

Audio Mixture Digital Matrix

MIAUDIO

David Pedrosa Branco
Universidade de Aveiro
dpbranco@ua.pt

Iouliia Skliarova
Universidade de Aveiro
iouliia@ua.pt

José Neto Vieira
Universidade de Aveiro
jnvieira@ua.pt

Abstract

Modern music is turning more and more to technologic solutions so that new composition styles and techniques are created. Sound movement is a concept that is gaining strength in this area. Multichannel sound diffusion systems are built to provide the user with the capability to independently control several input channels through the desired output channels. This project (MIAUDIO) allows using up to 8 input channels that can be mixed in real-time through 32 output speakers. A hardware solution was adopted. Eight input analogue audio signals are conditioned, converted to digital format and sent to a Field Programmable Gate Array (FPGA). A host computer communicates with the FPGA via USB and supplies the parameters that define the audio mixture matrix. The FPGA processes this information and sends the resulting signals to digital-to-analogue converters so that the analogue signals are then filtered and reproduced. MIAUDIO was successfully implemented. This is a low-cost solution and its developing time was relatively short. A signal analysis has been made and good results have been achieved.

1. Introduction

Electroacoustic is turning more and more to sound diffusion techniques. With resource to new technologies multichannel sound systems are constructed. These systems allow creating different sound diffusion scenarios, i.e., immersion and the possibility of movement of the sound around the audience. SARC [1] and BEAST [3] [4] are some of 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 [12]. In this speaker distribution, the listening room is divided in four sections: Main, Wide, Rear and Distant. The section Main gives us 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 us the perception of what is further than the main image.

With resource to software tools, the mixture of the input channels is made and most of the times hardware is also used to define the mixing parameters. As we will see, these systems differ from the implementation topology used in MIAUDIO.

2. State of the Art

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. The 112 loudspeakers are strategically installed along four levels. This sound diffusion system is controlled using three Digidesign 192 I/O audio interfaces [6] that interact with a Pro Tools HD3 Accel system. A personal computer, Apple PowerMac G5, runs the software (Pro Tools [7]) and, using the information provided by the Digidesign mixing surfaces, creates the mixture with the audio signals involved.

Birmingham Electroacoustic Theater

The Birmingham ElectroAcoustic Theater (BEAST) is another multichannel sound diffusion system. 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 [2] [5] [13]. 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.

Conclusions and Comparisons

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 article (MIAUDIO) has its mixing algorithm implemented in hardware. A Field Programmable Gate Array (FPGA) is used to receive the audio signals and process them according to the 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 introduced, 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. This project was developed in a relatively short amount of time when compared to similar systems. The cost of the components used to assemble the system is under 500 Euros.

3. MIAUDIO – Audio Mixture Digital Matrix

System Description

MIAUDIO is a multichannel sound diffusion system built around an FPGA of Spartan-3E family [14]. 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 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 so that they can be reproduced.

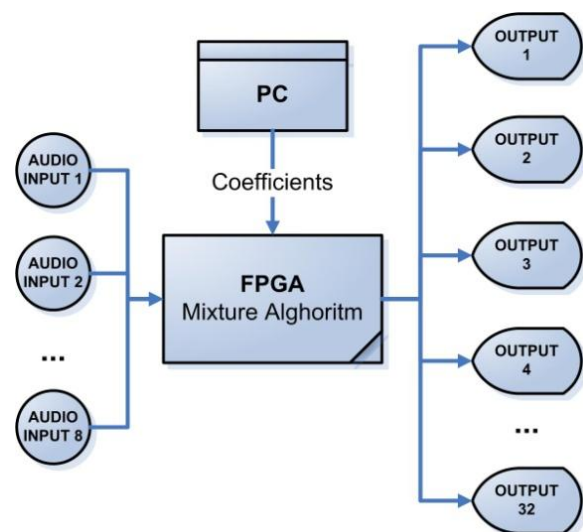


Fig. 1 - MIAUDIO's System Diagram

Internal Logic

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 (ADC). 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 communication 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.

