Reconfigurable Digital Audio Mixer for Electroacoustic Music

David Pedrosa Branco, Iouliia Skliarova, José Vieira
Department of Electronics, Telecommunications and Informatics, IEETA
University of Aveiro
Aveiro, Portugal
dpbranco@ua.pt, iouliia@ua.pt, jnvieira@ua.pt

***Abstract*—Sound diffusion techniques are currently widely employed in the modern music allowing new composition styles and sound movement scenarios to be developed. Multichannel sound diffusion systems are built so as to provide the user with an opportunity to independently control several input channels through the desired output channels. The system MIAUDIO described in the paper allows using up to 8 input channels that can be mixed in real-time through 32 output speakers. The core part of the system which performs the desired audio mixture algorithm was implemented in a Spartan-3E Field Programmable Gate Array (FPGA). The mixture parameters are supplied by a host computer communicating with the FPGA via USB port. The complete system was successfully implemented and tested. The resulting solution has very low cost and was developed in relatively short time.**

Keywords-electronic music; audio mixture; multichannel sound diffusion system; FPGA-based prototyping

I. INTRODUCTION

Multichannel sound diffusion is currently an essential part of the electroacoustic music performance [1, 2]. Modern electronic music composers need to be able to control large arrays of audio loudspeakers distributed over a concert hall (or a laboratory) allowing sound to be placed in any location and with the desired intensity. Besides, recently new applications for sound spatialization have been identified such as virtual reality, multimedia computing, films, videos, and computer games [3]. Different sound scenarios, such as immersion and the possibility of movement of the sound around the audience, are supported by multichannel sound diffusion systems.

SARC [4] and BEAST [5] are some of the existing multichannel sound diffusion systems. These systems use several loudspeakers that are strategically positioned around the audience. The most common disposition is known as the *Main Eight* concept [6]. In this speaker distribution, the listening room is divided in four sections: *Main*, *Wide*, *Rear* and *Distant*. The section *Main* gives the frontal image while the *Wide* is used to stretch that image. The section *Rear* is positioned behind the audience allowing a 360° rotation of the sound. Finally, *Distant*, gives the perception of what is further than the main image.

Although there are exist several electroacoustic theaters, such as SARC, BEAST and ICAST [1], they are out of reach of musicians working in other countries (different to those where the respective theaters are installed) [2]. In particular, there is a strong group of musicians who work in the area of electroacoustic music in the Department of Communication and Art (DCA) in the University of Aveiro in Portugal. This group suggested constructing a digital audio mixture matrix in order to create a multichannel system that:

- would allow spatial movement of the sound;
- would be cheap;
- could be progressively expanded to include new functionalities;
- could be installed in a laboratory in DCA and used by the musicians.

As a result, a MIAUDIO system was developed from scratch in 2009. The core part of the system is implemented in a Field Programmable Gate Array (FPGA) of Spartan-3E family of Xilinx. The system is reconfigurable and therefore new functionalities can be introduced as well as system's parameters can easily be adjusted to new requirements.

The rest of this paper is organized as following. Section II describes and analyzes the existing multichannel sound diffusion systems and points out the main distinctions of MIAUDIO. All the components of MIAUDIO are characterized in Section III. The implementation details of the sound mixture algorithm are explained in Section IV. Section V reports the results of experiments. Conclusion is given in section VI.

II. STATE OF THE ART

From all the electroacoustic theaters the most known are SARC [4] and BEAST [5], which will be briefly described below.

A. *Sonic Arts Research Center*

Sonic Arts Research Centre (SARC), located in Belfast, is a sound diffusion system that features 112 loudspeakers that reproduce the mixture of 24 audio input channels through 48 different outputs [4]. The 112 loudspeakers are strategically installed along four levels. This sound diffusion system is controlled using three Digidesign 192 I/O audio interfaces [7] that interact with a Pro Tools HD3 Accel system. A personal computer, Apple PowerMac G5, runs the software (Pro Tools [8]) and, using the information provided by the Digidesign mixing surfaces, creates the mixture with the audio signals involved.

