters were set according to Section III, obtaining a reverberation time of approximately 2 s.

V. IMPLEMENTATION

There are several ways of implementing a Simulink model on a FPGA board: Hardware Co-Simulation, EDK Export, Bitstream generation, ISE Project (NGC code), ISE Project (HDL code).

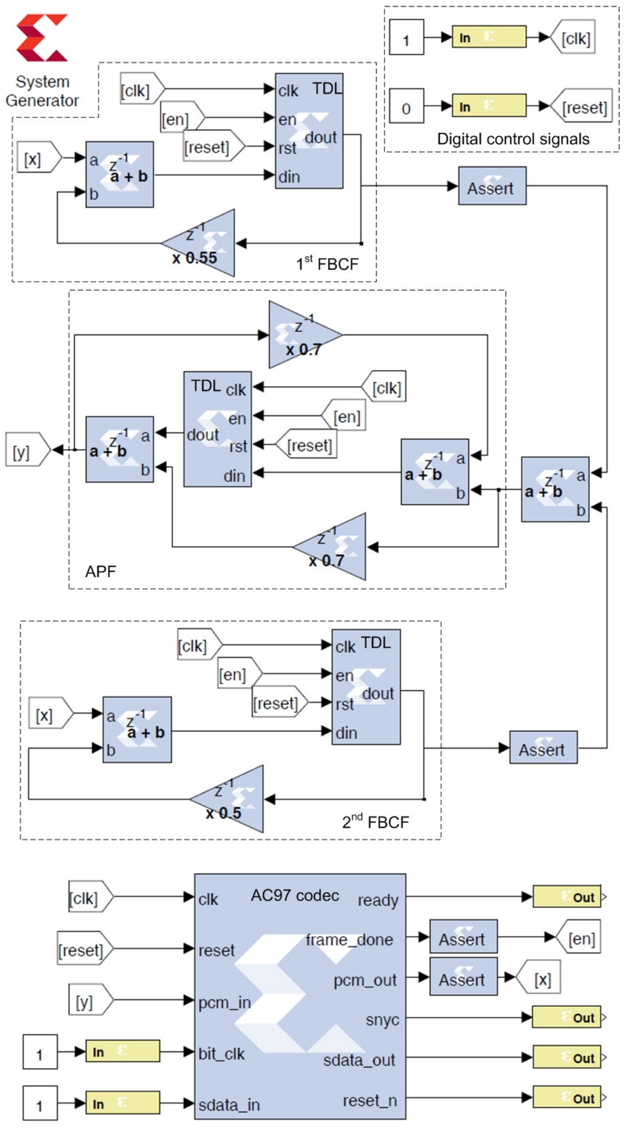

Figure 11. Schroeder reverberator with two FFCF and an APF using AC97 codec

The solution chosen in this case was an ISE implementation using HDL code. For this purpose, we used a controller from a codec developed for the paper in reference number [5], and we built an interface that configures the codec at the startup. This interface can be modeled in XSG, using a state machine and memory. A more direct implementation could be made using Verilog code. XSG allows block implementation in Verilog with the BlackBox constructive block [6]. The implementation of an entire Schroeder AR on an FPGA board encountered some difficulties, due to the fact that the amount of memory needed exceeded the board's resources. Therefore, the real-time processing was made for a partial AR consisting of two FBCF and one APF. In *Figure 11*, the important portions of the AR are framed with dashed lines. Digital control signals (global clock and reset) are fed to the design through "Gateway In" blocks, as well as AC97 codec specific signals (bit clock and serial data input). The delays of the filters are implemented with a taped delay line (TDL), which is described in [7], and it was integrated into the design using the facilities offered by the BlackBox block. The gain and addition blocks have registered outputs (or 1 clock cycle latency). The registering of the outputs results in higher operation frequency than an equivalent combinational logic circuit. Besides the two filter an AC97 audio codec interface is integrated into the design. The "pcm_out" and "frame_done" are the interesting output signals for this application. The "pcm_out" signal provides the signal samples, which are valid if "frame_done" signal is active, processed by the AR. These signals are interfaced with the design using "Assert" block that converts the sample rate between the schematic components. The area usage of the implementations is reported in *Table III*.

The transition diagram for the finite state machine (FSM) used for the AC97 codec configuration is shown in *Figure 12*. The reset signal sets the FSM into the *"Idle"* state, waiting for the AC97 codec to become available. The following six states are used to configure the codec for the desired operation. The names of states follow the next convention: *'Send'*, *'value'* to be written to a status register and the *'address'* of the register (the latter two values are in hexadecimal) separated with underscore character. First, the Master, Headphone, PCM Out and Microphone volumes are unmuted. Next the Record Gain is set to 0 dB. Finally the Microphone is selected for audio acquisition. When the initialization is done, the FSM remains in *"Ready"* state.

In order to make an idea about the efficiency of the XSG toolbox, we compared the synthesis results of an AR described in HDL, using the Xilinx ISE design environment, and the one created with the toolbox. The area usage of both implementations is listed in *Table IV*. As a conclusion we state that implementing with XSG, the used area is not wasted. Therefore, even though the code generated from XSG is a structural description of the system, this type of implementation leads to an efficient utilization of the FPGA device. The reason is that XSG uses optimized blocks, while the synthesizer in a HDL code may interpret the code differently and it may generate non-optimized components.

