FPGA-ENABLED REAL-TIME AUDIO SAMPLING, PROCESSING, AND RECORDING FOR AN ELECTRONIC DRUM SET

Matthew A. Taylor and Mark Rau

Music Technology
Massachusetts Institute of Technology
Cambridge, USA
{tmatthew|mrau}@mit.edu

ABSTRACT

Processing and recording multitrack audio from an electronic drum set is demanding of computational power and hardware resources. In this paper, we present a complete musical instrument system capable of up to 16-channel percussion sampling, processing, and recording, all in real time. The system leverages a field programmable gate array (FPGA) for parallel audio processing and includes audio effects such as pitch shift, delay, reverb, distortion, a virtual analog low-pass filter, and bit crush. The FPGA also provides interfaces for other system hardware, including an Ethernet audio interface and various audio effect control interfaces. The final design has a cost of under $500 and utilizes about half of the hardware resources on an entry-level FPGA, providing a future platform for more advanced percussion synthesis using real-time physical modeling.

1. INTRODUCTION

Processing and recording audio produced by an acoustic drum set is often a resource-intensive task. The drum set typically consists of eight or more drums and cymbals, and it is desirable to record using a similar number of microphones to facilitate post-processing [1]. Audio bleed between the recorded tracks is also challenging to avoid [2]. Research on drum source separation using a small number of recorded tracks is actively being pursued, but state-of-the-art work requires expensive hardware to achieve real-time performance [3].

Electronic drum sets do not require microphones for recording and therefore alleviate the challenges described above. However, multitrack audio recording is generally only a feature available in higher-end electronic drum sound modules. Additionally, there is ongoing research to more accurately reproduce the behavior of acoustic percussion instruments in an electronic system [4, 5, 6].

Previous developments of more realistic percussion physical modeling algorithms [5] and real-time percussion physical modeling [6] led the way towards affordable electronic drums that provide a more authentic playing experience. The percussion synthesizer developed in [6] utilizes an FPGA to efficiently implement a physical modeling algorithm. FPGAs are programmable logic devices which may be used to implement any custom digital circuit design, as long as the timing constraints of the design are satisfied and the hardware utilization does not exceed the resources available on the device. The parallel computing capabilities of FPGAs make them well-suited for physical modeling tasks, and their deterministic latency is beneficial for real-time audio processing. Recently, an FPGA has also been used to implement real-time audio transmission over Ethernet [7], demonstrating the possibility for low-cost and low-latency multitrack audio recording.

In this paper, we demonstrate a complete musical instrument system that uses an FPGA to implement an electronic drum sampler, audio effects processor, and Ethernet audio interface, all of which provide real-time performance. The drum sampler produces ten audio channels (one for each drum and cymbal in a drum set), and four more audio channels are used for four external audio inputs. Any of these audio source channels may be mixed for input to an audio effects processor, which produces a stereo audio output. The Ethernet audio interface supports 16-channel, 48 ksps, 16-bit audio recording, which includes the 14 unprocessed source channels and two channels of processed stereo audio. The system also provides MIDI input for an electronic drum set or other MIDI controller, expression pedal input for live control of audio effects, and a digital audio workstation (DAW) interface for automating audio effect parameters. The total cost of the system hardware, including the FPGA development board and two custom printed circuit boards (PCBs) is less than $500. To our knowledge, there is little prior work integrating real-time audio effects, performance controls, and multitrack audio recording in a low-cost electronic drum system.

The proposed design utilizes roughly half of the resources available on an entry-level FPGA, suggesting the possibility for future integration with a physical modeling synthesis engine. The Ethernet audio interface developed for this system improves on the work in [7] by using a low-latency multirate resampler implemented on the FPGA to suppress the effect of clock drift between the FPGA and a computer.

This project is released as open-source (GPLv3 license) on GitHub, including the PCB designs, FPGA hardware description language (HDL) code, and software code[1].

2. METHODS

2.1. Hardware Design

For this project, we used the Digilent Nexys A7-100T FPGA development board[2], which includes an entry-level AMD Artix 7 FPGA, as well as other features, including an Ethernet PHY, a USB-UART bridge, and DDR2 DRAM.