B. *Brimingham ElectroAcoustic Theater*

The Birmingham ElectroAcoustic Theater (BEAST) is another multichannel sound diffusion system [5]. It was created in the Birmingham University in 1982. This system provides more than 100 speakers where each one can be independently addressed. Similarly to the SARC system, BEAST uses a digital multichannel sound interface that is controlled via specially written applications using MIDI faders with resource to a software known as SuperCollider [9-10]. Using the software, the MIDI faders can be assigned so that they control a single, a pair or a set of speakers. This configuration offers good flexibility to this system.

C. *Analisys of the Existing Systems*

Both systems presented use software based solutions. There is a software tool responsible for the mixture of the audio signals that uses information provided by digital mixture surfaces, or similar hardware. In this implementation method, a fast and reliable operating system is necessary so that real-time processing is guaranteed. The operating system has a great amount of resources dedicated to the sound system control leaving therefore little space to accomplish other possible tasks.

The project described in this paper (MIAUDIO) has its mixing algorithm implemented in hardware. An FPGA is used to receive the audio signals and process them according to parameters that are sent by software. Being so, the software's responsibility is to send the information that defines the audio mixture – a task much simpler and less demanding than processing the mixture itself. This is one of the advantages in MIAUDIO. In software based solutions like in BEAST and SARC, the operating system that produces the mixture has to be extremely reliable and efficient but above all, has to have a great processing power. In MIAUDIO, given the simplicity of the task assigned to the operating system, there is space to introduce several new functionalities as masterization, sound effects, etc.

By adopting a hardware solution implementation, new functionalities can be added, in MIAUDIO, without changing the core of the system. Changes can be made at a higher level. It is possible to add software that interacts with the module responsible for sending the mixture parameters as well as to introduce additional hardware. The FPGA can also be reconfigured to add new features without having to change the rest of the hardware.

Another relevant fact in this project is related to its development time and cost. The MIAUDIO was developed in a relatively short amount of time (approximately 10 person/months) when compared to similar systems. The cost of the components used to assemble the system is under 500 Euros (not including the loudspeakers). The cost of SARC and BEAST is not disseminated but we suppose it is significantly higher.

Different characteristics of the three systems (SARC, BEAST and MIAUDIO) are summarized in Table I.

TABLE I. COMPARISON OF DIFFERENT SYSTEMS

	BEAST	**SARC**	**MIAUDIO**
support software	SuperCollider	Pro Tools	to define
number of speakers	~100	112	32
input/output type	analog/analog	analog/analog	analog/analog
mixture	in software	in software	in hardware
cost	high	high	low
development time	long	long	short

III. MIAUDIO – AUDIO MIXTURE DIGITAL MATRIX

A. *System Description*

MIAUDIO is a multichannel sound diffusion system built around an FPGA of Spartan-3E family [11]. This system has the ability of mixing up to 8 analog input channels through 32 output channels. The analogue input audio signals are conditioned, converted to digital format by several analogue-to-digital converters (ADC) and then sent to the FPGA that performs the mixing algorithm. The host computer connects to the FPGA and is responsible for sending the parameters that define the audio mixture, i.e., send the information that represents the intensity level of each input channel on each output. This topology can be interpreted as a matrix where each coefficient represents the level of each audio input on each output channel.

Fig. 1 represents the system diagram. The host computer sends the parameters that define the audio mixture while the input channels, after the analogue-to-digital conversion, are sent to the FPGA. The resulting output channels are then converted to analogue format so that they can be reproduced.

