TOWARDS HIGH SAMPLING RATE SOUND SYNTHESIS ON FPGA

Romain Michon,[a] Julien Sourice,[b] Victor Lazzarini,[b] Joseph Timoney,[b] and Tanguy Risset[c]

[a]Univ. Lyon, Inria, INSA Lyon, CITI, EA3720, 69621 Villeurbanne, France
[b]Maynooth University, Maynooth, Ireland
[c]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
romain.michon@inria.fr

ABSTRACT

This "Late Breaking Results" paper presents an ongoing project aiming at providing an accessible and easy-to-use platform for high sampling rate real-time audio Digital Signal Processing (DSP). The current version can operate in the megahertz range and we aim to achieve sampling rates as high as 20 MHz in the near future. It relies on the Syfala compiler which can be used to program Field Programmable Gate Array (FPGA) platforms at a high level using the FAUST programming language. In our system, the audio DAC is directly implemented on the FPGA chip, providing exceptional performances in terms of audio latency as well. After giving an overview of the state of the art of this field, we describe the way this tool works and we present ongoing and future developments.

1. INTRODUCTION

Sampling rate selection in the context of real-time audio Digital Signal Processing (DSP) is impacted by a wide range of factors. Beside psychoacoustic considerations (i.e., Nyquist frequency, human hearing range, etc.), aliasing, hardware, and computational power all play an important role. While 48 kHz is fairly standard as it puts the Nyquist frequency (24 kHz) well above the human hearing range, it is fairly common nowadays to see professional audio systems running at 96, 192, 384, and even 768 (in some rare cases) kHz, minimizing aliasing and audio latency (to the detriment of computational power).

Until recently, the use of sampling rates in the megahertz range for real-time applications was mostly reserved to researchers with very specific needs. At this rate, real-time constraints are such that standard processor architectures which are traditionally used for audio DSP (i.e., CPUs, DSPs, and microcontrollers) can't really keep up. That's why ASICs[1] and FPGAs[2] are used when such performances are needed. On this kind of platform, the speed at which an algorithm can be run is mostly limited by the maximum clock tolerated by the system and the "length" of the corresponding electronic circuit (i.e., the time it takes to go from point A to point B in the algorithm).

In the music technology industry, FPGAs have been used for high sampling rate applications in a few rare products. A good example is the Novation Summit[3] which is a "hybrid" analog/digital synthesizer where basic digital oscillators (e.g., sine, etc.) are implemented on an FPGA and computed at 24 MHz. They "market" this as "digital approximating analog," which is certainly appealing in the analog synth community.

Beyond simple waveform oscillators, implementing more complex audio DSP algorithms on an FPGA is a notoriously hard task. The use of Hardware Description Languages (HDLs) such as Verilog or VHDL combined with fixed point arithmetic makes the programming of this kind of platform completely out of reach to most audio DSP programmers. While some high-level environments are available such as Simulink[4] or Vivado Block Design,[5] they're almost all based on the combination of pre-"compiled" ("synthesized"[6] using FPGA terminology) objects, significantly limiting the scope of what can be implemented.

We recently released Syfala[7] [1], the first "audio DSP to FPGA compiler" relying on the FAUST programming language[8] [2]. This open source tool allows us to fully program a series of Digilent development boards (i.e., Zybo Z7-10/20 and Genesys) based on Xilinx/AMD FPGAs to carry out real-time audio signal processing tasks. In this context, we explored a wide range of target applications leveraging the unique performances of FPGAs for audio DSP: ultra-low latency processing [1, 3], spatial audio [4], etc.

While FPGAs can easily keep up with very high audio sampling rates (in the megahertz range, as mentioned above), the fastest audio codec[9] chips available on the market such the Analog Devices ADAU1787 don't go beyond 768kHz. There are two potential solutions to this problem: (i) using general-purpose Analog to Digital/Digital to Analog Converters (ADC/DAC), etc., (ii) implementing audio ADC/DACs directly on the FPGA. Both methods require the use of additional circuitry to implement reconstruction filters, carry out impedance matching, etc. The main disadvantage of (i) is that interfacing an external chip operating in the megahertz range with an FPGA through its GPIOs can be challenging for prototyping if the circuit is not properly shielded. On the other hand, (ii) offers an extremely robust and reliable solution since everything happens directly on the FPGA. If the sampling rate is high enough, there's no need for complex reconstruction filters and a simple RC lowpass filter consisting of a resistor and a capacitor (providing a very slow roll-off) is sufficient for this task.

