Modular Delay Audio Effect System on FPGA

Dean Cannon, Tianyang Fang and Jafar Saniie

Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu/)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, U.S.A.

***Abstract*—This paper describes a System-on-Chip implementation of a Modular delay audio effect system. The Zedboard Zynq development board was used to implement the design. The programmable logic on the Zynq processor was used to handle the signal processing and the processing system was used to communicate between the programmable logic and software running on a computer to control different audio effect parameters in real-time.**

Keywords—Audio effect, FPGA, modular design, system on a chip.

I. Introduction

Audio mixers have a wide range of casual and professional applications. However, analog mixers are expensive, bulky, and have no reconfiguration potential. Modular audio effects are audio devices that allow users to change the routing of different audio signal processing elements and control signals, allowing for greater flexibility in sound design than fixed signal devices. It also allows individual modules to be swapped out or replaced with others. This implementation attempts to create a modular system to allow different audio signal processing modules to be routed in any order. The signal router module was designed with 16 audio inputs and outputs. This module can route any of its inputs to any of the outputs. Two different signal processing modules were designed to connect to the signal router, a delay line, and an audio mixer. The software was then written for the processing system of the Zynq SoC to handle communication between the programmable logic and a PC.

FPGAs and other digital hardware have increased use in this topic and related topics over the past decade. Paper [1] presented a reconfigurable digital audio mixer using a Spartan-3E FPGA, which has eight inputs and 32 outputs, and the system was concluded to be cost-effective; In paper [2], a Software-Defined Radio based modular audio mixer is proposed and evaluated to be inexpensive and meets professional use; Paper [3] designed two audio mixing algorithms with automatic gain control on the software-based multipoint control unit(MCU); In paper [4], an integrated-circuit implementation of an eight inputs/2 outputs audio mixer is presented, which dates back to mid-90s; In paper [5], the limitation of analog circuits for audio mixing was discussed, and a VLSI chip design for audio mixing and signal processing was presented. Based on the established research, our research contributes to providing the design flow of a modular audio effect system on an affordable FPGA platform with low demand on the logic unit utilization, which is friendly to low-end FPGAs, and a low delay time throughout the system. We also case studied several published papers[6-10] involving audio mixing techniques for the audio effects used in this research.

In the next chapter, we will outline the design of the audio effect system, including the audio interface and the control units. Chapter III discusses the implementation of the audio processing modules on the FPGA. Then in Chapter IV, we introduce the software design to the control system. In Chapter V, we will evaluate the performance of the designed system. Finally, Chapter VI concludes the paper.

II. Audio Effect System Design

The System on a Chip is implemented on a Zedboard, a development board for the Xilinx Zynq XC7020 SoC. This SoC contains a dual ARM Cortex A9 processing system, referred to as the PS, and an Artix-7 FPGA referred to as the PL. The AXI interconnect is a PS-PL interface that allows fast communication between the PL and PS. An AXI Interface can be created using the Xilinx IP Integrator and added to the project creating registers that the PL and PS can read and write to. The software drivers for the AXI Interface can then be automatically generated and imported into a Vitis project for use in the PS's software. For this system, an AXI-Lite interface is used. There are eight 32-bit ports implemented as registers that the PS can write to, and the PL can read from. These eight values are then sent throughout the PL to control various modules in the design.

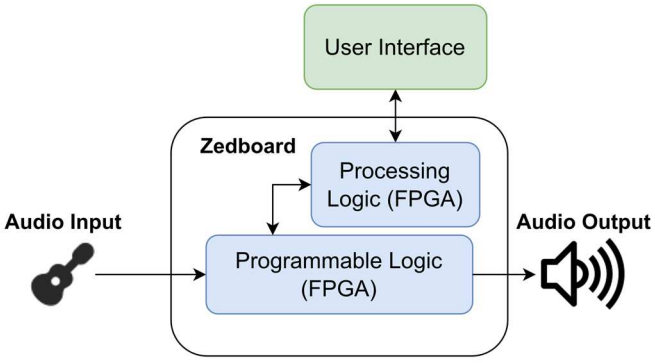

Figure 1. Audio Interface and Control

The AXI interface is integrated into an IP Block called the Audio System on the PL. On top of the AXI Interface and registers, the Audio System also contains an IP block for interfacing to the audio codec on the Zedboard. The Zedboard