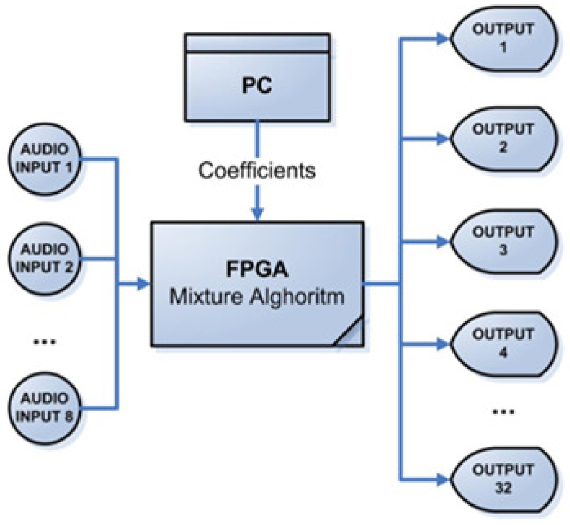

Figure 1. MIAUDIO's system diagram.

B. *Principal Logic Blocks*

Fig. 2 represents a block diagram of the several modules implemented in the FPGA. The *Input block* is in charge of the communication with the analogue-to-digital converters. After receiving a sample of each audio channel, this information is sent to the *Arithmetic block* whose responsibility is to generate the 32 output signals according to the current mixture matrix. To obtain the parameters of the matrix, this block communicates with the *Memory Control block* that manages memory banks embedded in the FPGA where that information is stored. Because the matrix is controlled by a computer, the *USB Communication block* is created to establish the USB interface between the FPGA and the PC. After generating the 32 output samples, the *Arithmetic block* sends this information to the *Output block* that is responsible for properly sending these samples to the digital-to-analogue converters.

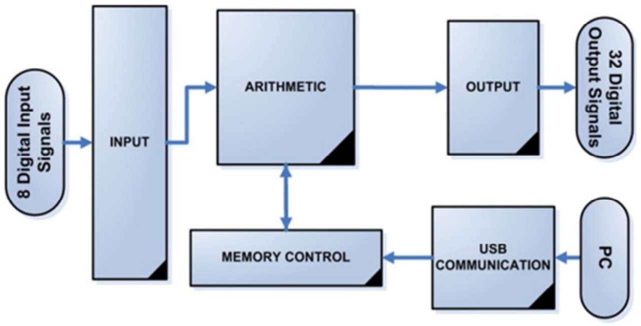

Figure 2. Logic blocks implemented in the FPGA.

C. *Mixture Algorithm*

Each audio input can have a different volume in each output channel. Being so, because there are 8 input channels and 32 outputs, 256 coefficients are necessary to define the audio mixture matrix. Each output can have information of any of the input channels, therefore each channel is multiplied by the coefficient that determines the weight of that input on the respective output and afterwards the 8 products associated with the same output are added.

Fig. 3 represents the relation between the inputs, coefficients and outputs. As mentioned before, there are 256 coefficients that define the audio mixture matrix. Eight input signals are introduced in the system and 32 outputs are generated, being possible that each one of them is different combination of the input audio signals.

D. *Hardware Components*

The system is built around an FPGA of Spartan-3E family [11]. To use this FPGA, the board NEXYS-2 [12] from Digilent was chosen as the design platform. This board has numerous interfaces around the FPGA such as a USB module and several expansion ports that are directly connected to the FPGA. Considering that the analogue input signals are processed digitally, it is necessary to use analogue-to-digital converters as well as digital-to-analogue converters (DAC). The converters selected for this project were PCM1802 ADC [13] and DAC8534 DAC [14], both designed by Burr-Brown Products. Additional hardware is also required to condition the signals to the system.