[1] *Application-Specific Integrated Circuit*.

[2] *Field-Programmable Gate Array.*

[3] https://novationmusic.com/en/synths/summit – All URLs provided in this paper have been verified on April 23d, 2023.

[4] https://www.mathworks.com/discovery/fpga-programming.html

[5] https://www.xilinx.com/products/design-tools/vivado.html

[6] In the context of FPGAs, the word "synthesized" is the equivalent to "compiled" on other platforms.

[7] https://github.com/inria-emeraude/syfala

[8] https://faust.grame.fr

[9] Throughout this paper, "audio codec" will refer to a hardware component implementing an audio ADC/DAC (not an audio compression algorithm).

In this "late breaking result" paper, we present the current status of an ongoing project aiming at providing built-in audio ADC/DAC support as part of the Syfala tool-chain without using any additional complex hardware. Our goal is to offer the same level of performances as commercial audio codecs while allowing for sampling rates in the megahertz range. This will potentially open the way to a broad range of new developments towards improving virtual analog systems, considering audio DSP from a more continuous standpoint, reducing aliasing and artifacts, etc.

First, we give a brief overview of the state of the art of the field of ADC and DAC design as well as of existing works around implementing ADCs and DACs on FPGAs. Then we demonstrate how we implemented a second order delta-sigma ($\Delta\Sigma$) DAC in our Syfala tool-chain and we present its performances. We finally discuss the prospect and potential challenges of implementing a higher order $\Delta\Sigma$ DAC as well as an ADC on an FPGA. We also briefly talk about some possible difficulties related to running audio DSP algorithms in real-time at a high sampling rate.

2. BACKGROUND

2.1. $\Delta\Sigma$ ADCs and DACs

$\Delta\Sigma$ modulation [5] is by far the most commonly used technique for implementing audio ADC and DAC nowadays [**?**]. Most audio codec chips available on the market rely on this method. A first order $\Delta\Sigma$ DAC is based on an integrator, a 1-bit DAC, and a comparator (see Figure 1). It converts a digital signal (i.e., a stream of samples) into a stream of pulses (bits) generated at a high frequency. The more pulses, the higher the analog voltage at the output of the DAC. The audio sampling rate, the clock of the DAC, and the order of the $\Delta\Sigma$ modulator are interconnected parameters which all influence the resulting Signal to Noise Ratio (SNR) [6]. A higher order $\Delta\Sigma$ modulator allows for a lower OverSampling Ratio (OSR) [6]. The OSR directly determines the SNR of the system. For example, if a first order $\Delta\Sigma$ modulator is used with an OSR of 32, then the SNR will be around 40 dB. With an OSR of 32, for an audio sampling rate of 48 kHz, the clock of the $\Delta\Sigma$ modulator has to be 1.536 MHz. On the other hand, if a fifth order $\Delta\Sigma$ (which is the standard used for audio codec chips) modulator is used with the same configuration, then the SNR will increase to 124 dB. Note that there exists many different $\Delta\Sigma$ modulator topology when going beyond second order presenting different advantages and tradeoffs in terms of numerical stability [6]. These SNR figures are independent from the bit depth of incoming samples which can also be a source of quantization noise. Hence, if 16 bits audio samples are provided to a fifth order $\Delta\Sigma$ DAC with an OSR of 32, the resulting SNR should be around 98 dB.

To summarize, a higher order $\Delta\Sigma$ modulator can help increasing the audio sampling rate of the system while preserving a reasonable SNR. Also, the higher the clock of the $\Delta\Sigma$ modulator, the better the performances of the system in all respects.

Once the stream of pulses is generated, it must of course be filtered (lowpass) to reconstruct the analog signal. Using a high audio sampling rate can help decrease the complexity of the filter needed for this task. The lower the order of the reconstruction filter, the more progressive its roll-off. Hence, the -6 dB per octave provided by a simple first order RC filter (which can be implemented with just a resistor and a capacitor) is enough if the audio sampling rate is in the megahertz range.