Two PCBs were designed for this project. The first PCB contains controls for the audio effect parameters, a patch bay for rout-

[1] https://github.com/MatthewATaylor/DigiDrum3000/

[2] https://digilent.com/shop/nexys-a7-amd-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/

ing audio effects, and a MIDI input. The second PCB contains a four-channel audio ADC (PCM1840) for balanced external audio input, a 2-channel audio DAC (TAD5242) for balanced stereo audio output, and a four-channel ADC (ADC104S021) for expression pedal input.

Fig. 1 shows a block diagram of the relevant system hardware, the external hardware interfaces, and a photograph of the assembled system.

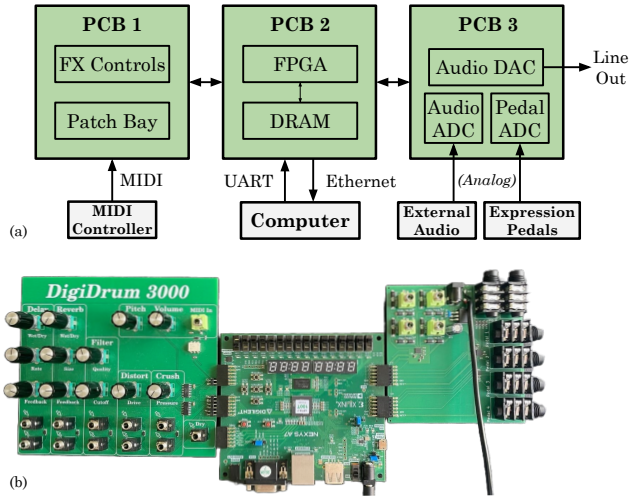

Figure 1: (a) Block diagram of the system hardware. (b) Photograph of the assembled hardware.

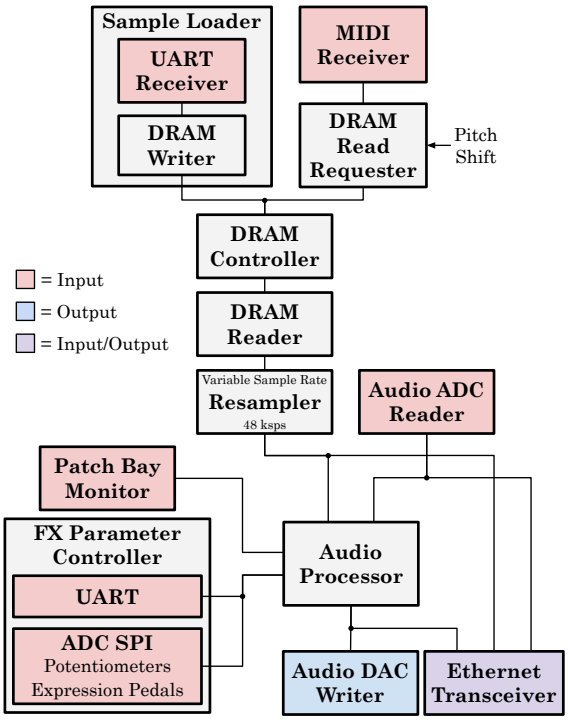

Figure 2: Block diagram of the FPGA design, with blocks colored to indicate input and output interfaces.

2.2. FPGA Design

The FPGA is used to implement a variety of audio digital signal processing algorithms, as well as the digital communication interfaces for the Ethernet PHY, USB-UART bridge, DDR2 DRAM, ADCs (I2S and SPI protocols), and DAC (I2S). A high-level block diagram of the FPGA architecture is provided in Fig. 2. The SystemVerilog HDL is used to program the FPGA hardware, and the Vivado toolchain is used to perform synthesis and implementation of the design. Audio samples are represented as a 16-bit signed integer, and all audio processing is performed in fixed point. The Ethernet transmitter logic is clocked at 50 MHz, the DRAM controller is clocked at 150 MHz, and all other logic is clocked at 120 MHz.