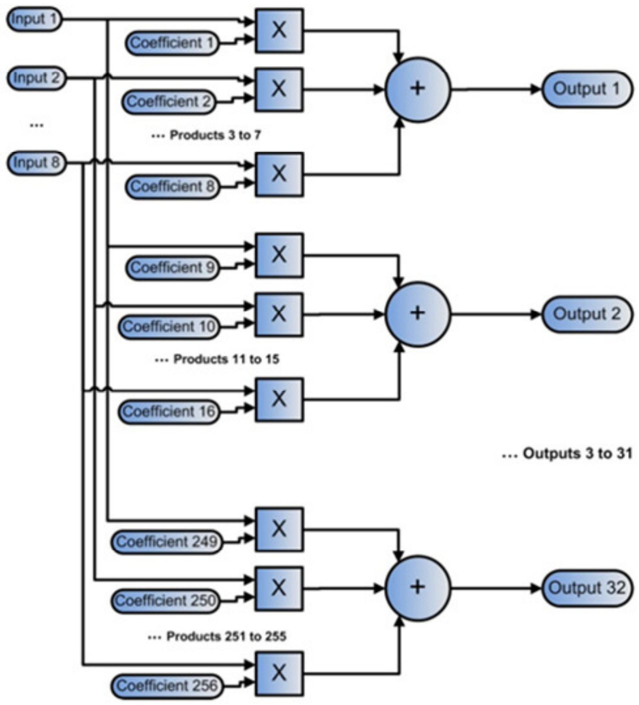

Figure 3. Mixture algorithm implemented in the FPGA.

Fig. 4 represents the input and output stages, for two and four channels, respectively, of the system and their interconnections with the FPGA. The input signal is delivered through XLR [15] cables and introduced into input buffers that convert the signal from the differential format to single-ended format. Then a second order antialiasing filter, implemented with resource to operational amplifiers, is used. The analogue-to-digital conversion is performed and then the resulting information is sent to the FPGA. The input signal is converted with a 24-bit resolution and it is sent by the analogue-to-digital conversion through a serial interface. This transfer is controlled by the ADC.

On the output stage, a similar but symmetric process occurs. The digital information is sent by the FPGA towards the DAC, also through a serial interface. In this case, data have 16-bit resolution. The analogue signal is low-pass filtered and then converted to differential format. To obtain the number of channels desired the input and output blocks depicted in Fig. 4 are replicated 4 and 8 times respectively.

IV. Details of the Mixture algorithm

Evaluating the *Arithmetic Block*, it is possible to observe that a total of 256 products are necessary once considered the products between each input signal and the respective 32 coefficients. Being so, it is crucial that this operation is optimized so that the processing time remains smaller than the ADCs' sampling period. Therefore, dedicated multipliers where used to enhance the system's performance. The received input signals are truncated to 18 bits.

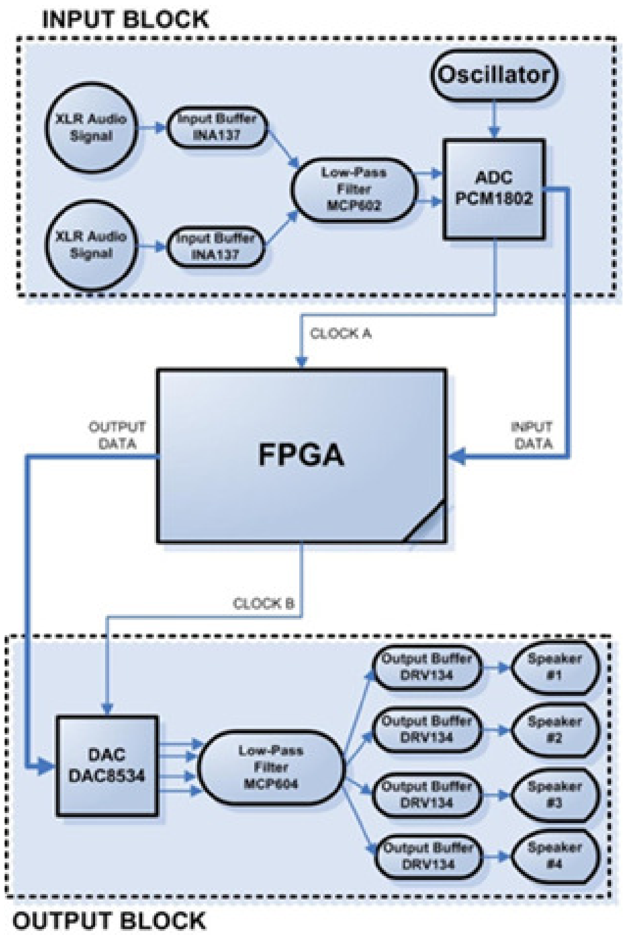

Figure 4. Input and output stages.