$\Delta\Sigma$ ADC work in a similar way but are usually slightly more challenging to implement as they imply the use of a hardware comparator which is not necessarily built-in/directly available on FPGAs [7].

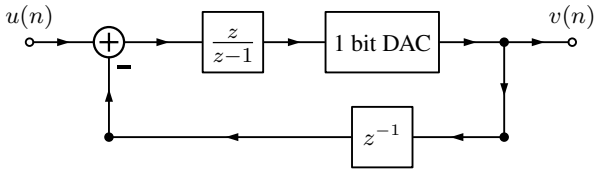

Figure 1: *First order $\Delta\Sigma$ DAC where $u(n)$ is the digital signal input and $v(n)$ is a stream of pulses.*

2.2. $\Delta\Sigma$ ADCs and DACs on FPGAs

FPGAs are a convenient platform for implementing $\Delta\Sigma$ DACs and ADCs (if the FPGA provides differentiated general purpose inputs). Indeed, running a $\Delta\Sigma$ modulator at a very high speed (more than 100 MHz) and connecting its output to a General Purpose Input/Output (GPIO) is trivial. In fact, coding a first order $\Delta\Sigma$ DAC is often a basic exercise/example when learning FPGA programming.[10] Implementing a second order $\Delta\Sigma$ DAC is not that much more complicated and examples of such projects can be easily found on the web.[11] Things get significantly more complex when considering third order and beyond because of stability issues. Hence, while constructing a third, forth, or fifth order $\Delta\Sigma$ DAC is fairly straightforward, formatting coefficients and preventing numerical/rounding errors is potentially very challenging. Various tools such as the Matlab delta-sigma toolbox[12] can help with that. Additionally, various papers on on this topic have been written over the years [8, 9, 10, 11].

$\Delta\Sigma$ ADCs face more or less the same challenges as their DAC counterparts. As mentioned previously, implementing a $\Delta\Sigma$ ADC on an FPGA is significantly simpler if the chip provides differential inputs as those can potentially be used to implement the required differentiator at the beginning of the algorithm [6]. If the FPGA chip doesn't provide differential inputs, then a hardware differentiator should be used, making the design significantly more complex and hence "defeating the purpose" of using an FPGA for this task.

3. IMPLEMENTATION IN THE CONTEXT OF SYFALA

Syfala [1] allows us to run FAUST programs on Xilinx FPGAs-based boards such as the Digilent Zybo Z7 or Genesys without having to write a single line of hardware description language code (which is normally used to program FPGAs).

The standard version of the Syfala compiler can target various audio codec chips including the one built-in on the Digilent FPGA Zybo (see Figure 2) and Genesys boards. We implemented a custom first and second order $\Delta\Sigma$ DAC integrating to the Syfala

[10] https://www.fpga4fun.com/PWM_DAC_2.html

[11] https://github.com/hamsternz/second_order_sigma_delta_DAC

[12] https://www.mathworks.com/matlabcentral/fileexchange/19-delta-sigma-toolbox

tool-chain and that can be used as an alternative for audio codecs. Hence, if the `-sd` option is used when calling the Syfala compiler, the audio output of the system is implemented through a second order $\Delta\Sigma$ DAC. The current version is multichannel which means that a new $\Delta\Sigma$ DAC is instantiated for each output of the FAUST DSP program and associated to a GPIO on the board.