2.2.1. Audio Sampling

On startup, up to about 20 minutes of 16-bit, 48 ksps audio data may be written to DRAM via a 1.5 Mbit/s USB UART link. For this project, we load ten audio files, each containing the recorded sound of a drum or cymbal from an acoustic drum set. After audio samples have been loaded, MIDI input is processed and mapped to a DRAM read request for a particular memory location. A counter increments this memory location at a rate that depends on the desired pitch shift (at most $\pm$ 2 octaves). The MIDI velocity is used to scale the audio received from DRAM.

Because audio samples are retrieved from DRAM at a variable rate, a multirate resampler is used to convert the variable-rate audio signal to a fixed 48 ksps audio signal for processing. The resampler consists of a third-order Farrow structure upsampler [8], which performs Lagrange interpolation on the input signal to provide a fixed 192 ksps output. A 4:1 polyphase downsampler [9] then follows, resulting in a 48 ksps signal. For anti-aliasing, the downsampler uses a Kaiser-windowed 512-tap FIR filter converted to minimum phase for reduced latency. Linear phase is not required in the context of this project because we do not support arbitrary mixing of audio signals in the audio processor.

2.2.2. Audio Effects Processing

A collection of five audio effects are implemented on the FPGA: delay, reverb, distortion, a virtual analog low-pass filter, and bit crush. The PCB patch bay may be used to select the order in which the effects are applied.

The delay effect uses BRAM resources on the FPGA to create a delay line ring buffer with a length of 8192 samples. An instance of the multirate resampler discussed previously is used at the input and output of the delay line to provide a maximum delay time of about 700 ms while preserving limited BRAM resources.

The reverb effect implements the Freeverb algorithm[3], which consists of a parallel combination of eight feedback comb filters followed by a series of four all-pass filters for each of two audio channels. BRAM resources are used to implement the delay lines within each of the filters.

The distortion effect applies a saturator to the input signal using an approximation of the $\tanh$ function that is computationally efficient and straightforward to implement on an FPGA:

$$\tanh(x) \approx \frac{x^3 + 27x}{9x^2 + 27}.$$

[3]https://github.com/sinshu/freeverb

The approximation is related to the $[3/2]$ Padé approximant [10] of the $\tanh$ function, with the coefficients modified to provide an output value of ± 1 at input $x = \pm 3$, as well as a first and second derivative of 0. Therefore, a smooth transition is maintained between the saturator function and a hard clipping region at $|x| > 3$. The distortion effect is oversampled by a factor of four for reduced aliasing of the harmonics generated by the nonlinearity.

The virtual analog low-pass filter effect implements an approximation of a four-pole transistor ladder filter [11] with the structure depicted in Fig. 3. The $\tanh$ block makes use of the $\tanh$ approximation described previously, and similar to the distortion effect, the filter is oversampled by a factor of four.

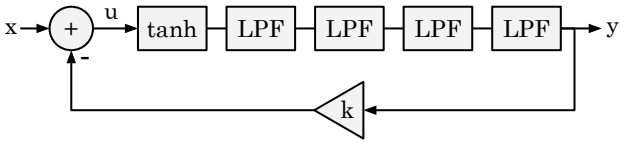

Figure 3: Block diagram of the resonant low-pass filter design. Each LPF block is a 1-pole low-pass filter.

The bit crush effect reduces the fidelity of the audio signal by hard clipping, reducing the sample bit depth, and downsampling without anti-aliasing.

2.3. Ethernet Audio Subsystem

Up to 16 channels of audio may be transmitted by Ethernet from the FPGA to a computer for recording. For this project, two channels are used for the processed stereo audio signal, and the remaining 14 channels contain the unprocessed audio of each drum and cymbal in the sampled drum set, plus the four external audio inputs. Data are transmitted at 100 Mbit/s in raw Ethernet packets consisting of a frame preamble, destination and source MAC address, payload length, 32-sample audio buffer payload, and CRC32 frame check sequence. An Ethernet interface was chosen due to its relatively straightforward implementation (compared to USB audio) and high-speed data rate.

We developed a driver for a Linux computer which processes the raw Ethernet packets that are received and writes the audio buffer payloads to a 512-sample ring buffer. The PipeWire API is used to read from this ring buffer and output an audio stream that may be used by other applications.