A cyclic Finite State Machine (FSM) was created so that 16 of the 20 dedicated multipliers available in the FPGA XC3S500e-4fg320 were used in each loop iteration. 16 iterations are required to obtain the 256 products. To generate one output sample, 8 multiplications are necessary (each output is a combination of the 8 input signals). Being so, each iteration produces two output samples. A rounding algorithm and overflow detection are also done while generating each output signal. Overflow detection is crucial because, after the described products, an eight operand addition takes place increasing therefore the probability of overflow occurrence. The *Arithmetic Block* has all the data represented in two's complement format. The dedicated multipliers as well as the analogue-to-digital converters require this format.

After performing multiplications, the 36-bit results have their 17 less significant bits truncated. Afterwards the sign bit is expanded, producing 21-bit partial result.

Fig. 5 represents the used rounding algorithm. This algorithm is applied after the addition operation is done. Considering that the data samples are, at this point, in two's complement format and are 21 bits wide, to perform the rounding operation, it is necessary to check the most significant bit (with index 20). First, the two less significant bits are evaluated. If they are equal to either "00" or "01", then no rounding is performed and these two bits are simply discarded. Otherwise, the two less significant bits are discarded and either "1" is added if the most significant bit is "0" or '1' is subtracted if the most significant bit is "1". The result of rounding is 19-bit wide.

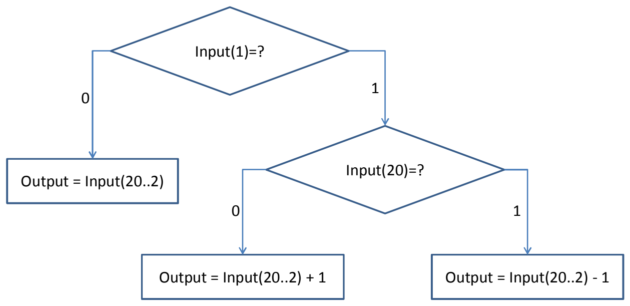

Figure 5. Rounding algorithm.

After rounding, an overflow detection technique is necessary to confirm that no overflow has occurred. To allow for overflow detection, an extra step was taken into account in the arithmetic addition phase. The most significant bit was replicated so that we would have four signal bits in the most significant data bits. This way, it is guaranteed that the resulting most significant bit is intact after adding the eight inputs referenced to a certain output. Fig. 6 describes the implemented overflow detection.

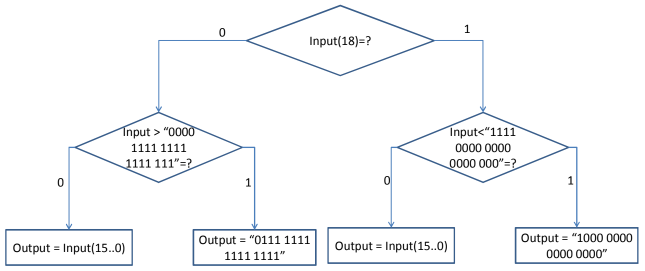
0
Input(18)=?
1
Input > "0000 1111 1111 1111 111"=?
0
1
Input<"1111 0000 0000 0000 000"=?
0
1
Output = Input(15..0)
Output = "0111 1111 1111 1111"
Output = Input(15..0)
Output = "1000 0000 0000 0000"

Figure 6. Overflow detection algorithm.

By evaluating the most significant bit (with index 18) it is determined if the data is bigger or smaller than zero. If the most significant bit is "0", the word is compared to the greatest positive value possible, i.e., to "0000 1111 1111 1111 111" in this example. If the most significant bit is "1" the data are compared with the greatest negative value possible, i.e., "1111 0000 0000 0000 000". When an anomaly is detected (data bigger than the maximum values) the data is assigned to the respective maximum value. We have therefore, saturated overflow detection.