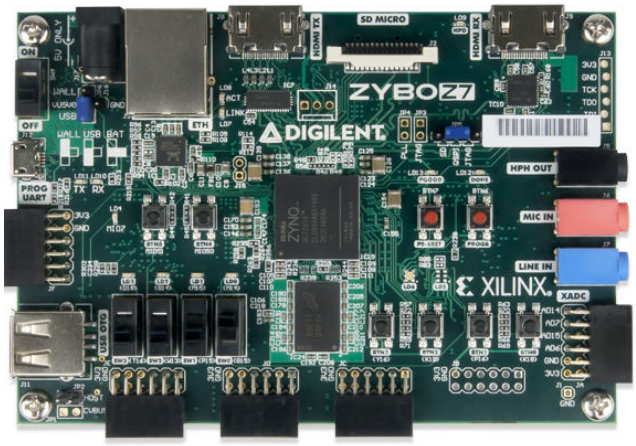

Figure 2: *The Digilent Zybo Z7 FPGA board used for this project.*

When the `-sd` option is used, the audio sampling rate of the system is automatically switched to 5 MHz. The master clock of the FPGA is 125 MHz yielding an OSR of 25 and hence providing a SNR of about 70 dB. At such a high sampling rate, using a one pole RC filter as described in §2.1 is acceptable and is enough to minimize aliasing. Hence, each DAC GPIO must be connected to such a filter. For a cut-off frequency of 20 kHz, a 880 Ω resistor and a 0.01 μF capacitor can be used. A 10 μF capacitor should also be put in series to get rid of DC. Different values for the sampling rate can be specified too using the appropriate Syfala option (`--sample-rate`), bearing in mind that diminishing its value increases the SNR and vice versa.

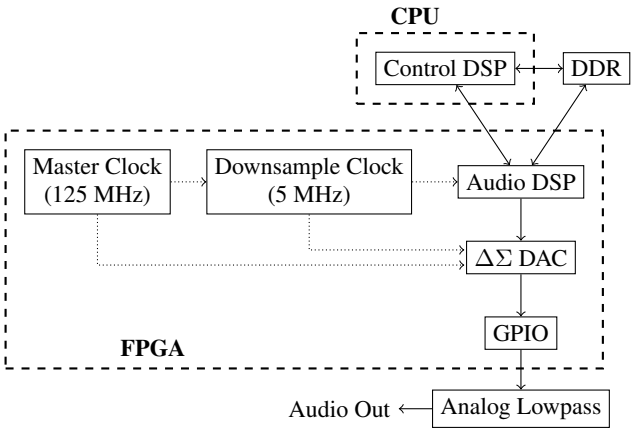

Figure 3: *Implementation overview of the system. Clock signals are depicted with dotted arrows.*

As the $\Delta\Sigma$ DAC is seamlessly integrated to the Syfala tool-chain (see Figure 3), all the other functionalities of this environment remain active/available (i.e., use of DDR for long delays, control computations happening on the ARM processor which is part of the board, etc.).

4. HIGH SAMPLING RATE AUDIO DSP

Running audio DSP algorithms in real-time at a high audio sampling rate can present various challenges which are often related to precision/numerical errors. This of course can greatly vary from one algorithm to another, but obviously running more samples through a filter or computing a wave-table oscillator index using a phasor based on a delta increment can all be significantly impacted by this. A good example of that is the default sine wave oscillator in FAUST which is based on a wave table (represented here by the `sin` function) and whose implementation takes the following form:

```
phasor(freq) = (+(freq/ma.SR) ~ ma.frac);
osc(freq) = sin(phasor(freq)*2*ma.PI);
```

The `~` in `phasor` represents a recursive signal and `ma.frac` yields the fractional part of a decimal number. In that case, `freq/ma.SR` is not precise enough at 5 MHz to provide an accurate frequency. This is just a simple example to demonstrate that high audio sampling rate can be really enabling on one side but can also creates many issues on the other.

5. FUTURE WORK

The ultimate goal for this project is to eventually integrate a 5th order $\Delta\Sigma$ DAC as well as ADC to the Syfala tool-chain.

As mentioned in §2.2, implementing a 5th order $\Delta\Sigma$ DAC on an FPGA is not trivial because of the potential instability of this kind of algorithms due to numerical/rounding errors. This problem is reinforced by the fact that these errors tend to get worse as the clock of the $\Delta\Sigma$ modulator increases.

VHDL-based solutions do exist though [11] and we plan to exploit them to potentially reach this goal. Alternatively, we're also particularly interested in investigating the potential use of FAUST for writing these algorithms and comparing their performances with their VHDL counterparts. The FAUST version would mitigate numerical errors because of the use of floating points whereas the VHDL version would probably be more efficient from a computational standpoint but more prompt to rounding errors. It would also be interesting to investigate the use of FloPoCo[13] [12] (which is a tool for generating arithmetic cores on FPGAs) in this context.