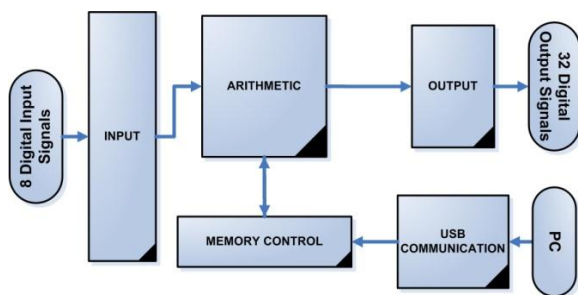


Fig. 2 - FPGA Internal Logic Blocks

Mixing 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.

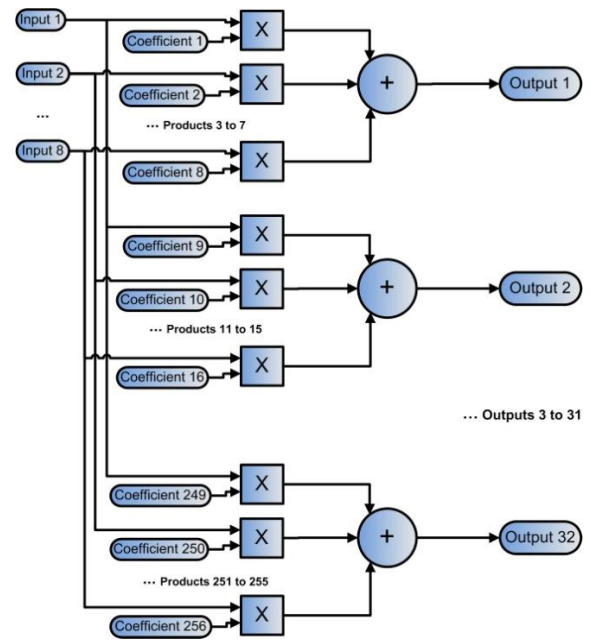


Fig. 3 – Arithmetic FPGA Logic

System Hardware

The system is built around an FPGA of Spartan-3E family [14]. To use this FPGA, the board NEXYS2 [11] 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 (ADC) as well as digital-to-analogue converters (DAC). The converters selected for this project were PCM1802 ADC [10] and DAC8534 DAC [9], both designed by Burr-Brown Products. Additional hardware is also required to condition the signals to the system.

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 [8] cables and introduced into input buffers that convert the signal from its differential format to single-ended. Then a second order antialiasing filter, implemented with resource to operational amplifiers, is used. The analogue-to-digital conversion is preformed 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 has a 16-bit resolution. The analogue signal is low-pass filtered and then converted to differential format. To obtain the number of channels desired these blocks are replicated 4 and 8 times respectively.

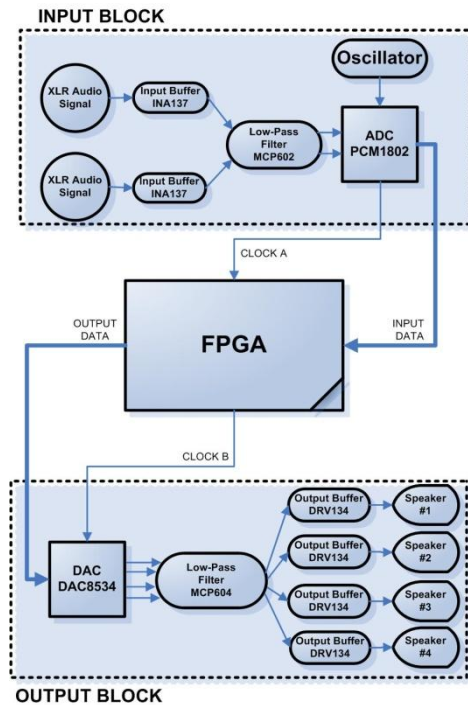


Fig. 4 - Input and Output Stages

Further Implementation Considerations

Evaluating the Arithmetic FPGA Logic, it is possible to observe that a total of 256 products are necessary once considered the products between each input signals 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 were used to enhance the system's performance. A cyclic Finite State Machine (FSM) was created so that 16 of the 20 dedicated multipliers available in this FPGA were used in each loop iteration. Sixteen iterations are necessary 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 is also accomplished 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. These algorithms will be

briefly explained further in this article. The arithmetic block has all the data represented in two's complement format. The dedicated multipliers require this format as well as the analogue-to-digital converters.