As can be seen in Fig. 4, the ADC PCM1802 generates a clock signal that controls the data transfer considering that the ADC is configured in *Master* mode. Being the clock signal external to the FPGA (it is created by the ADC with resource to an external oscillator, and, in this case, has a frequency different from the 50MHz clock that controls the FPGA logic circuits), a *First-In-First-Out* (FIFO) stack was created. This FIFO is provided by Xilinx (*Xilinx*

LogiCORE™ IP) and has the particularity of having, if desired, different write and read clocks. This module is highly effective and hides possible synchronization concerns from the user. On the output stage, this issue is no longer a problem once the data transfer clock is generated by the FPGA. The digital-to-analogue converter works in *Slave* mode.

Embedded in NEXYS-2 there is a module responsible for managing the USB communication between the connected device and the FPGA. Cypress CY7C68013 [16] is an integrated circuit that interprets the USB communication signals and converts them to a sort of parallel communication. If the respective communication circuit (interacting with the Cypress module) is correctly implemented in the FPGA, the signals generated by the Cypress module are well interpreted and data can be transferred from a computer equipped with USB2.0.

A source file that allows using this communication was provided by Digilent Inc. (manufacturer of NEXYS-2) and adapted to this project. The adaptation consisted in storing the sent information in memory banks embedded in the FPGA. Previously, this information was stored in registers and there were only 16 register available. Considering that 256 registers would be necessary to store the matrix coefficients, it would be a waste of resources. While processing each group of 8 input samples, the memory banks are accessed so that the latest 256 coefficients are used.

V. EXPERIMENTS

To evaluate the MIAUDIO's behavior, several tests were made during and after the final implementation. With the aid of a logic analyzer it was possible to determine the time interval between the beginning of the ADCs' sample transfer and the instant where the DACs receive the corresponding sample. This time interval can be seen in Fig. 7 and Table II, and corresponds to the FPGA processing time. It is equal to 13µs as shown in Table III. Observing t_2 and t_3 duration, it is possible to verify that the sampling frequency is 96KHz. This matches the sampling frequency configured in the analogue-to-digital converters.

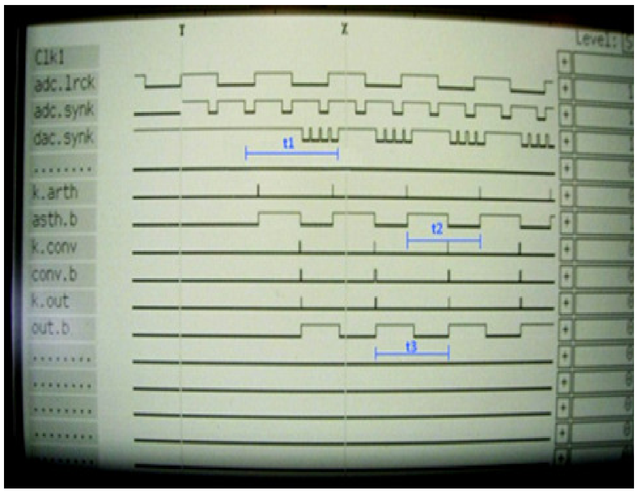

Figure 7. Test time diagram.