VI. CONCLUSIONS

In this paper, we studied and implemented the feedforward comb filters, feedback comb filters and allpass filters using Xilinx System Generator, with the purpose of creating an artificial reverberator.

Earlier implementations based on programmed logic proved to be insufficient for realtime reproduction of reverberations, thus a dedicated system was designed on an FPGA platform. The classic FPGA design flow implies the knowledge of an HDL language. An alternative design flow is oggered by the Xilins System Generator toolbox which allows the designer to use a graphical interface.

A typical Schroader reverberator model was created in Simulink using the mentioned toolbox. The created model can be configured to emulate different enclosures. We used the model to emulate an ordinary room and a concert hall. The simulation results were reported in this paper. The target FPGA was a XC5VLX110T device from the Vitex-5 family. The reverberator was interfaced with an AC97 audio codec, which was used for audio signal aquisition. The configuration of the codec is achieved by a finite state machine and its state diagram is reported too.

A brief study of the XSG efficiency in terms of area usage was carried out. We created a similar HDL model, which was created using the Xilinx ISE design environment. It was found that the area usage of both models are comparable.

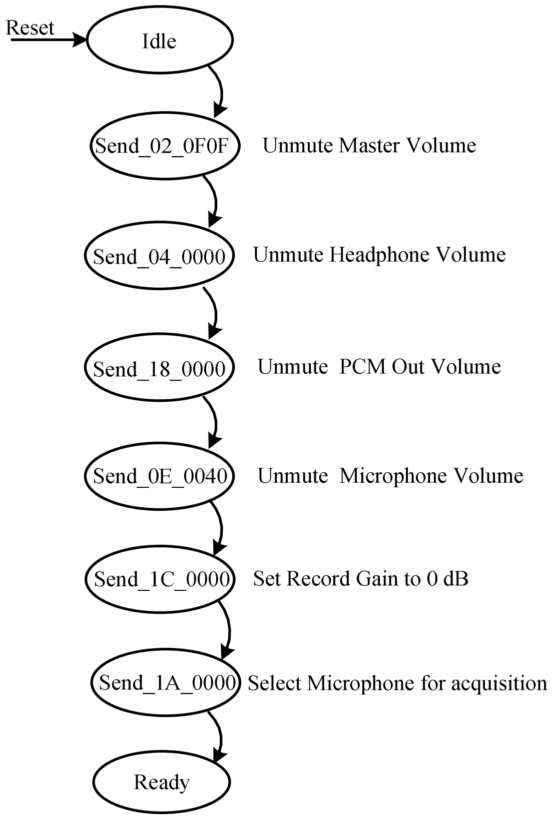

Figure 12. Transition diagram of the FSM used to initialize the AC97 audio codec

Table IV. Area usage comparison of two reverberators designed in Xilinx ISE and XSG

	ISE	Matlab
BUFGs	6%	6%
External IOBs	1%	1%
ILOGICs	0	1%
LOCed IOBs	100%	100%
OLOGICs	0	1%
Slices	1%	1%
Slice Registers	1%	1%
Flip Flops	360	353
LUTS	1%	1%
LUT-Flip Flop pairs	1	1%

ACKNOWLEDGEMENTS

This paper was supported by the project "Develop and support multidisciplinary postdoctoral programs in primordial technical areas of national strategy of the research - development - innovation" 4D-POSTDOC, contract nr. POSDRU/89/1.5/S/52603, project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013 and the grant ID 2534 funded by the Romanian National University Research Council.

REFERENCES

[1] J.O. Smith, "*Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects*", Stanford University, 2007

[2] M.S. Marco Jeun, P. Vary, "A Binaural Room Impulse Response Database for the Evaluation of Dereverberation Algorithms", *Proceeding DSP'09 Proceedings of the 16th international conference on Digital Signal Processing*, Santorini, Greece. July 2009

[3] F. Menzer, F. Christof, "Binaural Reverberation Using a Modified Jot Reverberator with Frequency-Dependent Interaural Coherence Matching", *126th AES Convention*, Munich, Germany, May 7-10, 2009

[4] I. Dornean, M. Ţopa, B. S. Kirei, "Digital Implementation of Artificial Reverberation Algorithms"*, Acta Technica Napocensis - Electronics and Telecommunications,* ISSN 1221-6542, Volume 49, Number 4, pp.1-4, 2008.

[5] S.D.Copacian, "*Aplicatii de achizitie si prelucrare de semnal audio pe placa de dezvoltare XUPV5*", Bachelor's degree dissertation, Basis of Electronics Department, Technical University of Cluj Napoca, 2010,

[6] B. Stewart, "*The Xilinx DSP Primer*", 2009.

[7] B. S. Kirei, M. Ţopa, A.C. Fazakas, N. Toma, "Novel FIR Implementation for Acoustic Signal Processing", *Proceedings of the 15th International Symposium for Design and Technology of Electronics Packages SIITME2009*, 17-20 September, 2009, Gyula, Hungary, pp 387-390.