If the FPGA were to transmit audio at a constant sample rate, then any clock drift between the FPGA and computer would eventually result in an overrun or underrun in the ring buffer, producing audible pops in the PipeWire audio stream. Therefore, a proportional-integral (PI) controller is used in the computer software to target a 1/2-full ring buffer fill level. The PI controller calculates the desired perturbation in the FPGA audio sample rate, and a resampler on the FPGA responds accordingly. A block diagram of the full Ethernet audio subsystem is shown in Fig. 4.

2.4. Audio Effect Control Interfaces

This project implements three control interfaces for the audio effects processor in order to improve the system's flexibility in music production or performance settings. The first control interface is the PCB containing a patch bay and potentiometers that adjust audio effects parameters. The second control interface allows the connection of up to four external expression pedals to control any

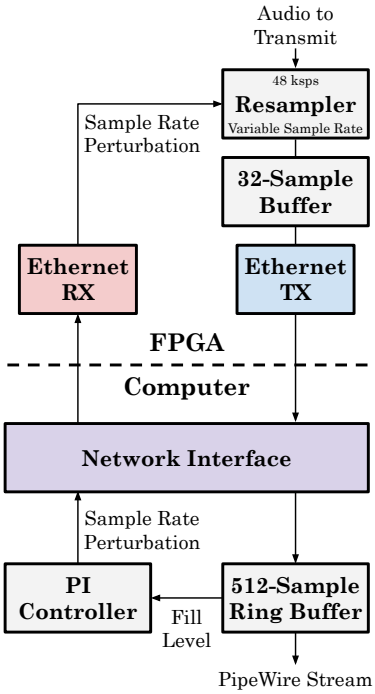

Figure 4: Block diagram of the Ethernet audio subsystem.

audio effect parameter. The scale and offset of each pedal may be individually adjusted for operation within a particular parameter range. The third control interface is a custom Python script which presents the FPGA as a virtual MIDI instrument. The Python script maps MIDI CC controls to each audio effect parameter, and any updates in parameter values are transmitted to the FPGA over USB UART. This control interface allows one to automate any of the audio effect parameters in a DAW.

3. RESULTS

3.1. FPGA Utilization

Table 1 depicts the overall resource utilization for the FPGA design. Each DSP block is a DSP48E1 resource, which contains a 25-bit by 18-bit signed multiplier. One BRAM block refers to either one RAMB18E resource or half of one RAMB36E resource and is equivalent to an 18-bit by 1,024-bit memory.

Table 1: Overall FPGA resource utilization.

Resource	Utilization Count	Utilization Percent
LUTs	36,715	57.9
Registers	39,547	31.2
DSP Blocks	154	64.2
BRAM Blocks	130	48.2

The FPGA resource utilization for each subsystem is shown in Table 2. The ten DSP blocks used in the DRAM interface provide velocity scaling for each of the ten drums and cymbals in the sampled drum set. When possible, the number of DSP blocks

utilized by each audio effect was reduced by performing time-division multiplexing of multiplication operations.

3.2. Latency

3.2.1. FPGA Audio Sampling and Processing

The latency of each FPGA subsystem is also shown in Table 2. To estimate the overall latency of the audio sampling and processing pipeline, we must sum the latencies of the DRAM interface, multirate resampler (consisting of one multirate upsampler and one 4:1 downsampler), and each of the five audio effects. The maximum latency of audio sampling and processing is therefore approximately 1.33 ms, which is subjectively insignificant. However, it must also be noted that the MIDI input latency is significant in comparison (approximately 960 µs per message) because MIDI uses a 31.25 kbit/s UART protocol.

Table 2: FPGA resource utilization and latency by subsystem.

Subsystem	DSP Blocks	BRAM Blocks	Maximum Latency [µs]
DRAM Interface	10	11	667
Multirate Upsampler	2	0	83.3
4:1 Downsampler	2	2	67.7
1:4 Upsampler	1	2	67.7
Delay	14	12	240
Reverb	6	36	0.867
Distortion	6	3	136
Filter	7	3	136
Bit Crush	0	0	0.00833

The relatively large latency in the DRAM interface is due to DRAM data being buffered in 8-sample packets. The number displayed in Table 2 represents the worst-case scenario when the pitch shift is -2 octaves and a sample is triggered just after an internal 8-sample-period counter has reset.