Along the same lines, we hope to be able to provide a $\Delta\Sigma$ ADC in Syfala using similar approaches to that described in [7]. This should be possible on the boards supported by Syfala which all have differential general purpose inputs. As for the DAC, it will be interesting to compare a FAUST implementation to a VHDL one.

6. CONCLUSIONS

The domain of high sampling rate real-time audio DSP is widely unexplored because it has been out of reach for a very long time. Many algorithms would benefit from running at a higher sampling rate, mitigating artifacts and potentially opening new possibilities such as improving virtual analog systems, considering audio DSP

[13] `https://www.flopoco.org/`

from a more continuous standpoint, etc. The system that we presented in this paper and that we're currently developing provides an accessible and easy-to-use platform for this kind of applications without making compromises in terms of performances, quality, etc. Beyond this, it might also allows us to simplify the overall design of Syfala by completely getting rid of audio codecs and providing even better latency performances.

7. ACKNOWLEDGMENTS

This project has been partially funded by the French ANR (Agence Nationale de la Recherche) through the FAST project[14] (ANR-20-CE38-0001).

8. REFERENCES

[1] Maxime Popoff, Romain Michon, Tanguy Risset, Yann Orlarey, and Stéphane Letz, "Towards an fpga-based compilation flow for ultra-low latency audio signal processing," in *Proceedings of the 2022 Sound and Music Computing Conference (SMC-22)*, Saint-Étienne, France, June 2022.

[2] Yann Orlarey, Stéphane Letz, and Dominique Fober, *New Computational Paradigms for Computer Music*, chapter "Faust: an Efficient Functional Approach to DSP Programming", Delatour, Paris, France, 2009.

[3] Loïc Alexandre, Pierre Lecomte, Marie-Annick Galland, and Maxime Popoff, "Feedback acoustic noise control with faust on fpga : application to noise reduction in headphones," in *Proceedings of the 2022 Sound and Music Computing Conference (SMC-22)*, Saint-Étienne, France, June 2022.

[4] Romain Michon, Joseph Bizien, Maxime Popoff, and Tanguy Risset, "Making frugal spatial audio systems using field-programmable gate arrays," in *Proceedings of the 2023 New Interfaces for Musical Expression (NIME-23)*, Mexico City, Mexico, 2023 (Paper accepted to the conference and to be published in June 2023).

[5] Francis DeJager, "Deltamodulation, a method of pcm transmission using the 1-unit code," *Philips Research Reports*, vol. 7, no. 6, pp. 442–466, 1952.

[6] Richard Schreier, Gabor C Temes, et al., *Understanding delta-sigma data converters*, vol. 74, IEEE press Piscataway, NJ, 2005.

[7] PA Harsha Vardhini, "Analysis of integrator for continuous time digital sigma delta adc on xilinx fpga," in *Proceedings for the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 2689–2693.

[8] AJ Magrath, IG Clark, and MB Sandler, "Design and implementation of a fpga sigma-delta power dac," in *Proceedings for the 1997 IEEE Workshop on Signal Processing Systems. SiPS 97 Design and Implementation formerly VLSI Signal Processing*. IEEE, 1997, pp. 511–521.

[9] Ralf Ludewig, Oliver Soffke, Peter Zipf, Manfred Glesner, Kong Pang Pun, Kuen Hung Tsoi, Kin Hong Lee, and Philip Leong, "Ip generation for an fpga-based audio dac sigma-delta converter," in *Field Programmable Logic and Application*, Jürgen Becker, Marco Platzner, and Serge Vernalde, Eds., Berlin, Heidelberg, 2004, pp. 526–535, Springer Berlin Heidelberg.

[10] R.C.C. Cheung, K.P. Pun, S.C.L. Yuen, K.H. Tsoi, and P.H.W. Leong, "An fpga-based re-configurable 24-bit 96khz sigma-delta audio dac," in *Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT) (IEEE Cat. No.03EX798)*, 2003, pp. 110–117.

[11] Zbigniew Kulka and Marcin Lewandowski, "An fpga-based sigma-delta audio dac," in *New Trends in Audio and Video / Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2008*, 2008, pp. 39–42.

[12] Florent De Dinechin and Bogdan Pasca, "Designing custom arithmetic data paths with flopoco," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[14] `https://fast.grame.fr`