has an Analog Devices ADAU1761 I2S stereo audio codec. The zedboard_audio IP block was used to implement an audio interface to and from the audio codec. This IP block generates the clocks necessary for the codec, configures it, and provides stereo audio to and from it. The line-in input on the Zedboard is used for the audio input signal, and the headphone out is used for the audio output. The processes audio in mono but the audio inputs and outputs are stereos. Only the left channel of the inputs and outputs is used from the audio interface to account for this. The right channel of the line input is connected directly to the right channel of the headphone output, which was useful early on in the design to test if the audio interface was correctly passing audio as the right channel could be listened to verify this. The ADAU1761 can be configured either using an I2C or SPI interface. In this case, I2C is used. The audio interface sends and receives audio data to the PL at a sample rate of 48KHz and in the format of Signed Fixed Point Two's complement. The audio interface also provides a 48KHz clock output synchronized to the audio samples used to clock all of the audio processing modules. The VHDL fixed_pkg and data type 'sfixed' were used to perform arithmetic operations on the audio data.

III. Implementation on FPGA

A. Signal Router

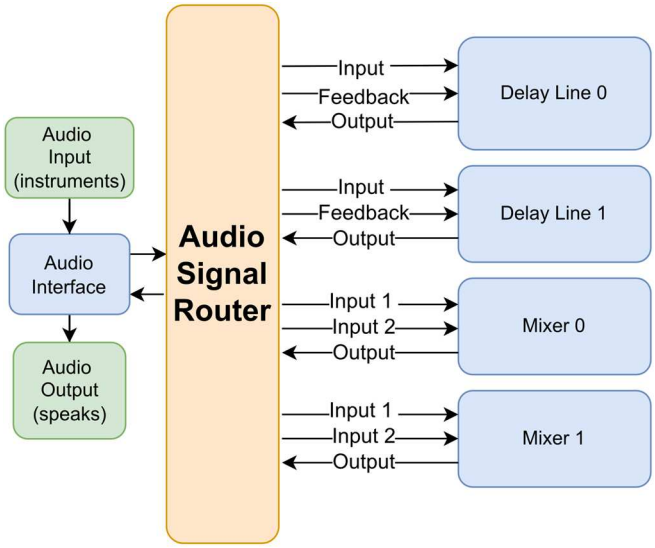

Figure 2. Audio Signal Routing layout. Note that the mixer modules take two Inputs to mix, while delay lines take one input and use a feedback input to create delay effects.

An audio signal router module was created to allow for a modular design. This module has 16 audio inputs and 16 audio outputs. Figure 2 shows the audio signal routing with each audio module connected to the Audio Signal Router. A 16-1 24-bit multiplexer is placed before each output that inputs each audio input to the signal router, which allows for an output to be any of the 16 input signals. Two of the Audio Control System registers are used to control the multiplexers. Each mux requires four bits for its selection signal, so 64 bits are needed to control all of them. A total of four audio processing blocks were implemented on the system, with nine inputs and five outputs. The inputs to the audio processing modules are connected to the outputs of the signal router and outputs of the processing blocks to the inputs of the signal router. Five of the audio signal routers inputs are used, and nine of the outputs. The five inputs include the outputs from the two delay modules, the outputs from the two audio mixers, and the audio input signal from the audio interface. The outputs of the audio signal router are the delay and feedback inputs to the delay modules, the two inputs to the mixer modules, and the audio interface's output.

B. Delay Module

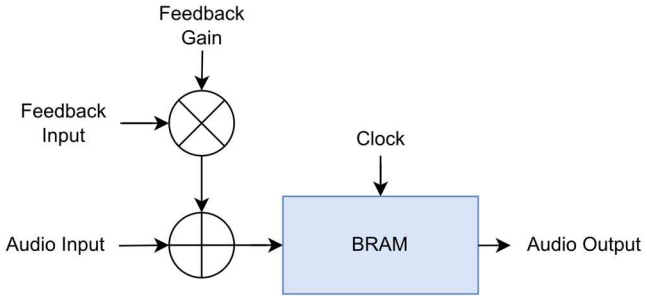

Figure 3. Audio Delay Line

The delay module implemented in this system is an audio effect delay line. Figure 3 shows the diagram for the delay module, which has a few additional features compared to a standard delay line, including a feedback input and a feedback gain control. In addition, the delay utilizes the BRAM on the PL to store the audio inputs. 65536 24-bit memory locations are used per delay line giving a max delay time per delay line of around 1.4 seconds. The delay line module has 24-bit audio and feedback inputs, two clock inputs for the 48KHz audio clock and 100 MHz BRAM clock, a 32-bit delay length input, a 32-bit feedback amount input, and 24-bit audio output. The 32-bit delay length and feedback amount inputs from the audio control interface register. Only 16 bits of the delay length input and 23 bits of the feedback amount are used.