TABLE II. SIGNAL DESCRIPTION FOR FIG. 7

Signal	Description
adc.lrck	Designates the channel being sent by the ADC (0 – channel 1; 1 – channel 2)
adc.synk	Represents the ADC data transmission state (1 – sending; 0 – stopped)
dac.synk	Represents the DAC data transmission state (1 – stopped; 0 – sending)
k.arth	Signals the beginning of the *Arithmetic block* processing
arth.b	Represents the *Arithmetic block* state (0 – standby; 1 – active)
k.out	Signals the beginning of the *Output block* processing
out.b	Represents the *Output block* state (0 – standby; 1 – active)

TABLE III. TIME INTERVALS

	Time Interval (µs)	Description
t_1	13.085	Processing Time
t_2	10.4	*Arithmetic Block* Activations Time Interval
t_3	10.4	*Output Block* Activations Time Interval

To measure the input/output delay, a 1KHz sinusoid was introduced at an input channel and forwarded to a certain output. Measuring the phase difference, a delay of 250µs was obtained. The input/output delay is even smaller than this value because the low-pass filter introduces a phase delay to the 1KHz sinusoid used to determine this value. This time interval corresponds to the processing time added to the conversion duration.

The power consumption of the system was another measured parameter. It was detected a maximum of 600mA. This value was obtained with all outputs carrying a signal introduced in one of the input channels. Finally, a spectral analysis was done and the harmonic distortion and noise were measured. A 20KHz cut frequency was obtained. The total harmonic distortion plus noise (THD+N) is equal to 0,09% (Vin=1,28V @1KHz).

Evaluating the results of synthesis and implementation, it was verified that few resources are allocated to implement this project. Sixteen of the twenty embedded multipliers are used to generate two output samples at each cycle iteration on the *Arithmetic Block*'s FSM. This value can be reduced from 16 to 8 by simply generating one instead of two samples per cycle. The number of occupied slices of XC3S500e-4fg320 FPGA is 69% according to the results calculated by Xilinx ISE 11.3 where the algorithm was synthesized, and further implemented. The maximum supported clock frequency of the mixture circuit implemented in the FPGA is 51.4 MHz.

VI. CONCLUSION

MIAUDIO was successfully implemented (see Fig. 8-10). A real-time multichannel diffusion system was created with a very compact and innovative architecture. A low-cost solution was achieved and the development time was relatively short.

Since the digital audio mixture is made in hardware, the computer that defines the parameters of the matrix has most of its resources free to engage in other possible tasks like producing effects over the audio signals, masterization, video synchronization, etc. This system is highly reconfigurable and new functionalities can easily be introduced without having to change the core of the system. The obtained results are quite good given that the input/output delay is extremely low and that the signal quality is assured.

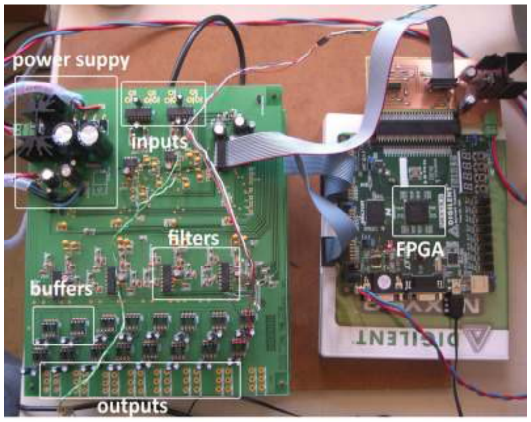

Figure 8. Top-view photo of the implemented system MIAUDIO.

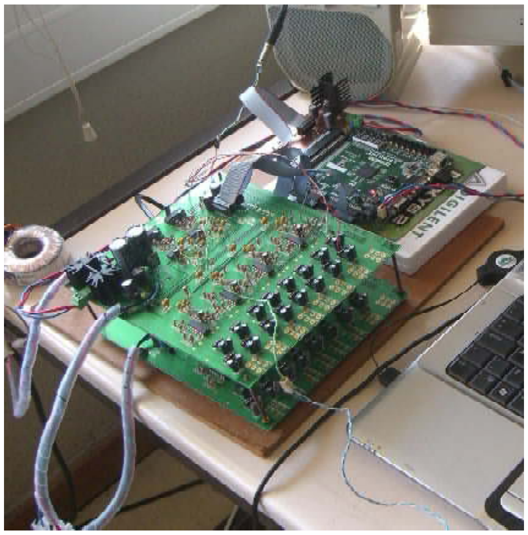

Figure 9. Side-view photo of the implemented system MIAUDIO.