Another crucial aspect is related to the clock synchronization. As we can see in Fig. 4, the ADC PCM1802 has a clock that controls the data transfer considering that it is configured in *Master* mode. Being this 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 extracts 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.

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, to perform the rounding operation, it is necessary to evaluate the most significant bit. First, the less significant bit is evaluated. If it is equal to "0", no rounding is performed and these two bits are simply discarded. Otherwise, "1" is added if the most significant bit is "0" or is subtracted if the most significant bit is "1". After rounding, an overflow detection technique is necessary to confirm that no overflow has occurred.

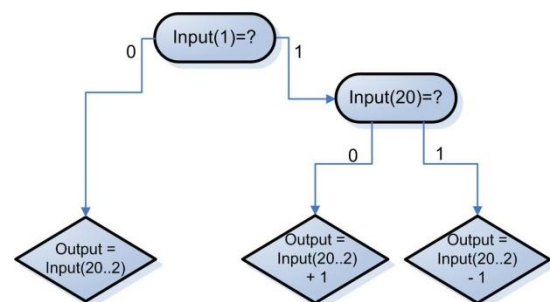


Fig. 5 - Rounding Algorithm

To allow 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.

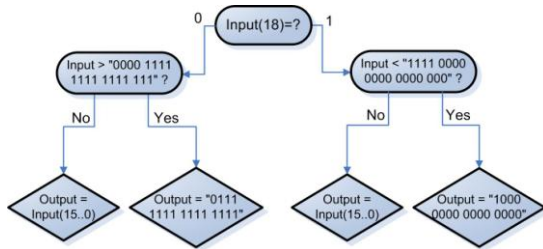


Fig. 6 - Overflow Detection Algorithm

By evaluating the most significant bit 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., “0000 1111 1111 1111 111” in this example. If the most significant bit is “1” the data is 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.

Embedded in NEXYS2 there is a module responsible for managing the USB communication between the connected device and the FPGA. Cypress CY7C68013 [15] 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 to the FPGA.

A source file that allows using this communication was provided by Digilent (manufacturer of NEXYS2) 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.

4. Results

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 ADC’s sample transfer and the instant where the DACs receive the corresponding sample. This time interval can be seen in Fig. 7 and corresponds to the FPGA processing time. It is equal to 13µs as shown in Table 2. 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.

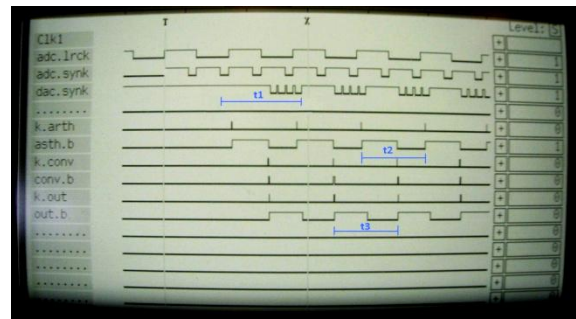


Fig. 7 - Test Time Diagram

| 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 1 - Signal Description

| | 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 |

Table 2 - Time Intervals

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 250 μ s 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% ($V_{in}=1,28V @1KHz$).

Evaluating the FPGA, it is 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. There were only used four of the twenty existing memory banks. As for Look Up Tables (LUT) and Flip Flops, the allocated resources are approximately 30% of the Spartan3E-500 FPGA according to the value presented by Xilinx ISE2.0 where the algorithm was synthesized.

5. Conclusions

MIAUDIO was successfully implemented (Fig. 8). A real-time multichannel diffusion system was created with a very compact and innovative architecture. A *low-cost* solution was achieved and its 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 were quite good given that the input/output delay is extremely low and that the

signal's quality is assured. The *know-how* contained in this project also allows the development of other audio systems like a digital mixing surface.