3.2.2. Ethernet Audio Subsystem

The main factors contributing to latency in the Ethernet audio subsystem are audio buffering in the Ethernet payload, Ethernet data transmission, and audio buffering in the PipeWire ring buffer. The 32-sample Ethernet payload buffer contributes about 667 µs of latency, and the 512-sample ring buffer contributes 5.33 ms of latency when its read pointer is 256 samples behind its write pointer (as enforced by the PI controller). Increasing the ring buffer size increases latency, and decreasing the ring buffer size decreases the maximum required quantum size of the follower PipeWire node, which increases CPU demand. While testing on an Intel i7-1255U CPU, audio buffer overruns and underruns were not observed when the ring buffer size was 512 samples or greater.

To determine the latency of Ethernet data transmission, the FPGA was configured to count the number of clock cycles passed between the first transmitted bit of an audio packet and the last received bit of a sample rate perturbation packet generated in response. On average, the latency of this round-trip communication was about 400 µs.

4. CONCLUSIONS

In this work, we demonstrate a low-cost musical instrument system that integrates real-time drum sampling, processing, and recording. The system provides various audio effects and control interfaces that the drummer may use for artistic expression, and the Ethernet audio interface enables real-time, multitrack audio recording to a computer.

The FPGA design utilizes roughly half of the resources available on the FPGA, so future integration with a real-time physical modeling synthesis engine is possible. In future work, it may also be worth exploring software such as Syfala and high-level synthesis tools [12] to reduce HDL development time.

5. ACKNOWLEDGMENTS

The authors would like to thank Charles Reischer for designing the patch bay PCB and associated control logic.

6. REFERENCES

[1] J. M. Eargle, “Popular recording and production,” in *Handbook of Recording Engineering*, 2nd ed. New York City, US: Springer Science+Business Media, 1992, pp. 365–392.

[2] M. J. Terrell and J. Reiss, “Automatic noise gate settings for multitrack drum recordings,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 1–4 2009.

[3] A. I. Mezza, R. Giampiccolo, A. Bernardini, and A. Sarti, “Toward deep drum source separation,” *Pattern Recognition Letters*, vol. 183, pp. 86–91, 2024.

[4] P. Williams and D. Overholt, “Design and evaluation of a digitally active drum,” *Personal and Ubiquitous Computing*, vol. 25, pp. 783–795, 2020.

[5] A. Torin, “Percussion instrument modelling in 3D: Sound synthesis through time domain numerical simulation,” Ph.D. dissertation, University of Edinburgh, 2016.

[6] K. Chuchacz, R. Woods, and S. O’Modhrain, “Towards a real-time implementation of a physical modeling based percussion synthesizer,” in *124th AES Conv. 2008*, vol. 2, Dec. 2008, pp. 692–697.

[7] P. Cochard, J. Weber, R. Michon, T. Risset, and S. Letz, “Ethernet real-time audio transmission to FPGA,” in *2024 IEEE 5th Int. Symp. Internet of Sounds (IS2)*, 2024.

[8] V. Välimäki, “Discrete-time modeling of acoustic tubes using fractional delay filters,” Ph.D. thesis, Helsinki University of Technology, Espoo, Finland, 1995.

[9] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Harlow, Essex, UK: Pearson Education Limited, 2014.

[10] G. A. Baker and P. Graves-Morris, *Padé Approximants*. Reading, Massachusetts, US: Addison-Wesley Publishing Company, 1984.

[11] V. Zavalishin, *The Art of VA Filter Design*. Berlin, Germany: Native Instruments, 2018.

[12] M. Popoff, R. Michon, T. Risset, P. Cochard, S. Letz, Y. Orlarey, and F. de Dinechin, “Audio DSP to FPGA compilation: The Syfala toolchain approach,” Univ Lyon, INSA Lyon, Inria, CITI, Grame, Emeraude, Tech. Rep. RR-9507, May 2023. [Online]. Available: https://inria.hal.science/hal-04099135