C. Audio Mixer

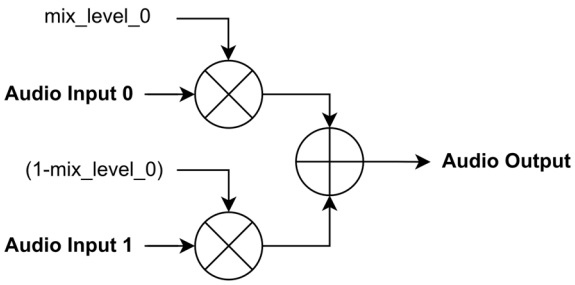

Figure 4. Audio Mixer design diagram

The Audio Mixer module shown in Figure 4 is a mixer that blends the two audio input signals based on its control input, a common control on many audio effects sometimes referred to as wet/dry mix. The "dry" signal is unaffected, while the "wet" is fully affected. The module has two audio inputs, a 32-bit control signal mix_level, a 48KHz audio clock, and a 24-bit audio output. The exact process is followed to ensure only a positive value is used. The two internal signals to this module are mix_level_0 and mix_level_1, which are what the

first and second audio input signals are multiplied by. Mix_level_0 comes from the mix_level control input to the module while mix_level_1 is set to one minus mix_level_0. There is one process for this module. At each rising clock edge, the two audio inputs are multiplied by their mix level values, added together, and then the module's audio output is set to the result.

IV. Software Setup and Performance Evaluation

The software that runs on the PS of the Zynq processor is a simple application that takes inputs from the Zedboard's USB-UART connection, parses them, and writes to the PL audio control interface via the AXI Interface. The drivers to write via the AXI interface were generated by Vivado/Vitis using the "Export Hardware" feature.

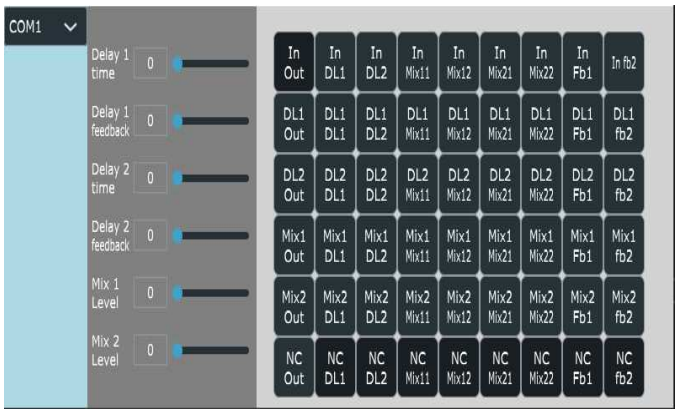

Figure 5. GUI Control Software for the Audio Effect System

The GUI control software, shown in Figure 5, is implemented using the JUCE framework to send commands to the PS of the Zedboard over USB serial traffic. JUCE is a C++ framework used for creating cross-platform applications. The juce_serialport module was used for the software to communicate to the Zedboard. A button matrix is used to control the signal router. The columns on the button matrix correspond to the outputs and the rows to the inputs. Clicking an individual button sends a command to the device to connect that button's input and output. Sliders are used to control each audio signal processing module's control input. A combo box is also used to select which serial port the software is connected to.

Figure 6 shows the FPGA utilization. The two delay line modules use up almost 70% of the PL's BRAM, which is not the most efficient utilization of the BRAM and could cause resource issues if many other modules are implemented. The processor that the Zedboard uses has 512 Mb of DDR3, which may be a better option to use for the delay buffer. The greater size can allow for longer delay times or increase the audio sample rate without worrying about a decrease in max delay time. The read and write speed of the BRAM is not necessary for the delay ram that only deals with a few reads and writes every audio sample.