References

- [1] <http://143.117.78.181/main.php?page=soniclab>.
- [2] <http://www.audiosynth.com/scfaq.html>.
- [3] <http://www.beast.bham.ac.uk/>.
- [4] <http://www.beast.bham.ac.uk/about/meet.shtml>.
- [5] http://www.computermusic.org/members_only/array_issues/spring98/sw_reviews.html.
- [6] <http://www.digidesign.com/index.cfm?itemid=4892>.
- [7] <http://www.digidesign.com/index.cfm?navid=349&langid=100&itemid=33116>.
- [8] <http://www.rane.com/par-c.html#xlr>.
- [9] Burr-Brown. DAC8534, Quad Channel, Low Power, 16-Bit, Serial Input, Digital-to-Analog Converter, September 2002.
- [10] Burr-Brown. PCM1802, Single-Ended Analog-Input 24-Bit, 96-KHz Stereo A/D Converter, January 2005.
- [11] Digilent. Digilent Nexys2 Board Reference Manual, June 2008.
- [12] Jonty Harrison. Diffusion: theories and practices, with particular reference to the beast system. <http://cec.concordia.ca/econtact/Diffusion/Beast.htm>.
- [13] James McCartney. A new real time synthesis language. <http://www.audiosynth.com/icmc96paper.htm>
- [14] Xilinx. Spartan-3E FPGA Family: Complete Data Sheet, April 2008.
- [15] Digilent (September 2004). Digilent USB 2 Module Reference Manual.

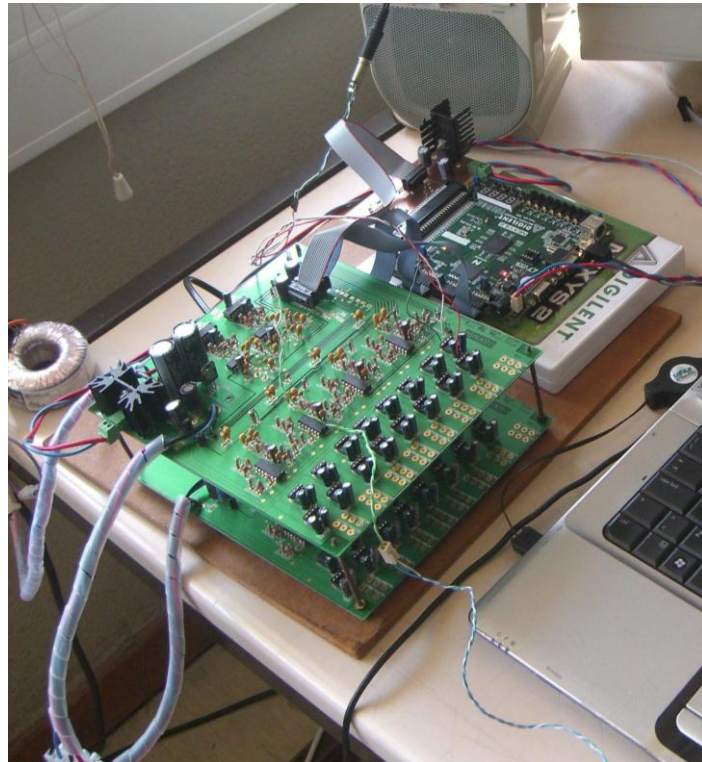
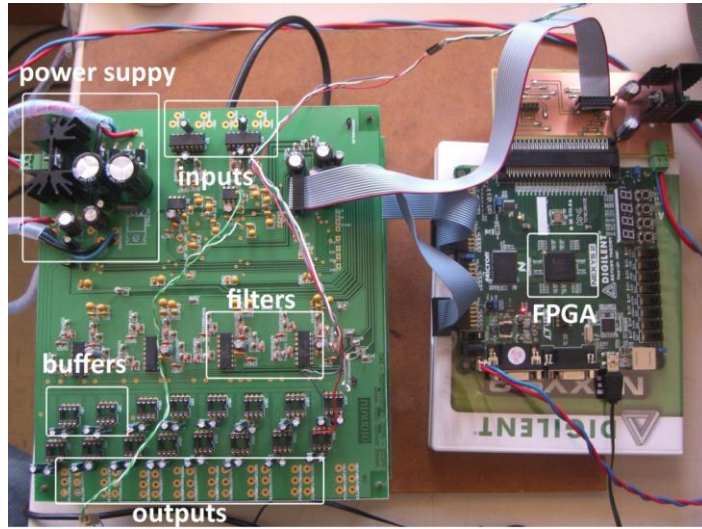


Fig. 8 – MIAUDIO