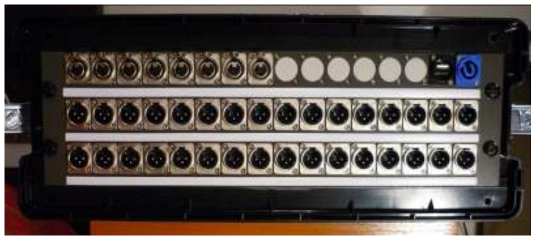

Figure 10. Final system MIAUDIO in the box with all the required input/output connectors.

One of the limiting aspects of MIAUDIO is the lack of efficient interface allowing a musician to interact with sound. Usually, artists recur to mixing consoles and manage the distribution of audio to as many channels as there are loudspeakers in such a way as each channel's fader controls the gain of the speaker to which it is assigned allowing for moving sound in the hall in a straightforward way [2]. What is currently implemented in MIAUDIO is a simple console application (developed in C) which permits coefficients to be sent one by one to the FPGA over a USB connection. This kind of interface is sufficient to test the system's performance but is unsuitable for musicians. Therefore, providing interface with either software which is used by musicians to create compositions or directly with some kind of intelligent controller is one of the directions of future work. Some examples of possible intelligent controllers are appointed in [2], such as a device which translates gestural actions of the composer into spatial location of sound or a strap-on handheld device equipped with 3-D accelerometers.

REFERENCES

[1] S.D. Beck, "The Immersive Computer-controlled Audio Sound Theater: History and current trends in multi-modal sound diffusion", 36th Int. Conf. and Exhibition on Computer Graphics and Interactive Techniques - SIGGRAPH 2009, New Orleans, Louisiana, August 3–7, 2009.

[2] C. Leider, "Multichannel Audio in Electroacoustic Music: an Aesthetic and Technical Research Agenda", IEEE International Conference on Multimedia and Expo - ICME'2007, Beijing, China, July 2-5, 2007.

[3] D.G.Malham and A. Myatt, "3-D Sound Spatialization using Ambisonic Techniques", Computer Music Journal, 19:4, 1995, pp. 58-70.

[4] Sonic Arts Research Center, Queen's University, Belfast, http://143.117.78.181/main.php?page=soniclab.

[5] BEAST and the Electroacoustic Music Studios, University of Birmingham, http://www.beast.bham.ac.uk/.

[6] J. Harrison, "Diffusion: theories and practices, with particular reference to the BEAST system", http://cec.concordia.ca/econtact/Diffusion/Beast.htm.

[7] Avid Technology, Digidesign, 192 I/O, http://www.digidesign.com/index.cfm?itemid=4892.

[8] Avid Technology, the Pro Tools Family, http://www.digidesign.com/index.cfm?navid=349&langid=100&itemid=33116.

[9] The SuperCollider FAQ, http://www.audiosynth.com/scfaq.html.

[10] J. McCartney, "SuperCollider: a new Real Time Synthesis Language", International Computer Music Conference - ICMC'96, pp. 257-258 Hong Kong, 1996.

[11] Xilinx, Spartan-3E FPGA Family: Complete Data Sheet, http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf.

[12] Digilent, Digilent Nexys-2 Board Reference Manual, http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf.

[13] Burr-Brown, PCM1802, Single-Ended Analog-Input 24-Bit, 96-KHz Stereo A/D Converter, http://focus.ti.com/lit/ds/symlink/pcm1802.pdf.

[14] Burr-Brown, DAC8534, Quad Channel, Low Power, 16-Bit, Serial Input, D/A Converter, http://focus.ti.com/lit/ds/symlink/dac8534.pdf.

[15] XLR cables, http://www.rane.com/par-c.html#xlr.

[16] Digilent, Digilent USB 2 Module Reference Manual, http://www.digilentinc.com/Data/Products/USB2/USB2_rm.pdf.