Resource	Utilization	Available	Utilization %
LUT	1926	53200	3.62
LUTRAM	66	17400	0.38
FF	2360	106400	2.22
BRAM	96.50	140	68.93
DSP	12	220	5.45
IO	10	200	5.00
MMCM	1	4	25.00

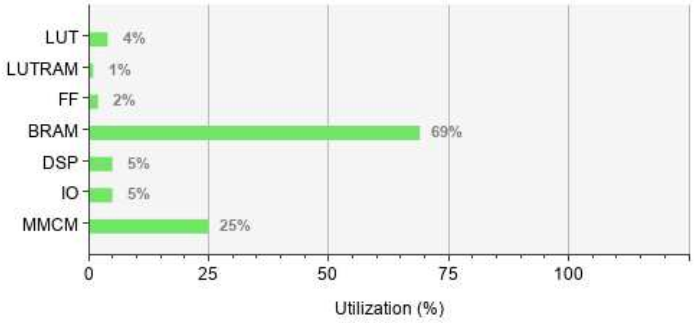

Figure 6. Zynq SoC hardware resources utilization

V. Conclusion

This research implemented a modular delay-based effect on a Zynq SoC. A signal router was used to achieve the ability to route audio signals; however, the user pleases using a GUI application running on a PC. The current signal processing modules for the device include a delay line and an audio mixer. While a modular audio effects processor was implemented, the audio processing modules that the system currently has are essential and can be handled by many cheaper embedded processors. In addition, the implementation provides a platform that allows future audio processing modules to be easily integrated through the signal router. For example, future models could include filters, distortion modeling, and oscillators for audio synthesis.

References

[1] D. P. Branco, I. Skliarova and J. Vieira, "Reconfigurable Digital Audio Mixer for Electroacoustic Music," 2010 International Conference on Reconfigurable Computing and FPGAs, 2010, pp. 132-137, doi: 10.1109/ReConFig.2010.28.

[2] S. Jaloudi, "Software-Defined Radio for Modular Audio Mixers: Making Use of Market-Available Audio Consoles and Software-Defined Radio to Build Multiparty Audio-Mixing Systems," in IEEE Consumer Electronics Magazine, vol. 6, no. 4, pp. 97-104, Oct. 2017, doi: 10.1109/MCE.2017.2714720.

[3] V. M. Baskaran and K. Wong, "Audio mixer with Automatic Gain Controller for software based Multipoint Control Unit," 2010 IEEE Asia Pacific Conference on Circuits and Systems, 2010, pp. 164-167, doi: 10.1109/APCCAS.2010.5774867.

[4] D. Thomson and E. M. Miranda, "A digitally controlled 8-input/2-output audio mixer IC," 38th Midwest Symposium on Circuits and Systems. Proceedings, 1995, pp. 1193-1196 vol.2, doi: 10.1109/MWSCAS.1995.510308.

[5] D. James, N. Mendelsohn and D. Fuchs, "Digital audio mixer: A VLSI approach," ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing, 1982, pp. 77-80, doi: 10.1109/ICASSP.1982.1171384.

[6] D. Koszewski and B. Kostek, "Low-level audio descriptors-based analysis of music mixes from different Digital Audio Workstations – case study," 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2018, pp. 213-216, doi: 10.23919/SPA.2018.8563428.

[7] J. K. Harris, A. J. L. Delin, J. R. Naylor, I. W. Stewart and J. M. Prince, "The application of embedded transputers in a professional digital audio mixing system," IEE Colloquium on Transputer Applications, 1989, pp. 2/1-2/3.

[8] A. Chen and M. A. Hasegawa-Johnson, "Mixed Stereo Audio Classification Using a Stereo-Input Mixed-to-Panned Level Feature," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 12, pp. 2025-2033, Dec. 2014, doi: 10.1109/TASLP.2014.2359628.

[9] P. F. La Rotta Santos, F. Moumtadi and A. Lambertt Lobaina, "A system for capture, transmission and mix digital audio," 2015 International Conference on Computing Systems and Telematics (ICCSAT), 2015, pp. 1-4, doi: 10.1109/ICCSAT.2015.7362961.

[10] P. de Smet and T. V. Stichele, "Subband based MPEG audio mixing for Internet streaming applications," 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221), 2001, pp. 1393-1396 vol.3, doi: 10.1109/ICASSP.2001.941189